Blockchain Service

Developer Guide

Issue 01

Date 2023-10-09

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

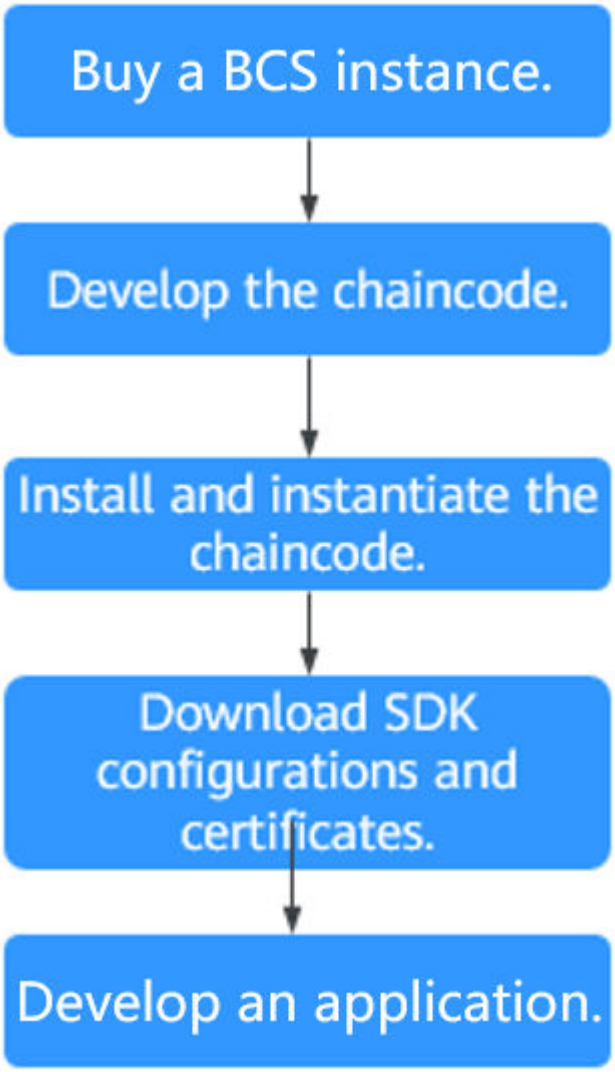
Contents

1 Overview

To use Blockchain Service (BCS), you must develop your own chaincodes and applications. This document describes chaincode development and application configuration for developers with Go or Java development experience.

The process is as follows:

Figure 1-1 Development process



1. Buy a BCS instance.

   BCS instances can be deployed in CCE clusters. For details, see **Deployment Using a CCE Cluster**.

2. Develop the chaincode.

   A chaincode is a program written in Go, Java, or Node.js for operating the ledger. For details, see **Chaincode Development**.

3. Install and instantiate the chaincode.

   BCS provides a graphical user interface (GUI) for chaincode management, including chaincode installation and instantiation. For details, see **Chaincode Management**.

4. Download SDK configurations and certificates.

   Before developing an application, obtain the required SDK configuration file and certificates. For details, see **Downloading SDK Configurations and Certificates**.

5. Develop an application.

Develop your own application code. You can use the **SDK** provided by BCS or the SDK provided by the official Fabric community that matches the version of your instance. For details, see **Application Development**.

You can also use **Homomorphic Encryption** for your instance.

NOTE

If you have requirements for service chaincode or client app design and development, reach us by **sales@huaweicloud.com**. BCS and its partners will provide you with comprehensive solutions based on your business and Huawei Cloud advantages and features.

2 Chaincode Development

2.1 Development Preparation

A chaincode, also called a smart contract, is a program written in Go, Java, or Node.js for operating the ledger. It is a code logic that runs on a blockchain and is automatically executed under specific conditions. Chaincodes are an important method for users to implement service logic when using blockchains. Due to the blockchain features, the execution results of smart contracts are reliable and cannot be forged or tampered with.

To use BCS, you must develop your own chaincodes and applications. Applications invoke chaincodes through peers in the blockchain network, and the chaincodes operate ledger data through peers.

NOTE

Ensure that no security risks (such as command injection) exist in the chaincodes you write and upload.

Preparing the Development Environment

Use the Go or Java development environment based on service requirements.

Preparing the Go Development Environment

1. Install the Go development environment. Download the installation package from **https://go.dev/dl/**. (Select a version later than 1.9.2.)

   The package name for each OS is as follows (version 1.11.12 is used as an example):

OS	Package
Windows	go1.11.12.windows-amd64.msi
Linux	go1.11.12.linux-amd64.tar.gz

- In Windows, you can use the .msi installation package for installation. By default, the .msi file is installed in **C:\Go**. You can add the **C:\Go\bin** directory to the **Path** environment variable. Restart the CLI for the settings to take effect.
- In Linux, decompress the downloaded binary package to the **/usr/local** directory. Add the **/usr/local/go/bin** directory to the **Path** environment variable.
  ```
  export PATH=$PATH:/usr/local/go/bin
  ```

After Golang is installed, you can run the **go version** command to view the version information and run the **go env** command to view the path configuration.

2. Install a Go editor of your choice. GoLand is recommended, which can be downloaded in **https://www.jetbrains.com/go/download**.

Preparing the Java Development Environment

This is applicable only to Fabric blockchain instances.

1. Install the Java development environment. Download the latest JDK version at **https://www.oracle.com/technetwork/java/javase/downloads/index.html** and install it.

   The package name for each OS is as follows (version 15.0.2 is used as an example):

OS	Package
Windows	jdk-15.0.2_windows-x64_bin.exe
Linux	jdk-15.0.2_linux-x64_bin.tar.gz

   - In Windows, you can use the .exe installation package for installation.
   - In Linux, decompress the downloaded binary package to the **/usr/local** directory.
     ```
     export PATH=$PATH:/usr/local/go/bin
     ```

   Set the environment variables (create them if they do not exist):

   - Set **JAVA_HOME** to the JDK installation directory, for example, **C:\Program Files (x86)\Java\jdk1.8.0_91** or **/usr/java/jdk1.8.0_91**.
   - Set **CLASSPATH** to **.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;**.
   - Add **%JAVA_HOME%\bin** and **%JAVA_HOME%\jre\bin** to **Path**.

   After the installation is complete, run the **java -version** command to view the version information.

2. Install a Java editor of your choice. **IntelliJ IDEA** is recommended.

Downloading the Source Code Package

Download the Fabric source code package as the third-party repository.

Download the required version from the following address:

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

NOTE

The version of the Fabric package should match that of the blockchain. For example, if a blockchain is v4.x.x, it uses Fabric v2.2, so you need to download the Fabric v2.2 package.

2.2 Development Specifications

Preventing Chaincode Container from Failing After a Panic

NOTE

This section applies only to the Go chaincode development.

When a panic exception occurs, the chaincode container may be suspended and restarted, logs cannot be found, and the problem cannot be located immediately. To prevent this case, add the defer statement at the entry point of the Invoke function. When a panic occurs, the error is returned to the client.

```
// Name the return value in the defer statement to ensure that the client can receive the value if a panic occurs.
//Use debug.PrintStack() to print the stack trace to the standard output for fault locating.
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
}
```

Querying Data in Batches

If too many records are returned in one query, too many resources will be occupied and the interface delay will be long. For example, if the interface delay exceeds 30s, the peer task will be interrupted. Therefore, estimate the data volume in advance, and query data in batches if the data volume is large.

To modify or delete the ledger data chaincode, consider performing operations in batches based on the data volume.

Using Indexes with CouchDB

Using indexes with CouchDB accelerates data querying, but slows data writing. Therefore, create indexes only for certain fields based on service requirements.

Verifying Permissions

Verify the permissions of the smart contract executor to prevent unauthorized users from executing chaincodes.

NOTE

If no specified organization is required for the endorsement, select at least two endorsing organizations to ensure that the chaincode data is not maliciously modified (such as installing invalid chaincode or processing data) by any other organizations.

Verifying Parameters

Before parameters (including input parameters and parameters defined in code) are used, the quantity, type, length, and value range of the parameters must be verified to prevent array out-of-range.

Processing Logs

During the development of services that have a complex logic and are prone to error, use **fmt** to print logs to facilitate debugging. **fmt** consumes a lot of time and resources. Delete the logs after the debugging is complete.

Configuring Dependencies

NOTE

This section applies only to the Java chaincode development.

Use Gradle or Maven to manage chaincode projects. If the chaincode project contains non-local dependencies, ensure that all nodes of the BCS instance are bound with EIPs. If the chaincode container runs in a restricted network environment, ensure that all dependencies in the project are configured as local dependencies. To obtain the chaincode used in this section, go to the BCS console and click **Use Cases**. Download **Chaincode_Java_Local_Demo** in the **Java SDK Demo** area.

2.3 Go Chaincode Development

2.3.1 Chaincode Structure

This section uses the Go language as an example. A chaincode is a Go file. After creating a chaincode, you can use it to develop functions.

Chaincodes can be compiled in two styles: 1.4 style (using package shim) and 2.2 style (using fabric-contract-api-go).

NOTE

BCS supports two chaincode styles.

Chaincode Interface

- To start a chaincode, you must call the Start function in the shim package (1.4 style). The input parameter is the Chaincode interface type defined in the shim package. During chaincode development, define a structure to implement the Chaincode interface.

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

- When developing chaincodes of the 2.2 style (using fabric-contract-api-go), define a structure to implement the Chaincode interface.

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

Chaincode Structure

- The Go chaincode structure (1.4 style) is as follows:

```
package main

//Import the required package.
import (
   "github.com/hyperledger/fabric/core/chaincode/shim"
   pb "github.com/hyperledger/fabric/protos/peer"
)

//Declare a structure.
type SimpleChaincode struct {}

//Add the Init method to the structure.
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
        //Implement the processing logic for chaincode initialization or update in this method.
    //stub APIs can be flexibly used during compilation.
}

//Add the Invoke method to the structure.
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
        //Implement the processing logic for responding to the call or query in this method.
    //stub APIs can be flexibly used during compilation.
}

//Main function. The shim.Start() method needs to be invoked.
func main() {
   err := shim.Start(new(SimpleChaincode))
   if err != nil {
      fmt.Printf("Error starting Simple chaincode: %s", err)
   }
}
```

- The Go chaincode structure (2.2 style) is as follows:

```
package main

//Import the required package.
import (
 "github.com/hyperledger/fabric/plugins/fabric-contract-api-go/contractapi")



//Declare a structure.
type Chaincode struct {
    contractapi.Contract
}

//Add the Init method to the structure.
func (ch * Chaincode) Init(ctx contractapi.TransactionContextInterface, args…) error {
```



```
        //Implement the processing logic for chaincode initialization or update in this method.
}

//Add the Invoke method to the structure.
func (ch * Chaincode) Invoke(ctx contractapi.TransactionContextInterface, args…) error {
        //Implement the processing logic for responding to the call or query in this method.
}

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

2.3.2 Chaincode APIs

The shim package in the Fabric source code package provides the following types of APIs:

- Parameter parsing APIs: used to parse parameters transferred to the invoked function or method during chaincode invocation
- Ledger data operation APIs: used to provide methods for performing operations on ledger data, including status data query and transaction processing
- Transaction obtaining APIs: used to obtain information about transaction proposals
- APIs for private data operations: used to perform operations on private data (the APIs are available since Hyperledger Fabric v1.2.0)
- Other APIs: used to set events and invoke other chaincodes

2.3.3 Sample Chaincode (1.4)

The following is an example of installing and instantiating the account transfer chaincode (1.4). For details about how to debug this chaincode, see **the official Fabric examples**.

```
package main

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

type SimpleChaincode struct {
}

//Automatically invoked during chaincode instantiation or update to initialize the data status.
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    //The output information of the println function is displayed in the logs of the chaincode container.
    fmt.Println("ex02 Init")

    //Obtain the parameters transferred by the user to invoke the chaincode.
    _, args := stub.GetFunctionAndParameters()

    var A, B string //Two accounts
    var Aval, Bval int //Balances of the two accounts
```



```
    var err error

    //Check whether the number of parameters is 4. If not, an error message is returned.
    if len(args) != 4 {
        return shim.Error("Incorrect number of arguments. Expecting 4")
    }

    A = args[0] //Username of account A
     Aval, err = strconv.Atoi(args[1]) //Balance in account A
    if err != nil {
        return shim.Error("Expecting integer value for asset holding")
    }

    B = args[2] //Username of account B
    Bval, err = strconv.Atoi(args[3])  //Balance in account B
    if err != nil {
        return shim.Error("Expecting integer value for asset holding")
    }

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

    //Write the status of account A to the ledger.
    err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
    if err != nil {
        return shim.Error(err.Error())
    }

    //Write the status of account B to the ledger.
    err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
    if err != nil {
        return shim.Error(err.Error())
    }

    return shim.Success(nil)
}

//The function is automatically invoked to query or invoke the ledger data.
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    fmt.Println("ex02 Invoke")
    //Obtain the function name and parameters transferred by the user to invoke the chaincode.
    function, args := stub.GetFunctionAndParameters()

     //Check the obtained function name.
    if function == "invoke" {
        //Invoke the invoke function to implement the money transfer operation.
        return t.invoke(stub, args)
    } else if function == "delete" {
        //Invoke the delete function to deregister the account.
        return t.delete(stub, args)
    } else if function == "query" {
        //Invoke the query interface to query the account.
        return t.query(stub, args)
    }
    //If the transferred function name is incorrect, shim.Error() is returned.
    return shim.Error("Invalid invoke function name. Expecting \"invoke\" \"delete\" \"query\"")
}

//Transfer money between accounts.
func (t *SimpleChaincode) invoke(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    var A, B string //Accounts A and B
    var Aval, Bval int //Account balance
    var X int //Transfer amount
    var err error

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

    A = args[0]       //Username of account A
```



```
    B = args[1]       //Username of account B

    //Obtain the balance of account A from the ledger.
    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))

    //Obtain the balance of account B from the ledger.
    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 indicates the transfer amount.
    X, err = strconv.Atoi(args[2])
    if err != nil {
        return shim.Error("Invalid transaction amount, expecting a integer value")
    }
    //Transfer
    Aval = Aval - X
    Bval = Bval + X
    fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)

    //Update the balance of account A after the transfer.
    err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
    if err != nil {
        return shim.Error(err.Error())
    }

    //Update the balance of account B after the transfer.
    err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
    if err != nil {
        return shim.Error(err.Error())
    }

    return shim.Success(nil)
}

//Account deregistration
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] //Username

    //Delete the account status from the ledger.
    err := stub.DelState(A)
    if err != nil {
        return shim.Error("Failed to delete state")
    }

    return shim.Success(nil)
}

//Account query
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]   //Username

    //Obtain the balance of the account from the ledger.
    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)
    //Return the transfer amount.
    return shim.Success(Avalbytes)
}

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

2.3.4 Sample Chaincode (2.0)

The following is an example of installing and instantiating the account transfer chaincode (2.0). For details about how to debug this chaincode, see **the official Fabric examples**.

```
package main

import (
        "errors"
        "fmt"
        "strconv"

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

// Chaincode implementation
type ABstore struct {
        contractapi.Contract
}

// Automatically invoked during chaincode instantiation or update to initialize the chaincode data.
func (t *ABstore) Init(ctx contractapi.TransactionContextInterface, A string, Aval int, B string, Bval int) error {
        // The output information of the println function is recorded in the logs of the chaincode container.
        fmt.Println("ABstore Init")
        var err error

        fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)
        // Write the status data to the ledger.
        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 transfers X to B.
func (t *ABstore) Invoke(ctx contractapi.TransactionContextInterface, A, B string, X int) error {
        var err error
        var Aval int
        var Bval int

        // Obtain the status data from the ledger.
        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))

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

        // Write the status data back to the ledger.
        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
}

// Account deregistration
func (t *ABstore) Delete(ctx contractapi.TransactionContextInterface, A string) error {

    // Delete the account status from the ledger.
        err := ctx.GetStub().DelState(A)
        if err != nil {
                return fmt.Errorf("Failed to delete state")
        }

        return nil
}

// Account query
func (t *ABstore) Query(ctx contractapi.TransactionContextInterface, A string) (string, error) {
        var err error
        // Obtain the status data from the ledger.
        Avalbytes, err := ctx.GetStub().GetState(A)
        if err != nil {
                jsonResp := "{\"Error\":\"Failed to get state for " + A + "\"}"
                return "", errors.New(jsonResp)
        }
```



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

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

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

2.3.5 Chaincode Debugging

To debug a chaincode, you can use MockStub to perform unit tests on the chaincode. Before chaincode debugging, import the shim package to the test code. You can refer to the sample test code provided below or the **test code provided by Fabric**.

Writing Test Code

The test code of **Sample Chaincode (1.4)** is as follows:

```
package main

import (
    "fmt"
    "testing"

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

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

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

func checkQuery(t *testing.T, stub *shim.MockStub, name string, value string) {
    res := stub.MockInvoke("1", [][]byte{[]byte("query"), []byte(name)})
    if res.Status != shim.OK {
        fmt.Println("Query", name, "failed", string(res.Message))
        t.FailNow()
    }
```

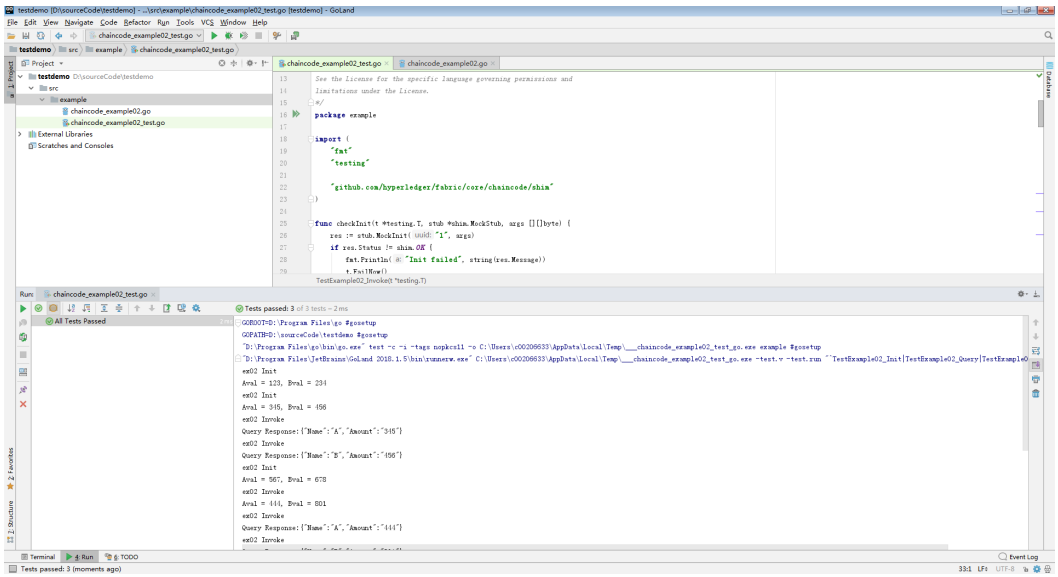


```
    if res.Payload == nil {
        fmt.Println("Query", name, "failed to get value")
        t.FailNow()
    }
    if string(res.Payload) != value {
        fmt.Println("Query value", name, "was not", value, "as expected")
        t.FailNow()
    }
}

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

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

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

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

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

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

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

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

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

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

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

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

Debugging

Execute the test function in the IDE.

Figure 2-1 Chaincode debugging



2.4 Java Chaincode Development

2.4.1 Chaincode Structure

This section uses the Java language as an example. A chaincode is a Java project. After creating the project, you can perform operations such as function development.

Notes and Constraints

The Java chaincode is supported only by Fabric v2.2 and later versions.

Chaincode Interface

A chaincode is invoked using the start function in the shim package. During chaincode development, define a class to extend ChaincodeBase. The following methods are overridden during the extension:

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

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

- **init** is called to initialize data during chaincode instantiation or update.
- **Invoke** is called to update or query the ledger. The service logic for responding to the call or query needs to be implemented in this method.

Chaincode Structure

The Java chaincode structure is as follows:


```
package main

//You only need to configure the required packages in Maven or Gradle. The packages are imported
automatically.
import org.hyperledger.fabric.shim.ChaincodeBase;
import org.hyperledger.fabric.shim.ChaincodeStub;

public class SimpleChaincodeSimple extends ChaincodeBase {
    @Override
    public Response init(ChaincodeStub stub) {
        //Implement the processing logic for chaincode initialization or update in this method.
        //stub APIs can be flexibly used during compilation.
    }

    @Override
    public Response invoke(ChaincodeStub stub) {
        //Implement the processing logic for responding to the call or query in this method.
        //stub APIs can be flexibly used during compilation.
    }

    //Main function. The shim.Start() method needs to be invoked.
    public static void main(String[] args) {
        new SimpleChaincode().start(args);
    }
}
```

2.4.2 Chaincode APIs

The shim package in the Fabric source code package provides the following types of APIs:

- Parameter parsing APIs: used to parse parameters transferred to the invoked function or method during chaincode invocation
- Ledger data operation APIs: used to provide methods for performing operations on ledger data, including status data query and transaction processing
- Transaction obtaining APIs: used to obtain information about transaction proposals
- Other APIs: used to set events and invoke other chaincodes

2.4.3 Sample Chaincode

The following is an example chaincode for reading and writing data. You can also refer to other chaincodes in the **official examples provided by Fabric**.

NOTE

When creating a Java project, you can create a Maven or Gradle project to import the dependency package. A Gradle project is used in the following example.

```
/* Before importing the following code to the build.gradle file of the project, delete or comment out the
original content in this file.*/
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

//Code repository where the dependency package is searched and downloaded.
repositories {
    mavenLocal()
    maven{ url "https://mirrors.huaweicloud.com/repository/maven/" }
    maven{ url "https://jitpack.io" }/* If JSON Schema cannot be imported, try to add it to this repository.*/
}

//Import the dependency package required by the code.
dependencies {
    compile group: 'org.hyperledger.fabric-chaincode-java', name: 'fabric-chaincode-shim', version: '2.2.+'
     testCompile group: 'junit', name: 'junit', version: '4.12' //testCompile indicates that this package is used
only for debugging.
    testCompile 'org.mockito:mockito-core:2.4.1'
}

shadowJar {
    baseName = 'chaincode'
    version = null
    classifier = null
    manifest {
        //The path must be the same as that of the class that extends ChaincodeBase.
        attributes 'Main-Class': 'org.hyperledger.fabric.example.SimpleChaincode'
     }
}
//The following is the content in the SimpleChaincode class.
package org.hyperledger.fabric.example;//The actual location of the chaincode file, which is usually
automatically generated.

//You only need to configure the required packages in Maven or Gradle. The packages are imported
automatically.
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);
    }
}
```

2.4.4 Chaincode Debugging

To debug a chaincode, you can use MockStub to perform unit tests on the chaincode. To obtain the chaincode used in this section, go to the BCS console and click **Use Cases**. Download **Chaincode_Java_Local_Demo** in the **Java SDK Demo** area.

Adding the Dependency

To use the mock() method, add the Mockito dependency.

- Gradle

  Add the following dependency to the **dependencies** block in the **build.gradle** file (not **dependencies** in the **buildscript** block):

  ```
  testCompile 'org.mockito:mockito-core:2.4.1'
  ```

- Maven

  Add the following configuration dependency to the **dependencies** block (add the block if it does not exist) in the **pom.xml** file:

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


Writing Test Code

If the **test** folder does not exist during project creation, create it under **src**. Select **test\java** under **Gradle Source Sets**, and create the **SimpleChaincodeTest.java** test file, as shown in the following figures:

Figure 2-2 Creating a test file

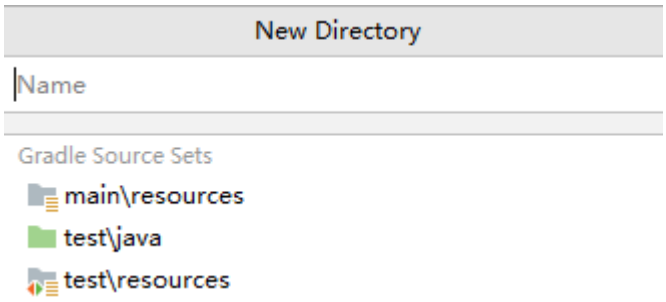
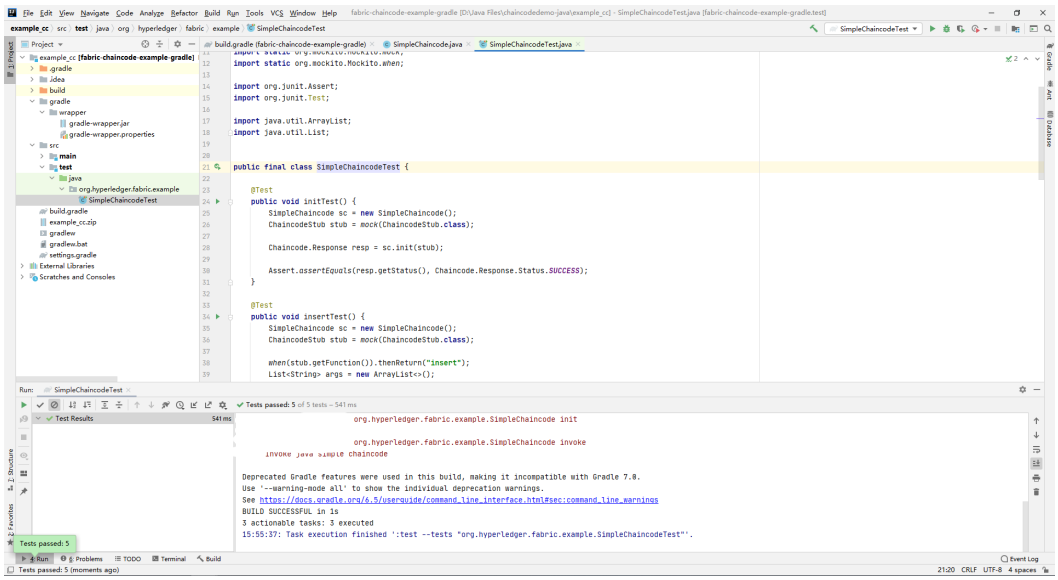


Figure 2-3 Test file



The content of the **SimpleChaincodeTest.java** test code is as follows:

```
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\"}");
    }
}
```

Debugging

In **SimpleChaincodeTest.java**, click **Run Test** on the left of **SimpleChaincodeTest**.

Figure 2-4 Executing the test

```
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      }
```

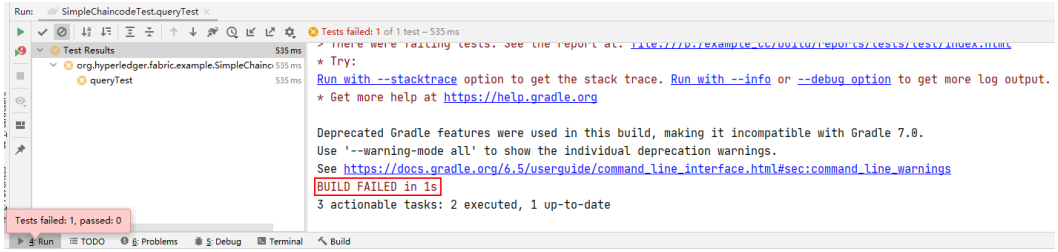
If the following information is displayed, the chaincode debugging is successful:

Figure 2-5 Successful

```
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"'.
```

If the following information is displayed, the chaincode debugging failed. Edit the chaincode or check the logic of the test code based on the displayed information.

Figure 2-6 Failed




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


3 Application Development

3.1 Overview

User applications interact with the ledger using chaincodes. An application can be written in a variety of languages, including Go, Solidity, Java, C++, Python, and Node.js. The languages used by the application and the chaincode do not necessarily have to be the same, as long as the application can invoke the chaincode using the SDK.

NOTE

- Enhanced Hyperledger Fabric blockchains open gRPC APIs to applications, just like the open-source version. These APIs are usually called by SDKs. For details, see **the definition of the SDK APIs**.

3.2 Preparations

User applications interact with the ledger using chaincodes. An application can be written in a variety of languages, including Go, Solidity, Java, C++, Python and Node.js. The languages used by the application and the chaincode do not necessarily have to be the same, as long as the application can invoke the chaincode using the SDK.

1. Buy a BCS instance.

   BCS instances can be deployed in CCE clusters. For details, see **Deployment Using a CCE Cluster**.

2. Obtain the required SDK configuration file and certificates. For details, see **Downloading SDK Configurations and Certificates**.

3.3 Development

Develop your own application code. You can use the **SDK** provided by BCS or the SDK provided by the official Fabric community that matches the version of your instance.

You can also use **Homomorphic Encryption** for your instance.

NOTE

The version of the Fabric package should match that of the blockchain. For example, if a blockchain is v4.x.x, it uses Fabric v2.2, so you need to download the Fabric v2.2 package.

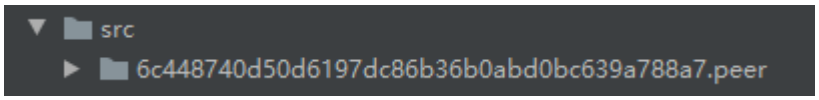
Configuring the Organization ID

Modify the application code for configuring the organization ID of the BCS instance. After the downloaded certificate file is decompressed, the peer file name contains the directory name and the organization ID.

The following figure is for reference only. Use the actual certificate file.

After the certificate file is decompressed, the directory name is **6c448740d50d6197dc86b36b0abd0bc639a788a7.peer** and the organization ID is **6c448740d50d6197dc86b36b0abd0bc639a788a7**.

Figure 3-1 Decompressing the certificate file



Configuring the SDK file

1. Modify the application code related to the SDK configuration file. As shown in the following example, you must configure the correct absolute path of the SDK configuration file.

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

2. If the paths in the SDK configuration file are different from the actual ones, you must manually change all certificate paths in the SDK configuration file.

4 Demos

4.1 Go SDK Demo

This section provides a Go SDK–based demo to help you develop your own Go client applications.

Preparations

- Prepare an ECS.
- Install the golang environment on the ECS. The Go version must be 1.12 or later, and earlier than 1.16.
- Obtain the Go SDK source code. To obtain it, log in to the BCS console, click **Use Cases** and download the source code in the **Go SDK Demo** area.

Buying a BCS Instance

For details, see **Deployment Using a CCE Cluster**.

Installing and Instantiating a Chaincode

To obtain the chaincode used in this demo, go to the BCS console and choose **Use Cases**. Download the example Go chaincode in the **Go SDK Demo** area.

For details, see **User Guide > Blockchain Management > Chaincode Management**.

Downloading SDK Configurations and Certificates

Step 1 Log in to the BCS console.

Step 2 On the **Instance Management** page, click **Download Client Configuration** on an instance card.

Step 3 Select **SDK Configuration File** and set the parameters as described in the following table.

Parameter	Setting
Chaincode Name	Enter **chaincode**. **NOTE** The chaincode name must be the same as the name specified during chaincode installation and instantiation.
Certificate Root Path	Enter **/root/gosdkdemo/config**.
Channel	Select **channel**.
Organization & Peer	Retain the default value.

Select **Orderer Certificate**.

Select **Peer Certificates**, select **organization** for **Peer Organization**, and select **Administrator certificate**.

Step 4 Click **Download**. The SDK configuration file and the administrator certificates for the **orderer** and **organization** organizations are downloaded.

----End

Deploying the Application

1. Download the Go SDK source code to the **/root** directory of the ECS and decompress the package.

   To obtain it, go to the BCS console and click **Use Cases**. Download the source code in the **Go SDK Demo** area.

2. Decompress the .zip package obtained in **Downloading SDK Configurations and Certificates** and copy the **orderer** and **peer** folders and the **sdk-config.json** and **sdk-config.yaml** files from the **configs** folder to the **/root/gosdkdemo/config/** directory.

3. Find the **/gosdkdemo/src/main.go** file in the code and modify it as follows:

   a. Change the value of **configFile** to the actual name of the SDK configuration file, for example, **demo-channel-sdk-config.yaml**.

   b. Change the value of **org** to the hash value of **organization**.

   On the **Channel Management** page, click **View Peer**. The organization ID is the value of **MSP ID** without "MSP".

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

4. Use **go.mod** to set GOPATH based on the actual installation path.

   a. Set the environment variable **GO111MODULE** to **on**.

   ```
   export GO111MODULE=on
   ```

   b. The following figure shows the **go.mod** file. Modify the "replace" line based on the actual installation path.

   ```
   module main
   go 1.15
   //Specify the dependency to be imported and its version.
   ```



```
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
)
//The project path /root/gosdkdemo/src is used as an example.
replace github.com/hyperledger/fabric-sdk-go => /root/gosdkdemo/src/github.com/hyperledger/fabric-sdk-go
```

5. Find the **main.go** file in the **gosdkdemo/src** directory and run the following command:

   go run main.go

Transferring Private Keys in Memory

Encrypt the private key file, decrypt it in the demo, and transfer it to Fabric SDK.

1. For a TLS private key:

   In the initializeSdk function of the **main.go** file, invoke the function:

   ```
   encryptedtlsKey,err := GetTlsCryptoKey(org)  // Retrieve the encrypted TLSprivate key from the specified directory in the configuration file.
   // Decrypt encryptedtlsKey using the specified encryption and decryption method to obtain the decryptedTlsKey string.
   SetClientTlsKey(decryptedTlsKey)     // Transfer the decrypted TLS private key to Fabric SDK.
   ```

2. For an MSP private key:

   In the insert function of the **main.go** file, invoke the function:

   ```
   encryptedbytekey,_:= GetPrivateKeyBytes(org)  // Retrieve the encrypted MSP private key from the specified directory in the configuration file.
   // Decrypt encryptedbytekey using the specified encryption and decryption method to obtain the decryptedKey string.
   SetPrivateKey(decryptedKey) //Assign the decrypted MSP private key to the global variable privateKey and import the variable.
   ```

Common APIs

fabric-sdk-go mainly uses the FabricSDK class, which can be constructed by using the NewSDK() method.

FabricClient, ChannelClient, ChannelMgmtClient, and ResourceMgmtClient can perform common operations of **fabric-sdk-go**.

- **FabricSDK**

  FabricSDK uses the New() method to generate objects in **pkg\fabsdk\fabsdk.go**. The New() method has an **Options** parameter. The following is an example of generating FabricSDK:

  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 indicates the configuration file path. **OrgId** is the organization ID in the SDK configuration file.

FabricSDK uses the NewSDK() method to generate objects in **def/fabapi/fabapi.go**. The NewSDK() method has an Options parameter. The following is an example of generating the **Options** parameter:

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

ConfigFile indicates the configuration file path. **LoggerFactory** is optional. If it is not specified, logs are printed to the console by default.

- **FabricClient**

  FabricClient provides the following common APIs:

API	Description	Setting	Returned Values
CreateChannel	Creates a channel.	request CreateChannelRequest	txn.TransactionID, error
QueryChannelInfo	Queries channel information.	name string, peers []Peer	Channel, error
InstallChaincode	Installs chaincodes in a blockchain.	request InstallChaincodeRequest	[]*txn.TransactionProposlResponse, string, error
InstallChaincode	Installs chaincodes in a blockchain.	request InstallChaincodeRequest	[]*txn.TransactionProposlResponse, string, error
QueryChannels	Queries created channels in a blockchain.	peer Peer	*pb.ChannelQueryResponse, error
QueryInstalledChaincodes	Queries installed chaincodes in a blockchain.	peer Peer	*pb.ChaincodeQueryResponse, error

- **ChannelClient**

  ChannelClient provides chaincode query and invoking APIs.

API	Description	Setting	Returned Values
Query	Queries data by using the chaincode.	request QueryRequest	[]byte, error
QueryWithOpts	Similar to the **Query** API, but can specify notifier, peers, and timeout with **QueryOpts**.	request QueryRequest, opt QueryOpts	[]byte, error
ExecuteTx	Invokes the chaincode.	request ExecuteTxRequest	TransactionID, error
ExecuteTxWithOpts	Similar to the **ExecuteTx** API, but can specify notifier, peers, and timeout with the **ExecuteTxOpts** parameter.	request ExecuteTxRequestopt ExecuteTxOpts	TransactionID, error

- **ChannelMgmtCLient**

  ChannelMgmtClient provides only two APIs: **SaveChannel(req SaveChannelRequest) error** and **SaveChannelWithOpts(req SaveChannelRequest, opts SaveChannelOpts) error**, which are used to create channels and need to invoke the **createChannel()** API of FabricClient.

- **ResourceMgmtClient**

  ResourceMgmtClient provides chaincode lifecycle management APIs and an API for adding peers to a channel.

  NOTE

  The chaincode deleting API is provided by BCS and can only delete the chaincode installation package.

API	Description	Setting	Returned Values
InstallCC	Installs chaincodes.	reqInstallCCRequest	[]InstallCCResponse, error
InstallCCWithOpts	Similar to the **InstallCC** API, but can specify peers with **InstallCCOpts**.	reqInstallCCRequest,opts InstallCCOpts	[]InstallCCResponse, error
InstantiateCC	Instantiates chaincodes.	channelID string,reqInstantiateCCRequest	error

API	Description	Setting	Returned Values
InstantiateCCWithOpts	Similar to the **InstantiateCC** API, but can specify peers and timeout with **InstantiateCCOpts**.	channelID string,reqInstantiateCCRequest, optsInstantiateCCOpts	error
UpgradeCC	Upgrades chaincodes.	channelID string,reqUpgradeCCRequest	error
UpgradeCCWithOpts	Similar to the **UpgradeCC** API, but can specify peers and timeout with **UpgradeCCOpts**.	channelID string,reqUpgradeCCRequest, optsUpgradeCCOpts	error
DeleteCC	Deletes chaincodes (only the chaincode installation packages).	channelID string,reqDeleteCCRequest	error
DeleteCCWithOpts	Similar to the **DeleteCC** API, but can specify peers and timeout with **DeleteCCWithOpts**.	channelID string,reqDeleteCCRequest,opts DeleteCCOpts	error
JoinChannel	Adds peers to a channel.	channelID string	error
JoinChannelWithOpts	Similar to the **JoinChannel** API, but can specify peers and timeout with **JoinChannelOpts**.	channelID string,optsJoinChannelOpts	error

NOTE

All APIs with options can specify peers. The peers can be generated through NewPeer(userName string, orgName string, url string, certificate string, serverHostOverride string, config config.Config) (fab.Peer, error) in **def/fabapi/pkgfactory.go**. Compared with the native NewPeer method, this method has two more parameters: **userName** and **orgName**, which are used by the peers to find the corresponding TLS certificate with bidirectional TLS.

Invoking a Contract

Main.go is a simple client application sample program, which is used to help you quickly get started with the client development process. The main steps are as follows:

```
//1. Import packages. The SDK package provides some APIs for user applications to access chaincodes.
import (
```



```
  "fmt"
  "github.com/hyperledger/fabric-sdk-go/pkg/client/channel"
  "github.com/hyperledger/fabric-sdk-go/pkg/fabsdk"    ......
)
//2. Create file configurations. This step encapsulates some common configurations required for application development, including the SDK configuration file path and organization name.
var (
  configFile = "/root/fabric-go-demo/config/go-sdk-demo-channel-sdk-config.yaml"
  org = "9103f17cb6b4f69d75982eb48bececcc51aa3125"
  ......
)
//3. Load the configuration file.
loadConfig()
//4. Initialize the SDK.
initializeSdk()
// 5. Execute the chaincode and write the data to the ledger. The key is "testuser", and the value is "100".
insert("insert",[][]byte{
  []byte("testuser"),
  []byte("100"),
})
// 6. Query the chaincode and output the query result. The key is "testuser".
query("query",
[][]byte{
[]byte("testuser"),
})
```

Table 4-1 Functions

Function	Description
getOptsToInitializeSDK	Parses the configuration file, and creates and returns the fabsdk.Option object.
GetDefaultChaincodeId	Parses the configuration file and returns **chaincodeID**.
GetDefaultChannel	Parses the configuration file and returns **channelID**.
UserIdentityWithOrgAndName	Verifies the identity of a user. The input parameters are the organization name and username. The verification result is returned.
ChannelClient	Create the *channel.Client object. The input parameters are the organization name, username, and channel ID. The *channel.Client object is returned.
insert	Writes data to the ledger. The input parameters are the method name of the chaincode and the key-value pair to be inserted. The write result is returned.
query	Queries chain data. The input parameters are the method name of the chaincode and the data to be queried. The query result is returned.

4.2 Java SDK Demo

This section provides a demo application that uses a Java SDK and supports the OSCCA-published cryptographic algorithms to help you quickly understand the concepts and process of using BCS.

NOTE

This is a demo only and is not for actual use.

Preparations

Action	Description
Install an integrated development environment (IDE).	Install Java Development Kit (JDK), Maven, and Eclipse. You can replace Eclipse with another IDE you prefer. The JDK version must be 1.8 (64-bit). If you have installed JDK, run the **java -version** command in the command line to check the JDK version.

Buying a BCS Instance

Step 1 Log in to the BCS console.

Step 2 On the **Instance Management** page, click **Buy** next to **Enhanced Hyperledger Fabric Instance**.

Step 3 Configure basic information about the BCS instance by referring to **Table 4-2**.

NOTICE

To ensure that the demo runs properly, set the parameters as described in the following table.

Table 4-2 Basic settings

Parameter	Setting
Billing Mode	Select **Pay-per-use**.
Region	Retain the default value.
Enterprise Project	Select an existing enterprise project, for example, **default**. If the enterprise management service is not enabled, this parameter is unavailable.
Instance Name	Enter **java-sdk-demo**.
Edition	Select **Basic**.
Blockchain Type	Select **Private**.
Enhanced Hyperledger Fabric Version	v2.2
Consensus Mechanism	Select **Raft (CFT)**.

Parameter	Setting
Resource Access Initial Password	Enter a password.
Confirm Password	-

Step 4 Click **Next: Configure Resources**. Table 4-3 describes the resource parameters.

Table 4-3 Resource configurations

Parameter	Example
Environment Resources	Select **Custom**.
Cluster	Select **Create a new CCE cluster**.
AZ	Select an AZ.
Cross-AZ Scheduling	Select **No**.
ECS Specifications	Select the flavor for **4 vCPUs | 8 GB**.
ECS Quantity	Enter **1**.
High Availability	Select **No**.
VPC	Select **Automatically create VPC**.
Subnet	Select **Automatically create subnet**.
ECS Login Method	Select **Password**.
Password of Root User	If you do not enter a password here, the previously specified resource access initial password will be used.
Confirm Password	-
Use EIP of a CCE Node	Select **Yes**.
EIP Billed By	Retain the default value.
EIP Bandwidth	Set it to 5 Mbit/s.

Step 5 Click **Next: Configure Blockchain**. Table 4-4 describes the blockchain parameters.

Table 4-4 Blockchain configurations

Parameter	Example
Blockchain Configuration	Select **Custom**.

Parameter	Example
Blockchain Mgmt. Initial Password	If you do not enter a password here, the previously specified resource access initial password will be used.
Confirm Password	-
Volume Type	Select **SFS Turbo** or select another one as prompted.
Storage Capacity of Peer Organization (GB)	Retain the default value.
Ledger Storage	Select **File database (GoLevelDB)**.
Peer Organization	A peer organization named **organization** has been automatically created. Change the peer quantity to 1.
Channel Configuration	The **organization** organization has been added to the channel automatically. Retain this default setting.
Orderer Quantity	Retain the default value.
Security Mechanism	Select **ECDSA**. **NOTICE** The **OSCCA-published cryptographic algorithms** option is available. If you select this option, other modifications are required for the demo deployment. Pay attention to the descriptions of modifications.
Configure Block Generation	Select **No**.
Enable Support for RESTful API	Select **No**.

Step 6 Click **Next: Confirm**.

Step 7 Confirm the configurations and finish the purchase process.

Wait for several minutes. After a message is displayed indicating successful installation, check the status of the instance. If it is **Normal**, the deployment is completed.

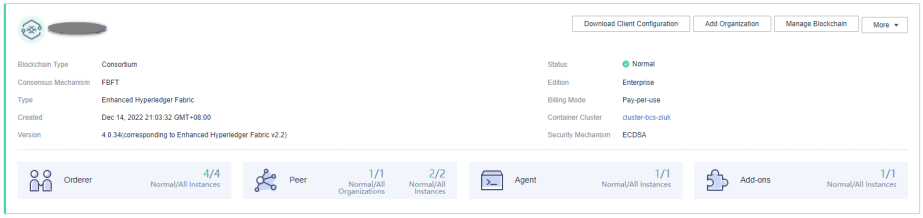
----End

Installing and Instantiating a Chaincode

Step 1 Log in to the BCS console.

Step 2 In the navigation pane on the left, click **Instance Management**.

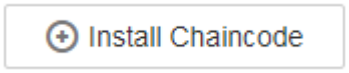
Step 3 Find the instance you just created and click **Manage Blockchain** to go to the Blockchain Management console.



Step 4 On the login page, enter the username and password, and click **Log In**.

NOTE

The username is **admin**, and the password is the **Blockchain Mgmt. Initial Password** set when you created the BCS instance. If you have not set this password, use the resource access initial password.

Step 5 Click 

in the upper left corner of the page.

The parameters for chaincode installation are as follows.

Table 4-5 Installation parameters

Parameter	Setting
Chaincode Name	Enter **chaincode**.
Chaincode Version	Select **2.0**.
Ledger Storage	**File database (goleveldb)**
Select All Peers	Check the box.
Organization & Peer	Select **peer-0**.
Language	Select **Golang**.
Chaincode File	Chaincode_Go_Demo: To obtain it, go to the BCS console and click **Use Cases**. Download the example Go chaincode in the **Java SDK Demo** area.
Chaincode Description	Enter a description of the chaincode.
Code Security Check	This option is displayed only when the chaincode language is Golang. Enable this option to check chaincode security.

Step 6 Click **Install**.

Step 7 After installing the chaincode, click **Instantiate** in the **Operation** column of the chaincode list.

The parameters for chaincode instantiation are as follows.

Table 4-6 Instantiation parameters

Parameter	Setting
Chaincode Name	Enter **chaincode**.
Channel	Select **channel**.
Chaincode Version	Select **2.0**.
Initialization Function	Enter **init**.
Chaincode Parameters	a,200,b,250
Endorsement Policy	Select **Endorsement from any of the following organizations**.
Endorsing Organizations	Select **organization**.
Privacy Protection Configuration	Select **No**.

Step 8 Click **Instantiate**.

Wait for 2 to 3 minutes and refresh the page. Click **View more** in the **Instantiation** column to check the instantiation status.

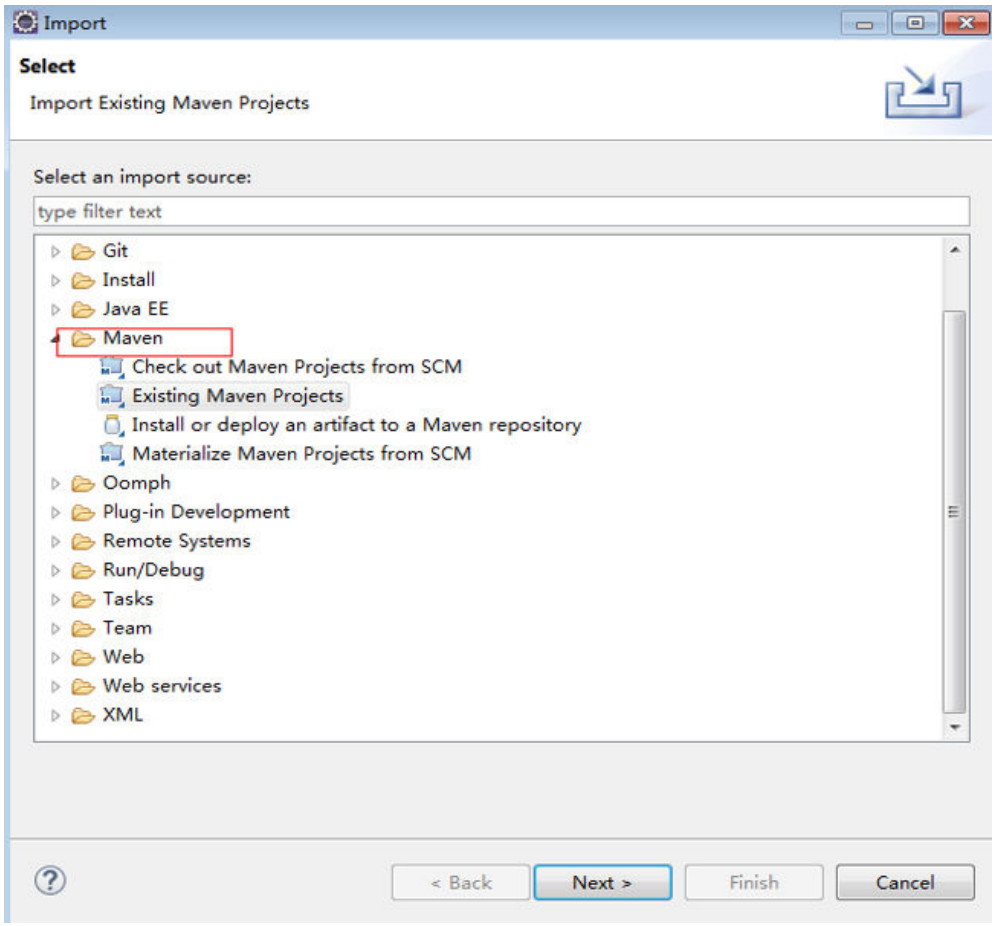
----End

Configuring the Application

Step 1 Import the project.

Obtain the project code and decompress it. To obtain the project code, go to the BCS console and choose **Use Cases**. Download the example Java chaincode in the **Java SDK Demo** area.

Right-click the Eclipse page, and choose **Import** from the shortcut menu to import the project code file (Maven project) to Eclipse.



Step 2 Download SDK configurations and certificates.

1. On the **Instance Management** page, click **Download Client Configuration** on an instance card.
2. Select **SDK Configuration File** and set the parameters as described in the following table.

Table 4-7 SDK parameters

Parameter	Description
Chaincode Name	Enter **chaincode**. **NOTICE** The chaincode name must be the same as the name specified during chaincode installation and instantiation.
Certificate Root Path	Enter the path to the **config** folder of the **javasdkdemo** project. **NOTICE** Change the backslashes (\) to slashes (/), for example, **D:/javasdkdemo/config**.
Channel	Select **channel**.
Organization & Peer	Select all peers in the channel.

Select **Orderer Certificate**.

Select **Peer Certificates**, select **organization** for **Peer Organization**, and select **Administrator certificate**.

3. Click **Download** to download the SDK configuration file and the administrator certificates for the **java-sdk-demo-orderer** and **organization** organizations to the **config** directory in the demo project.

Step 3 Copy and decompress the package.

Step 4 Decompress **demo-config.zip** and copy the contents in the **java-sdk-demo-orderer-admin-cert**, **organization-admin-cert**, and **sdk-config** folders to the **config** directory in the demo project.

----End

Deploying the Application

Step 1 In the Maven project, find the **Main.java** file in the **/javasdkdemo/src/main/java/handler/** directory, and change the file path in the following code of the Main class to the absolute path of the **java-sdk-demo-sdk-config.yaml** file. The path can be found in **Step 2.3**. Change the backslashes (\) to slashes (/).

```
helper.setConfigCtx("E:/yourdir/huawei.yaml");
```

For example, change the path to **helper.setConfigCtx("D:/javasdkdemo/config/java-sdk-demo-channel-sdk-config.yaml")**.

NOTICE

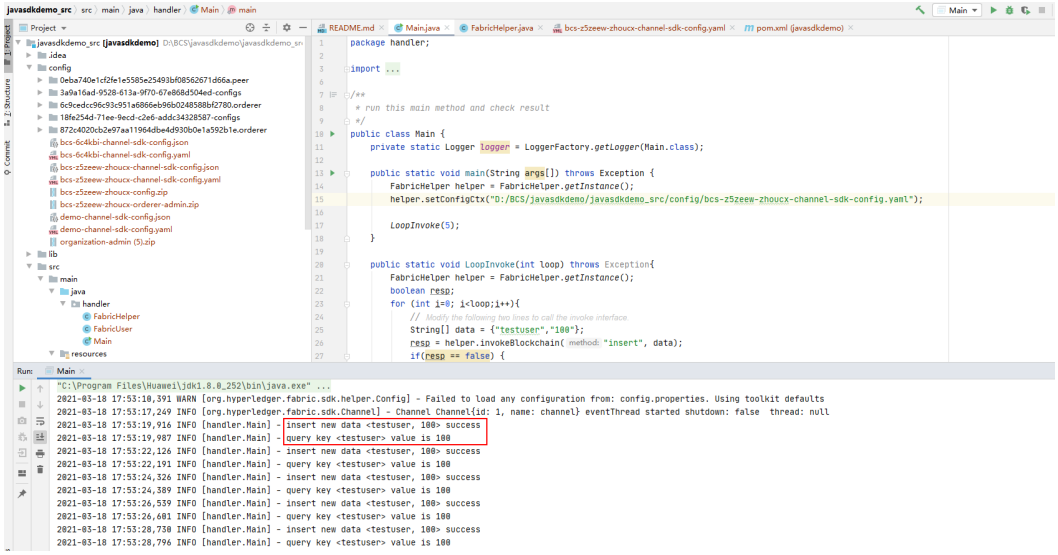
Add the dependency upon **fabric-sdk-java-1.4.1-jar-with-dependencies.jar** in the **lib** folder of the project to **pom.xml** because OSCCA-published cryptographic algorithms and other cryptographic algorithms need to refer to the **fabric-sdk** dependency package provided by Huawei. To add the dependency, remove the comments on the dependency as shown in the following figure. Otherwise, the project may not run properly.

Figure 4-1 File details

Step 2 Run the main function.

Each time the command is successfully executed, the key-value pair <**testuser,100**> is saved to the blockchain. If you query key **testuser**, the value is **100**. You can also view the transaction records in **Block Browser**.

Figure 4-2 Transaction records



----End

Transferring Private Keys in Memory

Encrypt the private key file, decrypt it in the demo, and transfer it to Fabric SDK.

For an MSP private key, in the genFabricUser function of the **FabricHelper** file, invoke the function:

```
// Retrieve the encrypted MSP private key from the specified directory in the configuration file.
String adminPrivateKeyString = extractPemString(msp, "keystore");
// Decrypt adminPrivateKeyString to obtain the decryptedKey string.
// Assign the decrypted MSP private key to the adminPrivateKeyString variable.
String adminPrivateKeyString = decryptedKey;
```

NOTE

Currently, TLS private keys cannot be transferred through the memory.

4.3 Gateway Java Demo

This section provides a demo based on Fabric Gateway for Java. Fabric Gateway Java encapsulates the Java SDK, which reduces the code amount and helps users develop Java client applications.

Common APIs

When you use Fabric-Gateway-Java to initiate transactions and query data, the Network and Contract interfaces are used. For more interfaces, see the **Fabric official website**.

- **Network**

  The common interfaces are as follows:

API	Description	Setting	Returned Values
getContract	Gets an instance of a contract.	String chaincodeId	Contract
addBlockListener	Adds a listener to listen to block events.	Consumer<org.hyperledger.fabric.sdk.BlockEvent> listener	Consumer<org.hyperledger.fabric.sdk.BlockEvent>
getChannel	Gets the channel associated with the network.	/	org.hyperledger.fabric.sdk.Channel
removeBlockListener	Removes a listener.	Consumer<org.hyperledger.fabric.sdk.BlockEvent> listener	void

- **Contract**

  The common interfaces are as follows:

API	Description	Setting	Returned Values
submitTransaction	Submits a transaction. The invocation method and parameters need to be entered.	String name, String... args	byte[]
evaluateTransaction	Evaluates a transaction. The invocation method and parameters need to be entered.	String name, String... args	byte[]
createTransaction	Creates a transaction. The transaction needs to be submitted.	String name	Transaction
addContractListener	Adds a listener to listen to events emitted by committed transactions.	Consumer<ContractEvent> listener	Consumer<ContractEvent> listener

API	Description	Setting	Returned Values
removeContractListener	Removes a listener.	Consumer<ContractEvent> listener	void

4.4 RESTful API Demo

Huawei Cloud BCS provides RESTful APIs to simplify the usage of blockchains. You only need to develop applications that support RESTful APIs to access blockchains without the need to learn Hyperledger Fabric SDKs for Golang, Java, and Node.js. This demo uses a Golang client to show how RESTful APIs are used to invoke a chaincode.

NOTE

This is a demo only and is not for actual use.

Creating a BCS Instance

Step 1 Log in to the BCS console.

Step 2 Click **Buy** next to **Enhanced Hyperledger Fabric Instance**.

Step 3 Configure basic information about the BCS instance by referring to **Table 4-8**.

NOTICE

To ensure that the demo runs properly, set the parameters as described in the following table.

Table 4-8 Basic settings

Parameter	Setting
Billing Mode	Select **Pay-per-use**.
Region	Retain the default value.
Enterprise Project	Select **default**.
Instance Name	Enter **demo**.
Edition	Select **Basic**.
Blockchain Type	Select **Private**.
Enhanced Hyperledger Fabric Version	v2.2
Consensus Mechanism	Raft(CFT)

Parameter	Setting
Resource Access Initial Password	Enter a password.
Confirm Password	Confirm the password.

Step 4 Click **Next: Configure Resources**. Table 4-9 describes the resource parameters.

Table 4-9 Resource configurations

Parameter	Example
Environment Resources	Select **Custom**.
Cluster	Select **Create a new CCE cluster**.
AZ	Select an AZ.
ECS Specifications	Select the flavor for **4 vCPUs | 8 GB**.
ECS Quantity	Enter **1**.
High Availability	Select **No**.
VPC	Select **Automatically create VPC**.
Subnet	Select **Automatically create subnet**.
ECS Login Method	Select **Password**.
Password of Root User	If you do not enter a password here, the previously specified resource access initial password will be used.
Confirm Password	-
Use EIP of a CCE Node	Select **Yes**.
EIP Billed By	Retain the default value.
EIP Bandwidth	Set it to 5 Mbit/s.

Step 5 Click **Next: Configure Blockchain**. Table 4-10 describes the blockchain parameters.

Table 4-10 Blockchain configurations

Parameter	Example
Blockchain Configuration	Select **Custom**.
Blockchain Mgmt. Initial Password	If you do not enter a password here, the previously specified resource access initial password will be used.

Parameter	Example
Confirm Password	-
Volume Type	Select **SFS Turbo**.
Storage Capacity of Peer Organization (GB)	Retain the default value.
Ledger Storage	Select **File database (GoLevelDB)**.
Peer Organization	A peer organization named **organization** has been automatically created. Change the peer quantity to 1.
Channel Configuration	The **organization** organization has been added to the channel automatically. Retain this default setting.
Orderer Quantity	Retain the default value.
Security Mechanism	Select **ECDSA**. **NOTICE** Only **ECDSA** can be selected.
Configure Block Generation	Select **No**.
Enable Support for RESTful API	Select **Yes**. If you select **No**, you can enable support for RESTful APIs later by performing the following steps: 1. In the navigation pane on the left, choose **Add-on Management**. 2. On the **Add-on Repository** tab page, hover the mouse pointer over the **baas-restapi** card. 3. Click **Install** and select the purchased BCS instance.

Step 6 Click **Next: Confirm**.

Step 7 Confirm the configurations and finish the purchase process.

Wait for several minutes. After a message is displayed indicating successful installation, check the status of the instance. If it is **Normal**, the deployment is completed.

----End

Installing and Instantiating a Chaincode

Step 1 Log in to the BCS console.

Step 2 Find the instance you just created and click **Manage Blockchain** to go to the Blockchain Management console.

Step 3 On the login page, enter the username and password, and click **Log In**.

NOTE

The username is **admin**, and the password is the **Blockchain Mgmt. Initial Password** set when you created the BCS instance. If you have not set this password, use the resource access initial password.

Step 4 Click 

in the upper left corner of the page.

The parameters for chaincode installation are as follows.

Parameter	Setting
Chaincode Name	Enter **bcsysq**.
Chaincode Version	Enter **1.0**
Ledger Storage	File database (goleveldb)
Select All Peers	Check the box.
Organization & Peer	Select **peer-0**.
Language	Select **Golang**.
Chaincode File	Add the downloaded chaincode file **chaincode_example02.zip**.
Chaincode Description	Enter a description of the chaincode.
Code Security Check	This option is displayed only when the chaincode language is Golang. Enable this option to check chaincode security.

Step 5 Click **Install**.

Step 6 After installing the chaincode, click **Instantiate** in the **Operation** column of the chaincode list.

The parameters for chaincode instantiation are as follows.

Parameter	Setting
Channel	Select **channel**.
Chaincode Version	Enter **1.0**
Initialization Function	Enter **init**.
Chaincode Parameters	Enter **a,200,b,250**.
Endorsement Policy	Select **Endorsement from any of the following organizations**.
Endorsing Organizations	Select **organization**.
Privacy Protection Configuration	Select **No**.

----End

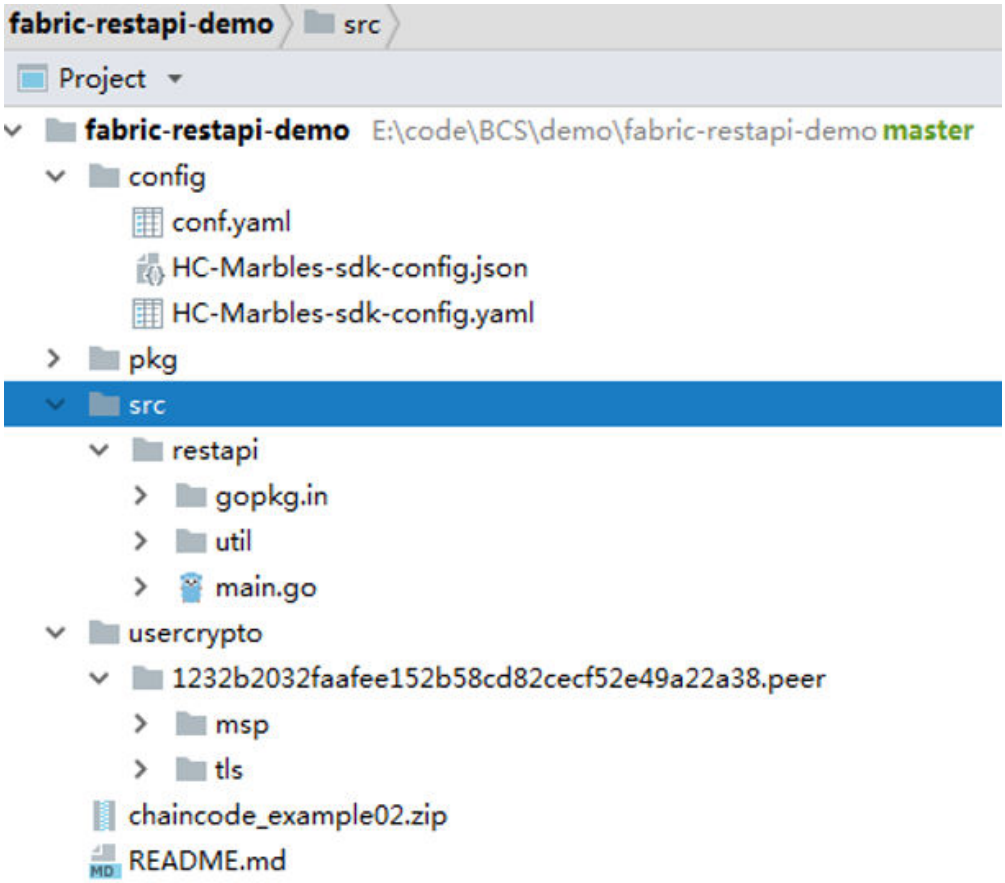
Configuring the Application

Step 1 On the **Instance Management** page, click **Download Client Configuration** on an instance card.

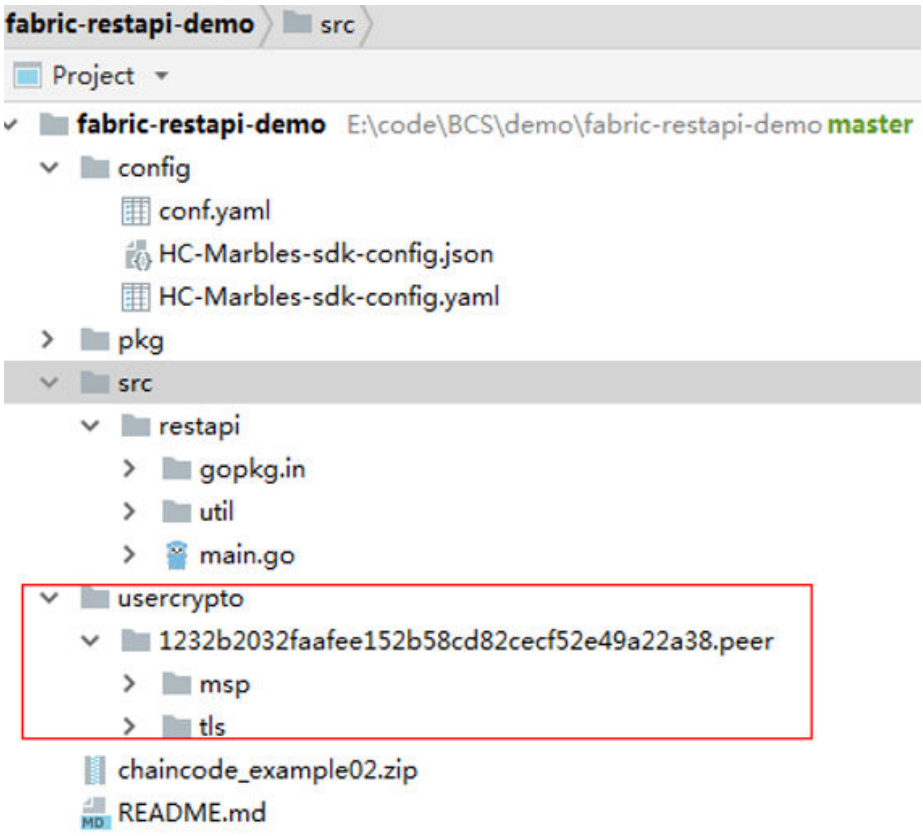
Step 2 Select **Peer Certificates**, select **organization** for **Peer Organization**, and select **User certificate** to download.

Step 3 Download and decompress the demo project code package **fabric-restapi-demo.zip** to the local PC, and use an IDE to open it.

This demo project is a RESTful client compiled using Golang. It enables chaincode invocation through RESTful APIs to achieve chaincode-based money transfer. Use an IDE such as GoLand to open the package. The following figure shows the content of the project.



Step 4 Decompress the downloaded user certificate to the **usercrypto** directory of the project, as shown in the following figure.



Step 5 Modify parameter settings.

1. Modify the parameters in the **conf.yaml** file in the **config** directory as shown and described in the following figure and tables.

2. Modify the **main.go** file in the **src/restapi** directory, as shown in the following figure and tables.

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


NOTE

For each peer that needs to participate in the endorsement, construct an OrgPeer structure including the organization ID and the domain name of the peer. Add the structure to an array of the OrgPeer type, convert the structure into a byte array using the json.Marshal() method, and then convert the structure into a character string. The OrgPeer structure is as follows:

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

Table 4-11 Parameters

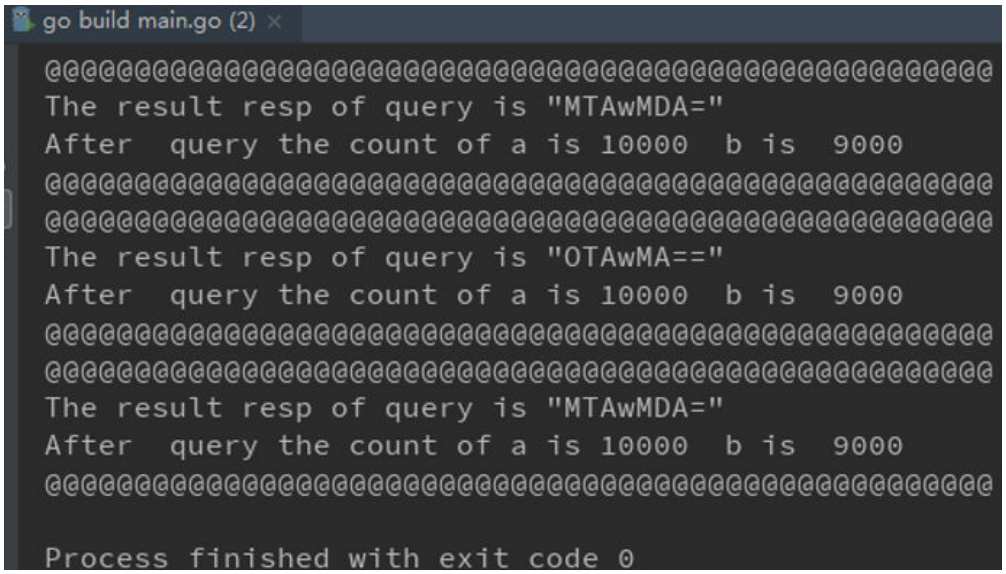
Parameter	Description
Endpoint	IP address and port number of the server bearing the RESTful service endpoint, which can be obtained by performing the following steps: 1. On an instance card, click **Container Cluster** to go to the CCE console. 2. Click an instance's cluster. In the navigation pane on the left, click **Workloads**. 3. On the **Deployments** page, click the **baas-restapi** workload. Click the node where the instance is located. On the **Nodes** page, obtain the value of EIP in the **IP Address** column. This value allows you to access the RESTful API service. The port is fixed to 32621.
Path	Path to the RESTful APIs service. Retain the default value.
CryptoMethod	Encryption algorithm. If the ECDSA algorithm is used, set this parameter to **SW**.
SignCert	Path to the signature in the downloaded certificate.
PrvKey	Private key in the downloaded certificate.
InvokeReq	Request body parameters. Set these parameters based on the deployed chaincode. The **InvokeReq** parameter descriptions in the following table are for your reference.
QueryReq	Similar to **InvokeReq**. Set this parameter based on the deployed chaincode.

Table 4-12 InvokeReq parameters

Parameter	Description	Example Value
SignGzipSwitch	Indication of whether GZIP compression is enabled. **0** indicates disabling, and **1** indicates enabling.	"1"
ChannelId	Blockchain channel name.	"channel"
ChaincodeId	Chaincode name.	"testcode"
Chaincode Version	Chaincode version.	"1.0"
UserId	User ID issued by the organization CA. The default value for BCS is **User1**.	"User1"
OrgId	Organization ID in a blockchain. **NOTE** On the **Channel Management** page, click **View Peer**. The organization ID is the value of **MSP ID** without "MSP". For example, if the MSP ID is **1232b2032faafee152b58cd82cecf52e49a22a38MSP**, the blockchain organization ID is **1232b2032faafee152b58cd82cecf52e49a22a38**.	"1232b2032faafee152b58cd82cecf52e49a22a38"
OrgPeers	Organization ID and domain name of each peer.	"[{OrgId:"1232b2032faafee152b58cd82cecf52e49a22a38", PeerDomainName:"peer-1232b2032faafee152b58cd82cecf52e49a22a38-0 .peer-1232b2032faafee152b58cd82cecf52e49a22a38.default.svc.cluster.local "}]"
Opmethod	Purpose, that is, to **invoke** or **query** chaincodes.	"invoke"
Args	Chaincode invoking parameters.	["invoke","a","b","100"]

Step 6 Build and run main().

The code will read the **QueryReq** and **InvokeReq** parameters in **conf.yaml** and **main.go** and call **/v1/chaincode/operation** of the RESTful APIs to invoke the chaincode. The code running result is as follows.



NOTE

This demo uses a simple REST client to invoke the chaincode through RESTful APIs. The returned invocation result is TransactionID encrypted using Base64, and the query result is data encrypted using Base64. The code is for reference only. You can use this project code to understand how to invoke RESTful APIs.

----End

5 Blockchain Middleware APIs

5.1 Overview

BCS provides blockchain middleware to support the development of upper-layer applications. You can easily use standard APIs to quickly access the middleware capabilities. **Table 5-1** lists the APIs.

This chapter describes data plane APIs. For details about management plane APIs, see **API Reference**.

The endpoint of data plane requests can be obtained from the value of the **basic_info->agent_portal_addrs** field returned by the management plane API for querying BCS instance details. An example request is **https://192.168.0.90:30603/v2/agent/apis/tokens**.

Table 5-1 APIs

API	Description	Middleware	Port
Chaincode Invoking (OBT)	Accessing the blockchain system to invoke and query chaincodes **NOTE** This function is under OBT.	baas-restapi add-on Installation steps: Log in to the BCS console. In the navigation pane, choose **Add-on Management**. On the **Add-on Repository** tab page, click **Install** in the card of the baas-restapi add-on.	32621
Chaincode Management (OBT)	Installing, instantiating, deleting, and querying chaincodes **NOTE** This function is under OBT.	baas-agent (installed by default)	30603

Distributed Identity (OBT)	Generating, applying for, and signing decentralized identifiers (DIDs) and verifiable credentials (VCs) **NOTE** This function is under OBT.	baas-restapi add-on Installation steps: Log in to the BCS console. In the navigation pane, choose **Add-on Management**. On the **Add-on Repository** tab page, click **Install** in the card of the baas-restapi add-on. On the **Install Add-on** page, enable DID APIs.	32621
Trusted Data Exchange (OBT)	Trusted data exchange based on the blockchain, involving data release, authorization, sharing, and decryption **NOTE** This function is under OBT.	baas-restapi add-on Installation steps: Log in to the BCS console. In the navigation pane, choose **Add-on Management**. On the **Add-on Repository** tab page, click **Install** in the card of the baas-restapi add-on. On the **Install Add-on** page, enable trusted data exchange APIs.	32621

5.2 Chaincode Invoking (OBT)

Function

This API is used to invoke and query the instantiated chaincodes of deployed BCS services.

URI

POST /v1/chaincode/operation

Request

Table 5-2 Request parameters

Parameter	Mandatory	Type	Description
channelId	Yes	String	Channel ID in a blockchain.

Parameter	Mandatory	Type	Description
chaincodeId	Yes	String	Chaincode ID.
chaincodeVersion	No	String	Chaincode version.
userId	Yes	String	User ID issued by the organization CA. Currently, the default value generated for BCS is **User1**.
orgId	Yes	String	Organization ID in a blockchain.
orgPeers	Yes	String	A character string consisting of the organization ID and domain name of each peer in an organization. The format is as follows: [{"orgId":"7258adda1803f4137eff4813e7aba323018200c5","peerDomainName":"peer-7258adda1803f4137eff4813e7aba323018200c5-0.peer-7258adda1803f4137eff4813e7aba323018200c5.default.svc.cluster.local"}]
opmethod	Yes	String	Purpose, that is, to **invoke** or **query** chaincodes.
args	Yes	String	Arguments, for example, ["Invoke", "a", "b", "1"]
timestamp	Yes	String	For example, 2018-10-31T17:28:16+08:00.
cert	Yes	String	User certificate file, which is uploaded in the form of a character string.

NOTE

For details about how to obtain the values of the preceding parameters, see **Chaincode Management** and **Block Browser**.

- On the **Chaincode Management** page, click in front of a chaincode to view its details, including the version, installation, and instantiation information.
- On the **Block Browser** page, select a channel to view real-time blockchain information, including the block quantity, transaction quantity, block details, transaction details, performance, and peer statuses.
- To ensure transaction security, you must use the private key in the Fabric user certificate (downloaded by following instructions in **Downloading the User Certificate**) to sign the request body. Currently, only the ECDSA encryption method is supported. Other encryption algorithms such as OSCCA-published cryptographic algorithms are not supported. Then, place the signature result in the **x-bcs-signature-sign** field in the request header.

Table 5-3 lists the request header parameters customized for the chaincode invoking RESTful API.

Table 5-3 Customized header parameters

Parameter	Mandatory	Description
x-bcs-signature-sign	Yes	Signature of the chaincode invoking request message body
x-bcs-signature-method	Yes	Encryption type, which is fixed at **SW** now.
x-bcs-signature-sign-gzip	Yes	Indication of whether GZIP compression is enabled. **0**: disabled; **1**: enabled.

NOTE

x-bcs-signature-sign: To ensure that only authorized invocation entities can invoke chaincodes, the user private key (downloaded by following instructions in **Downloading the User Certificate**) and the ECDSA encryption method must be used to encrypt and sign the SHA256 hash of the entire request body. The value of **x-bcs-signature-sign** is the encrypted and signed hash.

Downloading the User Certificate

Download the user certificate that is configured in BCS to call the APIs.

Step 1 Log in to the Huawei Cloud BCS console.

Step 2 On the **Instance Management** page, check the BCS instances.

Step 3 Click **Download Client Configuration**, and select **Peer Certificates**. Specify the peer organization and select **User certificate**.

Step 4 Click **Download**.

Step 5 Decompress the certificates. In the **msp** folder, the private key of the organization is stored in **keystore**, and the user certificate (public key) in **signcerts**.

----End

Response

- If **opmethod** is **invoke**, the **transactionID** encrypted and encoded using Base64 is returned.
- If **opmethod** is **query**, the query result returned by the chaincode is also encrypted and encoded using Base64.

Examples

The following is an example of invoking a chaincode:

- Example request

```
{
  "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"
}
```

- Example response

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

Error Codes

See **Error Codes**.

5.3 Chaincode Management

5.3.1 Obtaining a Token

Function

Obtaining Token from Blockchain Browser through user name and password.

URI

GET /v2/agent/apis/tokens

Request Parameters

Table 5-4 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-User	Yes	String	Username.
X-Auth-Pass	Yes	String	Password.

Response Parameters

Status code: 200

Table 5-5 Response body parameters

Parameter	Type	Description
user	String	Username.
token	String	token
expire_time	Long	Expiration time.

Status code: 400

Table 5-6 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 200

Success

```
{
  "user" : "admin",
  "token" :
"ACiUxRlk2Jcu2d5bp2SyfP7abHV1nVKlo9Mm2AxI9dDN7mdHXXHBdg=kaIglHeZLTu6UeIu5YRcXTlpUPXylEG3
ESNu0vIukrikG6SVhO393ds8rG+aRnZ+mMyookApP",
```



```
  "expire_time" : 1605867860701
}
```

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.2 Installing a Chaincode

Function

This API is used to install a chaincode on blockchain nodes. In some scenarios, only Go chaincodes are supported.

URI

POST /v2/agent/apis/chaincode/install

Request Parameters

Table 5-7 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Table 5-8 FormData parameters

Parameter	Mandatory	Type	Description
chaincode_name	Yes	String	Chaincode name, which must start with a letter and can contain lowercase letters and digits. Minimum: **6** Maximum: **25**
chaincode_version	Yes	String	Chaincode version. Only digits, periods (.), and hyphens (-) are allowed. The value must start and end with a digit, and the periods (.) and hyphens (-) cannot be adjacent. Minimum: **1** Maximum: **14**
target_peers	Yes	String	List of peers where the chaincode is installed, for example, [{"org_id":"9fb42c91458763990a45b62af92546a21f168dae","org_name":"organization3","peer_id":"peer-9fb42c91458763990a45b62af92546a21f168dae-0.peer-9fb42c91458763990a45b62af92546a21f168dae.default.svc.cluster.local","peer_name":"peer-0"}].
description	No	String	Chaincode description.
chaincode_language	Yes	String	Programming language of the chaincode, for example, Go.
db_type	No	String	Database type used by the chaincode. Set this parameter based on the ledger data storage mode set during instance creation. Example: **goleveldb** or **couchdb**.
security_check	No	String	Chaincode security check option. This parameter is valid only for Golang chaincodes. The value **true** indicates that the function is enabled, and the value **false** indicates that the function is disabled. The default value is **false**.

Parameter	Mandatory	Type	Description
file	Yes	File	Chaincode ZIP file.

Response Parameters

Status code: 200

Table 5-9 Response body parameters

Parameter	Type	Description
total_peer_num	Integer	Total number of peers where the chaincode is installed.
success_peer_num	Integer	Number of peers where the chaincode is installed successfully.
fail_peer_num	Integer	Number of peers where the chaincode fails to be installed.
fail_peers	Array of strings	Details of peers where the chaincode fails to be installed.

Status code: 400

Table 5-10 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

```
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}
```

Example Responses

Status code: 200

Success

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

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.3 Instantiating a Chaincode

Function

Instantiating a Chaincode

URI

POST /v2/agent/apis/chaincode/instantiate

Request Parameters

Table 5-11 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Table 5-12 Request body parameters

Parameter	Mandatory	Type	Description
chaincode_name	Yes	String	Chaincode name, which can contain 6 to 25 including lowercase letters and digits, and must start with a letter. Minimum: **6** Maximum: **25**
chaincode_version	Yes	String	Chaincode version. Only digits, periods (.), and hyphens (-) are allowed. The value must start and end with a digit, and the periods (.) and hyphens (-) cannot be adjacent.
channel_name	Yes	String	Channel name.
endorsement_policy	Yes	**Policy** object	Endorsement policy.
init_variable	Yes	**InitArgs** object	Initialization function and parameters.
private_data	No	String	Privacy protection configuration data, which is a JSON string. For example: [{"name":"kvstore-collection","policy":"OR('9b4ea44913e8eed42cb056177b46191e1d316678MSP.member','9b4ea44913e8eed42cb056177b46191e1d316678MSP.member')","requiredPeerCount":0,"maxPeerCount":2,"blockToLive":0,"memberOnlyRead":true}].

Table 5-13 Policy

Parameter	Mandatory	Type	Description
operation	Yes	String	Operator. **OR**: endorsement from any organization. **AND**: endorsement from all organizations.
group	Yes	Array of **OrgPolicy** objects	Endorsing organizations.

Table 5-14 OrgPolicy

Parameter	Mandatory	Type	Description
org_id	Yes	String	Organization ID.
category	No	String	Type, which can be **member** or **admin*.

Table 5-15 InitArgs

Parameter	Mandatory	Type	Description
func_name	Yes	String	Initialization function.
args	Yes	Array of strings	Initialization parameters.

Response Parameters

Status code: 400

Table 5-16 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

```
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}]"
}
```

Example Responses

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.4 Listing Installed Chaincodes

Function

This API is used to list installed chaincodes.

URI

GET /v2/agent/apis/chaincodes

Table 5-17 Query Parameters

Parameter	Mandatory	Type	Description
offset	No	Integer	Offset to start querying the chaincode list. The default value is **0**.
limit	No	Integer	Number of listed chaincodes. The default value is **10**.

Request Parameters

Table 5-18 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-19 Response body parameters

Parameter	Type	Description
count	Integer	Total number of chaincodes.
chaincodes	Array of **ChaincodeInfo** objects	Chaincodes.

Table 5-20 ChaincodeInfo

Parameter	Type	Description
chaincode_name	String	Chaincode name.
chaincode_language	String	Chaincode programming language.
update_time	String	Chaincode update time.
chaincode_version	String	Chaincode version. Separate multiple chaincode versions with commas (,).
install_org_infos	Array of **PeerInfo** objects	Chaincode installation information.
instantiated_channel	**instantiated_channel** object	Chaincode channel information.
instantiated_info	**instantiated_info** object	Instantiation information.

Table 5-21 PeerInfo

Parameter	Type	Description
org_name	String	Organization name.
org_id	String	Organization ID.
peer_name	String	Peer name.
peer_id	String	Peer ID.
status	String	Peer status.
channels	Array of strings	Peers where the chaincode is not instantiated.
url	String	URL of the peer.
peer	String	Internal domain name of the peer.

Table 5-22 instantiated_channel

Parameter	Type	Description
error	Array of **CCInstantiatedChannelError** objects	Instantiation error information.
success	Array of strings	Channels where the instantiation succeeded.
inprogress	Array of strings	Instantiation progress.

Table 5-23 CCInstantiatedChannelError

Parameter	Type	Description
channel_name	String	Name of the channel where the error occurred.
error_detail	String	Error details.

Table 5-24 instantiated_info

Parameter	Type	Description
channels	Array of **channels** objects	Channel information.

Table 5-25 channels

Parameter	Type	Description
channel_id	String	Channel name.
orgs	Array of **orgs** objects	Channel organization information.
versions	Array of strings	Version.

Table 5-26 orgs

Parameter	Type	Description
org_name	String	Organization name.
org_id	String	Organization ID.

Status code: 400

Table 5-27 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 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"
  } ]
}
```


Status code: 400

Bad Request


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


Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.5 Querying Version of a Specified Chaincode

Function

Querying Version of a Specified Chaincode

URI

GET /v2/agent/apis/chaincode/versions

Table 5-28 Query Parameters

Parameter	Mandatory	Type	Description
chaincode_name	Yes	String	Chaincode name, which can contain 6 to 25 including lowercase letters and digits, and must start with a letter. Minimum: **6** Maximum: **25**

Request Parameters

Table 5-29 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-30 Response body parameters

Parameter	Type	Description
versions	Array of **ChaincodeVersion** objects	Chaincode version information.

Table 5-31 ChaincodeVersion

Parameter	Type	Description
version	String	Chaincode version.
hash_code	String	Hash value of the chaincode version.
description	String	Chaincode version description.
install_time	String	Chaincode version installation time.
update_time	String	Chaincode version update time.
instantiate_status	Boolean	Chaincode version instantiation status.

Parameter	Type	Description
security_check_status	Integer	Chaincode security check status. The options are as follows: **0**: The check does not exist; **1**: The check is running; **2**: The check is completed; **3**: The check failed.
uninstantiated_peer_infos	Array of **PeerInfo** objects	Peers where the chaincode is not instantiated.

Table 5-32 PeerInfo

Parameter	Type	Description
org_name	String	Organization name.
org_id	String	Organization ID.
peer_name	String	Peer name.
peer_id	String	Peer ID.
status	String	Peer status.
channels	Array of strings	Peers where the chaincode is not instantiated.
url	String	URL of the peer.
peer	String	Internal domain name of the peer.

Status code: 400

Table 5-33 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 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
  } ]
}
```


Status code: 400

Bad Request


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


Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.6 Querying Chaincode Installation Information

Function

This API is used to query the information about chaincode installation on peers.

URI

GET /v2/agent/apis/chaincode/install

Table 5-34 Query Parameters

Parameter	Mandatory	Type	Description
chaincode_name	Yes	String	Chaincode name, which can contain 6 to 25 including lowercase letters and digits, and must start with a letter. Minimum: **6** Maximum: **25**

Request Parameters

Table 5-35 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-36 Response body parameters

Parameter	Type	Description
result	Array of **PeerInstallInfo** objects	Information about chaincode installation on peers.

Table 5-37 PeerInstallInfo

Parameter	Type	Description
org_name	String	Name of the organization to which the peer belongs.
org_id	String	ID of the organization to which the peer belongs.
peer_name	String	Peer name.
peer_id	String	Peer ID.
install_status	String	Chaincode installation result, which can be **installed** or **uninstalled**.
version	String	Chaincode version.

Status code: 400

Table 5-38 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 200

Success

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

Status code: 400

Bad Request

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


Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.7 Querying Chaincode Instantiation Information

Function

This API is used to query the information about chaincode instantiation on a channel.

URI

GET /v2/agent/apis/chaincode/instantiate

Table 5-39 Query Parameters

Parameter	Mandatory	Type	Description
chaincode_name	Yes	String	Chaincode name, which can contain 6 to 25 including lowercase letters and digits, and must start with a letter. Minimum: **6** Maximum: **25**

Request Parameters

Table 5-40 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-41 Response body parameters

Parameter	Type	Description
result	Array of **ChannelInstantiateInfo** objects	Information about chaincode instantiation on a channel.

Table 5-42 ChannelInstantiateInfo

Parameter	Type	Description
channel_name	String	Channel name.
instantiate_info	**InstantiateInfo** object	Instantiation information.
endorsement_policy	String	Endorsement policy.
version	String	Chaincode version.
orgs_info	Array of **OrgInfo** objects	Channel organization information.
has_private_data	Integer	Indication of whether privacy data exists. **1**: Privacy data exists; **0**: Privacy data does not exist.

Table 5-43 InstantiateInfo

Parameter	Type	Description
status	String	Instantiation status, which can be **CHAINCODE_INSTANTIATED**, **CHAINCODE_INSTANTIATION_INPROGRESS**, and **CHAINCODE_INSTANTIATION_FAILED**.
code	String	Code of the instantiation result.
reason	String	Cause of the instantiation result.
detail	String	Details of the instantiation result.

Table 5-44 OrgInfo

Parameter	Type	Description
org_name	String	Organization name.
org_id	String	Organization ID.

Status code: 400

Table 5-45 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 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
  } ]
}
```

Status code: 400

Bad Request

```
{
  "error_code" : "BCS.4000013",
```



```
  "error_message" : "request body is too large"
}
```

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.8 Querying an Appchain

Function

This API is used to query the channel information.

URI

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

Table 5-46 Path Parameters

Parameter	Mandatory	Type	Description
channel_name	Yes	String	Channel name.

Request Parameters

Table 5-47 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token, which is obtained by calling an API.

Response Parameters

Status code: 200

Table 5-48 Response body parameters

Parameter	Type	Description
peer_num	Number	Peer quantity.
chaincodes	Number	Contract quantity.
block_num	Number	Block quantity.
transaction_num	Number	Transaction quantity.
bph	Array of **bph** objects	Statistics on the number of blocks generated each hour.
bpm	Array of **bpm** objects	Statistics on the number of blocks generated every 5 minutes.
tph	Array of **tph** objects	Statistics on the number of contract transactions by hour.
tpm	Array of **tpm** objects	Statistics on the number of contract transactions every 5 minutes.
orgs_map	Array of **orgs_map** objects	Organization transaction statistics.
peers_list	Array of **peers_list** objects	Peer list.

Table 5-49 bph

Parameter	Type	Description
block	Number	Block quantity.
time	String	Time.

Table 5-50 bpm

Parameter	Type	Description
block	Number	Block quantity.
time	String	Time.

Table 5-51 tph

Parameter	Type	Description
tx	Number	Transaction quantity.
time	String	Time.

Table 5-52 tpm

Parameter	Type	Description
tx	Number	Transaction quantity.
time	String	Time.

Table 5-53 orgs_map

Parameter	Type	Description
org_name	String	Organization name.
tx_num	Number	Transaction quantity.

Table 5-54 peers_list

Parameter	Type	Description
org_name	String	Organization name.
org_id	String	Organization ID.
peer	String	Peer name.
url	String	PeerURL
status	String	Peer status.

Status code: 400

Table 5-55 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.

Parameter	Type	Description
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.9 Listing Blocks

Function

Listing Blocks

URI

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

Table 5-56 Path Parameters

Parameter	Mandatory	Type	Description
channel_name	Yes	String	Channel name. The value can contain 4 to 24 characters and cannot be the same as the system channel name (**testchainid**).

Table 5-57 Query Parameters

Parameter	Mandatory	Type	Description
is_pagination	No	Boolean	Whether to display records on different pages. By default, pagination is not enabled. Default: **false**
offset	No	Integer	Offset. The default value is **0**. Minimum: **0** Default: **0**
limit	No	Integer	Number of records displayed on each page. The default value is **10**. The value ranges from 1 to 50. Minimum: **1** Maximum: **50** Default: **10**

Request Parameters

Table 5-58 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-59 Response body parameters

Parameter	Type	Description
count	Integer	Total number of blocks.
data	Array of **BlockHeader** objects	Block details.

Table 5-60 BlockHeader

Parameter	Type	Description
block_number	Integer	Block ID.
block_hash	String	Hash of the block.
transaction_count	Integer	Total number of transactions.
data_hash	String	Data hash value.
previous_hash	String	Hash of the previous block.
timestamp	String	Timestamp.
organization_maps	Map<String,Integer>	The key is the creator MSP and the value is the quantity.

Status code: 400

Table 5-61 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

- Default mode request.

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

- Paging mode request.

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

Example Responses

Status code: 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}"
  } ]
}
```

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.10 Listing Transactions

Function

Listing Transactions

URI

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

Table 5-62 Path Parameters

Parameter	Mandatory	Type	Description
channel_name	Yes	String	Channel name. The value can contain 4 to 24 characters and cannot be the same as the system channel name (**testchainid**).

Table 5-63 Query Parameters

Parameter	Mandatory	Type	Description
is_pagination	No	Boolean	Whether to display records on different pages. By default, pagination is not enabled. Default: **false**
offset	No	Integer	Offset. The default value is **0**. Minimum: **0** Default: **0**
limit	No	Integer	Number of records displayed on each page. The default value is **10**. The value ranges from 1 to 50. Minimum: **1** Maximum: **50** Default: **10**

Request Parameters

Table 5-64 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-65 Response body parameters

Parameter	Type	Description
count	Integer	Total number of transactions.
data	Array of **TransactionSummary** objects	Transaction.

Table 5-66 TransactionSummary

Parameter	Type	Description
organization_name	String	Creator organization.
type	String	Transaction type.
transaction_id	String	Transaction ID.
chaincode_name	String	Chaincode name.
timestamp	String	Timestamp.
channel_name	String	Channel name.
creator_msp	String	Identity information.
chaincode_version	String	Chaincode version.
block_number	Integer	Block ID.

Status code: 400

Table 5-67 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

- Default mode request.

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

- Paging mode request.

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

Example Responses

Status code: 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"
 } ]
}
```


Status code: 400

Bad Request


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


Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.11 Querying Transaction Quantity

Function

Querying Transaction Quantity

URI

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

Table 5-68 Path Parameters

Parameter	Mandatory	Type	Description
channel_name	Yes	String	Channel name.

Request Parameters

Table 5-69 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-70 Response body parameters

Parameter	Type	Description
channel_id	String	Channel ID.
block_height	Integer	Block height.
transaction_num	Integer	Transaction quantity.

Status code: 400

Table 5-71 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 200

Success

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


Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.12 Listing Block Transactions

Function

Listing Block Transactions

URI

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

Table 5-72 Path Parameters

Parameter	Mandatory	Type	Description
channel_name	Yes	String	Channel name. The value can contain 4 to 24 characters and cannot be the same as the system channel name (**testchainid**).
block_num	Yes	String	Block ID.

Request Parameters

Table 5-73 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-74 Response body parameters

Parameter	Type	Description
[items]	Array of **ShowTransactionDetailRes** objects	Success

Table 5-75 ShowTransactionDetailRes

Parameter	Type	Description
read_set	Map<String,Array<**KVRead**>>	Read set "map[string][]*kvrwset.KVRead key:chaincode value".
write_set	Map<String,Array<**KVWrite**>>	Write set "map[string][]KVWrite key:chaincode value".
validation_code	String	Validation code.
endorser_organizations	Array of strings	Endorsing organizations.
proposal_hash	String	Request data hash.
transaction_summary	Object	Transaction details.

Table 5-76 KVRead

Parameter	Type	Description
key	String	Read key.
version	**version** object	Version of the key in the read set.

Table 5-77 version

Parameter	Type	Description
block_num	Integer	Block quantity.
tx_num	Integer	Transaction quantity.

Table 5-78 KVWrite

Parameter	Type	Description
key	String	Write key.
is_delete	Boolean	Indication of whether to delete.
value	String	Write value.

Table 5-79 TransactionSummary

Parameter	Type	Description
organization_name	String	Creator organization.
type	String	Transaction type.
transaction_id	String	Transaction ID.
chaincode_name	String	Chaincode name.
timestamp	String	Timestamp.
channel_name	String	Channel name.
creator_msp	String	Identity information.
chaincode_version	String	Chaincode version.
block_number	Integer	Block ID.

Status code: 400

Table 5-80 Response body parameters

Parameter	Type	Description
error_code	String	Error code.

Parameter	Type	Description
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 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" : [ ]
  }
} ]
```


Status code: 400

Bad Request


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


Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.13 Querying Transaction Details

Function

Querying Transaction Details

URI

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

Table 5-81 Path Parameters

Parameter	Mandatory	Type	Description
channel_name	Yes	String	Channel name.
transaction_id	Yes	String	Transaction ID.

Request Parameters

Table 5-82 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-83 Response body parameters

Parameter	Type	Description
read_set	Map<String,Array<**KVRead**>>	Read set "map[string][]*kvrwset.KVRead key:chaincode value".
write_set	Map<String,Array<**KVWrite**>>	Write set "map[string][]KVWrite key:chaincode value".
validation_code	String	Validation code.
endorser_organizations	Array of strings	Endorsing organizations.
proposal_hash	String	Request data hash.
transaction_summary	Object	Transaction details.

Table 5-84 KVRead

Parameter	Type	Description
key	String	Read key.
version	**version** object	Version of the key in the read set.

Table 5-85 version

Parameter	Type	Description
block_num	Integer	Block quantity.
tx_num	Integer	Transaction quantity.

Table 5-86 KVWrite

Parameter	Type	Description
key	String	Write key.
is_delete	Boolean	Indication of whether to delete.
value	String	Write value.

Table 5-87 TransactionSummary

Parameter	Type	Description
organization_name	String	Creator organization.
type	String	Transaction type.
transaction_id	String	Transaction ID.
chaincode_name	String	Chaincode name.
timestamp	String	Timestamp.
channel_name	String	Channel name.
creator_msp	String	Identity information.
chaincode_version	String	Chaincode version.
block_number	Integer	Block ID.

Status code: 400

Table 5-88 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 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" : [ ]
 }
}
```


Status code: 400

Bad Request


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


Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.14 Querying Peers

Function

Querying Peers

URI

GET /v2/agent/apis/peers

Request Parameters

Table 5-89 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-90 Response body parameters

Parameter	Type	Description
peers	Map<String,PeerInfo>	The key is the peer domain name, and value is the peer details.

Table 5-91 PeerInfo

Parameter	Type	Description
org_name	String	Organization name.
org_id	String	Organization ID.
peer_name	String	Peer name.
peer_id	String	Peer ID.
status	String	Peer status.
channels	Array of strings	Peers where the chaincode is not instantiated.
url	String	URL of the peer.
peer	String	Internal domain name of the peer.

Status code: 400

Table 5-92 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.

Parameter	Type	Description
error_msg	String	Error description.

Example Requests

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

Example Responses

Status code: 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"
    }
  }
}
```

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.15 Querying diskUsage of a Node

Function

This API is used to query the diskUsage of a node.

URI

GET /v2/agent/apis/peers/disk-usage

Request Parameters

Table 5-93 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Response Parameters

Status code: 200

Table 5-94 Response body parameters

Parameter	Type	Description
[items]	Array of **PeerDiskUsageRes** objects	Success

Table 5-95 PeerDiskUsageRes

Parameter	Type	Description
peer_name	String	Peer name.
volume_total	Long	Total storage.
volume_used	Long	Storage usage.
volume_free	Long	Available storage.

Status code: 400

Table 5-96 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

```
GET  https://192.168.0.90:30603/v2/agent/apis/peers/disk-usage
```

Example Responses

Status code: 200

Success

```
[ {
  "peer_name" : "peer-ee6d5dd863f7e24675f2f2db76f8dfdbaa9a07cf-0",
  "volume_total" : 105152176128,
  "volume_used" : 63434752,
  "volume_free" : 105088741376
} ]
```

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.16 Querying the System-Hosted Certificate Status

Function

This API is used to query the status of a certificate that is hosted by system.

URI

POST /v2/agent/apis/cert/status/{channel_id}

Table 5-97 Path Parameters

Parameter	Mandatory	Type	Description
channel_id	Yes	String	Channel name. The value can contain 4 to 24 characters and cannot be the same as the system channel name (**testchainid**).

Request Parameters

Table 5-98 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Table 5-99 Request body parameters

Parameter	Mandatory	Type	Description
user_name	No	String	Username used for generating a user certificate. Either user_name or cert_value needs to be transferred. The service preferentially reads the value of user_name. Users can use the username or directly transfer the certificate content to query the certificate status.
cert_value	No	String	Certificate content returned by the **Generating a User Certificate** API. The content is the content of the signcerts file in the file returned by the API for generating a user certificate, and \n is added to the end of the parameter value.
org_name	Yes	String	Organization name used for generating a user certificate.

Response Parameters

Status code: 200

Table 5-100 Response body parameters

Parameter	Type	Description
[items]	Array of booleans	Request sent. Certificate status contained in the response, which can be **true** (normal) or **false** (frozen).

Status code: 400

Table 5-101 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

```
POST https://192.168.0.90:30603/v2/agent/apis/cert/status/channeltest
```

Example Responses

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Request sent. Certificate status contained in the response, which can be **true** (normal) or **false** (frozen).
400	Bad Request

Error Codes

See **Error Codes**.

5.3.17 Deleting a Chaincode

Function

This API is used to delete a chaincode from blockchain nodes.

URI

DELETE /v2/agent/apis/chaincode/uninstall

Request Parameters

Table 5-102 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Table 5-103 Request body parameters

Parameter	Mandatory	Type	Description
chaincode_name	Yes	String	Chaincode name, which can contain 6 to 25 lowercase letters and digits and must start with a letter. Minimum: **6** Maximum: **25**
chaincode_version	Yes	String	Chaincode version. Only digits, periods (.), and hyphens (-) are allowed. The value must start and end with a digit, and the periods (.) and hyphens (-) cannot be adjacent.
target_peers	Yes	Array of **TargetPeer** objects	Details of peers where the chaincode is uninstalled.

Table 5-104 TargetPeer

Parameter	Mandatory	Type	Description
org_id	Yes	String	ID of the organization to which the peer belongs.
peer_id	Yes	String	Peer ID

Response Parameters

Status code: 200

Table 5-105 Response body parameters

Parameter	Type	Description
total_peer_num	Integer	Total number of peers where the chaincode is installed.
success_peer_num	Integer	Number of peers where the chaincode is installed successfully.
fail_peer_num	Integer	Number of peers where the chaincode fails to be installed.
fail_peers	Array of strings	Details of peers where the chaincode fails to be installed.

Status code: 400

Table 5-106 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

```
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"
  } ]
}
```

Example Responses

Status code: 200

Success

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

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

5.3.18 Downloading a Report

Function

This API is used to download the chaincode security check report.

URI

POST /v2/agent/apis/chaincode/report

Request Parameters

Table 5-107 Request header parameters

Parameter	Mandatory	Type	Description
X-Auth-Token	Yes	String	User token.

Table 5-108 Request body parameters

Parameter	Mandatory	Type	Description
chaincode_name	Yes	String	Chaincode name, which must be the same as that specified during chaincode installation. This parameter is mandatory when chaincode security check is enabled.
chaincode_version	Yes	String	Chaincode version, which must be the same as that specified during chaincode installation.

Response Parameters

Status code: 200

Table 5-109 Response body parameters

Parameter	Type	Description
-	File	Success

Status code: 400

Table 5-110 Response body parameters

Parameter	Type	Description
error_code	String	Error code.
error_message	String	Error description.
error_msg	String	Error description.

Example Requests

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

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

Example Responses

Status code: 400

Bad Request

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

Status Codes

Status Code	Description
200	Success
400	Bad Request

Error Codes

See **Error Codes**.

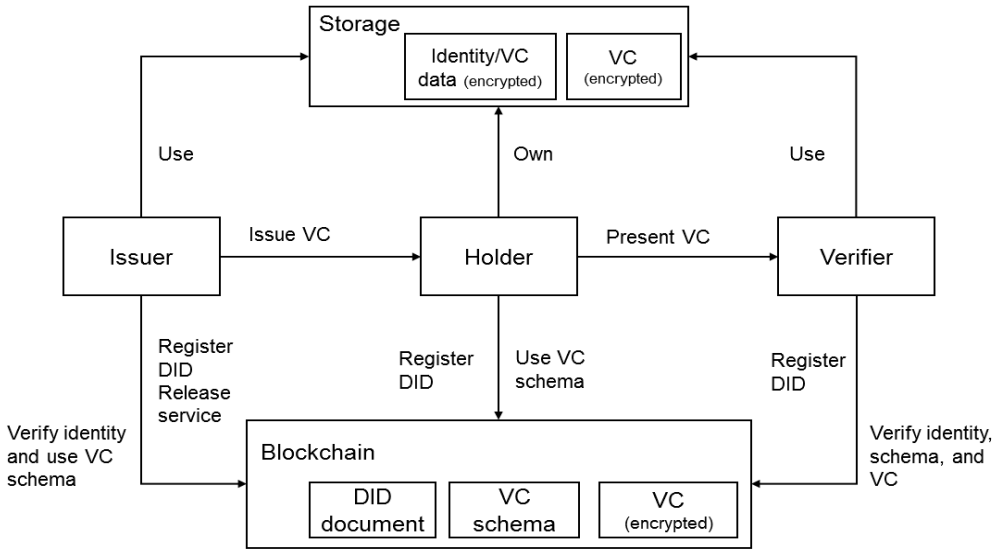
5.4 Distributed Identity (OBT)

5.4.1 Overview

Distributed identity (DID) is a blockchain-based identity management technology. It allows you to create user identities, and register, issue, and verify verifiable credentials (VCs). BCS's DID is implemented based on the W3C DID and VC specifications. It provides unified, self-explainable, and portable distributed identifiers for individual and enterprise users to address privacy issues and identity authentication across departments, enterprises, and regions.

Figure 5-1, **Figure 5-2**, **Figure 5-3**, and **Figure 5-4** illustrate the implementation and usage of DID.

Figure 5-1 DID implementation



Implementation

1. Each role can call the **Enterprise Identity Registration (with Service)** and **Registering a DID** APIs to generate their own DID which they fully control, and then publish the DID document to the blockchain to complete identity registration. The DID of an enterprise includes information about the services the enterprise can provide in various application scenarios.

   NOTE

   There are three roles: issuer, holder, and verifier. Each role can be a device, an application, an individual, or an organization.

2. VCs are used to authenticate identities. Relevant entities register and continuously maintain credential schemas on the blockchain. Holders initiate authentication requests to issuers, obtain credentials, and provide the credentials to verifiers to complete verification.
3. Verifiers verify VCs presented by holders by using an API to ensure that holders are qualified and permitted to proceed with relevant services.

Usage

The DID middleware is a set of microservices deployed on the user side to simplify the calling of blockchain APIs. When calling the DID API, the user private key and the Fabric-signed root certificate must be transferred.

NOTE

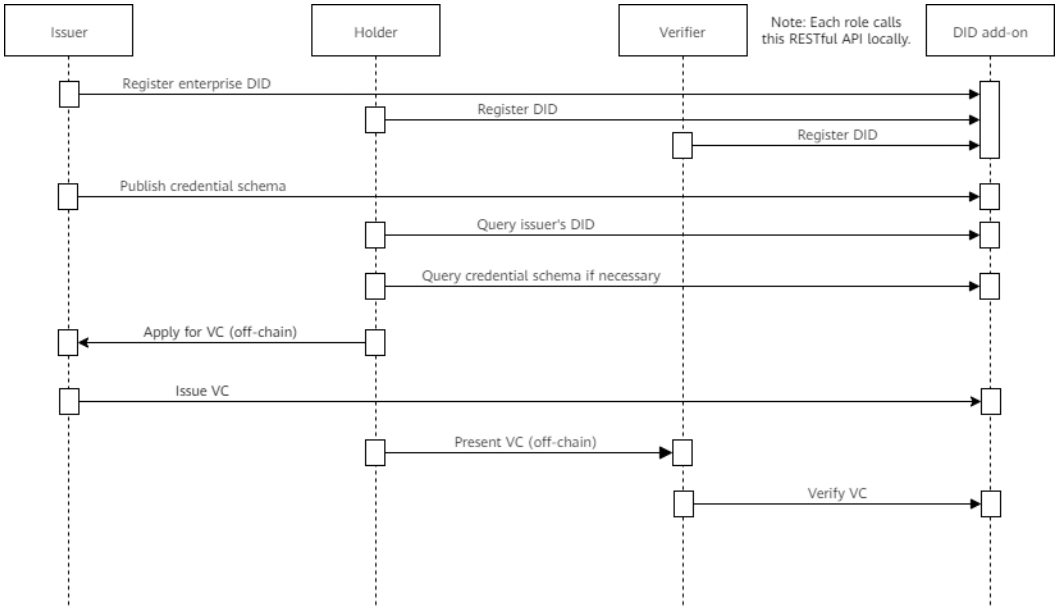
You can download the private key and certificate on the BCS console or generate them using OpenSSL. For details, see **How Can I Obtain Private Keys and Certificates of Fabric Users?**

Holders can apply for VCs on or off the blockchain.

- Off-chain application: The holder sends the identity or VC data for applying for a VC to the issuer.

- On-chain application: The holder encrypts and stores the identity or VC data for applying for a VC on the blockchain.

Figure 5-2 DID usage (off-chain application)



On-chain application can be online or offline, depending on whether communication channels are required between the holder and the issuer.

Figure 5-3 DID usage (on-chain, online application)

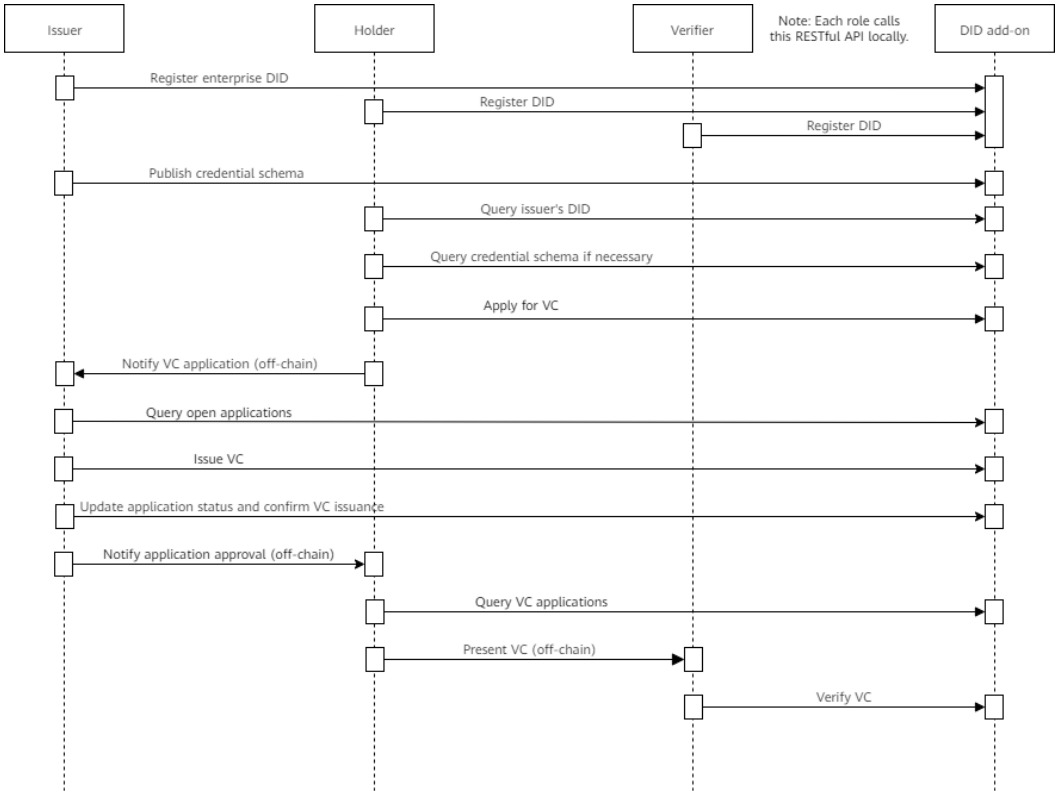
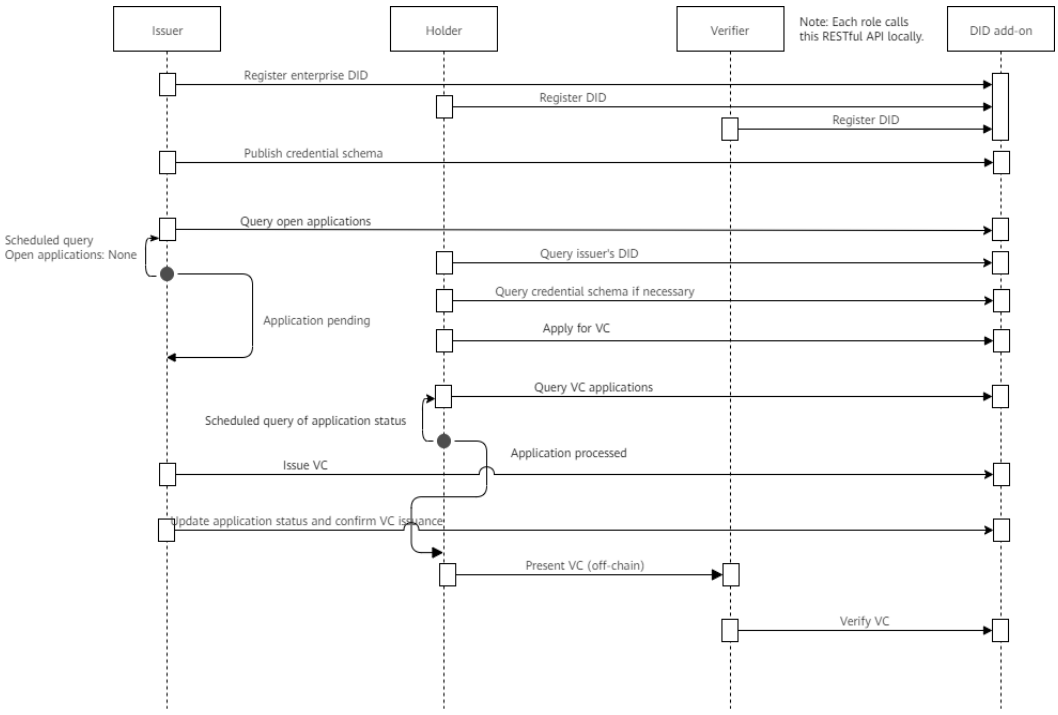


Figure 5-4 DID usage (on-chain, offline application)



5.4.2 Decentralized Identity (DID) Management

5.4.2.1 Enterprise Identity Registration (with service)

Function

This API is used to register a DID. Before calling this API, use the OpenSSL tool to generate a private key and the Fabric CA certificate for each user. You can also download the user certificate on the Service Management page of the BCS console. During registration, claim the list of available services.

URI

POST /v1/identity/firm-did

Request Parameters

Table 5-111 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.

Parameter	Mandatory	Type	Description
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
service	Yes	Array of **DIDService** objects	Service.

Table 5-112 DIDService

Parameter	Mandatory	Type	Description
type	Yes	String	Type.
serviceEndpoint	Yes	String	Endpoint.
credentialApplySchema	No	**CredentialApplySchema** object	Schema used for applying for a VC.

Table 5-113 CredentialApplySchema

Parameter	Mandatory	Type	Description
type	No	String	Type.
name	No	String	Name.
description	No	String	Description.
attributes	No	Array of **Attribute** objects	Attributes.

Table 5-114 Attribute

Parameter	Mandatory	Type	Description
name	No	String	Name.
type	No	String	Type.

Parameter	Mandatory	Type	Description
description	No	String	Description.

Response Parameters

Status code: 200

Table 5-115 Response body parameters

Parameter	Type	Description
did	String	Decentralized identifier.

Status code: 500

Table 5-116 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
      } ]
    }
  } ]
}
```


Example Responses

Status code: 200

Decentralized identifier.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Decentralized identifier.
500	Error response.

Error Codes

See **Error Codes**.

5.4.2.2 Registering a DID

Function

This API is used to register a DID. A DID can be conveniently registered and released. The DID does not provide services and has only one public key.

URI

POST /v1/identity/did

Request Parameters

Table 5-117 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.

Parameter	Mandatory	Type	Description
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.

Response Parameters

Status code: 200

Table 5-118 Response body parameters

Parameter	Type	Description
did	String	Decentralized identifier.

Status code: 500

Table 5-119 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
}
```


Example Responses

Status code: 200

Decentralized identifier.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Decentralized identifier.
500	Error response.

Error Codes

See **Error Codes**.

5.4.2.3 Updating a DID

Function

This API is used to update the services published in a DID document.

URI

PUT /v1/identity/did

Request Parameters

Table 5-120 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.

Parameter	Mandatory	Type	Description
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
did	Yes	String	Decentralized identifier.
service	No	Array of **DIDService** objects	Service list.

Table 5-121 DIDService

Parameter	Mandatory	Type	Description
type	Yes	String	Type.
serviceEndpoint	Yes	String	Endpoint.
credentialApplySchema	No	**CredentialApplySchema** object	Schema used for applying for a VC.

Table 5-122 CredentialApplySchema

Parameter	Mandatory	Type	Description
type	No	String	Type.
name	No	String	Name.
description	No	String	Description.
attributes	No	Array of **Attribute** objects	Attributes.

Table 5-123 Attribute

Parameter	Mandatory	Type	Description
name	No	String	Name.

Parameter	Mandatory	Type	Description
type	No	String	Type.
description	No	String	Description.

Response Parameters

Status code: 200

Table 5-124 Response body parameters

Parameter	Type	Description
context	String	context
id	String	Decentralized identifier.
publicKey	Array of **DocPublicKey** objects	Public keys.
authentication	Array of strings	Identifier of the DID master public key.
recovery	String	Identifier of the DID backup public key, which can be used to modify the master key.
service	Array of **Service** objects	Service list.
proof	**Proof** object	Proof structure, which can be left empty.
created	String	Creation time.
updated	String	Update time.
status	String	Status.

Table 5-125 DocPublicKey

Parameter	Type	Description
id	String	Public key identifier.
type	String	Public key type.
controller	String	Public key controller identifier.
publicKeyPem	String	Public key certificate.

Table 5-126 Service

Parameter	Type	Description
id	String	Service identifier.
type	String	Service type.
serviceEndpoint	String	Service endpoint.
credentialApplySchema	**CredentialApplySchema** object	Schema used for applying for a VC.

Table 5-127 CredentialApplySchema

Parameter	Type	Description
type	String	Type.
name	String	Name.
description	String	Description.
attributes	Array of **Attribute** objects	Attributes.

Table 5-128 Attribute

Parameter	Type	Description
name	String	Name.
type	String	Type.
description	String	Description.

Table 5-129 Proof

Parameter	Type	Description
creator	String	Creator identifier.
type	String	Signature type.
created	String	Signature creation time.
signatureValue	String	Signature.

Status code: 500

Table 5-130 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
      } ]
    }
  } ]
}
```

Example Responses

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	DID document structure.

Status Code	Description
500	Error response.

Error Codes

See **Error Codes**.

5.4.2.4 Querying a DID

Function

This API is used to query the document of a specified DID.

URI

POST /v1/identity/query-did

Request Parameters

Table 5-131 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
did	Yes	String	Decentralized identifier.

Response Parameters

Status code: 200

Table 5-132 Response body parameters

Parameter	Type	Description
context	String	context
id	String	Decentralized identifier.
publicKey	Array of **DocPublicKey** objects	Public keys.
authentication	Array of strings	Identifier of the DID master public key.
recovery	String	Identifier of the DID backup public key, which can be used to modify the master key.
service	Array of **Service** objects	Service list.
proof	**Proof** object	Proof structure, which can be left empty.
created	String	Creation time.
updated	String	Update time.
status	String	Status.

Table 5-133 DocPublicKey

Parameter	Type	Description
id	String	Public key identifier.
type	String	Public key type.
controller	String	Public key controller identifier.
publicKeyPem	String	Public key certificate.

Table 5-134 Service

Parameter	Type	Description
id	String	Service identifier.
type	String	Service type.
serviceEndpoint	String	Service endpoint.

Parameter	Type	Description
credentialApplySchema	**CredentialApplySchema** object	Schema used for applying for a VC.

Table 5-135 CredentialApplySchema

Parameter	Type	Description
type	String	Type.
name	String	Name.
description	String	Description.
attributes	Array of **Attribute** objects	Attributes.

Table 5-136 Attribute

Parameter	Type	Description
name	String	Name.
type	String	Type.
description	String	Description.

Table 5-137 Proof

Parameter	Type	Description
creator	String	Creator identifier.
type	String	Signature type.
created	String	Signature creation time.
signatureValue	String	Signature.

Status code: 500

Table 5-138 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

DID document structure.

```
{
  "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"
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	DID document structure.
500	Error response.

Error Codes

See **Error Codes**.

5.4.3 Verifiable Credential (VC) Management

5.4.3.1 Publishing Credential Schemas

Function

This API is used to publish schemas for professional VCs. A schema enforces a specific structure on the credential data.

URI

POST /v1/identity/credential-schema

Request Parameters

Table 5-139 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.

Parameter	Mandatory	Type	Description
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
title	No	String	Name.
identifier	Yes	String	Identifier.
attributes	No	Array of **Attribute** objects	Attribute information.
issuer	Yes	String	Issuer identifier.

Table 5-140 Attribute

Parameter	Mandatory	Type	Description
name	No	String	Name.
type	No	String	Type.
description	No	String	Description.

Response Parameters

Status code: 200

Table 5-141 Response body parameters

Parameter	Type	Description
schemaIndex	String	Index of the schema stored on the blockchain.
credentialSchema	**CredentialSchema** object	CredentialSchema

Table 5-142 CredentialSchema

Parameter	Type	Description
creator	String	Creator identifier.

Parameter	Type	Description
title	String	Name.
identifier	String	Credential schema identifier.
attributes	Array of **Attribute** objects	Attribute information.
version	Integer	Version.

Table 5-143 Attribute

Parameter	Type	Description
name	String	Name.
type	String	Type.
description	String	Description.

Status code: 500

Table 5-144 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

VCSchemaResponeParams

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	VCSchemaResponeParams
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.2 Querying a Credential Schema

Function

This API is used to query a Credential schema by index.

URI

POST /v1/identity/query-credential-schema

Request Parameters

Table 5-145 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
schemaIndex	Yes	String	Schema index.

Response Parameters

Status code: 200

Table 5-146 Response body parameters

Parameter	Type	Description
creator	String	Creator identifier.
title	String	Name.
identifier	String	Credential schema identifier.
attributes	Array of **Attribute** objects	Attribute information.
version	Integer	Version.

Table 5-147 Attribute

Parameter	Type	Description
name	String	Name.
type	String	Type.

Parameter	Type	Description
description	String	Description.

Status code: 500

Table 5-148 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

CredentialSchema Information

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

Status code: 500

Error response.

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


Status Codes

Status Code	Description
200	CredentialSchema Information
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.3 Applying for Verifiable Credentials

Function

This API is used to apply for verifiable credentials. The applicant must provide relevant data according to the fields claimed in the service.

URI

POST /v1/identity/apply-vc

Request Parameters

Table 5-149 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
applyer	Yes	String	Applicant identifier.
serviceID	Yes	String	Service identifier. Service ID declared by the service provider in the DID.

Parameter	Mandatory	Type	Description
data	No	String	data

Response Parameters

Status code: 200

Table 5-150 Response body parameters

Parameter	Type	Description
orderIndex	String	Order index.

Status code: 500

Table 5-151 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

VCOrderResponeParams Information

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


Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	VCOrderResponeParams Information
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.4 Confirming Credential Issuance

Function

This API is used for the issuer to confirm that the credential has been issued and to update the credential index to the order.

URI

POST /v1/identity/vc-order

Request Parameters

Table 5-152 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.

Parameter	Mandatory	Type	Description
timestamp	Yes	String	Timestamp.
orderIndex	Yes	String	Order index.
vcIndex	Yes	String	VC index.

Response Parameters

Status code: 200

Table 5-153 Response body parameters

Parameter	Type	Description
orderIndex	String	Order index.
result	String	Result.

Status code: 500

Table 5-154 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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
}
```

Example Responses

Status code: 200

ConfirmOrderResponeParams Information

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	ConfirmOrderResponeParams Information
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.5 Querying Credential Applications

Function

This API is used to query VC application orders.

URI

POST /v1/identity/query-vc-order

Request Parameters

Table 5-155 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.

Parameter	Mandatory	Type	Description
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
orderIndex	Yes	String	Order index.

Response Parameters

Status code: 200

Table 5-156 Response body parameters

Parameter	Type	Description
applyer	String	Applicant identifier.
orderSeq	String	Order number.
applyTime	String	Application time.
price	String	Price.
status	String	Status.
service	String	Service.
reason	String	Reason.
dataUri	String	VC data URI.
encryptedAesKey	String	Encrypted AES key.
uriType	String	URI type.
dataHash	String	Data hash value.
lockProof	String	Lock proof.
vcIndex	String	VC index.

Status code: 500

Table 5-157 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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
}
```

Example Responses

Status code: 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"
}
```

Status code: 500

Error response.

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


Status Codes

Status Code	Description
200	VCOrder Information
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.6 Querying Open Application Orders

Function

This API is used to query open VC application orders based on the service identifier. Only the service provider has the permission to query orders.

URI

POST /v1/identity/query-vc-orders

Request Parameters

Table 5-158 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
serviceID	No	String	Service identifier.

Response Parameters

Status code: 200

Table 5-159 Response body parameters

Parameter	Type	Description
items	Array of **VCOrder** objects	List.

Table 5-160 VCOrder

Parameter	Type	Description
applyer	String	Applicant identifier.
orderSeq	String	Order number.
applyTime	String	Application time.
price	String	Price.
status	String	Status.
service	String	Service.
reason	String	Reason.
dataUri	String	VC data URI.
encryptedAesKey	String	Encrypted AES key.
uriType	String	URI type.
dataHash	String	Data hash value.
lockProof	String	Lock proof.
vcIndex	String	VC index.

Status code: 500

Table 5-161 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**

Parameter	Type	Description
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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
}
```

Example Responses

Status code: 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"
  } ]
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	UntreatedVCOrder Information List
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.7 Issuing Verifiable Credentials

Function

This API is used to issue credentials. The time of issuance is the current time by default.

URI

POST /v1/identity/issue-vc

Request Parameters

Table 5-162 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
credentialInfo	Yes	**CredentialSubjectInfo** object	Credential subject details.

Table 5-163 CredentialSubjectInfo

Parameter	Mandatory	Type	Description
applyer	Yes	String	Applicant identifier.
issuer	Yes	String	Issuer identifier.
sequence	Yes	String	Credential sequence number.
schemaIndex	Yes	String	Credential schema index.

Parameter	Mandatory	Type	Description
expirationDate	No	String	Expiration time.
data	Yes	String	Plaintext credential data.

Response Parameters

Status code: 200

Table 5-164 Response body parameters

Parameter	Type	Description
vcIndex	String	VC index.

Status code: 500

Table 5-165 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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}"
}
```

Example Responses

Status code: 200

VCResponeParams Information

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	VCResponeParams Information
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.8 This API is used to query a credential by index.

Function

This API is used to query a credential by index.

URI

POST /v1/identity/query-vc

Request Parameters

Table 5-166 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.

Parameter	Mandatory	Type	Description
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
vcIndex	Yes	String	VC index.

Response Parameters

Status code: 200

Table 5-167 Response body parameters

Parameter	Type	Description
context	String	Content.
sequence	String	Serial number of the credential of the issuer.
type	Array of strings	Verifiable credential type.
issuer	String	Issuer identifier.
issuanceDate	String	Issuance date.
expirationDate	String	Credential validity period.
credentialSubject	**CredentialSubject** object	Credential subject.
revocation	**Revocation** object	Revocation.

Table 5-168 CredentialSubject

Parameter	Type	Description
owner	String	Applicant identifier.
type	String	Credential type.
schemaID	String	schema ID
dataURI	String	Data URI.
encryptedAeskey	String	Encrypted symmetric key.

Parameter	Type	Description
uriType	String	Data index type.
dataHash	String	Data hash value.

Table 5-169 Revocation

Parameter	Type	Description
id	String	API for querying revocation or URL of the revocation list.
type	String	Revocation type.

Status code: 500

Table 5-170 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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
}
```

Example Responses

Status code: 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"
  }
}
```


Status code: 500

Error response.


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


Status Codes

Status Code	Description
200	VerifiableCredential Information
500	Error response.

Error Codes

See **Error Codes**.

5.4.3.9 Verifying a Credential

Function

This API is used to verify that a credential exists and is valid.

URI

POST /v1/identity/verify-vc

Request Parameters

Table 5-171 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.

Parameter	Mandatory	Type	Description
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	User certificate. Add an explicit line break (\n) at the end of each line.
sk	Yes	String	User private key. Add an explicit line break (\n) at the end of each line.
timestamp	Yes	String	Timestamp.
vcIndex	Yes	String	VC index.
owner	Yes	String	VC holder identifier.

Response Parameters

Status code: 200

Table 5-172 Response body parameters

Parameter	Type	Description
isOwned	Boolean	Whether the holder owns the VC.

Status code: 400

Table 5-173 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Status code: 500

Table 5-174 Response body parameters

Parameter	Type	Description
errorCode	String	Error code. Minimum: **8** Maximum: **36**
errorMsg	String	Error description. Minimum: **2** Maximum: **512**

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

VCVerifyResponeParams Information

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

Status code: 400

Error response.

```
{
  "errorCode" : "BCS.4002030",
  "errorMsg" : "Owner(bol) does not have credential ..."
}
```

Status code: 500

Error response.

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


Status Codes

Status Code	Description
200	VCVerifyResponeParams Information
400	Error response.
500	Error response.

Error Codes

See **Error Codes**.

5.5 Trusted Data Exchange (OBT)

5.5.1 Overview

Data is crucial to business. In distributed applications, data is exchanged to break data silos and maximize data value. Trusted data exchange based on the blockchain can ensure data privacy and trustworthy data sharing. BCS's trusted data exchange middleware is integrated with the RESTful APIs add-on, which can be quickly installed and uninstalled, and supports elastic scaling. Users can access the blockchain system through RESTful APIs to quickly integrate data release, authorization, sharing, encryption, decryption, and fine-grained access control capabilities.

Function

- Trusted data exchange is implemented based on DID. Users who participate in data exchange must have DIDs. For details, see **Distributed Identity (OBT)**.
- Trusted data exchange involves two main data structures: data sets and data orders. Data sets contain data description and access control information (attributed-based encryption policies). Data orders contain data application and review information.
- Trusted data exchange supports three modes: application-authorization, proactive sharing, and fine-grained access control, as described in **Exchange Modes**.

  NOTE

  You can choose from multiple storage services to store the encrypted data to be exchanged. The one calling the API is responsible for storing ciphertexts to publicly accessible storage devices.

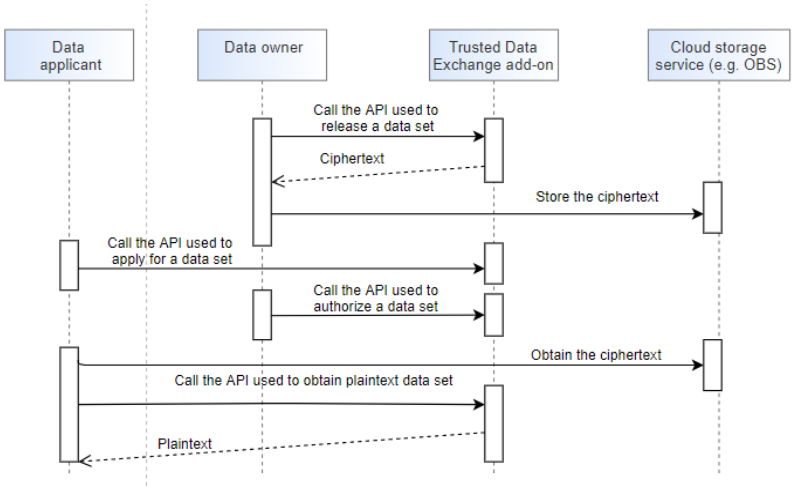
Roles

Trusted data exchange involves two roles: data owner and data applicant. Each user can be both an owner and an applicant.

Exchange Modes

- Application-authorization, which is illustrated in **Figure 5-5**.
  - The data owner calls the API used to release a data set to encrypt plaintext user data and register and release data description information.
  - The data applicant calls the API used to apply for a data set to invoke a chaincode to trigger the application-authorization process.
  - The data owner decides whether to authorize or reject the application based on the application information and the applicant's DID and VC information.

  Figure 5-5 Application-authorization

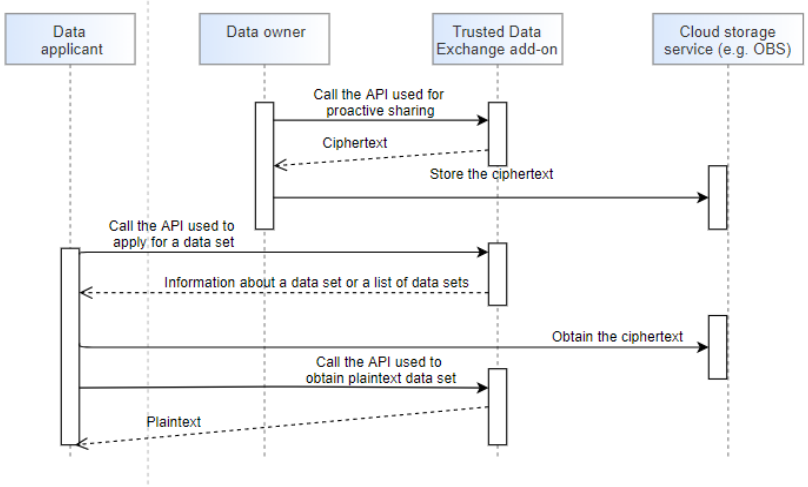


- Proactive sharing, which is illustrated in **Figure 5-6**.

  The API used to proactively sharing data sets is a combination of the APIs used to release and authorize data sets. The data owner releases a dataset to the blockchain and authorizes an applicant to decrypt the dataset. The authorized applicant can then directly decrypt the dataset. Other users can obtain the data description information by calling the APIs used to query a specific dataset and the dataset list, and then obtain the data decryption permission through the application-authorization process.

  For details about the APIs, see "Data Set Management" and "Data Order Management".

Figure 5-6 Proactive sharing



- Fine-grained access control, which is illustrated in **Figure 5-7**.

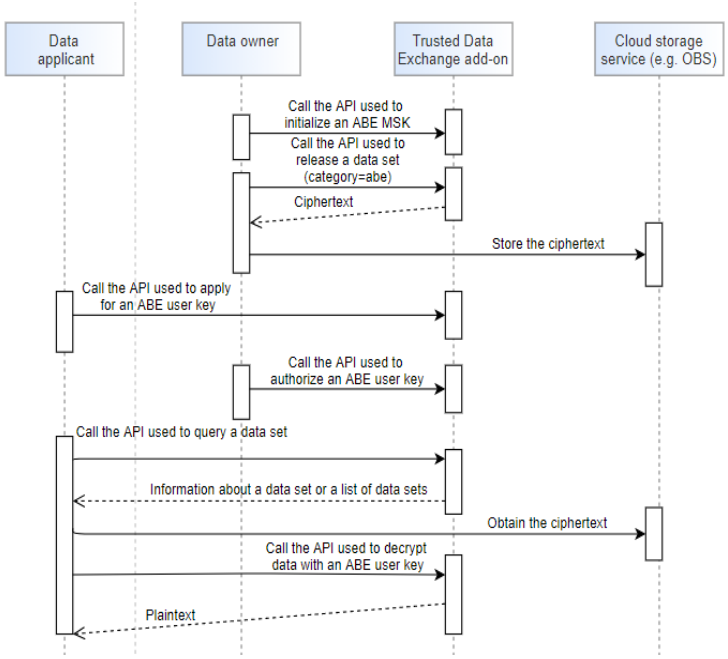
  NOTE

  Attribute-based encryption (ABE) achieves fine-grained, attribute-level access control for data exchange. Each data set is configured with an appropriate, owner-defined sharing policy. Data applicants with sufficient attributes are allowed to access the ciphertext.

  Fine-grained access control is implemented through ciphertext-policy ABE (CP-ABE). The policy is embedded in the ciphertext and the attribute is embedded in the user key.

  - Each data owner initializes its own master public key and private key only once.
  - When a data applicant needs to use some data, the applicant applies to the owner of the data for a user key. If attributes do not change, the applicant only applies for a user key once.
  - With a user key and sufficient attributes, the data applicant can asynchronously decrypt all data released by the data owner at any time as required.
  - For details about the APIs, see "Attribute-based Encryption Key Management".

Figure 5-7 Fine-grained access control



NOTE

Basic concepts:

- Attribute: An attribute describes the association between an entity and its nature.
- Policy: A policy is a combination of attribute sets and logical relationships. A policy is defined by the data owner and embedded in the ciphertext. For example, the policy "age>26 && gender=man" indicates that the age must be greater than 26 and the gender must be man.
- ABE master key: An ABE master key consists of a master public key (MPK) and a master secret key (MSK). They are owned by the data owner and are used to encrypt data and generate a user (data applicant) key.
- User key: A data applicant applies to data owners for a user key after submitting a set of attributes to the data user. A user key contains the user attribute information and is used to decrypt a ciphertext.

5.5.2 Data Set Management

5.5.2.1 Publishing a Data Set

Function

This API is used to publish a data set. Watermarking is not supported.

URI

POST /v1/datashare/dataset

Request Parameters

Table 5-175 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
provider	Yes	String	Data publisher identifier.
providerName	No	String	Data publisher name.
productName	Yes	String	Product name.
productID	Yes	String	Product ID.
sampleUrl	Yes	String	Sample data storage location.
sampleSize	No	String	Sample data size.
sampleType	No	String	Sample data type.
sampleName	No	String	Sample data name.
fileType	No	String	Data file type.
dataUrl	Yes	String	Data storage location.
dataHash	No	String	Data hash value.
dataSize	No	String	Data size.
dataName	No	String	Data name.
description	No	String	Data set description. When using ABE for encryption, describe the encryption policy in detail.
plainData	Yes	String	Plaintext of the Base64-encoded data.
category	No	String	Encryption type. The options are **abe** and **symmetric**. The default value is **symmetric**. If **abe** is used, **policy** is mandatory.

Parameter	Mandatory	Type	Description
watermarkType	No	String	Watermark type, which can be **visible** or **blind**. Specify this parameter if you want to embed a watermark. The embedded watermark is in the format of **Publisher DID_Product ID**.
file	No	File	File to embed a watermark in. If the file has a watermark, the **plainData** parameter does not take effect.
policy	No	**policy** object	ABE policy.

Table 5-176 policy

Parameter	Mandatory	Type	Description
Threshold	Yes	Integer	Attribute threshold that a policy must meet.
Children	Yes	Array of **policy-children** objects	Sub-attribute list.

Table 5-177 policy-children

Parameter	Mandatory	Type	Description
name	Yes	String	Attribute name.
type	Yes	String	Attribute type (**plain**, **comparable**, or **policy**.)
value	Yes	**policy-children-comparablevalue** object	Attribute value, which is determined based on the attribute type. When the attribute type is **plain**, the value is a string. When the attribute type is **policy**, the value is a sub-policy. When the attribute type is **comparable**, the value contains **op**, **value**, and **maxValue**.

Table 5-178 policy-children-comparablevalue

Parameter	Mandatory	Type	Description
op	No	String	Comparison operator (>, <, or =). This parameter is mandatory when **type** is set to **comparable**.
value	Yes	String	Attribute value of the comparison type. The value must be an integer.
maxValue	No	String	Upper limit of value. This parameter is mandatory when the attribute type is comparable.

Response Parameters

Status code: 200

Table 5-179 Response body parameters

Parameter	Type	Description
provider	String	Data set provider identifier.
providerName	String	Data set provider name.
productName	String	Data set product name.
productID	String	Data set product ID.
sampleUrl	String	Sample data URL.
sampleSize	String	Sample data size.
sampleType	String	Sample data type.
sampleName	String	Sample data name.
fileType	String	File type.
dataUrl	String	Data URL.
dataHash	String	Data hash value.
dataSize	String	Data size.
dataName	String	Data name.
description	String	Data description.
price	String	Data price.

Parameter	Type	Description
encryptedAesKey	String	Private key.
status	String	Status.
publishTime	String	Data publishing time.
dataFiles	Array of **DataFile** objects	Data file list.
sampleFiles	Array of **DataFile** objects	Sample file list.
category	String	Encryption type.
encryptData	String	Encrypted data.

Table 5-180 DataFile

Parameter	Type	Description
fileType	String	File type.
dataUrl	String	Data URL.
dataHash	String	Data hash value.
dataSize	String	Data size.
dataName	String	Data name.

Status code: 500

Table 5-181 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

An ABE example policy. In this example policy, decryption can be performed only when the decrypting party matches the "att1==hello" attribute or conforms to a sub-policy, where the value of attribute **att3** is larger than 16.

```
{
  "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" : "huawei",
 "productName" : "prodname",
 "productID" : "product2",
 "sampleUrl" : "http://hwcloud.com/sample.com/prodname2",
 "sampleSize" : "10KB",
 "sampleType" : "csv",
 "sampleName" : "data_sub1",
 "fileType" : "csv",
 "dataUrl" : "http://hwcloud.com/prodname2",
 "dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
 "dataSize" : "100MB",
 "dataName" : "mydata1",
 "description" : "this is my second prod",
 "plainData" : "base64 encoding string",
 "category" : "abe",
 "watermarkType" : "string",
 "file" : "string",
 "policy" : "{\"threshold\" : 1,\"children\" : [{\"name\" : \"att1\", \"type\" : \"plain\",\"value\" : \"hello\"},
{\"name\" : \"policy\", \"type\" : \"policy\",\"value\" : {\"Threshold\" : 1,\"Children\" : [{\"name\" : \"att3\",
\"type\" :\"comparable\", \"value\" : {\"op\" : \">\", \"value\" : \"16\", \"maxValue\" : \"10000\"}}]}}]}"
}
```

Example Responses

Status code: 200

Data set information.

```
{
 "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"
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Data set information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.2 Deleting a Data Set

Function

Deleting a Data Set

URI

DELETE /v1/datashare/dataset

Request Parameters

Table 5-182 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.

Parameter	Mandatory	Type	Description
provider	Yes	String	Data set provider identifier.
productID	Yes	String	Data product ID.

Response Parameters

Status code: 200

Table 5-183 Response body parameters

Parameter	Type	Description
result	String	Operation result.

Status code: 500

Table 5-184 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Operation result.

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

Status code: 500

Error response.

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



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

Status Codes

Status Code	Description
200	Operation result.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.3 Closing a Data Set

Function

Closing a Data Set

URI

PUT /v1/datashare/dataset

Request Parameters

Table 5-185 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
provider	Yes	String	Data set provider identifier.
productID	Yes	String	Data product ID.

Response Parameters

Status code: 200

Table 5-186 Response body parameters

Parameter	Type	Description
result	String	Operation result.

Status code: 500

Table 5-187 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Operation result.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Operation result.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.4 Querying a Data Set

Function

This API is used to query a data set based on the publisher identifier and product ID.

URI

POST /v1/datashare/query-dataset

Request Parameters

Table 5-188 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
provider	Yes	String	Data set publisher identifier.
productID	Yes	String	Data set product ID.

Response Parameters

Status code: 200

Table 5-189 Response body parameters

Parameter	Type	Description
provider	String	Data set provider identifier.
providerName	String	Data set provider name.
productName	String	Data set product name.
productID	String	Data set product ID.

Parameter	Type	Description
sampleUrl	String	Sample data URL.
sampleSize	String	Sample data size.
sampleType	String	Sample data type.
sampleName	String	Sample data name.
fileType	String	File type.
dataUrl	String	Data URL.
dataHash	String	Data hash value.
dataSize	String	Data size.
dataName	String	Data name.
description	String	Data description.
price	String	Data price.
encryptedAesKey	String	Private key.
status	String	Status.
publishTime	String	Data publishing time.
dataFiles	Array of **DataFile** objects	Data file list.
sampleFiles	Array of **DataFile** objects	Sample file list.
category	String	Encryption type.

Table 5-190 DataFile

Parameter	Type	Description
fileType	String	File type.
dataUrl	String	Data URL.
dataHash	String	Data hash value.
dataSize	String	Data size.
dataName	String	Data name.

Status code: 500

Table 5-191 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Operation result.

```
{
  "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"
}
```


Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Operation result.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.5 Listing Data Sets

Function

This API is used to query data sets by pages and specific conditions.

URI

POST /v1/datashare/query-datasets

Request Parameters

Table 5-192 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
currentPage	No	String	Pagination parameter: page number (default value: **1**).

Parameter	Mandatory	Type	Description
pageSizeNum	No	String	Pagination parameter: number of records on each page (default value: **100**).
provider	No	String	Filtering condition: publisher identifier.
searchText	No	String	Filtering condition: keywords (names and descriptions of products in the data set).
status	No	String	Filtering condition: data set status (**ready** or **closed**).

Response Parameters

Status code: 200

Table 5-193 Response body parameters

Parameter	Type	Description
items	Array of **DatasetResponse** objects	List.
pagination	**PaginationResp** object	Pagination information.

Table 5-194 DatasetResponse

Parameter	Type	Description
provider	String	Data set provider identifier.
providerName	String	Data set provider name.
productName	String	Data set product name.
productID	String	Data set product ID.
sampleUrl	String	Sample data URL.
sampleSize	String	Sample data size.
sampleType	String	Sample data type.
sampleName	String	Sample data name.
fileType	String	File type.

Parameter	Type	Description
dataUrl	String	Data URL.
dataHash	String	Data hash value.
dataSize	String	Data size.
dataName	String	Data name.
description	String	Data description.
price	String	Data price.
encryptedAesKey	String	Private key.
status	String	Status.
publishTime	String	Data publishing time.
dataFiles	Array of **DataFile** objects	Data file list.
sampleFiles	Array of **DataFile** objects	Sample file list.
category	String	Encryption type.

Table 5-195 DataFile

Parameter	Type	Description
fileType	String	File type.
dataUrl	String	Data URL.
dataHash	String	Data hash value.
dataSize	String	Data size.
dataName	String	Data name.

Table 5-196 PaginationResp

Parameter	Type	Description
currentPage	Integer	Current page.
pageSizeNum	Integer	Number of records on each page.
totalItems	Integer	Total number of records.

Status code: 500

Table 5-197 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Data set pagination information.

```
{
  "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
 }
}
```


Status code: 500

Error response.


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


Status Codes

Status Code	Description
200	Data set pagination information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.6 Sharing a Data Set

Function

This API is used to publish a data set and create authorized orders. Watermarking is not supported.

URI

POST /v1/datashare/dataset/share

Request Parameters

Table 5-198 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.

Parameter	Mandatory	Type	Description
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
provider	Yes	String	Data publisher identifier.
providerName	No	String	Data publisher name.
productName	Yes	String	Product name.
productID	Yes	String	Product ID.
sampleUrl	Yes	String	Sample data storage location.
sampleSize	No	String	Sample data size.
sampleType	No	String	Sample data type.
sampleName	No	String	Sample data name.
fileType	No	String	Data file type.
dataUrl	Yes	String	Data storage location.
dataHash	No	String	Data hash value.
dataSize	No	String	Data size.
dataName	No	String	Data name.
description	No	String	Data set description. When using ABE for encryption, describe the encryption policy in detail.
plainData	Yes	String	Plaintext of the Base64-encoded data.
consumer	Yes	String	Order applicant identifier.
orderSeq	No	String	Order number. If this parameter is left empty, the value is generated randomly.
watermarkType	No	String	Watermark type, which can be **visible** or **blind**. Specify this parameter if you want to embed a watermark. The embedded watermark is in the format of **Publisher DID_Product ID**.

Parameter	Mandatory	Type	Description
file	No	String	File to embed a watermark in. If the file has a watermark, the **plainData** parameter does not take effect.
productIDKeywords	No	Array of **productIDKeywordsJson** objects	Index keyword contained in the product ID, which is in JSON and is used for querying orders by condition.
onChainStore	No	String	Whether to store ciphertext data on the chain. The value can be **true** or **false** (default). If this parameter is set to **true**, **sampleUrl** and **dataUrl** will be automatically set. Watermarking is not supported when chain storage is used.
consumerName	No	String	Name of the user.
creatorDID	No	String	DID of the creator of the data set sharing process. This parameter does not need to be set when **action** is set to **new**.
processID	No	String	ID of the data set sharing process. This parameter does not need to be set when **action** is set to **new**.
stageName	No	String	Name of the data set sharing stage.
action	No	String	Process action. The options are **new**, **append**, and "" (N/A). If this parameter is not set, the value is regarded as N/A.
category	No	String	Encryption type, which can be **symmetric** (default) or **abe**. If **abe** is used, specify **policy**.
policy	No	**policy** object	ABE policy.

Table 5-199 productIDKeywordsJson

Parameter	Mandatory	Type	Description
value	No	String	Keyword value.

Table 5-200 policy

Parameter	Mandatory	Type	Description
Threshold	Yes	Integer	Attribute threshold that a policy must meet.
Children	Yes	Array of **policy-children** objects	Sub-attribute list.

Table 5-201 policy-children

Parameter	Mandatory	Type	Description
name	Yes	String	Attribute name.
type	Yes	String	Attribute type (**plain**, **comparable**, or **policy**.)
value	Yes	**policy-children-comparablevalue** object	Attribute value, which is determined based on the attribute type. When the attribute type is **plain**, the value is a string. When the attribute type is **policy**, the value is a sub-policy. When the attribute type is **comparable**, the value contains **op**, **value**, and **maxValue**.

Table 5-202 policy-children-comparablevalue

Parameter	Mandatory	Type	Description
op	No	String	Comparison operator (>, <, or =). This parameter is mandatory when **type** is set to **comparable**.

Parameter	Mandatory	Type	Description
value	Yes	String	Attribute value of the comparison type. The value must be an integer.
maxValue	No	String	Upper limit of value. This parameter is mandatory when the attribute type is comparable.

Response Parameters

Status code: 200

Table 5-203 Response body parameters

Parameter	Type	Description
consumer	String	Order applicant identifier.
consumerName	String	Order applicant name.
orderSeq	String	Order number.
provider	String	Order provider identifier.
providerName	String	Order provider name.
productID	String	Data set product ID.
productName	String	Data set product name.
price	String	Order amount.
applyTime	String	Application time.
encryptedAesKey	String	Private key.
status	String	Order status.
reason	String	Order application reason.
lockProof	String	Order lock-up proof.
creatorDID	String	DID of the process creator. If there is no process, the value is "".
processID	String	ID of the process of the current order. If there is no process, the value is "".
encryptData	String	Ciphertext of the Base64-encoded data.

Status code: 500

Table 5-204 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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" : "huawei",
  "productName" : "prodname",
  "productID" : "product2",
  "sampleUrl" : "http://hwcloud.com/sample.com/prodname2",
  "sampleSize" : "10KB",
  "sampleType" : "csv",
  "sampleName" : "data_sub1",
  "fileType" : "csv",
  "dataUrl" : "http://hwcloud.com/prodname2",
  "dataHash" : "2282ba7a1a2ef5700609214a997d3d4237a03bfd3632c6d089e57e7b6f467969",
  "dataSize" : "100MB",
  "dataName" : "mydata1",
  "description" : "this is my second prod",
  "plainData" : "base64 encoding string",
  "consumer" : "did:example:3TMWx8owKHARgNwbj4ywmG",
  "orderSeq" : "1",
  "watermarkType" : "string",
  "file" : "string",
  "productIDKeywords" : "[{\"value\":\"taiyuan\"},{\"value\":\"renmin_hospital\"},{\"value\":\"medicine\"}]",
  "onChainStore" : "string",
  "consumerName" : "string"
}
```

Example Responses

Status code: 200

Order information.

```
{
  "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"
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.7 Obtaining Plaintext Data

Function

This API is used to obtain the decrypted plaintext. Watermarking is not supported.

URI

POST /v1/datashare/dataset/query-plaintext

Request Parameters

Table 5-205 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.

Parameter	Mandatory	Type	Description
consumer	Yes	String	Order applicant identifier.
orderSeq	Yes	String	Order number.
encryptData	No	String	Ciphertext data. This parameter is optional when **onChainStore** is set to **true**.
watermarkType	No	String	Watermark type, which can be **visible** or **blind**. Specify this parameter if you want to embed a watermark. If a blind watermark has already been embedded into the data set to be published or shared, no more blind watermarks can be embedded into the data set. The embedded watermark is in the format of **User DID_Order ID**.
onChainStore	No	String	Whether to store ciphertext data on the chain. The value can be **true** or **false** (default). If you set this parameter to **true**, you do not need to specify **encryptData**. The ciphertext data can be automatically obtained from the chain.

Response Parameters

Status code: 200

Table 5-206 Response body parameters

Parameter	Type	Description
provider	String	Order provider identifier.
productID	String	Data set product ID.
plaintext	String	Plaintext of the Base64-encoded data.

Status code: 500

Table 5-207 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Order information.

```
{
  "provider" : "did:example:DHkJjyD5wZwya6sd6BNBnG",
  "productID" : "product1",
  "plaintext" : "base64 encoding string"
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.8 Extracting a Blind Watermark from a File

Function

This API is used to extract a blind watermark from a file. Watermarking is not supported.

URI

POST /v1/datashare/dataset/watermark/extract

Request Parameters

Table 5-208 FormData parameters

Parameter	Mandatory	Type	Description
file	Yes	File	File from which a blind watermark is to be extracted.

Response Parameters

Status code: 200

Table 5-209 Response body parameters

Parameter	Type	Description
watermark	String	Blind watermark in the file. If no blind watermark is embedded, no content is displayed.

Status code: 500

Table 5-210 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

None

Example Responses

Status code: 200

Content returned for extracting a blind watermark.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Content returned for extracting a blind watermark.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.9 Querying a Data Set Sharing Process

Function

This API is used to query a data set sharing process.

URI

POST /v1/datashare/dataset/query-process

Request Parameters

Table 5-211 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.

Parameter	Mandatory	Type	Description
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
creatorDID	Yes	String	Process creator identifier.
processID	Yes	String	Process ID.

Response Parameters

Status code: 200

Table 5-212 Response body parameters

Parameter	Type	Description
creatorDID	String	Process creator identifier.
processID	String	Process ID.
stages	Array of **StageInProcess** objects	Information about a stage in the process.

Table 5-213 StageInProcess

Parameter	Type	Description
stageName	String	Stage name.
createTime	String	Timestamp when the stage information is stored to the blockchain.
consumer	String	Consumer identifier.
orderSeq	String	Order number.

Status code: 500

Table 5-214 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
/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"
}
```

Example Responses

Status code: 200

Parameters in the response to the request for querying a specific data set sharing process.

```
{
  "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"
  } ]
}
```

Status Codes

Status Code	Description
200	Parameters in the response to the request for querying a specific data set sharing process.
500	Error response.

Error Codes

See **Error Codes**.

5.5.2.10 Querying All Processes of a Process Creator

Function

This API is used to query all processes of a process creator.

URI

POST /v1/datashare/dataset/query-processes

Request Parameters

Table 5-215 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
currentPage	No	String	Pagination parameter: page number (default value: **1**).
pageSizeNum	No	String	Pagination parameter: number of records on each page (default value: **100**).
creatorDID	Yes	String	Process creator identifier.

Response Parameters

Status code: 200

Table 5-216 Response body parameters

Parameter	Type	Description
items	Array of **DatasetShareProcessResponseBody** objects	List of data set sharing processes.

Parameter	Type	Description
pagination	**PaginationResp** object	Pagination information.

Table 5-217 DatasetShareProcessResponseBody

Parameter	Type	Description
creatorDID	String	Process creator identifier.
processID	String	Process ID.
stages	Array of **StageInProcess** objects	Information about a stage in the process.

Table 5-218 StageInProcess

Parameter	Type	Description
stageName	String	Stage name.
createTime	String	Timestamp when the stage information is stored to the blockchain.
consumer	String	Consumer identifier.
orderSeq	String	Order number.

Table 5-219 PaginationResp

Parameter	Type	Description
currentPage	Integer	Current page.
pageSizeNum	Integer	Number of records on each page.
totalItems	Integer	Total number of records.

Status code: 500

Table 5-220 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.

Parameter	Type	Description
errorMsg	String	Error description.

Example Requests

```
/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"
}
```

Example Responses

Status code: 200

Parameters in the response to the request for querying all processes of a process creator.

```
{
  "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
  }
}
```

Status Codes

Status Code	Description
200	Parameters in the response to the request for querying all processes of a process creator.
500	Error response.

Error Codes

See **Error Codes**.

5.5.3 Data Order Management

5.5.3.1 Applying for a Data Set

Function

Applying for a Data Set

URI

POST /v1/datashare/dataset/dataset-order

Request Parameters

Table 5-221 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.

Parameter	Mandatory	Type	Description
consumer	Yes	String	Order applicant identifier.
orderSeq	Yes	String	Order number.
provider	Yes	String	Data set publisher identifier.
productID	Yes	String	Data set product ID.
reason	No	String	Reason.
consumerName	No	String	Data set applicant name.

Response Parameters

Status code: 200

Table 5-222 Response body parameters

Parameter	Type	Description
consumer	String	Order applicant identifier.
consumerName	String	Order applicant name.
orderSeq	String	Order number.
provider	String	Order provider identifier.
providerName	String	Order provider name.
productID	String	Data set product ID.
productName	String	Data set product name.
price	String	Order amount.
applyTime	String	Application time.
encryptedAesKey	String	Private key.
status	String	Order status.
reason	String	Order application reason.
lockProof	String	Order lock-up proof.
creatorDID	String	DID of the process creator. If there is no process, the value is "".
processID	String	ID of the process of the current order. If there is no process, the value is "".

Status code: 500

Table 5-223 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Order information.

```
{
  "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" : ""
}
```

Status code: 500

Error response.

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


Status Codes

Status Code	Description
200	Order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.3.2 Authorizing a Data Set

Function

Authorizing a Data Set

URI

POST /v1/datashare/dataset/authorize-dataset

Request Parameters

Table 5-224 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
consumer	Yes	String	Order applicant identifier.
orderSeq	Yes	String	Order number.

Response Parameters

Status code: 200

Table 5-225 Response body parameters

Parameter	Type	Description
consumer	String	Order applicant identifier.
consumerName	String	Order applicant name.
orderSeq	String	Order number.
provider	String	Order provider identifier.
providerName	String	Order provider name.
productID	String	Data set product ID.
productName	String	Data set product name.
price	String	Order amount.
applyTime	String	Application time.
encryptedAesKey	String	Private key.
status	String	Order status.
reason	String	Order application reason.
lockProof	String	Order lock-up proof.
creatorDID	String	DID of the process creator. If there is no process, the value is "".
processID	String	ID of the process of the current order. If there is no process, the value is "".

Status code: 500

Table 5-226 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Order information.

```
{
  "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" : ""
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.3.3 Updating Order Status

Function

Updating Order Status

URI

PUT /v1/datashare/dataset/order

Request Parameters

Table 5-227 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
consumer	Yes	String	Order applicant identifier.
orderSeq	Yes	String	Order number.
orderStatus	Yes	Integer	Order status. **0**: completed; **1**: failed; **2**: canceled.
reason	No	String	Reason.

Response Parameters

Status code: 200

Table 5-228 Response body parameters

Parameter	Type	Description
result	String	Operation result.

Status code: 500

Table 5-229 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Operation result.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Operation result.
500	Error response.

Error Codes

See **Error Codes**.

5.5.3.4 Deleting an Order

Function

Deleting an Order

URI

DELETE /v1/datashare/dataset/order

Request Parameters

Table 5-230 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
consumer	Yes	String	Order applicant identifier.
orderSeq	Yes	String	Order number.

Response Parameters

Status code: 200

Table 5-231 Response body parameters

Parameter	Type	Description
result	String	Operation result.

Status code: 500

Table 5-232 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Operation result.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Operation result.
500	Error response.

Error Codes

See **Error Codes**.

5.5.3.5 Querying an Order

Function

This API is used to query an order based on the specified applicant DID and order number.

URI

POST /v1/datashare/dataset/query-order

Request Parameters

Table 5-233 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.

Parameter	Mandatory	Type	Description
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
consumer	Yes	String	Order applicant identifier.
orderSeq	Yes	String	Order number.

Response Parameters

Status code: 200

Table 5-234 Response body parameters

Parameter	Type	Description
consumer	String	Order applicant identifier.
consumerName	String	Order applicant name.
orderSeq	String	Order number.
provider	String	Order provider identifier.
providerName	String	Order provider name.
productID	String	Data set product ID.
productName	String	Data set product name.
price	String	Order amount.
applyTime	String	Application time.
encryptedAesKey	String	Private key.
status	String	Order status.
reason	String	Order application reason.
lockProof	String	Order lock-up proof.
creatorDID	String	DID of the process creator. If there is no process, the value is "".

Parameter	Type	Description
processID	String	ID of the process of the current order. If there is no process, the value is "".

Status code: 500

Table 5-235 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Order information.

```
{
  "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" : ""
}
```

Status code: 500

Error response.

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



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

Status Codes

Status Code	Description
200	Order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.3.6 Listing Orders

Function

This API is used to query orders by pages and specific conditions.

URI

POST /v1/datashare/dataset/query-orders

Request Parameters

Table 5-236 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
currentPage	No	String	Pagination parameter: page number (default value: **1**).
pageSizeNum	No	String	Pagination parameter: number of records on each page (default value: **100**).

Parameter	Mandatory	Type	Description
provider	No	String	Filtering condition: data set publisher identifier.
searchText	No	String	Filtering condition: keywords (names of products in the data set).
status	No	String	Filtering condition: order status (**ready**, **finished**, **failed**, or **canceled**.)
consumer	No	String	Filtering condition: order applicant identifier.

Response Parameters

Status code: 200

Table 5-237 Response body parameters

Parameter	Type	Description
items	Array of **DataOrderResponse** objects	List.
pagination	**PaginationResp** object	Pagination information.

Table 5-238 DataOrderResponse

Parameter	Type	Description
consumer	String	Order applicant identifier.
consumerName	String	Order applicant name.
orderSeq	String	Order number.
provider	String	Order provider identifier.
providerName	String	Order provider name.
productID	String	Data set product ID.
productName	String	Data set product name.
price	String	Order amount.

Parameter	Type	Description
applyTime	String	Application time.
encryptedAesKey	String	Private key.
status	String	Order status.
reason	String	Order application reason.
lockProof	String	Order lock-up proof.
creatorDID	String	DID of the process creator. If there is no process, the value is "".
processID	String	ID of the process of the current order. If there is no process, the value is "".

Table 5-239 PaginationResp

Parameter	Type	Description
currentPage	Integer	Current page.
pageSizeNum	Integer	Number of records on each page.
totalItems	Integer	Total number of records.

Status code: 500

Table 5-240 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Order pagination information.

```
{
  "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
  }
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Order pagination information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.4 Attribute-based Encryption Key Management

5.5.4.1 Initializing an ABE Master Key

Function

This API is used to initialize an ABE master key. If the owner has never initialized the ABE master key, the ABE master key is automatically generated and stored on the chain. If the owner already has an ABE master key, it will not be overwritten.

URI

POST /v1/datashare/abe-setup

Request Parameters

Table 5-241 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
owner	Yes	String	Key generator identifier.
keyManagerMode	No	String	When using ABE for the first time, select the **central** or **distributed** (default) mode. The mode cannot be changed once configured.

Response Parameters

Status code: 200

Table 5-242 Response body parameters

Parameter	Type	Description
secretJson	String	ABE master private key.
publicKeyJson	String	ABE master public key.

Status code: 500

Table 5-243 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

ABE master key information.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	ABE master key information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.4.2 Updating an ABE Master Key

Function

Updating an ABE Master Key

URI

POST /v1/datashare/abe-update

Request Parameters

Table 5-244 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
secretKeyJson	No	String	ABE master private key in JSON format.
publicKeyJson	No	String	ABE master public key in JSON format.
owner	Yes	String	Key generator identifier.

Response Parameters

Status code: 200

Table 5-245 Response body parameters

Parameter	Type	Description
oldSecretJson	String	Old ABE master private key.
oldPublicKeyJson	String	Old ABE master public key.
newSecretJson	String	New ABE master private key.

Parameter	Type	Description
newPublicKeyJson	String	New ABE master public key.

Status code: 500

Table 5-246 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

Original and updated ABE master key information.

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

Status code: 500

Error response.

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


Status Codes

Status Code	Description
200	Original and updated ABE master key information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.4.3 Querying an ABE Master Key

Function

Querying an ABE Master Key

URI

POST /v1/datashare/abekey

Request Parameters

Table 5-247 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
owner	Yes	String	Key generator identifier.
keyManagerMode	No	String	When using ABE for the first time, select the **central** or **distributed** (default) mode. The mode cannot be changed once configured.

Response Parameters

Status code: 200

Table 5-248 Response body parameters

Parameter	Type	Description
secretKeyJson	String	ABE master private key in JSON format.
publicKeyJson	String	ABE master public key in JSON format.

Status code: 500

Table 5-249 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```

Example Responses

Status code: 200

ABE master key information.

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

Status code: 500

Error response.

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


Status Codes

Status Code	Description
200	ABE master key information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.4.4 Applying for an ABE User Key

Function

Applying for an ABE User Key

URI

POST /v1/datashare/abekey-order

Request Parameters

Table 5-250 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
applyer	Yes	String	Applicant identifier.
provider	Yes	String	Authorizer identifier.
attrJson	Yes	Array of **attribute** objects	Attributes.

Table 5-251 attribute

Parameter	Mandatory	Type	Description
name	Yes	String	Attribute name.
type	Yes	String	Attribute type (plain or comparable).
value	Yes	String	Attribute value. When the attribute type is plain, the value is the attribute value. When the attribute type is comparable, the value must be an integer.
maxValue	No	String	Upper limit of the attribute value. This parameter can be used only when the attribute type is comparable.

Response Parameters

Status code: 200

Table 5-252 Response body parameters

Parameter	Type	Description
applyer	String	Applicant identifier.
applyerName	String	Applicant name.
provider	String	Authorizer identifier.
providerName	String	Authorizer name.
service	String	Service name of the authorizer.
price	Integer	Price.
applyTime	String	Application time.
encryptedABEKey	String	Encrypted ABE key.
status	String	Application status. The value **request** indicates that the application is not authorized, and the value **ready** indicates that the application has been processed.
reason	String	Reason.
lockProof	String	Proof.
attributesJson	String	Attribute.

Status code: 500

Table 5-253 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
Example: { "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\"}]"}
```

Example Responses

Status code: 200

ABE user key order information.

```
{
  "applyer" : "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "provider" : "did:example:Mb4SshJeN5ukWXkbMJK8xC",
  "applyTime" : "1622166512",
  "status" : "ready",
  "attributesJson" : "{\"att1\": \"YXR0MW5hbWU=\",\"att2\": \"YXR0Mm5hbWU=\",\"att3\": \"NQ==\"}"
}
```

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	ABE user key order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.4.5 Authorizing an ABE User Key

Function

Authorizing an ABE User Key

URI

PUT /v1/datashare/abekey-order

Request Parameters

Table 5-254 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
applyer	Yes	String	Applicant identifier.
provider	Yes	String	Authorizer identifier.
attrJson	No	Array of **attribute** objects	Attributes.

Table 5-255 attribute

Parameter	Mandatory	Type	Description
name	Yes	String	Attribute name.
type	Yes	String	Attribute type (plain or comparable).
value	Yes	String	Attribute value. When the attribute type is plain, the value is the attribute value. When the attribute type is comparable, the value must be an integer.

Parameter	Mandatory	Type	Description
maxValue	No	String	Upper limit of the attribute value. This parameter can be used only when the attribute type is comparable.

Response Parameters

Status code: 200

Table 5-256 Response body parameters

Parameter	Type	Description
applyer	String	Applicant identifier.
applyerName	String	Applicant name.
provider	String	Authorizer identifier.
providerName	String	Authorizer name.
service	String	Service name of the authorizer.
price	Integer	Price.
applyTime	String	Application time.
encryptedABEKey	String	Encrypted ABE key.
status	String	Application status. The value **request** indicates that the application is not authorized, and the value **ready** indicates that the application has been processed.
reason	String	Reason.
lockProof	String	Proof.
attributesJson	String	Attribute.

Status code: 500

Table 5-257 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests


```
{
  "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\"}]"
}
```


Example Responses

Status code: 200

ABE user key order information.


```
{
  "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\"}]"
}
```


Status code: 500

Error response.


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


Status Codes

Status Code	Description
200	ABE user key order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.4.6 Querying an ABE User Key Application

Function

Querying an ABE User Key Application

URI

POST /v1/datashare/query-abekey

Request Parameters

Table 5-258 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
provider	Yes	String	Authorizer identifier.
applyer	Yes	String	Applicant identifier.

Response Parameters

Status code: 200

Table 5-259 Response body parameters

Parameter	Type	Description
applyer	String	Applicant identifier.
applyerName	String	Applicant name.
provider	String	Authorizer identifier.
providerName	String	Authorizer name.
service	String	Service name of the authorizer.
price	Integer	Price.
applyTime	String	Application time.

Parameter	Type	Description
encryptedABEKey	String	Encrypted ABE key.
status	String	Application status. The value **request** indicates that the application is not authorized, and the value **ready** indicates that the application has been processed.
reason	String	Reason.
lockProof	String	Proof.
attributesJson	String	Attribute.

Status code: 500

Table 5-260 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

None

Example Responses

Status code: 200

ABE user key order information.

```
{
  "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==\"}"
}
```

Status code: 500

Error response.

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



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

Status Codes

Status Code	Description
200	ABE user key order information.
500	Error response.

Error Codes

See **Error Codes**.

5.5.4.7 Decrypting Data with ABE User Keys

Function

Decrypting Data with ABE User Keys

URI

POST /v1/datashare/abe-decrypt

Request Parameters

Table 5-261 Request body parameters

Parameter	Mandatory	Type	Description
orgID	Yes	String	Organization ID.
channelID	Yes	String	Channel ID.
cryptoMethod	Yes	String	Encryption method, which is fixed at **SW**.
cert	Yes	String	Certificate.
sk	Yes	String	Private key.
timestamp	Yes	String	Timestamp.
encryptData	Yes	String	Ciphertext data. This parameter is optional when **onChainStore** is set to **true**.
applyer	Yes	String	Applicant identifier.
provider	Yes	String	Authorizer identifier.

Parameter	Mandatory	Type	Description
orderSeq	No	String	Order number. This parameter is mandatory when **onChainStore** is set to **true**.
onChainStore	No	String	Whether to store ciphertext data on the chain. The value can be **true** or **false** (default). If you set this parameter to **true**, do not specify **encryptData**. The ciphertext data can be automatically obtained from the chain.

Response Parameters

Status code: 200

Table 5-262 Response body parameters

Parameter	Type	Description
plainData	String	Base64-decoded data.

Status code: 500

Table 5-263 Response body parameters

Parameter	Type	Description
errorCode	String	Error code.
errorMsg	String	Error description.

Example Requests

```
{
  "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"
}
```


Example Responses

Status code: 200

Base64-encoded data after ABE decryption.

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

Status code: 500

Error response.

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

Status Codes

Status Code	Description
200	Base64-encoded data after ABE decryption.
500	Error response.

Error Codes

See **Error Codes**.

6 Appendix

6.1 Encryption Using OSCCA-Published Cryptographic Algorithms

6.1.1 Overview

OSCCA-published cryptographic algorithms are cryptographic technologies and products used for the encryption or authentication of information that does not involve state secrets.

OSCCA-published cryptographic algorithms are independent and controllable algorithms developed by the State Cryptography Administration. They can improve the encryption strength and encryption and decryption performance. OSCCA-published cryptographic algorithms meet the requirements of government agencies, public institutions, large state-owned enterprises, financial banks, and other industries for reconstruction.

Huawei Cloud BCS provides the OSCCA-published cryptographic algorithm SDKs for you to develop client programs while protecting your private keys.

Downloading Resources

Table 6-1 SDK list

Matching Hyperledger Fabric Version	Language	Link
Fabric v1.4.0 and Fabric v2.2	Go	Go to the BCS console and click **Use Cases**. Download **Fabric_SDK_Go**.
	Java	Go to the BCS console and choose **Interactive Walkthroughs** > **Use Cases**. Download **Fabric_SDK_Gateway_Java** or **Fabric_SDK_Java**.

Matching Hyperledger Fabric Version	Language	Link
NOTE • The Go version must be v1.12 or later but must be earlier than v1.16. • The OSCCA-published cryptographic algorithm SDKs offer all functions of common SDKs and additionally support the OSCCA-published cryptographic algorithms. • Fabric_SDK_Gateway_Java encapsulates some interfaces of the SDK, covering Fabric_SDK_Java and making it easier to use. Therefore, Fabric_SDK_Gateway_Java is recommended.		

After decompressing the downloaded package, you will obtain the directories listed in the following table.

Directory	Description
src (Golang only)	Stores the Go SDK source code.
jar (Java only)	Stores the JAR package of the Java SDK.

6.1.2 Using SDKs

Installing the SDK

For details about how to obtain the Golang and Java packages, see **Overview**.

- Golang: Decompress the downloaded package to the **$GOPATH** directory.
- Java: Add the JAR file in the downloaded package to the dependency of the project as follows:
  a. Run the following command to register the downloaded SDK JAR package to the local Maven repository:
  ```
  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. Add the SDK dependency in the project by using the following code:
  ```
  <dependency>
      <groupId>org.hyperledger.fabric-sdk-java</groupId>
      <artifactId>fabric-sdk-java</artifactId>
      <version>2.2.6-BCS</version>
  </dependency>
  ```

Developing a Client Program

Develop your own application code. To use OSSCA-published cryptographic algorithms, simply choose them as the security mechanism when creating the BCS instance, and use OSCCA-published cryptographic algorithm SDKs instead of Fabric SDKs. The usage is the same.

Running the Client Program

To run the client program, you need to set the configuration file path, channel name, chaincode name, and organization ID.

- Configuration file path: the path for storing the downloaded configuration file
- Channel name: the channel name specified for the BCS instance
- Chaincode name: the name specified when the chaincode is installed for the BCS instance
- Organization ID: 02f23ab00f6e1ffcde8a27bfd3ac2290edc18127 in the following example configuration file

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

6.1.3 Appendix

The following table lists the third-party packages that **fabric-sdk-client/go** depends on.

No.	Package
1	github.com/Knetic/govaluate
2	github.com/VividCortex/gohistogram
3	github.com/cloudflare/cfssl
4	github.com/go-kit/kit
5	github.com/golang/mock
6	github.com/golang/protobuf
7	github.com/hashicorp/hcl
8	github.com/hyperledger/fabric-config
9	github.com/hyperledger/fabric-lib-go
10	github.com/hyperledger/fabric-protos-go
11	github.com/magiconair/properties
12	github.com/miekg/pkcs11
13	github.com/mitchellh/mapstructure
14	github.com/pelletier/go-toml
15	github.com/pkg/errors
16	github.com/prometheus/client_golang
17	github.com/spf13/afero
18	github.com/spf13/cast
19	github.com/spf13/jwalterweatherman

No.	Package
20	github.com/spf13/pflag
21	github.com/spf13/viper
22	github.com/stretchr/testify
23	github.com/tjfoc/gmsm
24	google.golang.org/grpc
25	gopkg.in/yaml.v2

6.2 Homomorphic Encryption

6.2.1 Overview

Huawei Cloud BCS provides a homomorphic encryption library to facilitate development. In addition to the encryption function offered by common encryption algorithms, homomorphic encryption also allows computation and conversion of cipher texts, which is important for security. Homomorphic encryption enables users without keys to perform computing on ciphertexts. In this way, the communication cost can be reduced, computing tasks can be transferred, and the calculation costs can be balanced among the involved parties. By using the homomorphic encryption technology, a decryption party obtains only the decryption result, but cannot obtain each message ciphertext. This improves information security.

BCS provides homomorphic encryption libraries for clients and chaincodes. These libraries are mainly used for transaction ciphertext computing to ensure user privacy and security.

- Library for clients: It provides the additive homomorphic encryption function and generates proofs of transaction amounts at clients.
- Homomorphic encryption chaincode **IDChaincode.go**: In a homomorphic encryption scenario, you need to download, install, and instantiate chaincodes for BCS instances before instance deployment.
- Library for chaincodes: It provides the zero knowledge proof function to verify the transaction proof of a user in a ciphertext case and generate post-transaction data. In this way, an endorser does not need to decrypt the transaction data of the user to determine the balance range.

Downloading Resources

Table 6-2 Library list

Matching Hyperledger Fabric Version	Library Version	Link
Fabric v1.1.0, v1.4.0, and v2.2	1.8.5	**Homomorphic encryption library**
	1.9.2	**Homomorphic encryption library**
	1.11.5	**Homomorphic encryption library**
• Homomorphic encryption chaincode: **IDChaincode.go** • Chaincode library API file: **api_ahe_cc.tar.gz**		
NOTICE • Select a package of the same version as the local compilation environment. For example, if the local Golang compiler is of version 1.8.5, download the library of version 1.8.5. • The OSCCA-published cryptographic algorithm SDK must be installed before using a homomorphic encryption library. • The **api_ahe_cc.tar.gz** package is used only for local compilation.		

6.2.2 Using the Homomorphic Encryption Library

This section describes how to use the homomorphic encryption library.

Procedure

Step 1 Buy a BCS instance and downloading files.

When buying a BCS instance, select **ECDSA** for **Security Mechanism**. After the subscription, download the homomorphic encryption library **ahex.x.x.tar.gz**, OSCCA-published cryptographic algorithm SDK **sdkx.x.x.tar.gz** in the client development kit, and the user certificates and configuration files of peers and orderers.

Step 2 Installing the client SDK library.

Decompress the downloaded OSSCA-published cryptographic algorithm SDK **sdkx.x.x.tar.gz** to the **$GOPATH** directory.

Step 3 Install the homomorphic encryption library.

Decompress the downloaded homomorphic encryption library **ahex.x.x.tar.gz** to the **$GOPATH** directory.

Step 4 Install the dependent library (only for Fabric v1.1).

The dependent library files are stored in the homomorphic encryption library directory. The decompressed homomorphic encryption library file is stored in the

$GOPATH/src/ahe/PSW/deps/lib directory. Copy all the files in this directory to the local **/usr/local/include/openssl/** directory. (If this directory does not exist, create it.) Then, run the following command to set the environment variables:

```
export LD_LIBRARY_PATH=/usr/local/include/openssl:$LD_LIBRARY_PATH
//The project path /usr/local/include/openssl is used as an example. Replace it with the actual directory, if any.
```

NOTE

If the system displays a message indicating that **libgmp** is missing, install the **gmp** library. For details about the installation method, see the Linux operating system package management tool. For example, if Ubuntu is used, run the **apt-get install libgmp10** command to install the library, or download the source code from https://gmplib.org/, then compile and install the library.

Step 5 Develop the client program and chaincode.

Refer to the **AHE Lib APIs** and **Chaincode Library APIs** to develop the application and chaincode (smart contract).

For example, the logic of an app client and chaincode is as follows:

- Typical service logic of an app client:
  a. Register a user.
     When a user is registered, the key generation function can be invoked to generate public and private keys for the user.
  b. Initialize the account balance.
     When the balance is initialized, the initial balance preparation function can be invoked to generate initial balance data with privacy protected.
  c. Initiate a transaction.
     During a transaction, the transaction preparation function can be invoked to generate transaction data with privacy protected.
- Typical service logic of a chaincode:
  a. Store the mapping between the user's public key and address.
  b. Verify the validity of the initial balance and generate an initial transaction.
  c. Verify the validity of transaction data and generate the transaction result.

     NOTE

     - The chaincode can invoke the initial balance verification function to verify the validity of the initial balance.
     - The transaction verification function is invoked to verify the validity of transaction data.

Step 6 Install a chaincode.

Install and instantiate the developed chaincode for the subscribed BCS instance on the management page.

Step 7 Deploy the application.

When developing the application, you can invoke the homomorphic encryption library to protect the transaction privacy. After the development is completed,

deploy the application on the purchased server. After deployment, ensure that the configuration files, certificate files (user certificates of peers and orderers), and openssl library are available in the environment.

----End

6.2.3 AHE Lib APIs

A homomorphic encryption library file is provided for integrating the homomorphic encryption function in the client application during development.

The code for importing the library is **import "ahe/PSW/api/ahelib"**.

GenerateKey

- **API prototype**

  func GenerateKey(pwd string) (privKeyStr string, pubKeyStr string, err error)

- **Function description**

  Generating a key pair for homomorphic encryption

- **Input**

Parameter	Type	Description	Mandatory
pwd	string	Used to encrypt the generated private keychain used for homomorphic encryption. AES-128 is used for the encryption. The **pwd** value must contain: 1. At least six characters 2. At least two types of the following characters: • Lowercase letters • Uppercase letters • Digits • Spaces or special characters including `~!@#$%^&*()-_=+|[{}];:'",<.>/?	Yes

- **Output**

Parameter	Type	Description
privKeyStr	string	Homomorphic private key encrypted using the **pwd**
pubKeyStr	string	Public keychain for homomorphic encryption

Parameter	Type	Description
err	error	Error message

- Note

  The complexity of keys depends on the actual application scenario. No restrictions need to be imposed on a key used to invoke an underlying library.

Encypt

- **API prototype**

  func Encypt (secretNumStr string, pubKeyStr string) (ciphertext string, err error)

- **Function description**

  Homomorphic encryption

- **Input**

Parameter	Type	Description	Mandatory
secretNumStr	string	Value to be encrypted. The value must be a positive integer greater than or equal to 0. If the number contains decimal places, the value must be multiplied to eliminate the decimal places.	Yes
pubKeyStr	String	Public key for homomorphic encryption	Yes

- **Output**

Parameter	Type	Description
ciphertext	string	Encrypted data
err	error	Error message

- **Note**

  None

Decrypt

- **API prototype**

  func Decrypt(ciphertext string, privKeyStr string, pwd string) (plainText *big.Int, err error)

- **Function description**

  Homomorphic decryption

- **Input**

Parameter	Type	Description	Mandatory
ciphertext	string	Ciphertext to be decrypted	Yes
privKeyStr	string	Private keychain encrypted using the **pwd** for protection	Yes
pwd	string	Character string used to protect the private key	No

- **Output**

Parameter	Type	Description
plainText	*big.int	Decrypted data
err	error	Error message

- **Note**

  None

Add

- **API prototype**

  func Add(cipher1, cipher2 string) (cipher string, err error)

- **Function description**

  Additive homomorphic encryption

- **Input**

Parameter	Type	Description	Mandatory
cipher1	string	Encrypted ciphertext 1	Yes
cipher2	string	Encrypted ciphertext 2	Yes

- **Output**

Parameter	Type	Description
cipher	string	Combined data
err	error	Error message

- **Note**

  None

InitBalance

- **API prototype**

  func InitBalance(balanceStr string, pubKey string) (string, error)

- **Function description**

  Balance initialization

- **Input**

Parameter	Type	Description	Mandatory
pubKey	string	Public keychain	Yes
balanceStr	string	Initial balance, which must be a positive integer greater than or equal to 0. If the number contains decimal places, the value must be multiplied to eliminate the decimal places.	Yes

- **Output**

Parameter	Type	Description
balanceInfo	string	Transaction amount information
err	error	Error message

- **Note**

  None

PrepareTxInfo

- **API prototype**

  func PrepareTxInfo(cipherBalanceA string, transNumStr string, pubKeyA, pubKeyB string, privKeyA string,pwd string) (string, error)

- **Function description**

  Transaction preparation

- **Input**

Parameter	Type	Description	Mandatory
cipherBalanceA	string	Current balance of user A (ciphertext), which is obtained from the blockchain.	Yes
transNumStr	string	Transfer amount (plaintext)	Yes

Parameter	Type	Description	Mandatory
pubKeyA	string	Public keychain of user A	Yes
pubKeyB	string	Public keychain of user B	Yes
PrivKeyA	string	Private keychain of user A	Yes
pwd	string	Character string used for encryption	No

- **Output**

Parameter	Type	Description
Txinfo	string	Transaction preparation data
err	error	Error message

- **Note**

  None

6.2.4 Chaincode Library APIs

The chaincode library is static and integrated in BCS instances. When developing a chaincode, you can use the API file to locally compile the chaincode.

Download the API file (for the download link, see **Downloading Resources**), and decompress it to the local **GOPATH** directory. Refer to the example chaincode provided in section 4.2.6 to import the homomorphic encryption library. After the chaincode is developed, install it in the BCS instance. The chaincode will automatically link to the library code in the instance to invoke the homomorphic encryption library for the chaincode.

The code for importing the homomorphic encryption library for the chaincode is **import "ahe/PSW/api/ChainCode"**.

NOTE

Do not modify this code line. Otherwise, the chaincode fails to invoke the homomorphic encryption library.

ValidateInitBalance

- **API prototype**

  func ValidateInitBalance(BalanceInfo, PubKey string) (InitBalance string, err error)

- **Function description**

  Verifies the validity of the balance proof in **balanceinfo** generated by **sdk.InitBalance**.

- **Input**

Parameter	Type	Description	Mandatory
BalanceInfo	string	Initial balance data	Yes
Pubkey	string	Public key for balance encryption	Yes

- **Output**

Parameter	Type	Description	Mandatory
InitBalance	string	Initial balance ciphertext	Yes
err	error	Error message	Yes

- **Processing**

  Balance ciphertext provided after the balance validity verification

- **Note**

  The account balance authenticity is ensured by the application logic. The chaincode can only check whether the amount is greater than 0 but cannot determine the actual balance of a user.

ValidateTxInfo

- **API prototype**

  ValidateTxInfo(txInfo, cipherBalanceA, cipherBalanceB string) (newCipherBalanceA,newCipherBalanceB,newCipherTxA,newCipherTxB string,err error)

- **Function description**

  Verifies the validity of the transaction proof in **Txinfo** generated by **PrepareTxInfo**.

- **Input**

Parameter	Type	Description	Mandatory
txinfo	string	Transaction proof data **NOTE** The transaction data includes the transaction proof data in addition to the ciphertext of the transaction data. It is obtained from **txinfo** returned using **sdk.PrepareTxInfo**.	Yes
cipherBalanceA	string	Current balance of user A (ciphertext)	Yes
cipherBalanceB	string	Current balance of user B (ciphertext)	Yes

- **Output**

Parameter	Type	Description
newCipherBalanceA	string	User A' balance to be updated after the transaction
newCipherBalanceB	string	User B' balance to be updated after the transaction
newCipherTxA	string	Transaction amount (encrypted using user A' homomorphic public-key encryption)
newCipherTxB	string	Transaction amount (encrypted using user B' homomorphic public-key encryption)
err	error	Error message

- **Note**

  In this money transfer transaction, user A is the payer, and B the payee.

6.2.5 IDChaincode

IDChaincode stores the public key and account data of a user. When a key pair for homomorphic encryption is generated for a user, the public keys must be registered with the IDchaincode so that the homomorphic-encryption public key of the payee can be queried using the account. For details about how to download **IDChaincode.go**, see **Downloading Resources**.

NOTE

IDChaincode.go is provided by Huawei Cloud BCS. You are advised not to modify it. If you modify it, its logic may be different from that of the example provided in **Sample Chaincode**.

Register

The account address is obtained by converting the public key hash into a hexadecimal character string.

Query

The account address is used to query the public key.

6.2.6 Sample Chaincode

A transaction chaincode implements the service logic. The example transaction chaincode provided in this section is used to transfer money between users. The validity of the transaction data in the form of a ciphertext is verified using the

homomorphic encryption library to detect unauthorized operations, if any. This chaincode implements the functions of balance initialization, balance query, and money transfer.

Initializing the Balance

After a user is registered successfully, the balance of the user must be initialized. The default value is 0.

- **Input**

Parameter	Type	Description	Mandatory
PubKey	string	Public key	Yes
Balance	string	Initial balance	Yes

- **Processing**

  Invoking the ValidateInitBalance API to verify the validity of the balance range for endorsement

```
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")
 }
```

- **Output**

Parameter	Type	Description
newCipherBalance	string	Balance ciphertext

Querying the Balance

The balance can be queried using the account address. For details, see the queryBalance API of the **transaction_demo**.

The following table lists the parameters of the queryBalance API.

-	Parameter	Type	Description
Input	addr	string	Account address
Output	cipherbalance	string	Ciphertext of the account balance
Output	err	error	Error message

Money Transfer

- **Input**

Parameter	Type	Description	Mandatory
AddrA	string	Address of user A (payer)	Yes
AddrB	string	Address of user B (payee)	Yes
txinfo	String	Transaction information PrepareTxInfo	Yes

- **Processing**

  a. Obtain the current balances of users A and B (cipherBalanceAKeyABlock and cipherBalanceBKeyBBlock) from the ledger based on the account addresses.

  b. Verify the validity of the certificates.

```
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. Provide the updated balances (ciphertext).

  The ledger content of a service needs to be customized. The encrypted balance amount is added to the users' ledgers. A storage structure is defined in the demo. After the storage structure is saved, a transaction record object is saved in JSON format.

```
type TransRecord struct {
    FromAddr string
    ToAddr   string
    TXType   string
    Balance  string
    TX       string
    remark   string
 }
```

6.2.7 Sample Application

This section provides an example application code and the corresponding chaincode to describe the use of the homomorphic encryption library. The main function of this application is money transfer between users, and the homomorphic encryption library is used to protect the transfer transaction information of the users.

The application involves three steps: user registration (including initializing the users' balances), money transfer, and balance query.

The application uses the command line interface (CLI) to perform service operations. The procedure is as follows.

Note

Query the domain names of orderers and peers in the downloaded **sdk.yaml** file, and add "*EIP* + *Domain name of each orderer*" and "*EIP* + *Domain name of each*

peer" to the **/etc/hosts** file. When Fabric v1.1 is used, set public network IP addresses as the EIPs of peers. When Fabric1.4 is used, set private network IP addresses for the peers. Otherwise, the network cannot be connected, causing transaction verification failures.

Perform the following steps to add the domain name mapping of the orderers and peers to the **/etc/hosts** file:

```
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
```

NOTE

Use real EIPs to replace *X.X.X.X*.

Registration

```
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
```

NOTE

If you do not set certain parameters, the default values are used. For the default values, see the **Flags** part. If a configured value is different from the default value, specify it for the parameter.

- Generating a pair of public and private keys for homogenous encryption

When a new user registers, a pair of public and private keys are generated for the user (identified by the user ID). This demo writes the key pair to the local **${userid} .data** file for each user.

```
privKeyStr, pubKeyStr, err := pswapi_sdk.GenerateKey(propwd)
check(err)
fmt.Println("key is nil")
userdata.PubKey = pubKeyStr
userdata.PriKey = privKeyStr
```

- Registering the public key

The Fabric SDK API is invoked to initiate registration with the IDChaincode, carrying the generated public key.

NOTE

The registration is implemented using the chaincode function of blockchains. Public keys are registered for sharing. During the money transfer transaction, the public key of the payer is used to encrypt the transaction data to protect privacy. Only the private key holder can decrypt the transaction data.

```
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))
 }
```

- Registering the initial balance
  a. The sdk.InitBalance API is used to initialize and encrypt the balance, generating the initial balanceinfo.
  b. The balanceinfo is sent to the transaction 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)
```

Transaction

```
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
// Note: The character string following -b is the account address of user B (displayed upon successful
registration of user B).
```

Query

```
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
```

Query code:

1. The InitBalance API provided by the SDK is invoked to initialize the balance.
2. The Fabirc SDK is invoked to send the transaction information.

```
get balance
 setup.ChainCodeID = txchaincode
 transRec := sdk_client.TransRecord{}

 fmt.Println("query balance")

 resps, err := sdk_client.Query(setup, "QueryBalance", [][]byte{[]byte(addrA)})
 if err != nil {
```



```
 fmt.Println("Fail to query balance :", err.Error())
 return err
}

err = json.Unmarshal(resps[0].ProposalResponse.GetResponse().Payload, &transRec)
if err != nil {
 fmt.Println("unmarshal query result error: ", err.Error())
 return err
}
```

```
decrypt balance

curbalance, err := pswapi_sdk.Decrypt(transRec.Balance, userdata.PriKey, propwd)
if err != nil {
 fmt.Println("sdk Decrypt error: ", err.Error())
 return err
}

fmt.Println("current balance:" + curbalance.String())
```

6.2.8 Transaction Verification with Homomorphic Encryption (Demo)

This section describes how to use the demo for transaction verification with homomorphic encryption.

NOTE

This is a demo only and is not for actual use.

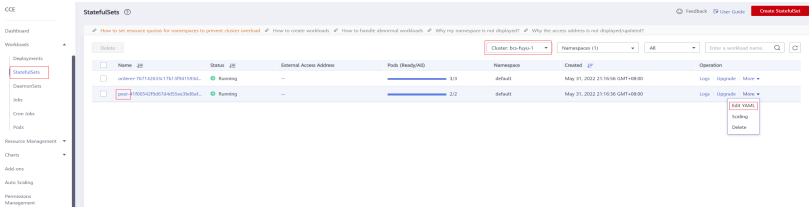
Procedure

Step 1 Buy a BCS instance.

Select a 4.x.x version (corresponding to Fabric v2.2) for **Version**, enter a blockchain name, and select **ECDSA** for **Security Mechanism**. Retain the default settings for the organization quantity and name.

Step 2 Change the chaincode container version.

1. Click **Instance Management**, and click a target instance cluster where the homomorphic encryption chaincode will be installed.
2. Choose **Workloads** > **StatefulSets**. Select a target cluster, click **More** in the **Operation** column of a target peer workload, and click **Edit YAML**.



3. Change the version number of **CORE_PEER_CCENV_IMAGE_NAME** to 3.0.5.

Step 3 Install and instantiate chaincodes.

Install and instantiate the demo chaincodes **transaction.zip** and **IDChaincode.zip**.

NOTE

- To facilitate subsequent operations, name the chaincodes **transaction** and **idchaincode**.
- During chaincode installation, enter **1.0** for **Chaincode Version** (as shown in the demo chaincode), and set **Golang** for **Language**.
- If the chaincode is developed by yourself, use the chaincode library interface file **api_ahe_cc.tar.gz**.

Step 4 On the **Instance Management** page, click **Download Client Configuration** on an instance card.

Step 5 Select configuration files to download and set the parameters as follows:

- Select **SDK Configuration File**.

  Chaincode Name: **transaction**

  Certificate Path: **/home/paas**

  Channel: **channel**

  Member: **organization**
- Select **Orderer Certificate**.
- Select **Peer Certificates**, retain the default selection for **Peer Organization**, and select **Administrator certificate**.

Step 6 Click **Download**.

Step 7 Install Golang on the local server.

1. Download the installation package **go1.11.5.linux-amd64.tar.gz**, upload it to the **/usr/local** directory on the local server, and decompress the package.


```
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. Add the following environment variables to the **/etc/profile** file:

```
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. Run the following command to make the environment variables take effect:

```
source /etc/profile
```

Step 8 Compile **appdemo**.

1. Go to the **/opt/gopath** directory (if this directory does not exist, create it manually), and upload the SDK library **(sdk1.11.5.tar.gz)**, homomorphic encryption library **(ahelib1.11.6.tar.gz)**, and OpenSSL library **(openssl.tar.gz)**. Run the following command to decompress the packages to the current directory:

```
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. Go to the **/opt/gopath/src/ahe/PSW/deps/lib** directory and copy the files in the directory to the **/usr/local/include/openssl/** directory (if this directory does not exist, create it manually).
3. Go to the **/opt/gopath/src/ahe/PSW/example/appdemo/** directory and run the **go build** command to compile the **appdemo** file.

```
[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]#
```

Step 9 Generate an image.

1. Create a folder, for example, **mkdir cj**, in the **/home/paas/** directory to generate an image.

2. Decompress the package downloaded in **Step 5**. In the **/home/paas/cj** directory, upload the administrator certificates of the orderer and peers, and copy the **appdemo** file compiled in **Step 6.3**, OpenSSL library, SDK library, and **Dockerfile** to the current directory and decompress them.

```
-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]#
```

NOTE

After decompression, add **libltdl.so.7** and **libltdl.so.7.3.0** to the **/home/paas/cj/openssl** directory.

3. Run the **unzip xxx.zip** command to decompress the administrator certificates of the orderer and peers. Move the decompressed certificate files respectively to the **/home/paas/cj/orderer** and **/home/paas/cj/peer** directories (if the directories do not exist, create them manually).

```
[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. Decompress the package downloaded in **Step 5** to obtain a .yaml file from the **sdk-config** folder and modify the certificate path configuration in the file.

   For example:

   a. Delete the hash prefixes from the peer addresses in all paths. Change the peer addresses to the addresses shown in the following figure.

```
organizations:

  aa73c757c9026fb623495d7058ca177f6152bcea:
    mspid: aa73c757c9026fb623495d7058ca177f6152bceaMSP

    cryptoPath:  /home/paas/peer/msp
    tlsCryptoKeyPath: /home/paas/peer/tls/server.key
    tlsCryptoCertPath: /home/paas/peer/tls/server.crt

    peers:

      - peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30605

      - peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30606

    certificateAuthorities:
      - ca-org1

  ordererorg:
      mspID: "2cf8802066c1c5011fd396c54c9126a17c9cfcc9MSP"

      cryptoPath: /home/paas/orderer/msp
```

   b. Check the paths of the decompressed certificates. After the modification, the SDK configuration file does not contain paths with hash values.

```
Search "/home/paas/" (7 hits in 1 file)
  C:\Users\c00206633\Desktop\test-sdk-config.yaml (7 hits)
    Line 81:     cryptoPath:  /home/paas/peer/msp
    Line 82:     tlsCryptoKeyPath: /home/paas/peer/tls/server.key
    Line 83:     tlsCryptoCertPath: /home/paas/peer/tls/server.crt
    Line 97:       cryptoPath: /home/paas/orderer/msp
    Line 109:        path: /home/paas/orderer/msp/tlscacerts/tlsca.2cf8802066c1c5011fd396c54c9126a17c9cfcc9-cert.pem
    Line 124:        path: /home/paas/peer/msp/tlscacerts/tlsca.aa73c757c9026fb623495d7058ca177f6152bcea-cert.pem
    Line 137:        path: /home/paas/peer/msp/tlscacerts/tlsca.aa73c757c9026fb623495d7058ca177f6152bcea-cert.pem
```

   c. Delete the code (if any) in the **certificateAuthorities** section from the SDK configuration file, save the file, and upload the file to the **/home/paas/cj** directory.


```
certificateAuthorities:
  ca-org1:
    url: https://ca_peerOrg1:7054
    httpOptions:
      verify: true
    tlsCACerts:
      path: $GOPATH/src/github.com/hyperledger/fabric-sdk-go/test/api-server/tls/fabricca/certs/ca_root.pem
      client:
       keyfile: $GOPATH/src/github.com/hyperledger/fabric-sdk-go/test/api-server/tls/fabricca/certs/client/client_fabric_client-key.pem
       certfile: $GOPATH/src/github.com/hyperledger/fabric-sdk-go/test/api-server/tls/fabricca/certs/client/client_fabric_client.pem

    registrar:
      enrollId: admin
      enrollSecret: adminpw
    caName: ca-org1
```

The following is an example of the modified SDK configuration file:

```
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-2cf8802066c1c5011fd396c54c91
26a17c9cfcc9.default.svc.cluster.local:
    url: grpcs://49.4.81.160:30805

    grpcOptions:
      ssl-target-name-override:
orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c91
26a17c9cfcc9.default.svc.cluster.local
      grpc-max-send-message-length: 15

    tlsCACerts:
      path: /home/paas/orderer/msp/tlscacerts/
tlsca.2cf8802066c1c5011fd396c54c9126a17c9cfcc9-cert.pem

peers:


  peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30605:
```



```
  url: grpcs://49.4.81.160:30605

  grpcOptions:
    ssl-target-name-override: peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
    grpc.http2.keepalive_time: 15

  tlsCACerts:
    path: /home/paas/peer/msp/tlscacerts/tlsca.aa73c757c9026fb623495d7058ca177f6152bcea-cert.pem


 peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local:30606:


  url: 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. Modify **Dockerfile**: Change the name to that of the .yaml file decompressed from the downloaded SDK.

```
[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. In the **/home/paas/cj/** directory, run the following commands to package all files into an image. Ensure that you have installed Docker in advance.

```
docker build -t tongtaidemotest:byl1 .
docker save tongtaidemotest:byl1>tongtaitest.tar
```

```
total 404140
-rwx------ 1 root root  23189440 Feb 28 22:07 appdemo
-rw------- 1 root root     10237 Feb 28 22:11 bcs-tongtaitest-orderer-admin.zip
drwxr-xr-x 2 root root      4096 Jul 19  2019 bin
-rw------- 1 root root       170 Feb 28 22:10 Dockerfile
drwxr-xr-x 2 root root      4096 Feb 28 22:20 openssl
-rw------- 1 root root   7891504 Feb 28 22:08 openssl.tar.gz
drwx------ 4 root root      4096 Feb 28 22:11 orderer
drwx------ 4 root root      4096 Feb 28 22:11 peer
drwxr-xr-x 3 root root      4096 Jul 19  2019 pkg
-rw------- 1 root root  27261385 Feb 28 22:09 sdk1.11.5.tar.gz
drwxr-xr-x 3 root root      4096 Jul 19  2019 src
-rw------- 1 root root     10048 Feb 28 22:11 testp-admin.zip
-rw------- 1 root root      3883 Feb 28 22:15 test-sdk-config.yaml
-rw------- 1 root root 355431424 Feb 28 22:22 tongtaitest.tar
```

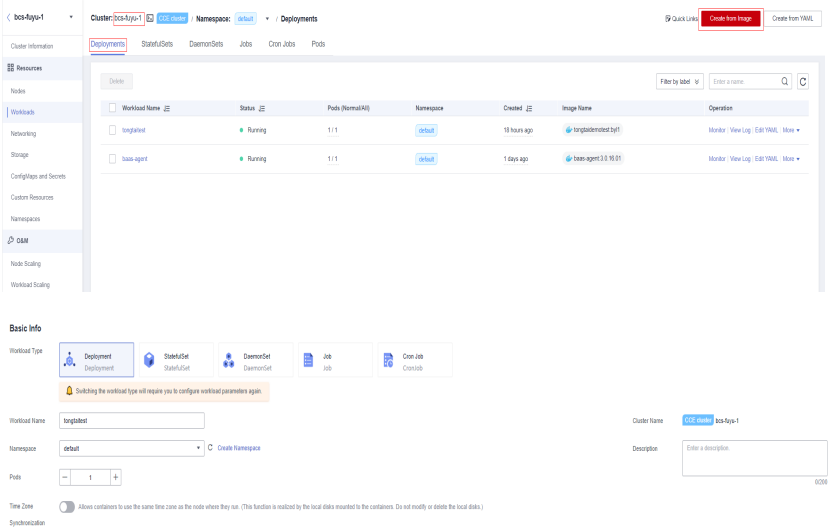
NOTE

Ensure that the EulerOS 2.2.5 image already exists on the local PC. Otherwise, the packaging will fail.

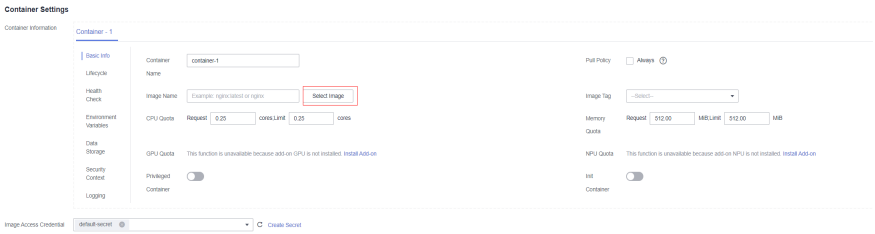

```
[root@cluster-bcs-064s-zeb9 ~]# docker images | grep euleros
swr.cn-north-1.myhuaweicloud.com/op_svc_bcs/euleroswithnodejs
      3.0.5               f22d1f510c4e        3 months ago        999MB
swr.cn-north-1.myhuaweicloud.com/op_svc_bcs/euleroswithgolang
      3.0.4               c8b11902bfc1        6 months ago        320MB
euleros
      2.2.5               b0f6bcd0a2a0        2 years ago         289MB
[root@cluster-bcs-064s-zeb9 ~]#
```

Step 10 Upload the image.

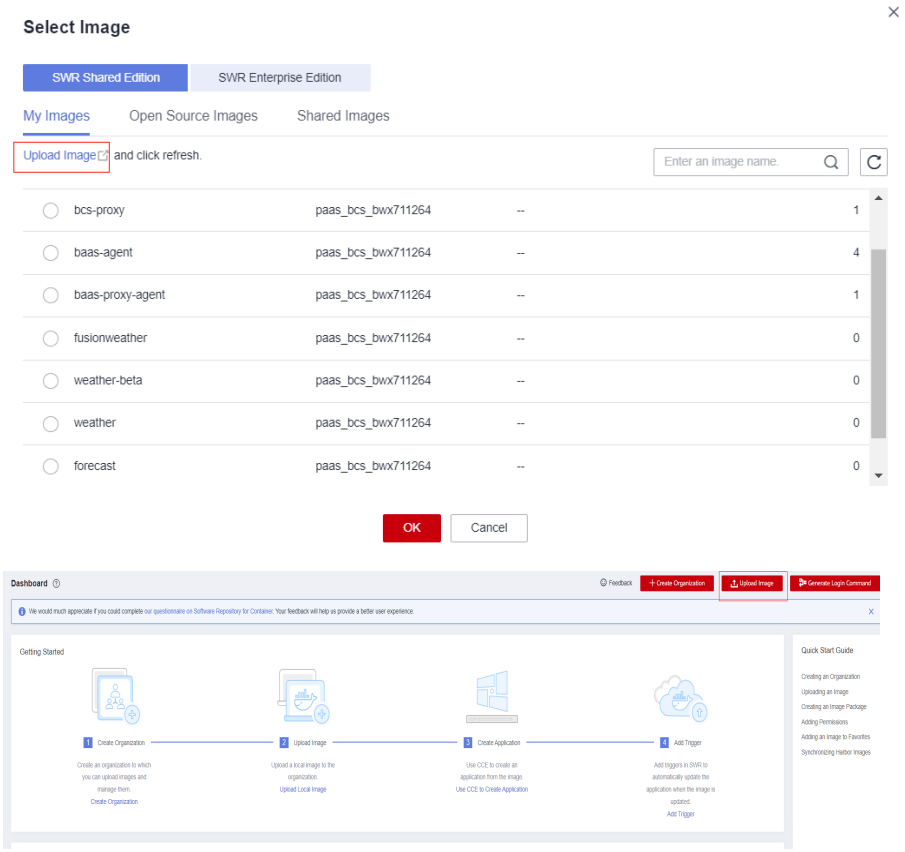
1. Download the packaged image to the local PC. Log in to Huawei Cloud and go to the CCE console. Create a Deployment in the cluster where the BCS instance is deployed, setting the pod quantity to **1**. Switch to the new CCE console in the upper right corner.



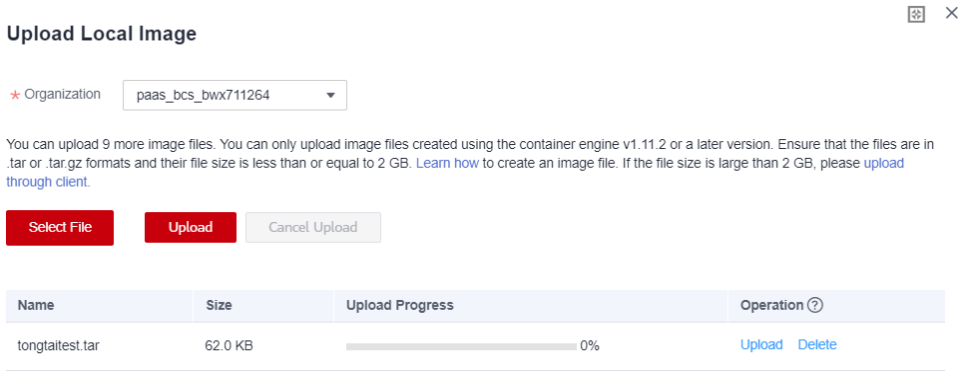
2. On the **Container Settings** tab page, select an image.



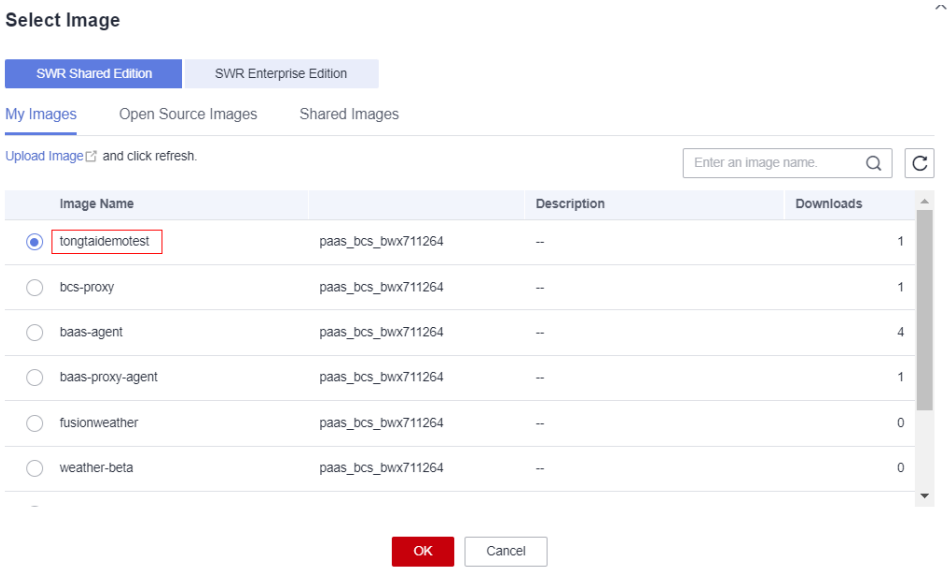
3. Click **Upload Image**. On the displayed **Dashboard** page, click **Upload Image**.



4. Select the packaged image file and wait until the upload is completed.



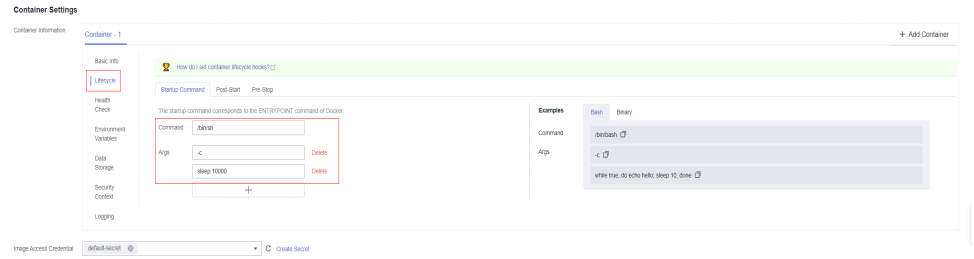
5. Then, go to the **Select Image** page, select the image, and click **OK**.



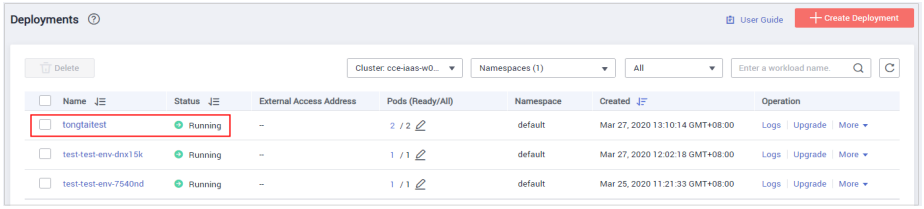
6. On the **Lifecycle** tab page, set the **Startup Command**.
   - **Command**: **/bin/sh**
   - **Args**:

     -c

     sleep 10000



7. Submit to create a Deployment. Save the settings.



Step 11 Verify transactions.

1. Log in to the ECS where the cluster is deployed, and run the following command to check whether the application container is normal:

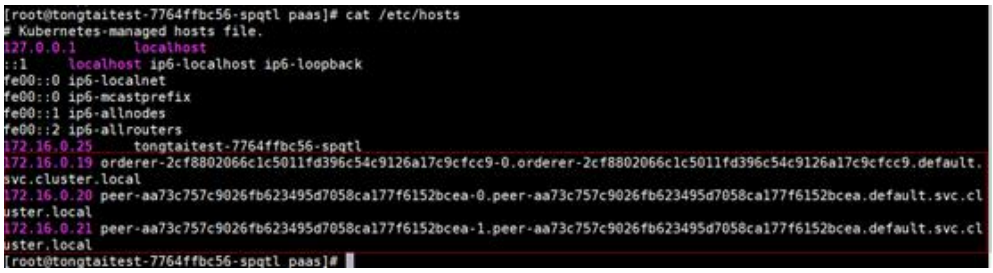
   ```
   docker ps -a | grep tongtai |grep container
   ```

2. You can view the name of the deployed application. Run the following command to access the container:

   ```
   docker exec -it Container ID bash
   ```

3. Add the domain name mapping between the orderer and peers to the **/etc/hosts** file.

Query the domains of the orderer and peers in the downloaded **sdk.yaml** file, and add "*IP address* + *Domain of the orderer*" and "*IP address* + *Domain of each peer*" to the end of the **/etc/hosts** file, as shown in the following figure.



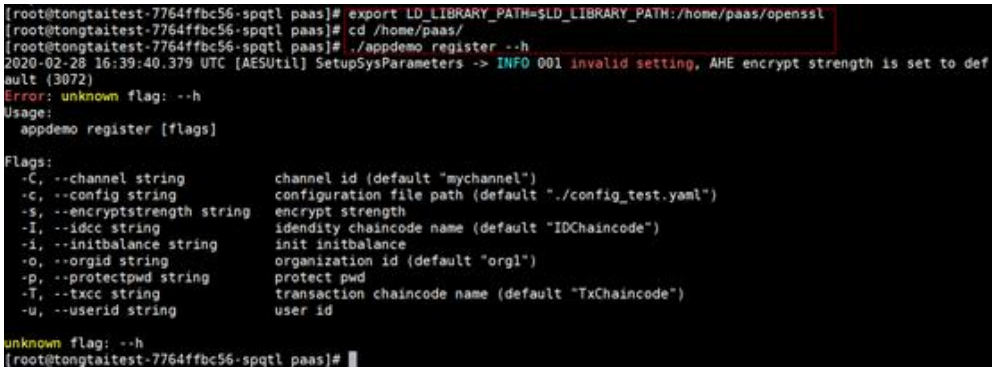

```
x.x.x.x
orderer-2cf8802066c1c5011fd396c54c9126a17c9cfcc9-0.orderer-2cf8802066c1c5011fd396c54c9126a17
c9cfcc9.default.svc.cluster.local
x.x.x.x peer-aa73c757c9026fb623495d7058ca177f6152bcea-0.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
x.x.x.x peer-aa73c757c9026fb623495d7058ca177f6152bcea-1.peer-
aa73c757c9026fb623495d7058ca177f6152bcea.default.svc.cluster.local
```

4. Run the following commands to configure environment variables and view the registration command:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/paas/openssl
cd /home/paas
./appdemo register -h
```

NOTE

- If you log out of the container during demo transactions, run the environment variable configuration commands again when logging in to the container next time.
- In the preceding command, **./appdemo register -h** is used to view the registration command parameters.



----End

Demo 1: Registering Accounts

Step 1 Run the following command to register account B with a balance of 100:

```
./appdemo register -u B -p tongtaitestB -i 100  -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T
transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```


NOTE

-u: the registered user name (**B**).

-p: user B's password (tongtaitestB). The password must contain at least two types of the following: uppercase letters, lowercase letters, digits, and special characters.

-c: the SDK configuration file name.

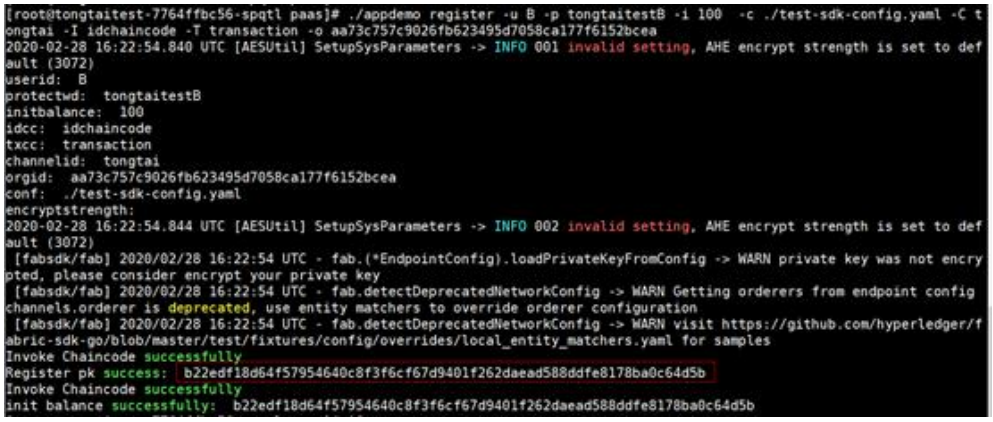
-C: the name of a channel for installing the chaincode.

-I: the actual chaincode name used after installing the demo chaincode **IDChaincode**.

-T: the actual chaincode name used after installing the demo chaincode **Transaction**.

-o: the organization ID of the peer, which can be queried on the **Channel Management** page.

The parameter description is the same in the following demos.

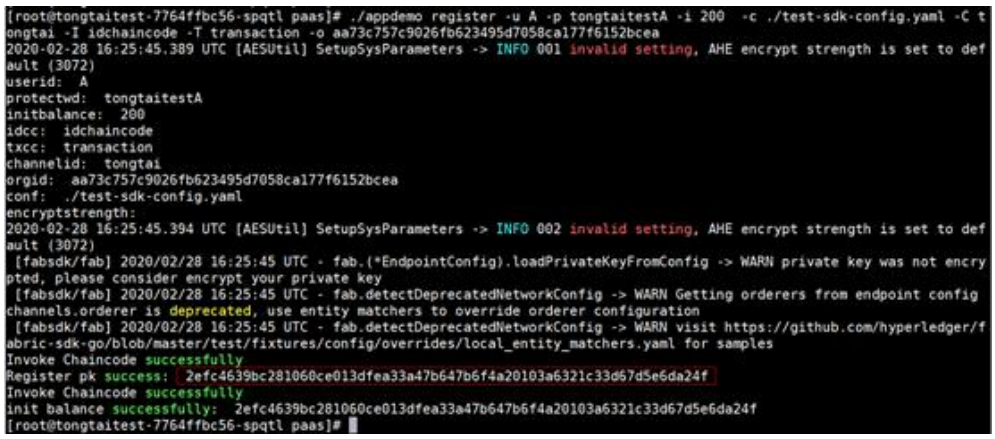


The returned value is an encrypted address, for example:

```
b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddfexxxxx
```

Step 2 Repeat the previous step to register account A with a balance of 200.

```
./appdemo register -u A -p tongtaitestA -i 200  -c ./test-sdk-config.yaml -C tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```



The returned value is an encrypted address, for example:

```
2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33dxxxxxx
```

----End

Demo 2: Transferring Money from A to B

Step 1 Run the following command to transfer money (10) from A to B:

```
./appdemo transaction -u A -p tongtaitestA -b
b22edf18d64f57954640c8f3f6cf67d9401f262daead588ddfe8178xxxx -t 10 -c ./test-sdk-config.yaml -C
tongtai -I idchaincode -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```

NOTE

The parameter after **-b** indicates the address of account B (the receiver), which is the value returned when registering account B.

```
[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]#
```

The return value is the address of account A: **2efc4639bc281060ce013dfea33a47b647b6f4a20103a6321c33d67d5xxxx**.

----End

Demo 3: Querying the Account Balance

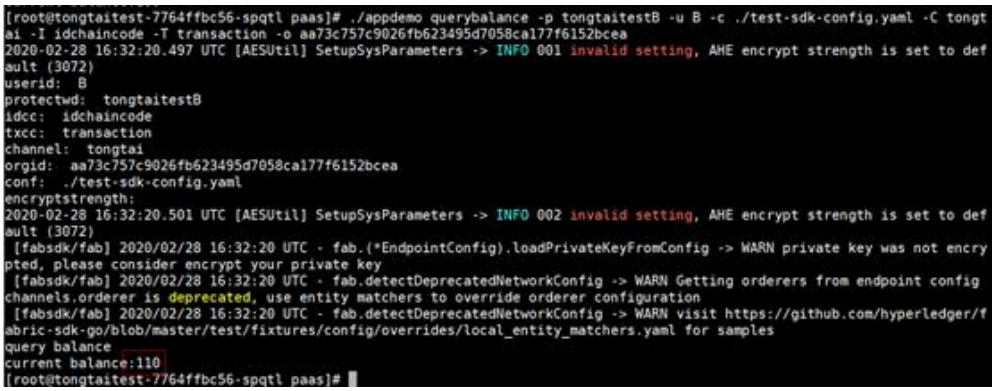
Step 1 Run the following command to query the balance of account A:

```
./appdemo querybalance -p tongtaitestA -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]#
```

Step 2 Run the following command to query the balance of account B:

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

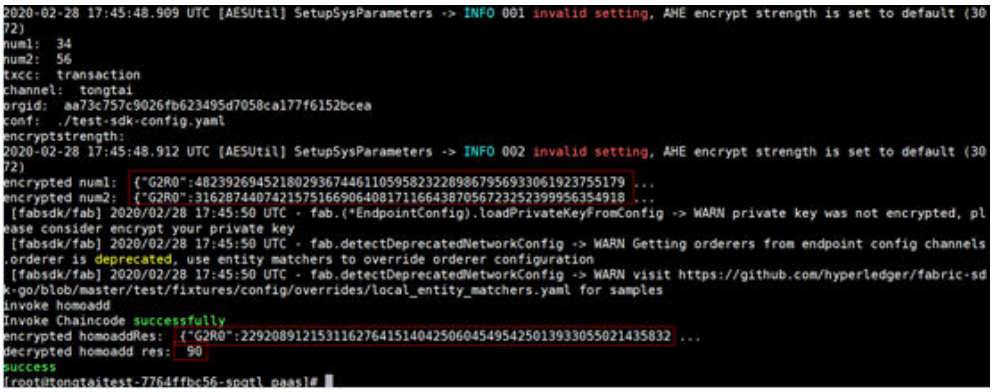



----End

Demo 4: Testing Additive Homomorphic Encryption

Step 1 Run the following command to test the additive homomorphic encryption:

```
./appdemo homoadd -c ./test-sdk-config.yaml -a 30 -b 60 -C tongtai -T transaction -o aa73c757c9026fb623495d7058ca177f6152bcea
```



NOTE

The parameters after **-a** and **-b** are two numbers for additive homomorphic encryption.

----End

Demo 5: Testing Multiplicative Homomorphic Encryption

Step 1 Run the following command to test the multiplicative homomorphic encryption:

```
./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":590474061597425199491667430880901668512467284038689194O2 ...
decrypted homomultiRes:  500
success
```

NOTE

The parameters after **-a** and **-b** are multipliers for multiplicative homomorphic encryption. The value after **-a** is encrypted, while the one after **-b** is in plaintext.

----End

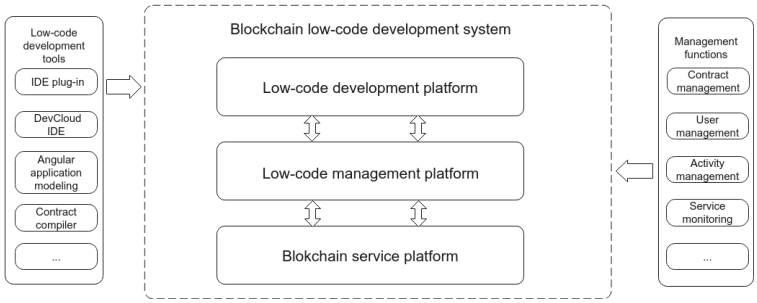
6.3 Low-Code Development

6.3.1 Overview

Currently, developing efficient and secure smart contracts poses high requirements on developers' understanding of basic blockchain knowledge and underlying contract languages, and expertise. In addition, operation personnel have to monitor the on-chain transaction progress in real time when processing service contracts, resulting a longer service management process and heavier development workloads.

BCS low-code development improves the blockchain usability by eliminating developers' dependency on blockchain knowledge and smart contract programming. It also allows operation personnel to focus more on services in a simple, easy, secure, and reliable way, with the blockchain service development and service-based process management functions.

Figure 6-1 System logic



NOTE

To use this function, contact technical support.

6.3.2 Developing Low-Code Contracts

When developing and testing traditional smart contract services, developers have to master the underlying contract languages, programming skills, and blockchain knowledge, to ensure the contract security. With the BCS low-code development function, only the business process modeling notations (BPMNs) and some assistant codes are required for compiling secure and reliable smart contracts.

The involved modules are as follows:

- BPMN diagram drawing

  On the easy-to-use BPMN drawing page, you can drag and combine start events, activities, gateways, attaching events, end events, and lanes to customize your service BPMN diagrams. And only a few service-based and query codes are required for developing smart contracts. On the drawing page, you can verify the BPMN diagram validity anytime before you click the **Build project** icon to generate a smart contract.

- Smart contract deployment

  Log in to the **Service Management** page, and the **Package Management** page is automatically displayed for you to view and manage the generated smart contracts. Specify the endorsement policies and target organizations for the generated contracts before installing and instantiating them. Smart contracts run in a Docker container on endorsing peers.

- Smart contract triggering

  The execution of instantiated smart contracts can be triggered by events or transactions. After it is triggered, endorsing peers start to reach a consensus to prevent attacks.

- Smart contract change

  After services are upgraded, you need to draw new BPMN diagrams and redeploy contracts of the new version. To ensure security, changes of a contract can be completed only after a specific quantity of endorsing peers reach a consensus.

6.3.3 Service Process Management

The common blockchain development and management require operation personnel to monitor on-chain transactions and process directions, and to filter correct transactions in the blockchain browser to proceed, increasing workloads.

BCS provides the service-based management process. Operation personnel can view the service progress and historical transactions intuitively. Also, users in different roles and groups can be assigned fine-grained permissions (not applicable to enhanced Hyperledger Fabric blockchains). For example, warehouse administrators can only distribute goods, and recipients can only confirm the acceptance. This function allows you to focus more on your services, enriching the upper-layer blockchain services and personnel management, and improving the usability of blockchains.

- User permission management

  Each smart contract initiator is also an administrator, who can manage user permissions on the **User Management** page. The initiator can set roles based on services, for example, buyers, sellers, and warehouse administrators; set

groups that contain one or more roles; and add users with specified usernames and passwords and assign them to different groups.

- Service monitoring

  Users can log in to the **Service Monitoring** page using the usernames and passwords to view the service progress, historical transactions, and process direction in graphics. Users can also check if the service is completed using the status flags.

- Service processing

  Services can be processed in the API-calling, automatic, and manual modes. API-calling mode: users can call contract SDKs to initiate transactions. Automatic mode: transactions are automatically initiated once certain requirements are met. Manual mode: users in different roles can perform certain operations based on their assigned permissions (not applicable to enhanced Hyperledger Fabric blockchains).

6.3.4 Usage Description

This section describes the process of low-code development. To use this function, contact technical support to obtain the tool package **BCS-BPMN.zip** and the license.

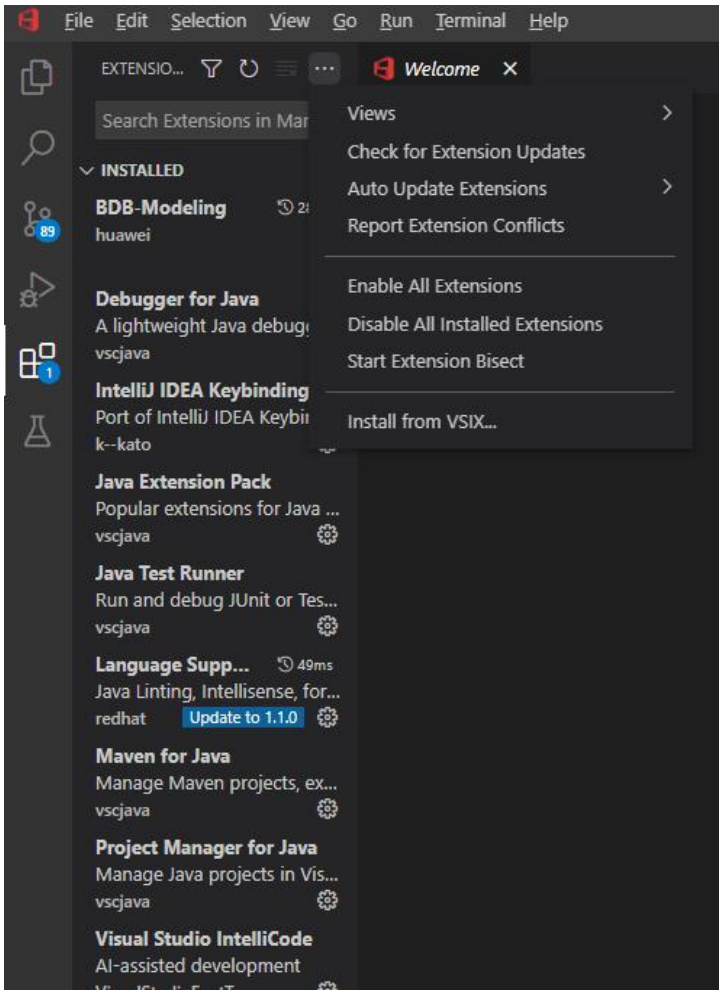
Step 1 Deploy the backend service.

1. Log in to the CCE console.
2. Create a CCE cluster. Buy an Ubuntu VM with 16-core CPU and 32 GB memory.
3. Select a cluster node, bind an EIP to the node, set security group rules for the node, add an inbound rule for TCP port 9096, and enable the Kubernetes service port. Set the port number to a value ranging from 1 to 32767.
4. Save the license file to the **/** directory on the VM, decompress the tool package **BCS-BPMN.zip** to the **/root** directory, and run the one-click deployment script **/root/BCS-BPMN/.build_config/one_step_deploy.sh**.

Step 2 Decompress the **BCS-BPMN.zip** tool package in Windows.

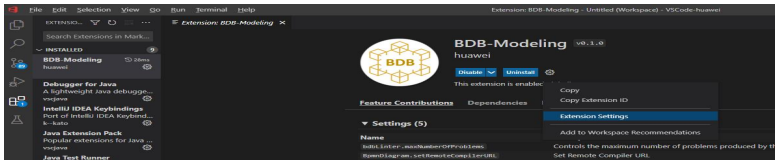
Step 3 Install the tool package.

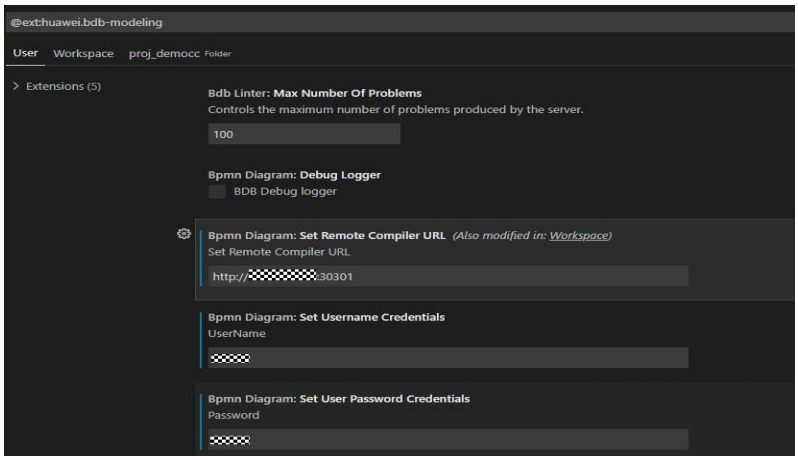
Start the VScode, click the expansion box on the left, click the three dots in the upper right, click **Install from VSIX**, and select **BCS-BPMN\extension\BDB-Modeling-0.1.0.vsix**.



Step 4 Configure the extension.

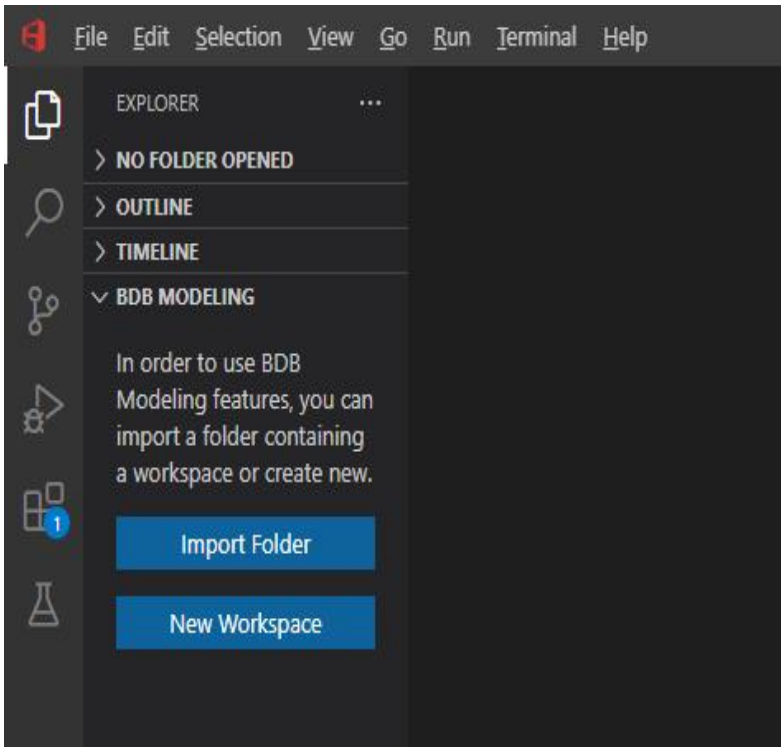
Click **Extension Settings** to set the **Remote Compiler URL** (**http://{***Tenant VM EIP***}:30301**), **UserName**, and **Password**.





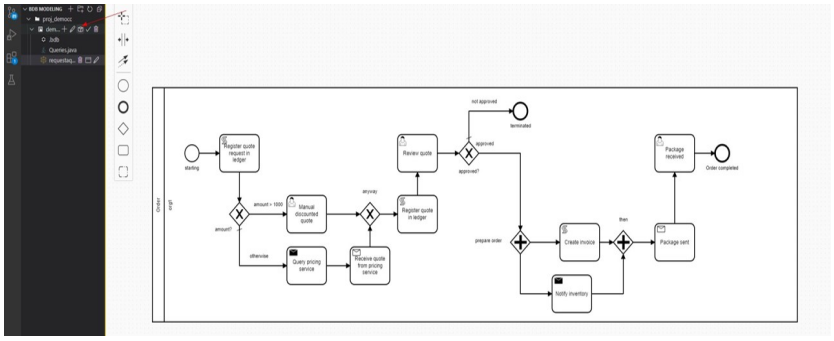
Step 5 Import the demo.

Choose **EXPLORER** > **BDB MODELING** > **Import Folder**, and select **BCS-BPMN \demo\projects\democc\modeller-workspace\proj_democc**.



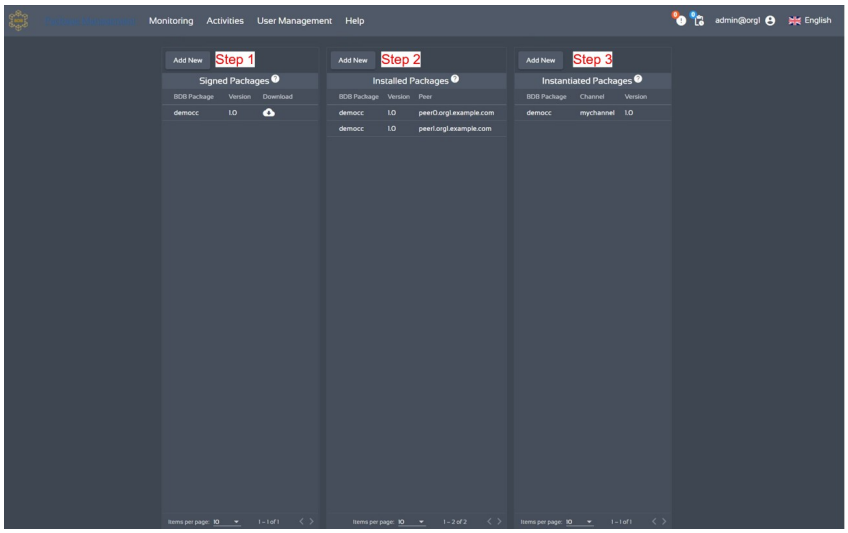
Step 6 Compile a BPMN diagram to generate a contract.

You can draw a BPMN diagram that meets your service requirements by referring to the example. After that, click **Compile**, wait for a few seconds until a message is displayed in the lower right, indicating that the compilation is successful.



Step 7 Install and instantiate the contract.

Log in to the management page by visiting **http://{***Tenant VM EIP***}:30300**. Click **Package Management**, and sign, install, and instantiate the contract in sequence.

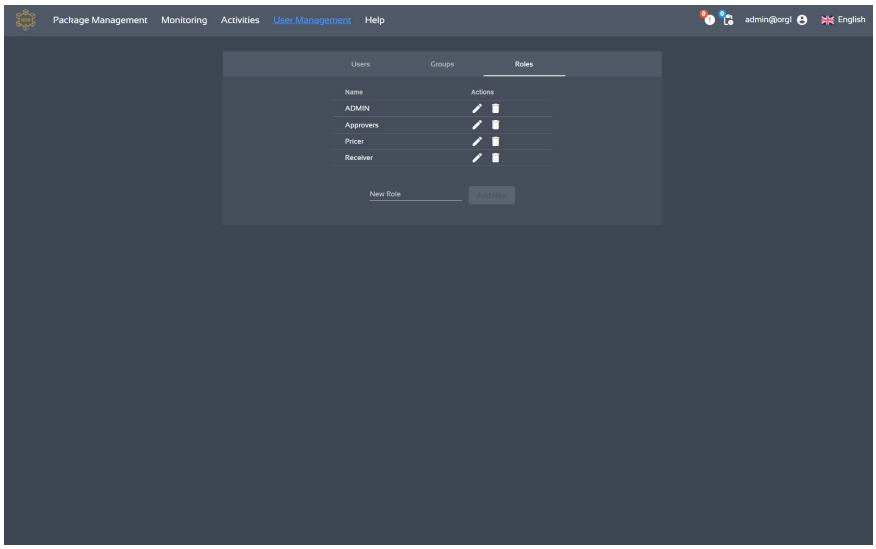


Step 8 Set roles, groups, and users on the **User Management** page.

Each role can be assigned fine-grained permissions to perform manual tasks. Each group can contain multiple roles. Each user can be added to multiple groups.

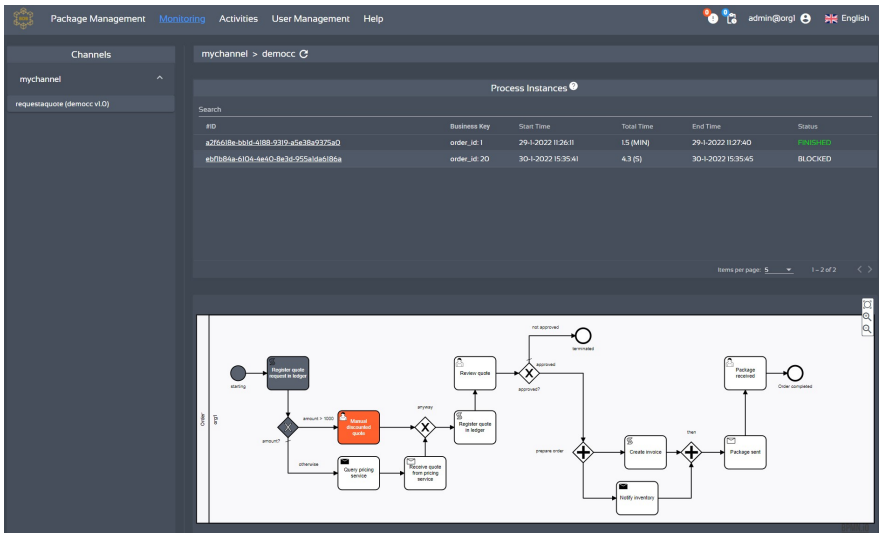
NOTE

Enhanced Hyperledger Fabric blockchains do not support fine-grained authorization.



Step 9 Monitor the services.

On this page, you can graphically check the service progress and direction, and use the status flags to check the service status.



Step 10 Access services using APIs.

BCS provides Java SDKs (now available) and RESTful APIs (coming soon) to easily call smart contracts and access the blockchain system.

----End

6.4 Error Codes

If an error code starting with APIGW is returned after you call an API, rectify the fault by referring to the instructions provided in **API Gateway Error Codes**.

Status Code	Error Codes	Error Message	Description	Solution
100	BCS.4006012	Invalid channel name.	Invalid channel name.	Enter a valid channel name.
400	BCS.2001001	BCS service modified.	BCS service modified.	The BCS service can work properly.
400	BCS.4001001	Operation failed. The BCS service is being created.	Operation failed. The BCS service is being created.	Wait until the service is created.
400	BCS.4001002	Operation failed. The BCS service is being upgraded.	Operation failed. The BCS service is being upgraded.	Wait until the upgrade is complete.
400	BCS.4001003	Operation failed. The BCS service is being deleted.	Operation failed. The BCS service is being deleted.	Wait until the deletion is complete.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001006	Failed to delete the SFS file system because the CCE API or the cluster status is abnormal.	Failed to delete the SFS file system because the CCE API or the cluster status is abnormal.	Manually delete the SFS file system. Procedure: Log in to the CCE console, choose **Resource Management** > **Storage**, and locate the file system in the cluster corresponding to the BCS service. For an SFS file system, click **Delete** in the **Operation** column. For an SFS Turbo file system, click **Unbind** and then go to the SFS console to delete the file system.
400	BCS.4001007	Services billed in the yearly/ monthly mode cannot be deleted.	Services billed in the yearly/ monthly mode cannot be deleted.	Services billed in the yearly/ monthly mode cannot be deleted. They can only be unsubscribed from on the console.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001008	Operation failed. Bind an EIP to at least one node in the cluster and try again.	Operation failed. Bind an EIP to at least one node in the cluster and try again.	Bind an EIP to at least one node in the cluster and try again. Procedure: Log in to the CCE console. Choose **Resource Management** > **Nodes**. Locate the nodes in the cluster corresponding to the service, and perform the following steps on any node: On the node details page, click the node name. On the displayed ECS console, click the ECS name. On the ECS details page, click the **EIPs** tab. Click **Bind EIP**.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001009	Operation failed. Bind an EIP to each node in the cluster and try again.	Operation failed. Bind an EIP to each node in the cluster and try again.	Bind an EIP to each node in the cluster and try again. Procedure: Log in to the CCE console. Choose **Resource Management** > **Nodes**. In the cluster corresponding to the service, locate the nodes that have not been bound with EIPs, and perform the following steps on each node: On the node details page, click the node name. On the displayed ECS console, click the ECS name. On the ECS details page, click the **EIPs** tab. Click **Bind EIP**.
400	BCS.4001010	Failed to download the BCS certificate because the etcd connection is abnormal.	Failed to download the BCS certificate because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001011	Operation failed. The service does not exist or has been deleted. Refresh the page.	Operation failed. The service does not exist or has been deleted. Refresh the page.	Wait for a few moments and refresh the page.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001015	Failed to obtain the peer information because the etcd connection is abnormal.	Failed to obtain the peer information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001018	Operation failed. A service with the same name already exists. Enter another name and try again.	Operation failed. A service with the same name already exists. Enter another name and try again.	Change the service name and try again.
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.	Failed to obtain the image version for upgrade because the required image is not in the image repository or the version configuration is incorrect.	Check the service version configuration and ensure that the image required for the upgrade is in the image repository.
400	BCS.4001020	Operation failed. The service status is abnormal or the service is being processed. Try again later.	Operation failed. The service status is abnormal or the service is being processed. Try again later.	Try again later.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001021	Failed to add the peer to the channel because the CCE API, cluster status, or etcd connection is abnormal.	Failed to add the peer to the channel because the CCE API, cluster status, or etcd connection is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001025	Invalid EIP. The EIP may have been unbound from the cluster. Check the EIP binding of the cluster.	Invalid EIP. The EIP may have been unbound from the cluster. Check the EIP binding of the cluster.	Log in to the CCE console, choose **Resource Management** > **Clusters**, and click the cluster corresponding to the service. Check whether the cluster nodes have been bound with EIPs. If not, bind EIPs to the cluster nodes.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001026	Failed to save the EIP to etcd because the etcd connection is abnormal.	Failed to save the EIP to etcd because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
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.	Failed to update the EIP information in the consortium member configuration because the CCE API, cluster status, or etcd connection is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001030	Operation failed because the etcd connection check failed.	Operation failed because the etcd connection check failed.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001031	Failed to query the member details because the etcd connection is abnormal.	Failed to query the member details because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001032	You do not have the required permission.	You do not have the required permission.	Configure the permissions by following the instructions provided in *BCS User Guide* > *Service Deployment* > *Prerequisites*.
400	BCS.4001033	Failed to delete the member information because the etcd connection is abnormal.	Failed to delete the member information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001036	Operation failed. The notification does not exist. Refresh the page.	Operation failed. The notification does not exist. Refresh the page.	Refresh the page later.
400	BCS.4001040	Failed to decline the invitation for joining a channel. Refresh the page and check the invitation status later.	Failed to decline the invitation for joining a channel. Refresh the page and check the invitation status later.	Refresh the page and check the invitation status later.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001042	Failed to query the member information because the etcd connection is abnormal.	Failed to query the member information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001043	Failed to modify the member information because the etcd connection is abnormal.	Failed to modify the member information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001044	Failed to accept the invitation. The acceptance is being processed. Refresh the page and check the invitation status later.	Failed to accept the invitation. The acceptance is being processed. Refresh the page and check the invitation status later.	Refresh the page later to view the processing result.
400	BCS.4001045	Weak password.	Weak password.	Enter a strong password.
400	BCS.4001047	Failed to modify the etcd data because the etcd connection is abnormal.	Failed to modify the etcd data because the etcd connection is abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001048	Failed to change the password because the CCE cluster is abnormal.	Failed to change the password because the CCE cluster is abnormal.	Log in to the CCE console and choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster nodes are normal. If they are abnormal, rectify the fault by referring to the CCE troubleshooting guide. After the cluster restores, try again. If the cluster nodes have been deleted, buy nodes and create a BCS service again.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001049	Operation failed. The cluster status is abnormal. Check the cluster status and try again.	Operation failed. The cluster status is abnormal. Check the cluster status and try again.	Log in to the CCE console and choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster nodes are normal. If they are abnormal, rectify the fault by referring to the CCE troubleshooting guide. After the cluster restores, try again. If the cluster nodes have been deleted, buy nodes and create a BCS service again.
400	BCS.4001050	Scaling failed because the CCE API or the cluster status is abnormal.	Scaling failed because the CCE API or the cluster status is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001051	Scaling failed due to incorrect request parameters. Check whether the number of peers in the organization has reached the limit (5).	Scaling failed due to incorrect request parameters. Check whether the number of peers in the organization has reached the limit (5).	Ensure that the number of peers in each organization does not exceed the upper limit (5) and try again.
400	BCS.4001052	Operation failed because it timed out.	Operation failed because it timed out.	View the result later.
400	BCS.4001053	Failed to query the network storage because the CCE API or the cluster status is abnormal.	Failed to query the network storage because the CCE API or the cluster status is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001054	Failed to add the peer to the channel because the CCE cluster status or etcd connection is abnormal.	Failed to add the peer to the channel because the CCE cluster status or etcd connection is abnormal.	Log in to the CCE console and check whether CCE is normal. Choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001058	Failed to delete the member information because the etcd connection may be abnormal.	Failed to delete the member information because the etcd connection may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001062	The maximum number of tenants that can be invited has been reached.	The maximum number of tenants that can be invited has been reached.	Do not invite more tenants after the invitation limit has been reached.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001063	The maximum number of invitation notifications that can be received has been reached.	The maximum number of invitation notifications that can be received has been reached.	Log in to the BCS console, go to the **Notification Management** page, and check whether the number of received invitations reaches the upper limit (100). Process the received notifications.
400	BCS.4001067	Failed to delete the notification. The etcd connection may be abnormal.	Failed to delete the notification. The etcd connection may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001069	The chaincode is being instantiated. Try again later.	The chaincode is being instantiated. Try again later.	Log in to the **Chaincode Management** page of the corresponding service to check the chaincode instantiation status. After the instantiation is complete, perform the operation again.
400	BCS.4001078	Failed to query the channel information because the etcd connection may be abnormal.	Failed to query the channel information because the etcd connection may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001080	Operation failed because the CCE cluster information failed to be obtained.	Operation failed because the CCE cluster information failed to be obtained.	Check if CCE is normal by trying to log in to the CCE console. If the login is successful, choose **Resource Management** > **Clusters** to check whether the cluster used by the service is normal. If the cluster is abnormal, try again after it restores.
400	BCS.4001081	Failed to obtain the invitee's organization because the etcd is abnormal.	Failed to obtain the invitee's organization because the etcd is abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001086	Permission verification failed.	Permission verification failed.	Configure the permissions by following the instructions provided in *BCS User Guide* > *Service Deployment* > *Prerequisites*.
400	BCS.4001088	Operation failed due to insufficient quota. Increase the quota and try again.	Operation failed due to insufficient quota. Increase the quota and try again.	Contact the administrator to increase the quota on ManageOne.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001089	Deletion failed. Failed to delete the BCS information.	Deletion failed. Failed to delete the BCS information.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001090	Operation failed. No node is available in the cluster. Try again later.	Operation failed. No node is available in the cluster. Try again later.	Check whether the node is abnormal or has been deleted. Log in to the CCE console. Choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster has nodes. If not, the nodes may have been deleted. In this case, buy nodes and create a BCS service again. If the nodes are abnormal, restore the nodes as instructed by the CCE troubleshooting document and try again later.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001095	Failed to delete block configurations due to etcd exceptions.	Failed to delete block configurations due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001096	Failed to delete the AgentConfig information due to etcd exceptions.	Failed to delete the AgentConfig information due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001097	Failed to delete the CCE cluster occupation information due to etcd exceptions.	Failed to delete the CCE cluster occupation information due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001098	Failed to delete the SDK information due to etcd exceptions.	Failed to delete the SDK information due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001099	Failed to delete the channel information due to etcd exceptions.	Failed to delete the channel information due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001100	Failed to delete the EIP information due to etcd exceptions.	Failed to delete the EIP information due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001101	Failed to delete the port information due to etcd exceptions.	Failed to delete the port information due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001102	Failed to delete the ipmapping information due to etcd exceptions.	Failed to delete the ipmapping information due to etcd exceptions.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001104	Failed to delete the deployment template because the CCE cluster status may be abnormal.	Failed to delete the deployment template because the CCE cluster status may be abnormal.	Check the CCE cluster status. Log in to the CCE console and choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, restore the cluster as instructed by the *CCE User Guide* and then try again.
400	BCS.4001105	Deleting the deployment template timed out because the CCE cluster status may be abnormal.	Deleting the deployment template timed out because the CCE cluster status may be abnormal.	Check the CCE cluster status. Log in to the CCE console and choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, restore the cluster as instructed by the *CCE User Guide* and then try again.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001106	Operation failed. The cluster does not exist. Try again later.	Operation failed. The cluster does not exist. Try again later.	Log in to the CCE console and choose **Resource Management** > **Clusters**. Check if the cluster used by the BCS service exists. If the cluster does not exist, create a cluster and buy a BCS service again.
400	BCS.4001109	Operation failed. Failed to obtain the peer organization status. Try again later.	Operation failed. Failed to obtain the peer organization status. Try again later.	Try again later.
400	BCS.4001110	Failed to create CertBitConf because the network may be faulty.	Failed to create CertBitConf because the network may be faulty.	Try again later.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001120	Failed to delete the service. The member failed to exit the consortium. The cluster or etcd connection is abnormal.	Failed to delete the service. The member failed to exit the consortium. The cluster or etcd connection is abnormal.	Log in to the CCE console and check whether CCE is normal. Choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001123	Failed to delete the service because the cluster has been hibernated.	Failed to delete the service because the cluster has been hibernated.	Wake up the cluster and try again. Log in to the CCE console and choose **Resource Management** > **Clusters**. Locate the cluster used by the service, choose **More** > **Wake**. After the cluster status is restored, delete the BCS service again.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001130	Failed to save the node and port information because the etcd, CCE API, or cluster is abnormal.	Failed to save the node and port information because the etcd, CCE API, or cluster is abnormal.	Log in to the CCE console and check whether CCE is normal. Choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001133	Failed to save the channel information because etcd may be abnormal.	Failed to save the channel information because etcd may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001134	Failed to save the BCS information because etcd may be abnormal.	Failed to save the BCS information because etcd may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001135	Failed to save the certificate because etcd may be abnormal.	Failed to save the certificate because etcd may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001136	Failed to save bitconf because etcd may be abnormal.	Failed to save bitconf because etcd may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001137	Failed to save the deployment template because etcd may be abnormal.	Failed to save the deployment template because etcd may be abnormal.	Check the etcd status. If etcd is abnormal, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001149	Failed to query the cluster details because the CCE service or API is abnormal.	Failed to query the cluster details because the CCE service or API is abnormal.	Check the CCE service and API status. Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001161	Failed to add the organization because the etcd connection or CCE API may be abnormal.	Failed to add the organization because the etcd connection or CCE API may be abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001171	Weak password.	Weak password.	Enter a strong password.
400	BCS.4001196	Operation failed because the current service or services in the consortium are being updated. Try again later.	Operation failed because the current service or services in the consortium are being updated. Try again later.	Try again later.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001202	Deletion failed. The service does not exist or has already been deleted. Refresh the page.	Deletion failed. The service does not exist or has already been deleted. Refresh the page.	Refresh the **Service Management** page.
400	BCS.4001203	Operation failed. Failed to obtain the deployment task.	Operation failed. Failed to obtain the deployment task.	Try again.
400	BCS.4001212	Operation failed. The ECS specified by node_flavor in the request cannot be purchased.	Operation failed. The ECS specified by node_flavor in the request cannot be purchased.	Specify an ECS with other flavors in the node_flavor.
400	BCS.4001234	Operation failed because the BCS service is in the Hibernating or Frozen state.	Operation failed because the BCS service is in the Hibernating or Frozen state.	Click **More** > **Wake** in the **Operation** column of the row containing the specified service and try again after the service is normal.
400	BCS.4001242	Failed to remove the organization from a channel.	Failed to remove the organization from a channel.	Remove the organization again until the channel and service statuses are normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001301	Failed to query the ROMA instance information because the ROMA instance may have been deleted or its status is abnormal.	Failed to query the ROMA instance information because the ROMA instance may have been deleted or its status is abnormal.	Log in to the ROMA console and check whether the instance status is normal. Wait until the instance status is normal or use an unused, normal instance.
400	BCS.4001302	Failed to query the cluster information because the cluster has been deleted or is abnormal.	Failed to query the cluster information because the cluster has been deleted or is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. If the cluster corresponding to the BCS service does not exist, the cluster has been deleted. In this case, use an existing, normal cluster to buy a BCS service again. If the cluster exists, check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored or use a normal cluster to create another BCS service.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001304	Failed to query the ROMA instance list because the ROMA MQS API is abnormal.	Failed to query the ROMA instance list because the ROMA MQS API is abnormal.	Log in to the ROMA console and check whether ROMA MQS and the ROMA instance is normal. If they are abnormal, try again after they are normal or rectify the fault as instructed by the ROMA troubleshooting document.
400	BCS.4001305	Failed to obtain the topic information of the ROMA instance because the ROMA instance is abnormal or has been deleted.	Failed to obtain the topic information of the ROMA instance because the ROMA instance is abnormal or has been deleted.	Log in to the ROMA console to check whether the ROMA instance used by the BCS service is normal. If it is abnormal, try again after it is normal. If the instance is deleted, purchase a new ROMA instance and create a BCS service again.
400	BCS.4001306	Operation failed because the ROMA instance is currently in use.	Operation failed because the ROMA instance is currently in use.	Select an unused, normal ROMA instance.
400	BCS.4001307	Operation failed because you do not have the permissions to view DMS data. Check your permissions in IAM.	Operation failed because you do not have the permissions to view DMS data. Check your permissions in IAM.	Check the account permissions in IAM. For details about the required permissions, see the user guide. Add the DMS permissions and try again.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001356	Operation failed. No network storage space has been configured for the cluster.	Operation failed. No network storage space has been configured for the cluster.	Buy a BCS service again. When configuring **Network Storage**, choose **Create SFS File System**.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001401	Failed to update the IPMapping information of the current tenant because etcd, the CCE API, or the cluster is abnormal.	Failed to update the IPMapping information of the current tenant because etcd, the CCE API, or the cluster is abnormal.	Log in to the CCE console and choose **Resource Management** > **Clusters**. Locate the cluster used by the BCS service, and check whether the cluster is normal. If the cluster is abnormal, rectify the fault by referring to the CCE troubleshooting document or use an available cluster to buy another BCS service. If the cluster status is normal, choose **Configuration Center** > **ConfigMaps**. Filter the configuration items by cluster, find the IPMapping of the cluster corresponding to the BCS service, and check whether the configuration is correct and whether the format is complete. If no fault if found in the preceding steps, rectify the etcd fault by referring to the *BCS Troubleshooting* document and try

Status Code	Error Codes	Error Message	Description	Solution
				again after etcd is normal.
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.	Failed to update the access address because the EIP information in the SDK configuration fails to be updated due to an etcd exception.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001404	Failed to update the EIP information of other consortium members because etcd may be abnormal.	Failed to update the EIP information of other consortium members because etcd may be abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001407	Try again or check whether ETCD is normal.	Obtain EIP data failed.	Try again or check whether ETCD is normal.
400	BCS.4001408	Do not use the domain name repeatedly. Apply for a new domain name or add a new domain name resolution set.	The domain name has been used in other BCS instances.	Do not use the domain name repeatedly. Apply for a new domain name or add a new domain name resolution set.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001520	Failed to obtain the topology information of the invitee because etcd may be abnormal.	Failed to obtain the topology information of the invitee because etcd may be abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001521	Failed to obtain the topology information of the inviting party because etcd may be abnormal.	Failed to obtain the topology information of the inviting party because etcd may be abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001708	Operation failed. The IP address of the IEF node cluster is incorrect.	Operation failed. The IP address of the IEF node cluster is incorrect.	1. Log in to the CCE console. 2. Choose Resource Management > Clusters, and click the cluster of a target BCS service. 3. Click Nodes, view the EIP bound to the cluster node. 4. Set the public IP addresses of these IEF nodes to the EIPs bound to the nodes.
400	BCS.4001717	Operation failed. The number of IEF nodes is less than that required by the consensus policy.	Operation failed. The number of IEF nodes is less than that required by the consensus policy.	Buy IEF nodes required by the consensus policy or buy more such nodes.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001750	Failed to query the quota from ManageOne because the ManageOne API may be abnormal.	Failed to query the quota from ManageOne because the ManageOne API may be abnormal.	Check the ManageOne API status. Log in to ManageOne and go to the quota management page. If the displayed page shows the service quotas in the specified region, the ManageOne API is normal. If the ManageOne API is abnormal, try again after it restores.
400	BCS.4001751	The BCS service quota is not found because the quota is not registered or the quota configuration is incorrect.	The BCS service quota is not found because the quota is not registered or the quota configuration is incorrect.	Log in to ManageOne and go to the quota management page. If the displayed page shows the BCS quota in the specified region, the BCS quota has been registered correctly. Otherwise, the BCS quota has not been registered or the quota configuration is incorrect. In this case, contact the administrator to register the BCS quota again.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001752	Failed to modify the quota on ManageOne because the ManageOne API may be abnormal.	Failed to modify the quota on ManageOne because the ManageOne API may be abnormal.	Check the ManageOne API status. Log in to ManageOne and go to the quota management page. If the displayed page shows the service quotas in the specified region, the ManageOne API is normal. If the ManageOne API is abnormal, try again after it restores.
400	BCS.4001753	Failed to query the product from ManageOne because the product does not exist, the product information is abnormal, or the ManageOne API is abnormal.	Failed to query the product from ManageOne because the product does not exist, the product information is abnormal, or the ManageOne API is abnormal.	Check the ManageOne API status and the product information. Log in to ManageOne and go to the quota management page. If the displayed page shows the service quotas in the specified region, the ManageOne API is normal. Then, check the BCS information. If the BCS information is missing, contact the administrator to register BCS. If the page is not displayed properly, the ManageOne API is abnormal. In this case, try again after it restores.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4001754	This operation cannot be performed on the order because the order parameters do not exist.	This operation cannot be performed on the order because the order parameters do not exist.	Buy a BCS service again.
400	BCS.4001755	Failed to save the order parameters because etcd may be abnormal.	Failed to save the order parameters because etcd may be abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4001756	The order processing cannot proceed because the order parameters have changed.	The order processing cannot proceed because the order parameters have changed.	Buy a BCS service again.
400	BCS.4001757	Operation verification failed. Only create and delete operations are supported.	Operation verification failed. Only create and delete operations are supported.	Check whether the operation is creation or deletion.
400	BCS.4001999	Failed to accept the invitation. Refresh the page and check the invitation status later.	Failed to accept the invitation. Refresh the page and check the invitation status later.	Refresh the page and check the invitation status later.
400	BCS.4003007	No authorization from CCE.	No authorization from CCE.	Obtain authorization on the CCE console.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4004001	You cannot select EVS storage because the CCE cluster version is earlier than 1.15.	You cannot select EVS storage because the CCE cluster version is earlier than 1.15.	Log in to the CCE console, choose **Resource Management** > **Clusters**, click the cluster where the BCS service is deployed, and check whether the cluster version is later than 1.15. If the cluster version is earlier than 1.15, create another cluster whose version is later than 1.15. If CCE does not support clusters whose version is later than 1.15, contact CCE support personnel.
400	BCS.4004002	EVS storage is not supported.	EVS storage is not supported.	Log in to CloudAutoDeploy-CDK, select a region, and choose **Cloud Service Management** > **Upgrade**. Select the cluster where the BCS service is deployed and select the baas-service pod for upgrade. Change the value of **evs_service_enable** to **true**, and perform the upgrade. Wait until the upgrade is complete.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4004003	You cannot select EVS storage because the EVS disk and the CCE node are located in different AZs.	You cannot select EVS storage because the EVS disk and the CCE node are located in different AZs.	Log in to the CCE console, choose **Resource Management** > **Clusters**, and click the cluster where the BCS service is deployed. Then, choose **Nodes** in the navigation pane, and view the AZs configured for the nodes. Choose **Storage** in the navigation pane, click the **EVS** tab, and click **Buy EVS Disk**. Check whether the AZ of the EVS disk is the same as that of the cluster nodes. If they are different, purchase another CCE cluster in the same AZ as the EVS disk.
400	BCS.4004004	You cannot select EVS storage when the Fabric version of the BCS service is earlier than 3.0.	You cannot select EVS storage when the Fabric version of the BCS service is earlier than 3.0.	Set the Fabric version of the BCS service to 3.0 or later.
400	BCS.4004005	You cannot change the billing mode of the BCS service to yearly/ monthly because the BCS service uses EVS storage.	You cannot change the billing mode of the BCS service to yearly/ monthly because the BCS service uses EVS storage.	BCS services billed in the yearly/ monthly mode do not support EVS storage. You can use SFS file systems instead.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4004006	Failed to query the EVS type.	Failed to query the EVS type.	Check whether the network is normal. Manually run the curl command with the EVS domain name. If the domain name cannot be accessed, contact the DNS service personnel. Log in to the EVS console, click **Buy Disk**, and check whether an error message is displayed. If an error message is displayed, contact the EVS O&M personnel. Check whether the back-end service of BCS is normal. If the back-end service is abnormal, view the corresponding logs.
400	BCS.400402	Query failed.	Query failed.	Check the entity type in the input parameter.
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.	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.	Check whether the AOM service is normal and whether the AOM monitoring API can be accessed.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4004025	Invalid request information.	Invalid request information.	Check the request information and ensure that it is correct.
400	BCS.4004029	Query failed.	Query failed.	Check the input parameters and try again.
400	BCS.4006005	Failed to query the BCS service because the etcd connection may be abnormal.	Failed to query the BCS service because the etcd connection may be abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006007	Failed to submit the request because no EIP is bound to the CCE cluster or the EIP fails to be queried.	Failed to submit the request because no EIP is bound to the CCE cluster or the EIP fails to be queried.	Log in to the CCE console. Choose **Resource Management** > **Nodes**. Locate the node corresponding to the BCS service and check whether the EIP is bound. For a private blockchain, at least one node must be bound with an EIP. For a consortium blockchain, all nodes must be bound with EIPs. Perform the following steps to bind an EIP: On the node details page, click the node name. On the displayed ECS console, click the ECS name. On the ECS details page, click the **EIPs** tab. Click **Bind EIP**. The **Bind EIP** dialog box is displayed.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006009	Failed to download the SDK configuration because the etcd connection or the CCE cluster status is abnormal.	Failed to download the SDK configuration because the etcd connection or the CCE cluster status is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service, and check whether the cluster status is normal and whether the cluster has available nodes. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006012	Failed to create the channel because the etcd connection or the CCE cluster status is abnormal.	Failed to create the channel because the etcd connection or the CCE cluster status is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service, and check whether the cluster status is normal and whether the cluster has available nodes. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006013	Failed to update the BCS service instance information because the etcd connection is abnormal.	Failed to update the BCS service instance information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006017	Failed to upgrade the BCS service because the etcd connection or the CCE cluster status is abnormal.	Failed to upgrade the BCS service because the etcd connection or the CCE cluster status is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service, and check whether the cluster status is normal and whether the cluster has available nodes. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006018	Failed to query data because the etcd connection is abnormal.	Failed to query data because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006019	Failed to change the password because the etcd connection, the CCE configuration update API, or the cluster status is abnormal.	Failed to change the password because the etcd connection, the CCE configuration update API, or the cluster status is abnormal.	Log in to the CCE console and check whether CCE is normal. If CCE is normal, choose **Resource Management** > **Clusters**. Locate the cluster corresponding to the BCS service and check whether the cluster status is normal. If the cluster is abnormal, try again after the cluster is restored. Alternatively, use a normal cluster to create another BCS service. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006025	Invitation failed because the tenant has already been invited or the etcd connection is abnormal.	Invitation failed because the tenant has already been invited or the etcd connection is abnormal.	Go to the **Member Management** page on the BCS console to check whether the tenant invitation information exists. If the invitation exists, the tenant has been invited and cannot be invited again. Wait for the invited tenant to reply. If the invitation does not exist, the etcd connection may be abnormal. In this case, proceed to the next step. Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006026	Some invitations failed because the etcd connection is abnormal.	Some invitations failed because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006027	Failed to list the members because the etcd connection is abnormal.	Failed to list the members because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006032	Failed to download the SDK configuration. The OrderEIP is empty because the etcd connection may be abnormal.	Failed to download the SDK configuration. The OrderEIP is empty because the etcd connection may be abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006034	Failed to obtain the notification information because the etcd connection is abnormal.	Failed to obtain the notification information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006036	Failed to generate the organization certificate because the etcd connection is abnormal.	Failed to generate the organization certificate because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006037	Failed to send the event information because the AOM API or the network connection is abnormal.	Failed to send the event information because the AOM API or the network connection is abnormal.	Log in to the AOM console to check whether AOM is normal and whether related information is reported. If AOM is abnormal, events can be reported only after AOM is restored. If AOM APIs are normal, the network connection may not be stable. Try again later.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4006038	Failed to update the notification information because the etcd connection is abnormal.	Failed to update the notification information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006039	Failed to submit the request because the cluster is in use.	Failed to submit the request because the cluster is in use.	Select an unused cluster and try again.
400	BCS.4006040	Failed to save the member information because the etcd connection is abnormal.	Failed to save the member information because the etcd connection is abnormal.	Rectify the etcd fault by referring to the *BCS Troubleshooting* document and try again after etcd is normal.
400	BCS.4006041	Failed to delete the notification because finished notifications cannot be deleted.	Failed to delete the notification because finished notifications cannot be deleted.	Do not delete finished notifications.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4009100	System error. Check the system or network status.	System error. Check the system or network status.	Switch from the BCS console to other service consoles, such as the CCE console. If multiple services are abnormal, the fault is caused by a common component fault in the environment. After the fault is rectified, try again. If only BCS is abnormal, the BCS backend node may be faulty. In this case, rectify the fault by following the instructions provided in "BCS Node Fault" in the *BCS Emergency Plan*.
400	BCS.4009101	The request URL does not exist. Check whether the URL has been registered.	The request URL does not exist. Check whether the URL has been registered.	Log in to the API registration page on DMK. Find BCS APIs, and check whether the API for performing the specified operation exists in the list. If it does not exist, register the API as instructed by the API Gateway document contained in the BCS installation package.

Status Code	Error Codes	Error Message	Description	Solution
400	BCS.4009102	Internal system error. Check the system status or network status.	Internal system error. Check the system status or network status.	Switch from the BCS console to other service consoles, such as the CCE console. If multiple services are abnormal, the fault is caused by a common component fault in the environment. After the fault is rectified, try again. If only BCS is abnormal, the BCS backend node may be faulty. In this case, rectify the fault by following the instructions provided in "BCS Node Fault" in the *BCS Emergency Plan*.
400	BCS.4009103	The maximum number (5) of invited members in the channel has been reached.	The maximum number (5) of invited members in the channel has been reached.	Do not exceed the maximum number of members that can be invited to each channel.
401	BCS.4010401	Permission verification failed.	Permission verification failed.	Configure the permissions by following the instructions provided in *BCS User Guide* > *Service Deployment* > *Prerequisites*.

Status Code	Error Codes	Error Message	Description	Solution
403	BCS.4030403	Insufficient permissions.	Insufficient permissions.	Configure the permissions by following the instructions provided in *BCS User Guide* > *Service Deployment* > *Prerequisites*.

