Blockchain Service

User Guide

Date **2021-01-15**

Contents

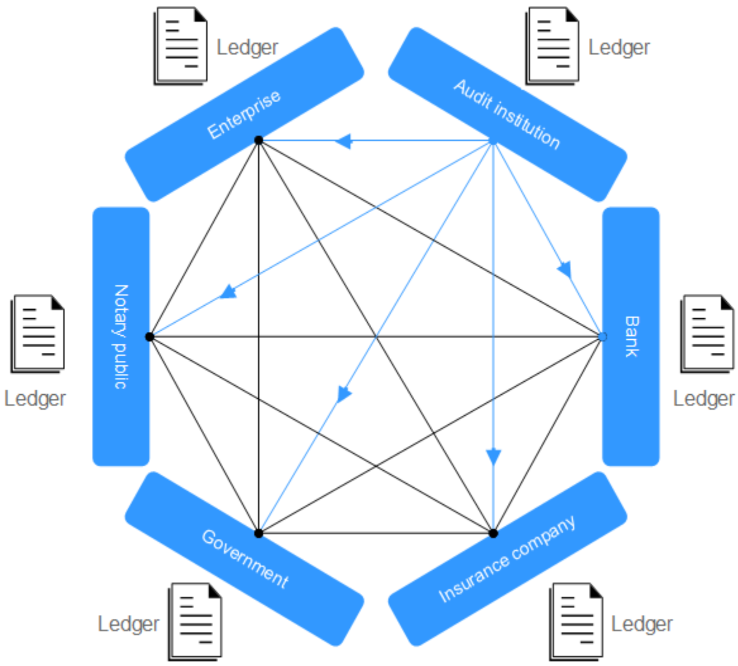
1 Service Overview

1.1 What Is Blockchain?

Traditional Business Network

In a traditional business network, participants such as business organizations, governments, and financial institutions do not have a shared system and separately maintain their own data. When a transaction is conducted, both parties modify their own ledgers and maintain the ledgers locally. This results in a network architecture shown in **Figure 1-1**.

Figure 1-1 Traditional business network architecture



Such a network faces the challenges of low efficiency, high cost, and being subject to attacks due to the following facts:

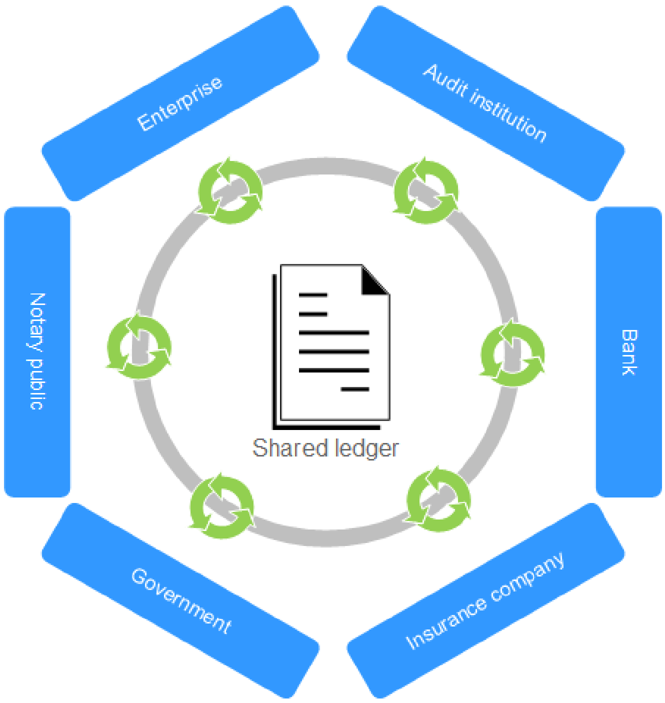
- Each participant maintains its own ledger. The transaction information is not transparently shared among participants, and it is not easy to discover data tampering, if any.
- If a transaction involves multiple parties, additional workload and costs are needed to reconcile disparate ledgers for data consistency.
- Data is scattered among participants, resulting in low efficiency of the overall business process.
- The network relies on a single or multiple central systems. Once the central systems experience fraud, attacks, or errors, operation of the entire network will become chaotic.

Basics of Blockchain

In a narrow sense, a blockchain is data blocks chained based on the block generation time and uses cryptography to ensure that distributed ledgers cannot be tampered with or forged. In a broad sense, the blockchain is a distributed architecture and computing mode that uses the blockchain data structure to verify and store data, distributed consensus algorithms to generate and update data, cryptography to ensure data transmission and access security, and smart contracts compiled by automated script code for programming and data operation.

Blockchain uses a set of technologies including shared ledgers, consensus algorithms, security and privacy protection, and smart contracts. It features multi-centralization, consensus and trust, immutability, and traceability. In a blockchain system, all participants share ledgers, which can solve the challenges in traditional business networks. **Figure 1-2** shows the architecture of a blockchain system.

Figure 1-2 Blockchain systems architecture



All participants in a business network share the ledger and update all the copies of the ledger upon each transaction.

Cryptography algorithms ensure that participants have access only to the ledger content related to them, thus ensuring transaction security.

Transaction-related contract clauses are embedded into the transaction database to comprise smart contracts, which are automatically executed when the business conditions are met.

Consensus algorithms ensure that transactions are validated by all involved parties and requirements of supervision and audit are met.

Benefits of Blockchain

Time saving: Shortens the time that a transaction takes from days to real time or quasi real time.

Reduced costs: Reduces extra costs and the participation of third parties required to ensure data consistency.

Lower risks: Precludes the possibility of tampering to reduce risks of frauds and cybercrimes.

Stronger trust: Builds up trust between transaction participants using shared ledgers, processes, and records.

1.2 What Is BCS?

Blockchain Service (BCS) is a blockchain platform for enterprises and developers. BCS helps you quickly deploy, manage, and maintain blockchain networks, lowering the threshold for using blockchain. In this way, you can focus on the development and innovation of your own business to quickly implement business using blockchain.

- **Infrastructure layer**

  The infrastructure layer offers underlying resources required for creating a blockchain network, including resources on nodes used to compute and store data in the network.
- **Blockchain platform**

  The blockchain service platform provides service management, channel management, and member management, among other modules. It helps you quickly create, manage, and efficiently maintain an enterprise-grade blockchain system for upper-layer applications.
- **Service application layer**

  BCS can be used in multiple scenarios of various industries, such as supply chain finance and tracing, digital assets, and notarization for crowdfunding. Industry-specific applications connect to the blockchain platform to ensure data reliability and security.
- **Security management**

  The cloud security system and innovative cryptography algorithms provide comprehensive security assurance for blockchain nodes, ledgers, smart contracts, and upper-layer applications.

1.3 Advantages

Open and Easy to Use

Building an enterprise-grade distributed blockchain network is not easy. It requires not only in-depth knowledge of blockchain but also complex design and configuration, which is error-prone and costly.

- BCS can help enterprises deploy blockchain networks within only 5 minutes, reducing the development and deployment costs by as much as 80%.
- BCS hosts functions of full-lifecycle management and GUI-based smart contract coding, commissioning, and deployment. Customers using BCS can focus on the innovation and development of their own service applications.

Flexible and Efficient

- BCS supports multiple efficient consensus algorithms and deeply optimizes existing algorithms to achieve balance between security and efficiency.
- Consensus within seconds (over 10,000 TPS) meets service performance requirements.
- Blockchain ledgers are stored in the efficient elastic storage files, satisfying the demand of fast storing massive amount of user data.
- Nodes of multiple roles and members can dynamically join or quit consortium blockchains.

Cost-Effective

- Functions such as service hibernation and waking at any time and data aging on orderers reduce costs.
- BCS is connected to the Application Operations Management (AOM) service of for comprehensive O&M on BCS services, including system status, performance, and transaction monitoring, maintenance, and alarming to reduce O&M costs.
- The peer scaling function allows auto scaling of peers on demand, greatly improving the cost-effectiveness.

Security and Privacy

Comprehensive approach to blockchain security:

- The security system ensures stable and secure running of blockchains.
- The Hyperledger-assured security system prevents data tampering and protects privacy by means of certificate management and the blockchain structure of data.
- Innovative algorithms such as homomorphic encryption and zero-knowledge proofs provide further privacy protection.
- Chinese cryptographic algorithms are used for encryption and decryption.

Trusted Cooperation

BCS provides the Trusted Computing Platform to facilitate trusted cooperation between multiple parties. This platform has the following core features:

- Decentralized identity (DID) management, which is in compliance with the W3C DID and W3C VC standards. This feature lowers the threshold of trust and improves cooperation efficiency.
- Blockchain-based, trusted data sharing, which ensures trusted data flow between multiple parities, breaks data silos, and realizes data value.
- Confidential computing, which is based on blockchain, Trusted Execution Environment (TEE), and federated learning technologies. The raw data can be computed without being revealed, ensuring data privacy.

1.4 Functions

BCS provides the following functions to help you quickly deploy blockchains featuring security, high efficiency, and cost-effectiveness.

Service Deployment

You can create resources when deploying a blockchain system, without a need to prepare resources required by the system in advance.

- The deployment time is reduced from days to minutes, and the blockchain network configuration is completed during deployment.
- Underlying technological details are masked. You do not need to care about the underlying technology implementation and platform construction.
- You can create a cross-region consortium blockchain or a private blockchain.

Ledger Storage

File database (GoLevelDB) and NoSQL (CouchDB) are available for ledger storage.

- File database: Historical transaction data is stored in the blockchain, and status data is stored in the LevelDB.
- NoSQL: A CouchDB database stores transaction and status data.

Consensus Algorithms

BCS supports multiple consensus algorithms for diverse scenarios.

- Solo: A simple consensus algorithm. In a Solo ordering service, only one orderer is available. Therefore, Solo does not support fault tolerance but features quick startup and resource saving. It is recommended for testing.
- Fast Byzantine consensus algorithm (FBFT): A highly available consensus algorithm with superb performance. It requires at least four orderers and tolerates faults at a maximum of (N – 1)/3 orderers, where N indicates the total number of orderers. It is recommended for production environment.

- Kafka (crash fault tolerant): A high-speed consensus algorithm, which tolerates crash faults on just under half of all orderers. It is recommended for production environment.
- Raft (crash fault tolerant): A high-speed consensus algorithm, which tolerates crash faults on just under half of all orderers. It is recommended for production environment.

Consortium Member and Organization Management

- A consortium initiator can dynamically invite other tenants to conveniently and quickly set up a consortium blockchain.
- Peer organizations can be dynamically added to a BCS service to avoid impact of insufficient peer organizations configured during service deployment.

Auto Scaling of Nodes

Peers can be scaled out dynamically based on user requirements, which does not require system reboot.

Chaincode Management

You can manage chaincodes on the graphical user interface (GUI) throughout the entire chaincode lifecycle, including installation, instantiation, and upgrade.

Block Browser

You can query blockchain information required for maintenance in the blockchain browser. The information includes the block quantity, transaction quantity, block details, transaction details, performance, peer statuses, block list, and transaction list.

Privacy Protection

Privacy is ensured within a channel because different members in a channel can have different access permissions. For example, member A can have the permissions to access certain data, but member B, who does not have relevant permissions, cannot access the specified data.

This is different from the privacy ensured through channel isolation because a channel isolates data for members in the channel.

Monitoring and O&M

BCS connects to the monitoring platform to monitor data and resources in real time and generate alarms and notifications when necessary.

- Automated O&M: BCS actively upgrades the underlying blockchain platform and updates patches to seamlessly integrate with the O&M system.
- Enterprise-grade monitoring: Multi-dimensional monitoring is performed on clusters 24/7, and user-defined alarms can be reported through multiple channels.

Inter-Blockchain Data Interaction

- Arbitration is introduced for inter-blockchain transaction results. The blockchain data structure is used to manage the inter-blockchain transaction results, ensuring atomicity of the transactions.
- Arbitration nodes only manage the verification results of inter-blockchain transactions and does not touch original transaction data, ensuring independency and security of the transactions.
- Behavior consistency between parties in inter-blockchain transactions is verified, ensuring consistency of the transaction information during distribution (such as in asset transfer).

NOTE

Currently, only the enterprise edition support inter-blockchain data interaction. To use this function, contact the administrator.

1.5 Solutions

1.5.1 Transactions Between Corporation Subsidiaries

BCS allows for a collaboration consortium composed of subsidiaries and audit organizations of a multinational corporation for inter-subsidiary transactions. Transaction parties within the consortium can develop trust and eliminate reconciliation and discrepancies, which offers end-to-end audit support.

Industry Status Quo and Pain Points

- **Lack of trust between corporation subsidiaries**

  Transaction parties do not fully trust in each other for ownership and fund transfer during contract execution and transactions.
- **Delayed financial settlement**

  Reconciliation of internal transactions requires a large amount of manpower and long time. The discrepancy in reconciliation may lead to delayed settlement and report issuance.
- **Low efficiency and high cost**

  Internal reconciliation is time-consuming and requires a large number of financial personnel's efforts. However, the reconciliation result may still be incorrect, and it is hard to perform supervision.
- **No simple method of data sharing**

  The financial data of corporation subsidiaries is distributed in different types of enterprise resource planning systems (ERPs), which are not integrated or connected.
- **Regulators lacking trust in corporations**

  A multinational corporation must keep data for many years (usually 10 or more years) and provide evidence to external auditors or authorities, demonstrating that data sources are trustworthy and the data has not been tampered with.

- **Regulation compliance issues**

  Inter-subsidiary transfer pricing and complex transactions may cause tax base erosion and profit shifting (BEPS) and may result in financial statement restatements.

Solution Architecture

The BCS-based inter-subsidiary transaction solution has the following features:

- **Unified ledger**

  Tamper-proof, consistent business transaction records are traceable, eliminating the necessity of reconciliation and meeting audit requirements.

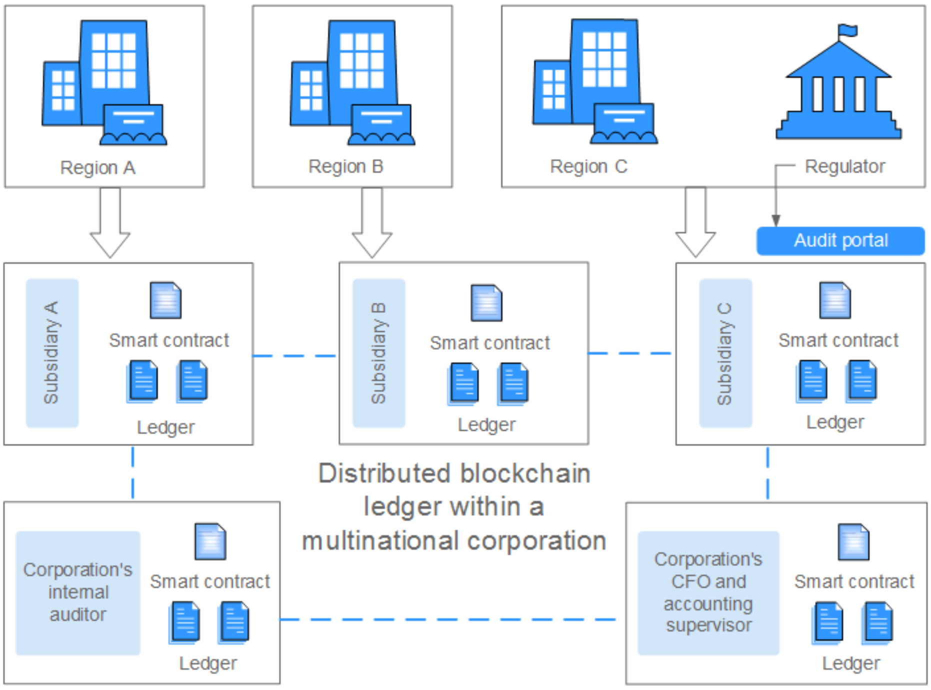
- **Digital assets**

  Tokens are used to record the transaction assets and rights to realize the life-cycle management of digital assets.

- **Smart contract fulfillment**

  Automated fulfillment ensures the fairness of transactions based on the contract terms and conditions.

Figure 1-3 Solution architecture



Solution Highlights

- Ensuring consistency of inter-subsidiary transaction records and the balance of accounting without the need for reconciliation
- Using tokens to track goods such as the statuses, physical locations, and ownership changes and strictly adhering to the contract clauses to carry out transactions, which improves the trust between transaction parties
- Simplifying and normalizing the inter-subsidiary supply chain processes

- Supporting transactions that involve different systems
- Providing end-to-end traceable and immutable information for internal and external audits

1.5.2 Supply Chain Logistics

Manufacturers, warehousing institutes, logistics providers, and customers can use BCS to comprise collaboration consortia and use IoT technologies to record all the logistics information of goods, including production, warehousing, line haul transportation, reselling, and local logistics. The consortia break down information silos, improves circulation of information, and build trust between parties.

Industry Status Quo and Pain Points

- **Disadvantage of using paper documents**

  Many phases of logistics still involve manual operations and paper documents. This causes long duration of the process, high costs, slow reconciliation, and risks of document losses or damage. The cost on maintaining and transferring documents accounts for 1/5 of the total logistics cost.
- **Low efficiency**

  Participants in a supply chain have their own information systems, independent from each other. There is no unified standard or tracking system. It is difficult for them to collaborate effectively.
- **Long duration**

  Electronic information can be easily tampered with. Therefore, paper documents are used as the only type of proof for settlement, but extend the accounting period and the carriers' average collection period of receivables.
- **Difficult financing**

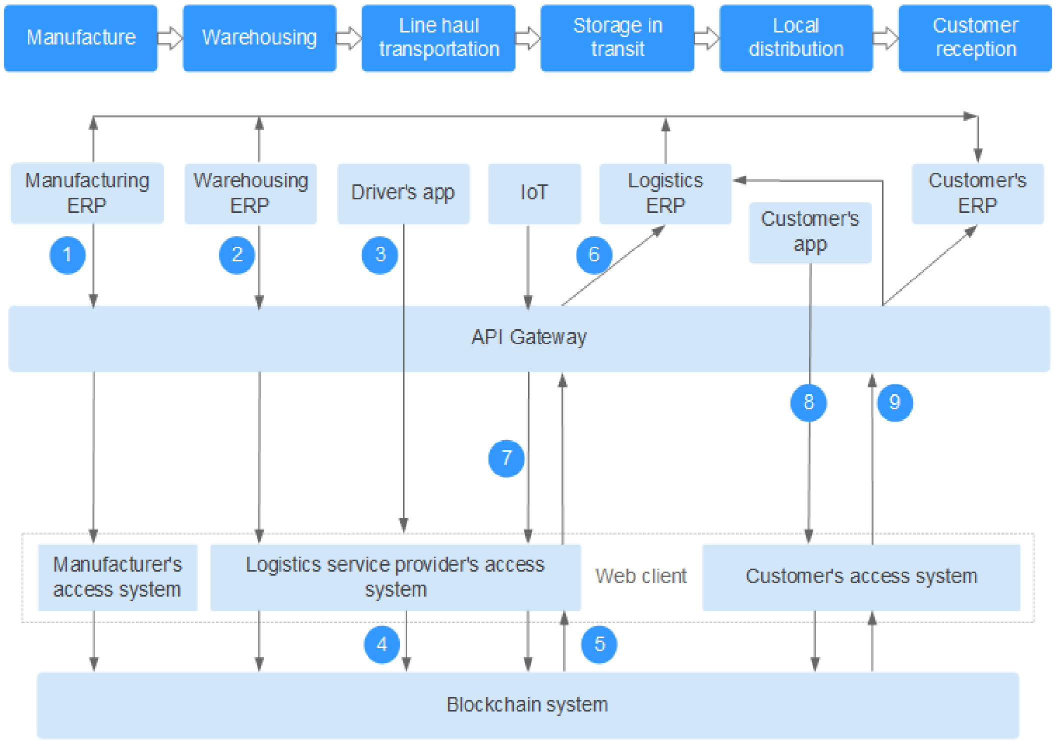
  Most carriers are small- and medium-sized enterprises, lacking credit records, scores, or credibility. Financing is difficult and requires high costs.

Solution Architecture

The supply chain logistics solution provided by BCS can be combined with the IT information systems of logistics participants to achieve the following:

- Jointly maintain unified ledgers, which store immutable and traceable goods transfer records to meet audit and tracing requirements.
- Provide common APIs for participants' IT systems to access BCS and input data, which cannot be tampered with. In this way, participants establish their credibility and trust in each other.
- Automatically store the geo-fence information reported by the driver's app to show in real time when, where, and by whom goods are processed.
- Fulfill smart contracts to automatically perform signing, settlement, and calculation to obtain the performance data, which is considered fair due to the automation.

Figure 1-4 Solution Architecture



Procedure:
1. Goods delivery information is sent to the blockchain through the access system.
2. Goods reception and delivery information is sent to the blockchain through the access system.
3. The driver collects goods information by scanning.
4. Logistics information is sent to the blockchain through the access system.
5. The blockchain system confirms that the received information has been stored.
6. The information is sent to the IT system through the API gateway.
7. The GPS data of trucks is sent to the blockchain through the access system.
8. The customer reception information is sent to the blockchain through the access system.
9. The blockchain system confirms that the received information has been stored and sent to the ERP's of the logistics service provider and manufacturer.

Solution Highlights

- **Reduced errors**

  Distributed, shared ledgers greatly improve the traceability and transparency of the supply chain and effectively reduce or eliminate changes of fraud and errors.

- **Increased efficiency**

  Electronic proofs of delivery (PODs) are used instead of paper documents to reduce the delay caused by paper works, and smart contracts allows automatic settlement to improve efficiency.

- **Lower costs**

  Quick settlement, automatic order reception, and goods tracking significantly lower the logistics costs of all the involved parties.

- **Transparent audit**

  Immutability of distributed ledgers and non-repudiation of signatures allow for quick discovery of problems in supply chain logistics.

- **Trust**

In addition to transparent rules and automated settlement, the blockchain technology enables end-to-end tracking of goods all the way through production and transparent to final reception. These mechanisms greatly improve the trust between consumers and partners in the supply chain.

1.5.3 Healthcare

BCS helps healthcare institutions, third-party organizations, and supervision departments to form a collaboration consortium. Healthcare information silos are broken down using electronic medical records that cannot be tampered with and can protect privacy. This builds trust between doctors and patients and provides comprehensive health and medical care information for telemedicine and referral.

Industry Status Quo and Pain Points

- **Insecure data**

  Most healthcare data is stored in the data center. If a natural disaster or hacking occurs patients' electronic medical records stored in the data center may be lost.

- **Information silos**

  There is no appropriate mechanism for mutual trust and data sharing between healthcare institutions, which leads to information silos and makes it difficult to obtain complete and comprehensive data. Data may be modified casually when shared and therefore, is considered unreliable.

- **Repeated medical treatment**

  A new electronic medical record is stored each time a patient goes to a different hospital due to a lack of data exchange or sharing channels between hospitals and other healthcare institutions. In this situation, there is no complete medical record for the patient, who may need to experience repetitive health checks. This leads to a waste of time, money, and healthcare resources.

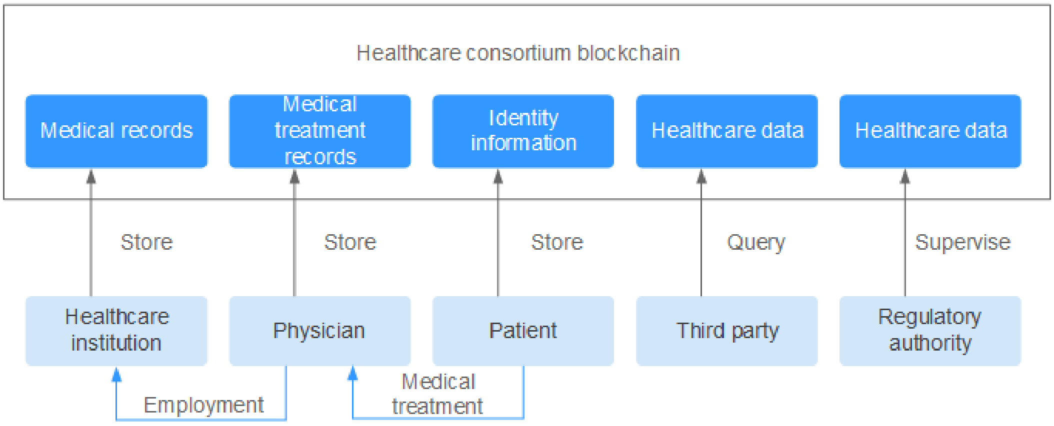
- **No access to personal medical data**

  Patients have no access to their medical care data, which is stored in the hospital systems. This affects medical treatment and health management.

Solution Architecture

A healthcare consortium blockchain is built, comprising healthcare institutions, third parties, physicians, patients, and regulators based on electronic medical records (EMRs). The medical and healthcare data is stored in the blockchain and offered to patients or for scientific research, with security and privacy protected by using encryption and smart contract-based authorization mechanisms.

Figure 1-5 Solution Architecture



Solution Highlights

- **Information silos broken down**

  The healthcare consortium blockchain connects information systems of healthcare institutions, so that regional inspection as well as ultrasound and radiological examination results can be securely exchanged for online healthcare, two-way referral, and remote consultation.

- **Immutable medical data**

  The EMRs, physicians' diagnosis process and results, medical record query histories, and patient identity information are stored in blockchains to ensure that they cannot be tampered with and can be traced. This can reduce medical disputes and construct a harmonious healthcare environment.

- **Protected privacy and right to know**

  Encryption and smart contract-based authorization mechanisms offers patients access to their own healthcare data while protecting their privacy. Others can access the data only when authorized.

- **Quick and effective supervision**

  Regulatory authorities can use the data on blockchains to effectively prevent healthcare treatment that violates regulations, reducing medical disputes.

1.6 Key Concepts

Blockchain

In a narrow sense, a blockchain is a list of data records (called blocks) linked in chronological order using cryptography and a distributed ledger to prevent data tampering and forging. In a broad sense, the blockchain technology is a new distributed infrastructure and computing paradigm that uses the blockchain data structure to verify and store data, distributed node consensus algorithms to generate and update data, cryptography to ensure security of data transmission and access, and smart contracts formed by automated scripts to implement programming and operate data.

Distributed Ledger

A distributed ledger is a database shared, replicated, and synchronized among network members. It records transactions between network participants, such as exchange of assets and data. Use of a distributed ledger eliminates the time and expenditure of ledger reconciliation. Any reference to ledgers in BCS documents means distributed ledgers.

- Decentralized and trustless: Data copies are stored on nodes. No central node or a third-party organization is responsible for data control.
- Collectively maintaining data consistency: Each participant uses a public key as its identity. Nodes independently check the data validity and collectively determine the data to be written to the ledger, by consensus.
- Reliable data, difficult to be tampered with: Data is stored in blocks. Each node stores all blocks. Data access permissions can be customized. Block chaining prevents data tampering.

Smart Contract

A smart contract, also called a chaincode, is a code logic that runs on a blockchain and is automatically executed under a specific condition. It is an important method for a user to implement service logic when using a blockchain. Thanks to the blockchain features, the execution results of smart contracts are reliable and cannot be forged or tampered with.

- Cheating is prevented. Smart contracts are automatically triggered when conditions are met. Execution results are verified independently.
- Results cannot be modified because the data is stored in the blockchain.
- Contract content is reliable because it is stored in the blockchain.
- Privacy is protected. Only specified participants can obtain contract content and data.

Peer

Peers are network nodes that maintain ledgers. One or more peers form a peer organization.

Orderer

Orderers are nodes that order transactions into a block.

Channel

A channel isolates the ledger data of a transaction from other transaction data in a consortium blockchain to ensure confidentiality. Each channel can be considered as a sub-blockchain and corresponds to a specific ledger. The ledger in a channel is invisible to other channels.

Distributed Consensus

A majority of independent participants in a system need to achieve consensus on a transaction or operation, for example, verification of double-spending transactions, verification of transaction validity, and the decision on whether to write verified data to the existing ledger.

Hash Algorithm

A hash value of a digital content segment can be used to verify data integrity. Any minor modification to digital content leads to a significant change in the hash value. A qualified hash algorithm can be used to easily obtain a hash value from digital content, but it is almost impossible to calculate the original digital content by using a hash value.

Organization

A channel includes multiple members (organizations). If identity certificates of two entities on the blockchain network can be traced back to a same Root CA, the two entities belong to a same organization.

1.7 Restrictions and Quotas

To use a BCS service, you need to create Cloud Cluster Engine (CCE) clusters, bind elastic IP addresses (EIPs) to servers, create a Scalable File Service (SFS) file system, deploy a BCS service, and build a blockchain application.

BCS has quota limitations. A maximum of five BCS services can be created. Service editions have different specifications. For details about the specifications of each edition, see **Edition Specifications**.

1.8 Edition Specifications

BCS provides professional and enterprise editions with different specifications.

Table 1-1 Comparison between editions

Item		Professional Edition	Enterprise Edition
Applicable scenario		Small-scale commercial use	Medium-scale commercial use
Consortium blockchain		Supported	Supported
Peak transaction performance		≤ 500 TPS	≤ 2000 TPS
Consensus algorithms	Solo	Supported	Supported
	Raft	Supported	Supported
	Kafka (CFT)	Supported	Supported
	FBFT	Not supported	Supported

Item		Professional Edition	Enterprise Edition
Node management	Maximum number of peer organizations	2	5
	Maximum number of peers	2	2
	Maximum number of orderers	3	4
	Maximum number of channels	2	4
	Automatic recovery from node faults	Supported	Supported
	Node auto scaling	Supported	Supported
	Maximum number of light nodes	Not supported	10
Security functions	ECDSA	Supported	Supported
	Chinese cryptographic algorithms	Not supported	Supported
	Additive homomorphic encryption	Not supported	Supported
	Zero knowledge proof	Not supported	Supported
	Encryption using Intel SGX	Not supported	Not supported
High availability	Invoking smart contracts through RESTful APIs	Supported	Supported
	Common deployment	Supported	Supported
	High-availability deployment	Not supported	Not supported
O&M and monitoring	O&M logging	Supported	Supported

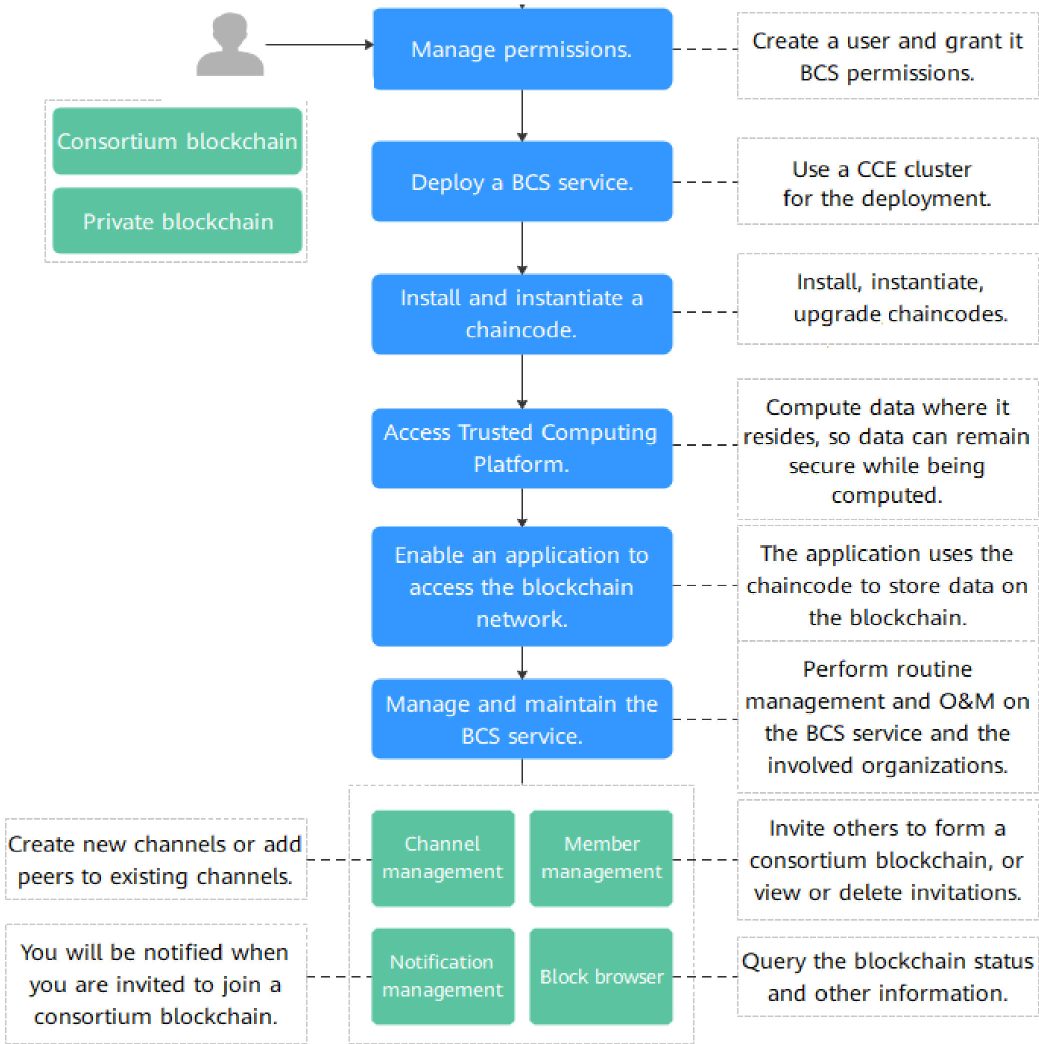
Item		Professional Edition	Enterprise Edition
	Node status monitoring	Supported	Supported
	Status alarming	Supported	Supported
Service support	Named service manager	Not supported	Not supported
	Remote technical support from the R&D team	Not supported	Not supported
	Onsite technical support	Not supported	Not supported

2 User Guide

2.1 Outline of the BCS Usage Process

Blockchain Service (BCS) provides functions such as service deployment, blockchain management, channel management, member management, and notification management. The following figure outlines the BCS usage process.

Figure 2-1 Outline of the BCS usage process



2.2 Service Deployment

2.2.1 Using a CCE Cluster

BCS services can be deployed using Cloud Container Engine (CCE). CCE is a high-performance, high-reliability service through which enterprises can manage containerized applications. CCE supports native Kubernetes applications and tools, allowing you to easily set up a container runtime environment on the cloud. For more information, see *Cloud Container Engine User Guide*. When creating a BCS service, you need to configure basic parameters and network nodes to quickly create and deploy the service.

Deploying a BCS Service

After the environment is ready, perform the following steps to create and deploy a BCS service:

Step 1 Create a BCS service.

- Log in to the BCS console, and click **Create BCS Service** in the upper right corner.
- To enable trusted computing for an existing BCS service, perform the following steps:
  a. Log in to the BCS console.
  b. In the navigation pane on the left, choose **Add-on Management**.
  c. On the **Add-on Repository** tab page, click **Install** in the **tc3-taskserver** card.

NOTE

– To enable trusted computing for a deployed BCS service, ensure that the service has been upgraded to the latest version.
– Click **Task Details** on the **Service Management** page to view service deployment records, and click **View Details** or **Delete** on the right of the records.

Step 2 Configure basic information about the BCS service by referring to **Table 2-1**.

Table 2-1 Configuring the basic information

Parameter	Description	Example Setting
Region	Select the region where the blockchain infrastructure is located. You are advised to select the same region as the service application system.	Retain the default value.
Enterprise Project	Select an existing enterprise project, to which the BCS service will be added. **NOTE** ● If you have not enabled the enterprise management service, this parameter is not displayed. ● To use an existing CCE cluster to deploy a BCS service, you are advised to add the BCS service to the enterprise project of the CCE cluster. If the BCS service and the CCE cluster where the BCS service is deployed belong to different enterprise projects, the BCS service may fail to be used.	Select **default**.
Service Name	A service name can contain 4 to 24 characters, including letters, digits, and hyphens (-). It cannot start with a hyphen (-).	Enter **bcs-wh**.
Edition	BCS provides professional and enterprise editions.	Select **Enterprise**.

Parameter	Description	Example Setting
Blockchain Type	A private blockchain is used only by the tenant that deploys the BCS service. A consortium blockchain can be used by the blockchain initiator and the tenants that the initiator invites to join the consortium.	Select **Private**.
Fabric Version	BCS service version. ● v4.x.x BCS services correspond to Hyperledger Fabric v2.0. ● v3.x.x BCS services correspond to Hyperledger Fabric v1.4.0.	Select **v2.0**.
Consensus Mechanism	The supported mechanisms for blockchain nodes reaching consensus include: SOLO (for testing), fast Byzantine fault tolerance (FBFT), Kafka (crash fault tolerant), and Raft (crash fault tolerant). **NOTE** ● If Raft is selected, a professional or enterprise edition service has three orderers by default. ● Fabric v1.4.0 does not support Raft (CFT), and Fabric v2.0 does not support SOLO and Kafka (CFT).	Select **FBFT**.
Resource Access Initial Password	Password of blockchain administration user admin, ECS user root, or CouchDB database user. It will be used as such a password if **Blockchain Mgmt. Initial Password**, **Initial Password** displayed when **NoSQL (CouchDB)** is selected for **Ledger Storage**, or **Password of Root User** is not set.	-
Confirm Password	Confirm the resource access initial password.	-

Step 3 (Optional) Click **Quick Config** and a BCS service will be automatically created with the specifications listed in **Table 2-2**.

Table 2-2 Default specifications

Item	Professional	Enterprise
Number of CCE cluster nodes	1	2

Item	Professional	Enterprise
CCE node specifications	4 vCPUs | 8 GB	4 vCPUs | 8 GB
	Note: If the default specifications are sold out, other high specifications will be used by default.	
High availability of the CCE cluster	No	No
Storage of the SFS file system	100 GB	100 GB
DMS for Kafka	Assured bandwidth: 100 MB/s; Type: Common I/O; Storage space: 600 GB Note: If the default specifications are sold out, other high specifications will be used by default.	
EIP	Type: Dynamic BGP; Bandwidth: 5 Mbit/s	

Step 4 Click **Next: Configure Resources**. Table 2-3 describes the resource parameters.

Table 2-3 Configuring resources

Parameter	Description	Example Setting
Environment Resources	Use the default environment or customize your environment resources.	Select **Custom**.
Cluster	Cluster where the BCS service will be deployed. You can use an existing CCE cluster or create a new one. **NOTE** CCE clusters of v1.15 or earlier are supported.	Select **Create a new CCE cluster**.
AZ	Select the AZ where the ECS is located.	Select **AZ1**.
ECS Specifications	Specifications of the ECSs in the CCE cluster.	Select the flavor for **4 vCPUs | 8 GB**.
ECS Quantity	Enter the required ECS quantity.	Enter **2**.
High Availability	If you have high requirements on system reliability, create high-availability ECSs.	Select **No**.
VPC	You can create a new virtual private cloud (VPC), select an existing VPC, or let the system automatically create a VPC.	Select **Automatically create VPC**.
Subnet	A subnet provides dedicated network resources that are logically isolated from other networks for network security.	Select **Automatically create subnet**.

Parameter	Description	Example Setting
ECS Login Method	Either a password or key pair can be used to log in to ECSs.	Select **Password**.
Password of Root User	Password of the root user for logging in to ECSs. If you do not enter a password here, the previously specified resource access initial password will be used.	-
Confirm Password	Confirm the ECS login password of the root user.	-
Kafka Premium	Retain the default setting. This parameter is displayed only if **Consensus Mechanism** is set to **Kafka (CFT)**.	A Kafka premium instance will be created automatically.
Kafka Instance AZ	Select an AZ as needed. This parameter is displayed only if **Consensus Mechanism** is set to **Kafka (CFT)**.	Select **AZ1**.
Bandwidth	Select a bandwidth as required. This parameter is displayed only if **Consensus Mechanism** is set to **Kafka (CFT)**.	Select **100 MB/s**.
Storage Space	The storage space is used for storing messages. When creating a topic, you can specify the number of replicas (3 replicas by default). For example, if the required disk size to store the data for the retention period is 100 GB, the disk capacity must be at least: 100 GB x Number of replicas + 100 GB (reserved). This parameter is displayed only if **Consensus Mechanism** is set to **Kafka (CFT)**.	Select Ultra-high I/O and 300 GB.
Use EIP of a CCE Node	● If you select **Yes**, an EIP bound to the cluster will be used as the blockchain network access address. If the cluster is not bound with any EIP, bind an EIP to the cluster first. ● If you select **No**, a private address of the cluster will be used as the blockchain network access address. Ensure that the application can communicate with the internal network of the cluster.	Select **Yes**.

Parameter	Description	Example Setting
EIP Billed By	Specifies whether the bandwidth is charged by fixed bandwidth or by traffic.	Select **Bandwidth**.
EIP Bandwidth	Select a bandwidth as required.	Set it to 5 Mbit/s.

Step 5 Click **Next: Configure Blockchain**. Table 2-4 describes the blockchain parameters.

Table 2-4 Blockchain configurations

Parameter	Description	Example Setting
Blockchain Configuration	Use the default blockchain configurations or customize your own blockchain configurations.	Select **Custom**.
Blockchain Mgmt. Initial Password	Enter the blockchain management initial password. If you do not enter a password here, the previously specified resource access initial password will be used.	-
Confirm Password	Enter the blockchain management initial password again for confirmation.	-
Volume Type	**SFS Turbo** provides low-latency and high-IOPS file storage.	Select **SFS Turbo**.
Peer Organization	Peer organizations to be added to the BCS service.	Add a peer organization named **organization** with 2 peers.
Storage Capacity of Peer Organization	Stores shared distributed ledger, consensus data, and other intermediate data of the blockchain system.	Set it to 100 GB.
Channel Configuration	Channels isolate business in a consortium blockchain. Business participants (some or all of the organizations in a consortium) are channel members. Each channel can be regarded as a sub-chain and corresponds to one distributed ledger.	By default, a channel named **channel** has been created, and the peer organization you just specified has been added to the channel.
Orderer Quantity	Number of nodes that order transactions into blocks in the blockchain network. When the consensus mechanism is Raft (CFT), the number of orderers is 3.	Enter **3**.

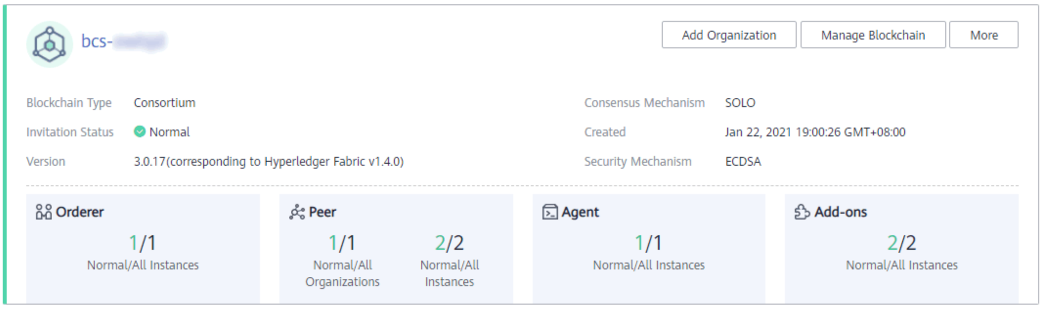
Parameter	Description	Example Setting
Enable Data Aging on Orderers	When the amount of data on an orderer reaches a specified threshold, the system automatically deletes the earliest data to prevent exceptions caused by insufficient storage space. This function can be enabled if **Consensus Mechanism** is set to **Kafka (CFT)**.	Select **No**.
Security Mechanism	Encryption algorithm used to ensure data security. ECDSA and Chinese cryptographic algorithm are supported.	Select **ECDSA**.
Ledger Storage	Multiple types of databases can be used for ledger storage. For details about their differences, see the corresponding tips on the GUI.	Select **File database (GoLevelDB)**.
Configure Block Generation	The configuration of block generation includes the block generation interval, maximum number of transactions in a block, and maximum size of a block. A new block is generated at the specified interval or when the transaction quantity or size of a block reaches the threshold. Configure these parameters based on the transaction frequency and service volume.	Select **No**.
Enable Support for RESTful API	If you need to use RESTful APIs to invoke chaincodes, select **Yes**.	Select **No**.

Parameter	Description	Example Setting
Enabling Trusted Computing	Trusted computing enhances full-lifecycle data security and privacy by using trusted execution environments (TEEs). It helps achieve trusted sharing of data assets, secure multi-party computation, and confidential computing. Trusted computing is not supported by BCS services that use the Chinese cryptographic algorithms. Select **Yes** or **No** as required. ● **Yes**: Enable trusted computing for the service. For details about Trusted Computing Platform, see **Trusted Computing Platform**. ● **No**: Do not enable trusted computing. After the BCS service is created, you can still enable Trusted Computing Platform by installing an add-on. For details, see **Add-on Management**.	Select **No**.

Step 6 Click **Next: Confirm**.

Step 7 Confirm the configurations and click **Submit**.

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



----End

Subsequent Operations (Optional)

You can configure an anti-affinity label for the cluster node where the BCS service is deployed. This label can be used to isolate the service from other applications in the same cluster to ensure normal running of the system.

Step 1 Log in to the CCE console. In the navigation pane, choose **Resource Management** > **Nodes**. The node list is displayed. Choose **Operation** > **Manage Label** in the **Operation** column.

Step 2 Click **Add Label**. Set **Key** to **nodeScope** and **Value** to **userApplication** for the label to be added.

Step 3 Click **OK**. After **Label updated successfully.** is displayed, click **Manage Labels** again. Then you can see the label that you have added.

----End

2.3 Inviting Tenants to a Consortium Blockchain

After creating a consortium blockchain, you can invite tenants to join it.

NOTE

- BCS services corresponding to Fabric v1.4.0 can be upgraded to the version corresponding to Fabric v2.0. If one member in a consortium blockchain has upgraded to Fabric v2.0, all consortium members must also upgrade to v2.0. Otherwise, transactions will fail. For details about upgrading the version, see **Step 3**.
  - BCS v3.x.x corresponds to Hyperledger Fabric v1.4.0.
  - BCS v4.x.x corresponds to Hyperledger Fabric v2.0.

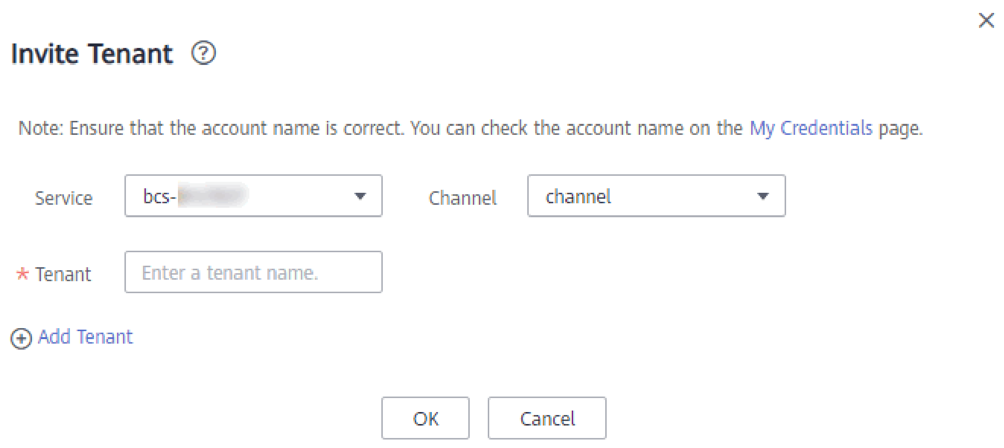
Inviting a Tenant

Step 1 Log in to the BCS console.

Step 2 Choose **Member Management** in the navigation pane on the left. Click **Invite Tenant** in the upper right corner of the page.

Step 3 In the **Invite Tenant** window, select your BCS service and channel, and enter the invited tenant's name.

Figure 2-2 Inviting a tenant



Step 4 (Optional) Click **Add Tenant** to invite multiple tenants.

NOTE

A maximum of 5 tenants can be invited.

Step 5 Click **OK**. An invitation notification is sent to the invited tenant.

----End

Accepting/Declining an Invitation

When you are invited to join a consortium blockchain, you will receive a notification. You can either accept or decline it.

Step 1 Log in to the BCS console.

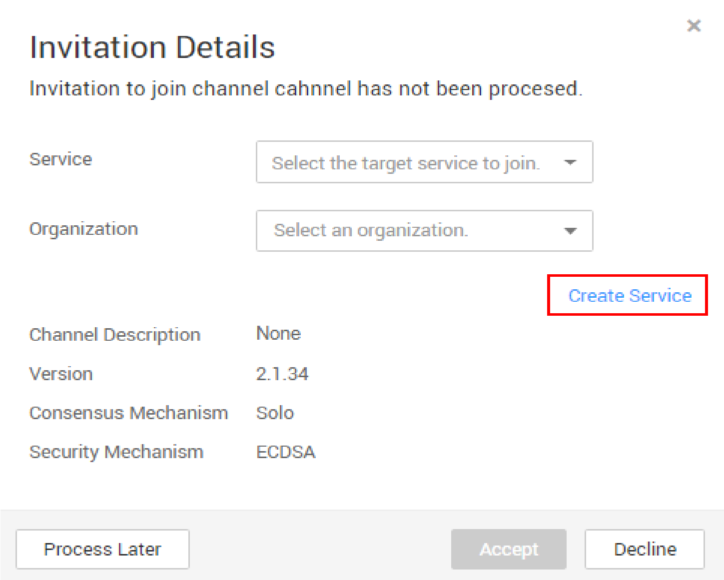
Step 2 Choose **Notification Management** in the navigation pane on the left. On the **Notification Management** page, locate the notification and click **View Details** in the **Operation** column.

- To accept the invitation, select the organization that you want to add to the consortium, and then click **Accept**.
- To decline the invitation, click **Decline**.

NOTE

– Click **Create Service** to create a service before selecting an organization. Otherwise, you cannot join the consortium.

Figure 2-3 Creating a BCS service after receiving an invitation



– For details about how to create a BCS service, see **Service Deployment Using a CCE Cluster**. To successfully join a consortium blockchain, certain parameters of your service must have the same settings as the inviter's BCS service, such as the blockchain type, consensus mechanism, and security mechanism. Therefore, these parameters are dimmed on the service configuration page and cannot be modified.

----End

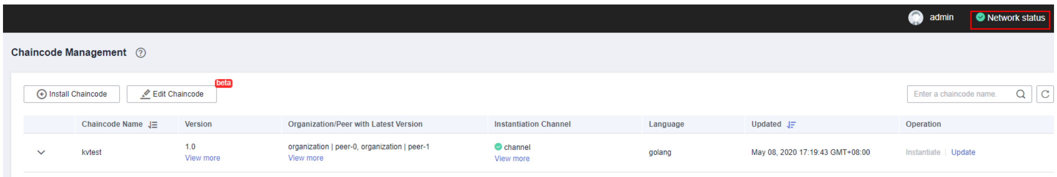
2.4 Blockchain Management

2.4.1 Chaincode Management

You can manage chaincodes on the web, including chaincode installation, instantiation, and update.

Note

If the **Network Status** displayed in the upper right corner of the **Blockchain Management** page is abnormal, do not perform any operations. Wait for a few minutes until the network is recovered.



Installing a Chaincode

Step 1 Log in to the **Blockchain Management** console.

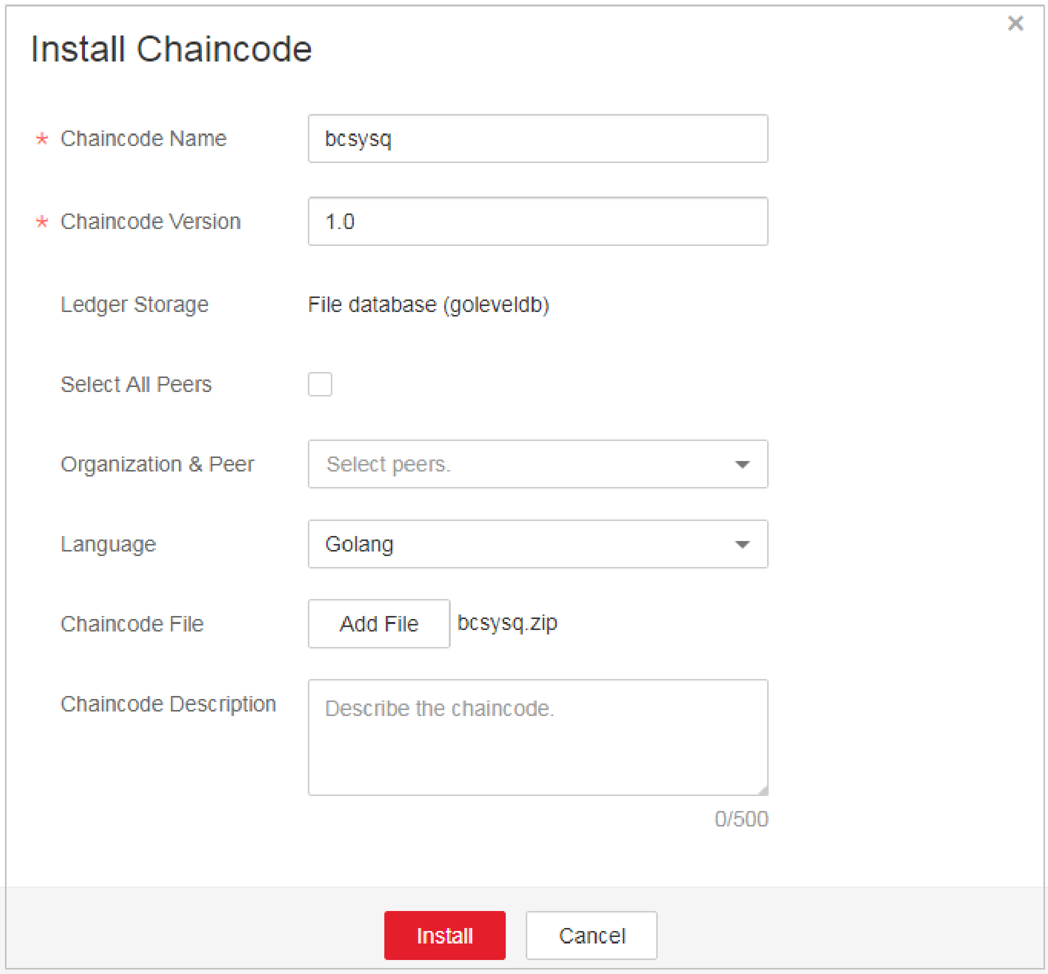
- Perform the following steps to go to the **Blockchain Management** console:
  a. Log in to the BCS console.
  b. Click **Manage Blockchain** in a service card.
  c. Enter the username, password, and verification code, and click **Log In**.

  NOTE

  The username is **admin**, and the initial login password is the password set when you buy the BCS service. For security purposes, change the password periodically.

Step 2 On the **Chaincode Management** page, click **Install Chaincode**.

Step 3 On the **Install Chaincode** dialog box, enter the chaincode name and version number, select the peers where the chaincode is to be installed, select the chaincode programming language, and add the chaincode file, as shown in the following figure.



Step 4 Click **Install**.

----End

Instantiating a Chaincode

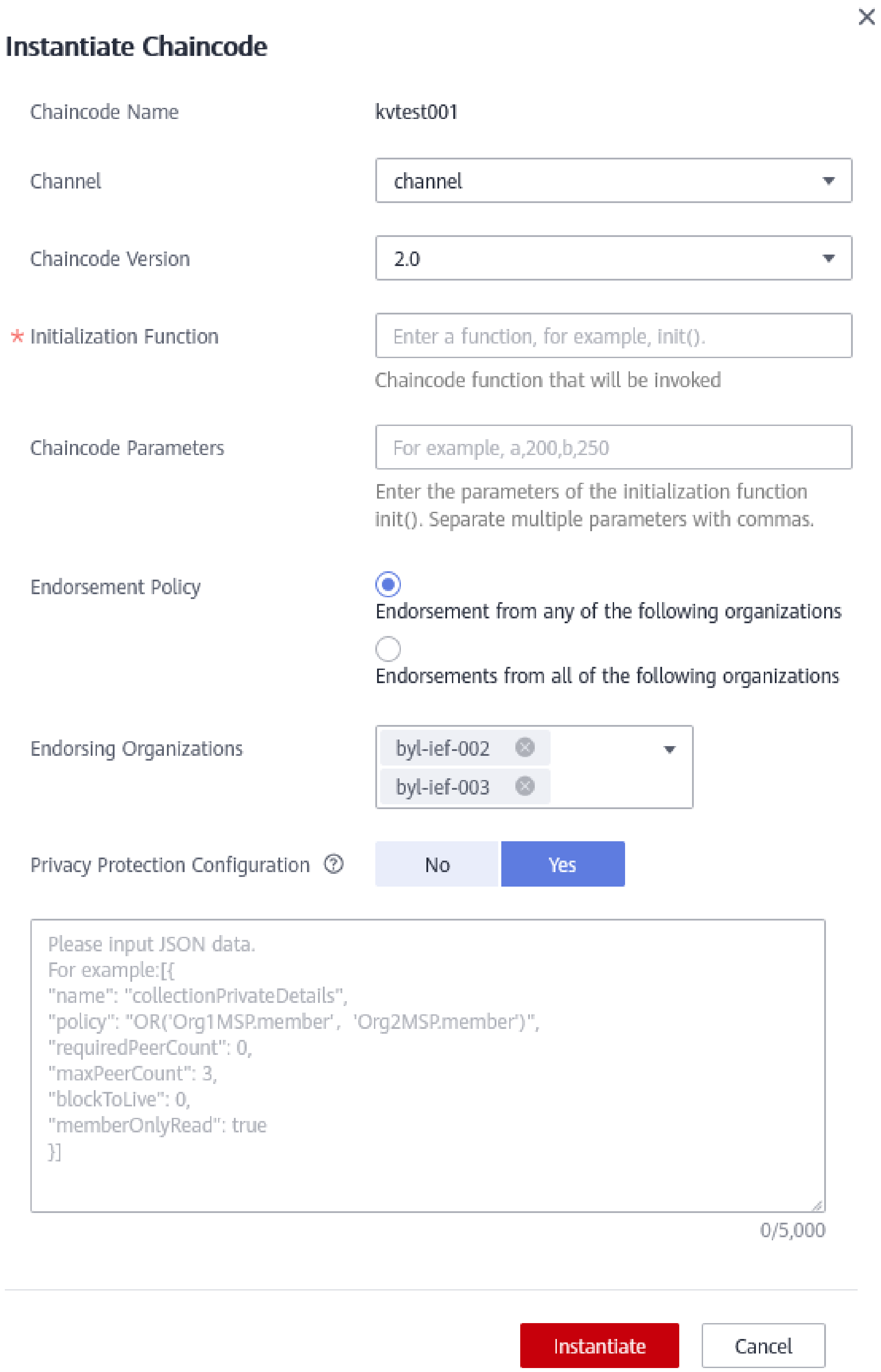
After a chaincode is installed, it must be instantiated on the channel so that the peers can interact with each other using the distributed ledger and the chaincode container.

Before instantiating a chaincode, you need to add the peers to the channel. Otherwise, the chaincode cannot be instantiated.

Step 1 Click **Instantiate** in the **Operation** column of the chaincode list.

Step 2 Specify the channel for instantiation, chaincode version, endorsement policy, endorsing organizations, initialization function and chaincode parameters.

Figure 2-4 Instantiating a chaincode



Step 3 Enter the private data (JSON format) to be protected in the text box below **Privacy Protection**.

If you want to restrict data in a shared channel to certain specified members, use the privacy protection function. Skip this step if privacy protection is not required for your chaincode.

Configure privacy protection by referring to the example and the following parameter description:

- **name**: Name of the collection of private data, for example, collectionPrivateDetails.
- **policy**: Peers allowed to access the data in the collection. In the example, only peers of organizations Org1 and Org2 are allowed to obtain the data in the collection.
- **requiredPeerCount**: Number of endorsing peers to which the private data can be disseminated. In the example, value **0** indicates that there is no endorsing peer.
- **maxPeerCount**: Maximum number of orderers, which is **3** in the example. Multiple orderers can be used for data redundancy. If one orderer is unavailable, other orderers can respond to requests for obtaining the private data.
- **blockToLive**: Maximum number of blocks that the private data can live for. If the number of blocks exceeds the threshold, the private data will be cleared. To keep private data indefinitely, set this parameter to **0**.
- memberOnlyRead: The default value is **true**. The access policy set in **policy** takes effect only when **memberOnlyReadis** is set to **true**.

Example of privacy protection configuration (JSON):

```
[
 {
   "name": "collectionPrivateDetails",
   "policy": "OR('Org1MSP.member','Org2MSP.member')",
   "requiredPeerCount": 0,
   "maxPeerCount": 3,
   "blockToLive":0,
   "memberOnlyRead": true
 }
]
```

This configuration indicates that the chaincode uses a private data collection called collectionPrivateDetails. Only the peers of organizations Org1 and Org2 have access to the data in this collection.

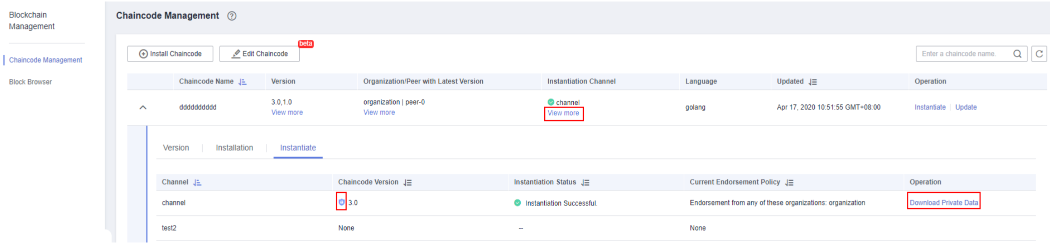
NOTE

The values of **name** and **blockToLive** cannot be modified during subsequent chaincode upgrade. For details, see **Using Private Data in Fabric**.

Step 4 Click **Instantiate**.

If privacy protection is configured, you can click **View More** after the chaincode is successfully instantiated to download the private data and check whether the privacy protection settings are correct.

Figure 2-5 Downloading private data



If chaincode instantiation fails, refer to **Chaincode Instantiation Error Codes** to determine the cause.

----End

Updating a Chaincode

If your chaincode is updated, you need to install and instantiate it again to meet new business requirements.

Step 1 Click **Update** in the **Operation** column of the chaincode list.

Step 2 Fill in the chaincode version, select peers, add a chaincode file, and click **Update**.

Step 3 Instantiate the updated chaincode. For details, see **Instantiating a Chaincode**.

Step 4 (Optional) Click in front of the chaincode name. You can see details about this chaincode, including the versions, and installation and instantiation information.

----End

Chaincode Instantiation Error Codes

Chaincode instantiation may fail due to various causes. When confronted with an instantiation failure, you can refer to the following table to determine the cause.

Table 2-5 Error codes

Error Code	Message
6001	Instantiation timed out.
6999	Unknown error.
6701	Client failed to connect to a peer.
6703	Endorsement signature failed verification.
6704	Failed to pull the ccenv image during chaincode compilation.
6705	Chaincode compilation failed.
6707	Failed to build a chaincode image.
6708	Failed to create a chaincode container.

Error Code	Message
6709	Failed to register the chaincode container.
6710	Client failed to connect to an orderer.
6712	Transaction recording in distributed ledgers failed.
6713	Request error determined by the orderer.
6714	The endorsement policy failed the verification.
6715	Instantiation failed because instantiation of another chaincode has already been started.
6716	Error detected in the init() function parameters.
6717	Error detected in the invoke() function parameters.
6720	Failed to create a chaincode certificate.
6721	Chaincode container startup timed out.
6722	Transaction timed out because init() execution abnormally terminates after startup of the chaincode container.
6723	A chaincode with the same schema has already been instantiated on this channel.
6725	The signature set does not satisfy the endorsement policy.
6726	The instantiation policy failed the verification. Select a peer of an organization that exists in the channel before chaincode instantiation to upgrade the chaincode.
6901	Instantiation failed. The chaincode to be instantiated must contain all the tables in the previously instantiated chaincode.
6902	Instantiation failed. The chaincode to be instantiated must contain all the fields in the previously instantiated chaincode.
6903	Instantiation failed. The chaincode to be instantiated must not contain any changes to the field attributes included in the previously instantiated chaincode.
6904	The schema file of the instantiated chaincode does not exist.
6905	Failed to resolve the schema file.
6906	Insufficient disk space.

2.4.2 Block Browser

You can query blockchain information required for maintenance, including the block quantity, transaction quantity, block details, transaction details, performance, and peer statuses.

Procedure

Step 1 Open the block browser page.

1. Log in to the BCS console.
2. Click **Manage Blockchain** in a service card.
3. Enter the username, password, and verification code, and click **Log In**.
4. Choose **Block Browser** in the navigation pane.

Step 2 Select a channel from the **Channel** drop-down list box. Real-time data is displayed in the lower part of the page.

Step 3 You can view the following data in the block browser.

Table 2-6 Data

Item	Description
Peers	Number of peers in the selected channel
Chaincodes	Number of chaincodes in the selected channel, that is, the number of the chaincode versions
Blocks	Number of generated blocks
Transactions	Number of transactions that have been performed
Block details	Click the **Block List** tab to view the block hash, data hash, and creation time of recent blocks.
Transaction details	● Click the **Transaction List** tab to view the information about recent transactions such as the transaction IDs, creators' MSPs, and creation time. ● Click **View Details** in the **Operation** column of the transaction list to view more details about the transaction.
Performance analysis	The line charts show the trends of performance data, helping you know the performance status. ● Block performance: Click **Block** to view changes in the block quantity. Move the pointer along the curve to view the number of blocks at different time points. ● Transaction performance: Click **Transaction** to view changes in the transaction quantity. Move the pointer along the curve to view the number of transactions at different time points. **NOTE** You can select a time granularity (hours or minutes) in the upper right corner of the chart.

Item	Description
Transaction quantity of organizations	The pie chart shows the percentage of each organization's transactions. **NOTE** Move the pointer on the pie chart to view the transaction quantity and percentage of each organization.
Peer status	You can view the running statuses of all peers in the selected channel to detect exceptions of peers in time.

----End

2.5 Trusted Computing Platform

2.5.1 Overview

Trusted computing ensures the security of blockchains. BCS provides the Trusted Computing Platform to address security and privacy issues in enterprise data cooperation. Data silos can be eliminated through trusted data sharing. Computing is performed where the data resides, so that data can be kept secure while being computed. The Trusted Computing Platform provides the following capabilities:

- Privacy protection: Federated learning and trusted execution environment (TEE) ensure that raw data can be computed without being revealed, ensuring the data privacy of all parties. As a part of the Trusted Computing Platform, TEE enables trusted data exchange, computing, credential management, and token management.
- Full-lifecycle trustworthiness: Trustworthiness is ensured throughout the lifecycle spanning resource registration, identity authentication, data release, computing review, computing scheduling, and asset settlement. Evaluation, auditing, and tracing are driven and documented by blockchains and smart contracts.

The Trusted Computing Platform is suitable for big data bureaus, economy bureaus, large enterprises, research institutions, and medical institutions.

Enabling Trusted Computing

You can enable trusted computing for a BCS service in either of the following ways:

- When creating a BCS service, select **Yes** for **Enable Trusted Computing**. For details, see **Using a CCE Cluster**.
- If you did not enable trusted computing during BCS service creation, you can still enable it by installing an add-on. For details, see **Add-on Management**.

NOTE

Trusted computing is not supported by the following BCS services:

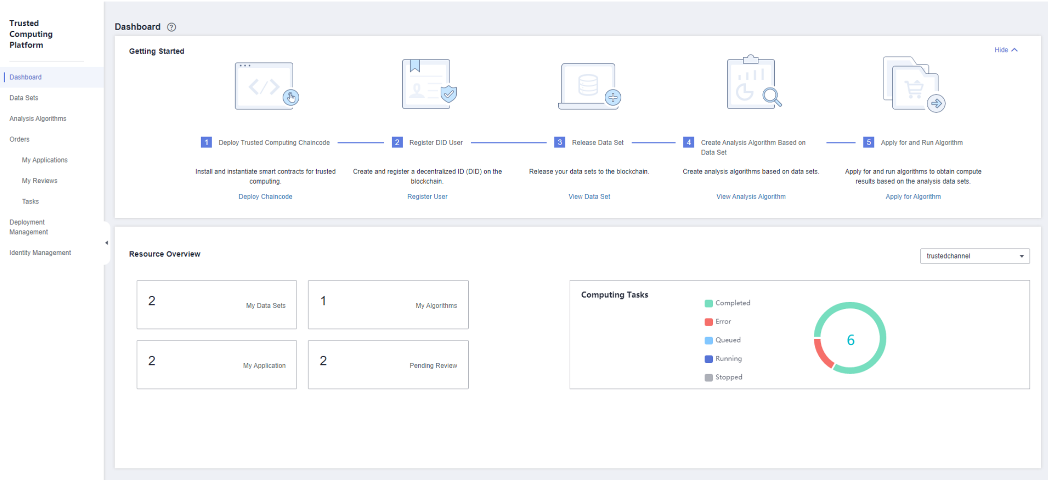
- BCS services that use Chinese cryptographic algorithms

Viewing the Dashboard of Trusted Computing Platform

The **Dashboard** page of the Trusted Computing Platform provides an overview of computing task statuses and resources, and the general procedure for using the trusted computing platform. On the **Dashboard** page, you can perform the following operations in sequence:

Deployment Management > **Identity Management** > **Data Set Management** > **Analysis Algorithm Management** > **Order Management**

Figure 2-6 Dashboard page



2.5.2 Data Set Management

You can upload data sets to the server. After being encrypted, the data sets will be released to the blockchain and can be used by others. You can also apply for data sets that have already been released.

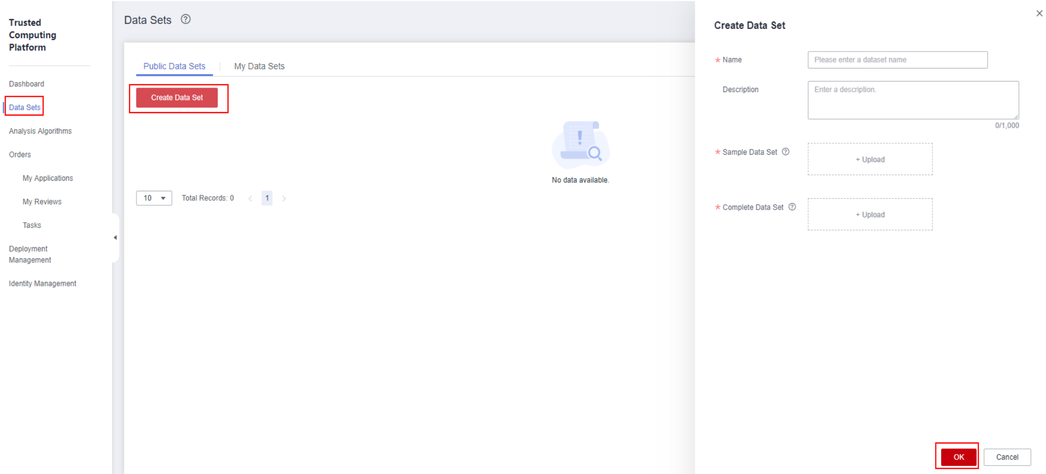
On the **Data Sets** page of Trusted Computing Platform, you can create data sets and apply for public data sets.

Step 1 Log in to Trusted Computing Platform. In the navigation pane on the left, choose **Data Sets**.

Step 2 Click **Create Data Set** in the upper left corner of the **Data Sets** page.

Step 3 Specify the name and description of the data set, and upload a sample data set and the complete data set.

Figure 2-7 Creating a data set



Step 4 Click **OK**.

Step 5 On the **Public Data Sets** tab page, apply for public data sets.

- Click **Apply** in the row containing the data set you want to apply for, and then click **Yes**.
- Click **Download Sample** in the row containing the data set that you want to download to use as a sample data set.

Step 6 On the **My Data Sets** tab page, view your data sets.

- Click **Released** to view the data sets you have released. You can download samples of these data sets or delete these data sets.
- Click **Applied** to view the data sets you have applied for. You can also download samples of these data sets.

----End

2.5.3 Analysis Algorithm Management

You can use an analysis algorithm to compute data of a data set that has already been released to the blockchain. When you apply for an analysis algorithm, you are requesting the right to use the data set on which the algorithm is based. You can compute the data set by using the analysis algorithm.

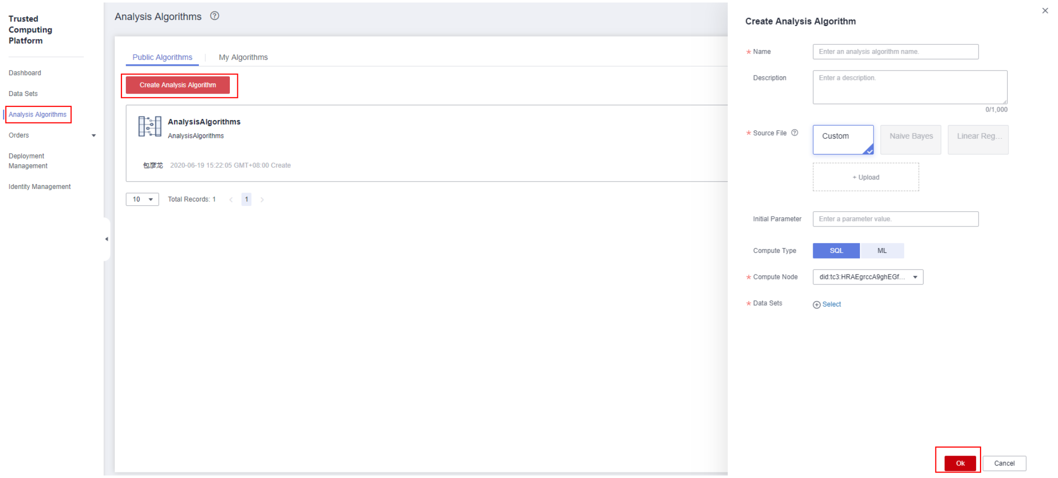
On the **Analysis Algorithm** page of Trusted Computing Platform, you can create and apply for analysis algorithms.

Step 1 Log in to Trusted Computing Platform. In the navigation pane on the left, choose **Analysis Algorithms**.

Step 2 Click **Create Analysis Algorithm** on the **Public Algorithms** or **My Algorithms** tab page.

Step 3 Specify the analysis algorithm name, description, and initial parameter, select or upload a source file, and specify the compute type, compute node, and the data set.

Figure 2-8 Creating an analysis algorithm



Step 4 Click **OK**.

Step 5 On the **Public Algorithms** tab page, apply for public algorithms.

Step 6 On the **My Algorithms** page, view your algorithms.

- Click **Released** to view the algorithms you have released. You can delete these algorithms.
- Click **Applied** to view the algorithms you have applied for.

----End

2.5.3.1 Writing an Analysis Algorithm

You can write your own algorithm scripts in Python and upload them by using PySpark. The computing jobs on computing nodes are submitted through **spark-submit**. You can design your algorithms based on the downloaded data set samples to implement SQL queries, data analysis, machine learning, and more.

The algorithm script needs to accept the following parameters:

- Path of the data file, which cannot be empty. You can configure multiple paths.
- Path for storing computing results, which cannot be empty. If you want to save the model, save the result to this folder.
- Computing dependency parameters, which can be empty. You can configure multiple dependency parameters.

The following example shows a simple implementation of k-means clustering. In lines 48 to 50, the function acceptance parameters are set according to the preceding parameter requirements. In lines 79 and 80, the computing results are saved to the specified file.

```python
if __name__ == "__main__":

    if len(sys.argv) != 5:
        print("Usage: kmeans <file> <savePath> <k> <convergeDist>", file=sys.stderr)
        sys.exit(-1)

    spark = SparkSession\
        .builder\
        .appName("PythonKMeans")\
        .getOrCreate()

    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
    data = lines.map(parseVector).cache()
    print(data)
    K = int(sys.argv[3])
    convergeDist = float(sys.argv[4])

    kPoints = data.takeSample(False, K, 1)
    tempDist = 1.0

    while tempDist > convergeDist:
        closest = data.map(
            lambda p: (closestPoint(p, kPoints), (p, 1)))

        pointStats = closest.reduceByKey(
            lambda p1_c1, p2_c2: (p1_c1[0] + p2_c2[0], p1_c1[1] + p2_c2[1]))
        newPoints = pointStats.map(
            lambda st: (st[0], st[1][0] / st[1][1])).collect()
        tempDist = sum(np.sum((kPoints[iK] - p) ** 2) for (iK, p) in newPoints)

        for (iK, p) in newPoints:
            kPoints[iK] = p

    with open(sys.argv[2],'a+') as f:
        f.write(str(kPoints))
    spark.stop()
```

2.5.4 Order Management

There are two types of orders:

- Data set orders: After a data set application is approved, the applicant can obtain the plaintext of the data set.
- Algorithm orders: After an analysis algorithm application is approved, the applicant can obtain the computing result, but cannot obtain the plaintext of the algorithm.

On the **Orders** pages, you can apply for data sets and analysis algorithms, run algorithms, and obtain computing results.

Viewing My Applications

Step 1 Log in to Trusted Computing Platform. In the navigation pane, choose **Orders** > **My Applications**.

Step 2 Click the **Data Sets** tab.

- Click **Approved** to view the successful data set applications. You can click **Download Result** to download the data set.
- Click **Pending** to view the data set applications pending approval. You can click **Withdraw Application** to withdraw the application.

- Click **Rejected** to view the rejected data set applications. You can click **Apply Again** to apply for the data set again.

Step 3 Click the **Analysis Algorithms** tab.

- Click **Approved** to view the successful algorithm applications. You can click **Run Algorithm** to run the algorithm and click **Download Result** to download the computing result.
- Click **Pending** to view the algorithm applications pending approval. You can click **Withdraw Application** to withdraw the application.
- Click **Rejected** to view the rejected algorithm applications. You can click **Apply Again** to apply for the algorithm again.

----End

Viewing My Reviews

Step 1 Log in to Trusted Computing Platform. In the navigation pane, choose **Orders** > **My Reviews**.

Step 2 Click the **Data Sets** tab.

- Click **Approved** to view the data set applications you have approved.
- Click **Pending** to view the data set applications pending approval. You can click **Approve** or **Reject** to approve or reject the application.
- Click **Rejected** to view the data set applications you have rejected.

Step 3 Click the **Analysis Algorithms** tab.

- Click **Approved** to view the algorithm applications you have approved.
- Click **Pending** to view the algorithm applications pending approval. You can click **Approve** or **Reject** to approve or reject the application.
- Click **Rejected** to view the algorithm applications you have rejected.

----End

Managing Tasks

After your algorithm application is approved, you can run the algorithm on the **My Applications** > **Analysis Algorithms** page. Then, a computing task is generated.

On the **Tasks** page, you can manage the computing tasks running based on the analysis algorithms.

Step 1 Log in to Trusted Computing Platform. In the navigation pane on the left, choose **Orders** > **Tasks**.

Step 2 View the task information on the **Tasks** page.

Task statuses include:

- **Completed**: The task is completed. You can download the results or delete tasks in this state.
- **Running**: The task is running. You can stop or delete tasks in this state.
- **Queued**: The task is waiting to run. You can stop or delete tasks in this state.

- **Abnormal**: An error occurs on the task. You can restart or delete tasks in this state.
- **Stopped**: The task has been stopped. You can restart or delete tasks in this state.

----End

2.5.5 Deployment Management

On Trusted Computing Platform, you can install and instantiate chaincodes for trusted, collaborative computing.

This section describes how to deploy trusted chaincodes.

Step 1 Log in to Trusted Computing Platform.

Perform the following steps to access Trusted Computing Platform:

1. Log in to the BCS console.
2. In the navigation pane on the left, choose **Add-on Management**.
3. On the **Add-on Instances** tab page, click **Access Trusted Computing Platform**.
4. Enter the username, password, and verification code, and click **Log In**.

NOTE

- The default username is **admin**, and the initial login password is the password set when you buy the BCS service. For security purposes, change the password periodically.
- Use Firefox 38.0 or later or Google Chrome 43.0 or later for login. If you use Internet Explorer, the redirection may fail and a message will be displayed indicating that the certificate is untrusted. In this case, resolve the problem according to the instructions provided on the **Microsoft official website**.

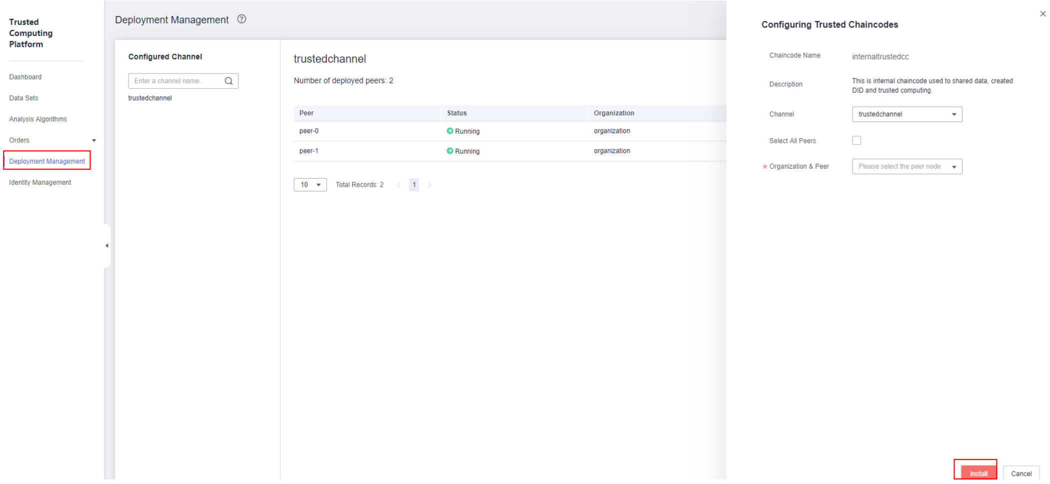
Step 2 The default trusted chaincode has already been installed and instantiated on the default channel **trustedchannel**. If there are additional channels, proceed with the next step.

Step 3 On Trusted Computing Platform, choose **Deployment Management** in the navigation pane on the left.

Step 4 Click **Configure Trusted Chaincode** on the right of the page.

Step 5 Select a channel. Then, select all peers or specify the organization and peers.

Figure 2-9 Configuring a trusted chaincode



Step 6 Click **Install**.

----End

2.5.6 Identity Management

Trusted computing depends on decentralized identity (DID). All operations on the Trusted Computing Platform are performed by DID users. After enabling trusted computing, an enterprise or organization can create a DID user and perform trusted computing as the DID user.

This topic describes how to register and update a DID user.

Registering a User

NOTE

- System user: the **admin** user for logging in to Trusted Computing Platform. The system user can configure trusted chaincodes, but all other operations are performed by the DID user.
- DID user: You must create a valid DID user to manage data sets, analysis algorithms, tasks, and orders.

Step 1 Log in to Trusted Computing Platform. In the navigation pane on the left, choose **Identity Management**.

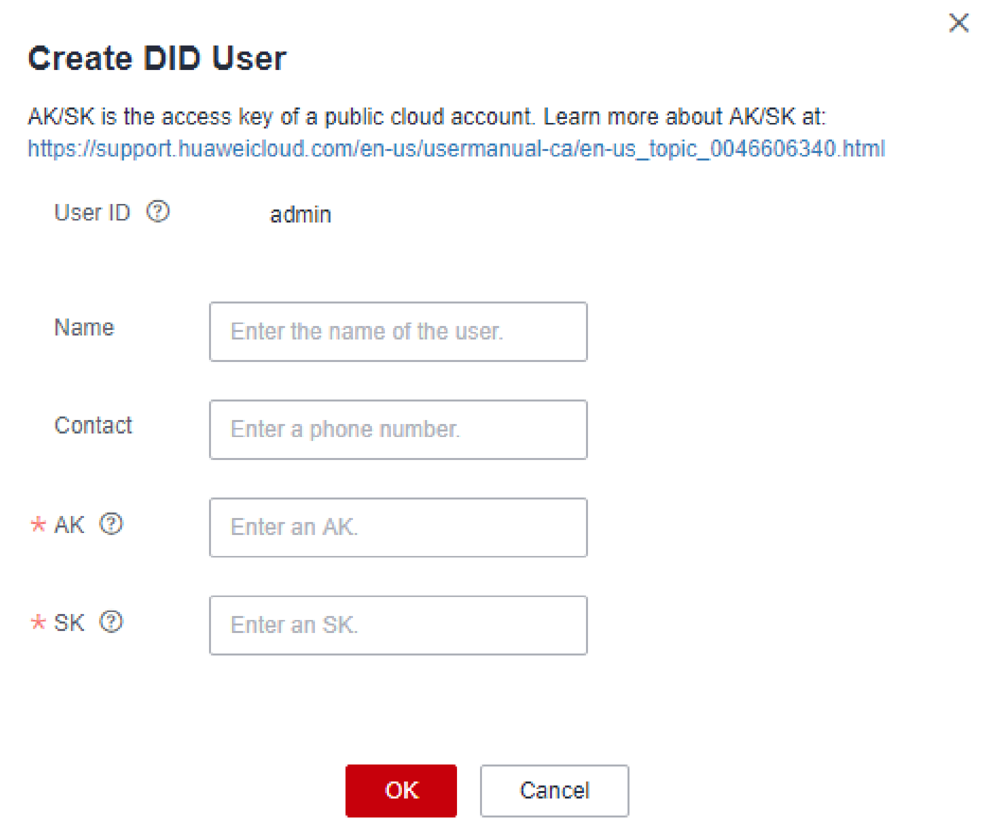
Step 2 Click **Register User**.

Step 3 Enter the name, phone number, access key ID (AK), and secret access key (SK).

NOTE

AK/SK constitute the access key of the account.

Figure 2-10 Registering a user



Step 4 Click **OK**.

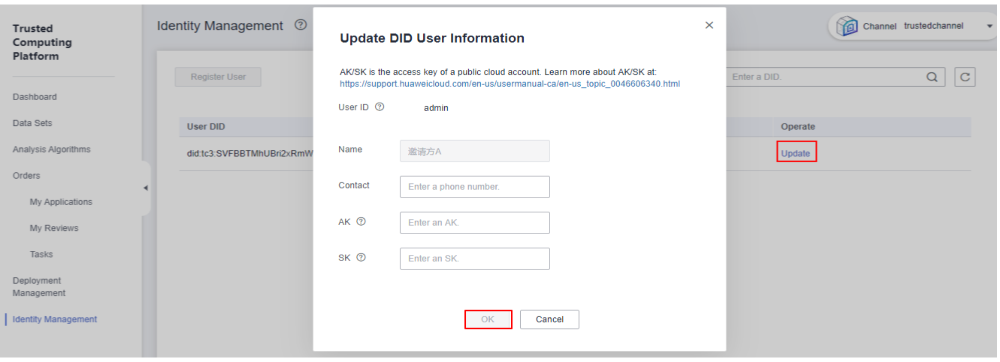
----End

Updating DID User Information

Step 1 Log in to Trusted Computing Platform. In the navigation pane on the left, choose **Identity Management**.

Step 2 Click **Update** in the **Operation** column of the row containing the user you want to update.

Figure 2-11 Updating DID user information



Step 3 Click **OK**.

----End

2.6 BCS Access

The BCS supports chaincode functions such as execution and query. Before developing an application, you need to download the certificates and SDK configuration. The SDKs can use the configuration file to easily access the blockchain network and complete transactions. You do not need to manually configure the SDKs.

Downloading Certificates

Two types of certificates are now supported: administrator certificate and user certificate. The administrator certificate is required to create, join, and upgrade a channel, and install, instantiate, update, and delete a chaincode. For transactions and query, you are advised to use the user certificate. Download the certificates of a service on the **Service Management** page.

NOTICE

- The administrator certificate differs between an orderer and a peer. For management within a channel, you need to use the administrator certificate for peers instead of that for orderers.
- Keep the private key in the downloaded certificate secure. You are advised to encrypt the private key for storage.

Step 1 Log in to the BCS console.

Step 2 In the navigation pane on the left, choose **Service Management**.

Step 3 Click the target service to view its details.

Step 4 In the **Blockchain Organizations** section of the service details page, click to download the required certificates.

Step 5 Decompress the downloaded certificate packages and store the files in an application directory for the application to access.

----End

Downloading the SDK Configuration

Step 1 On the **Service Management** page, choose **More** > **Download SDK Configuration**.

Step 2 Configure the SDK file parameters, as shown in the following figure.

Figure 2-12 Configuring the SDK file

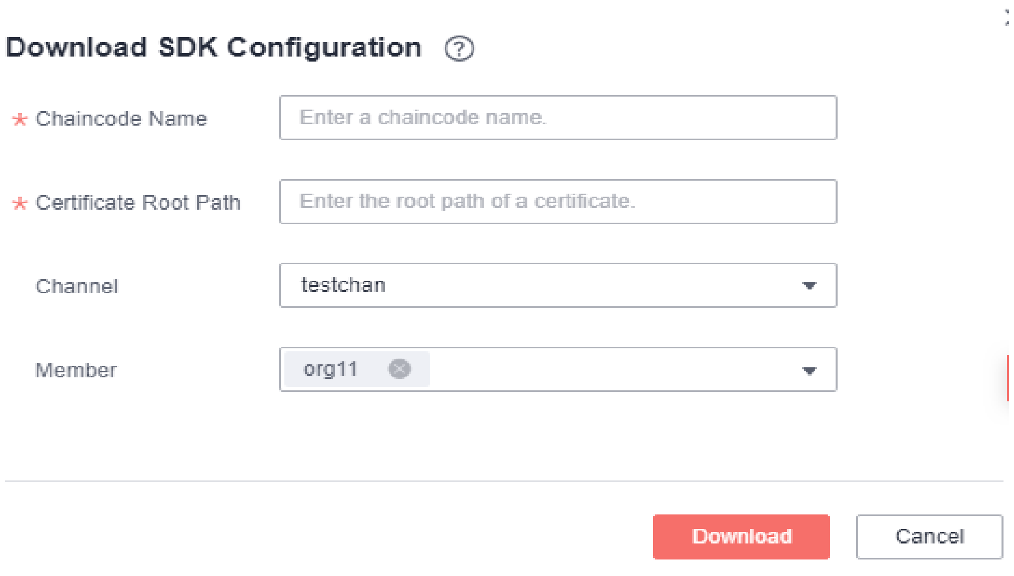


Table 2-7 Parameters

Parameter	Setting
Chaincode Name	Set it as required. The chaincode name must be the same as the name specified during chaincode installation and instantiation.
Certificate Root Path	Enter the root path of the certificates specified during application compilation.
Channel	Select a channel.
Member	Select peer organizations in the channel.

Step 3 Click **Download** to download the .zip package.

Step 4 Decompress the file package and store the retrieved .yaml file.

----End

2.7 Service Management

You can view the running status of your BCS services and perform operations on your BCS services.

Procedure

Step 1 Log in to the BCS console.

Step 2 In the navigation pane, choose **Service Management**. You can view the overall running status of your services. For details about the parameters, see **Table 2-8**.

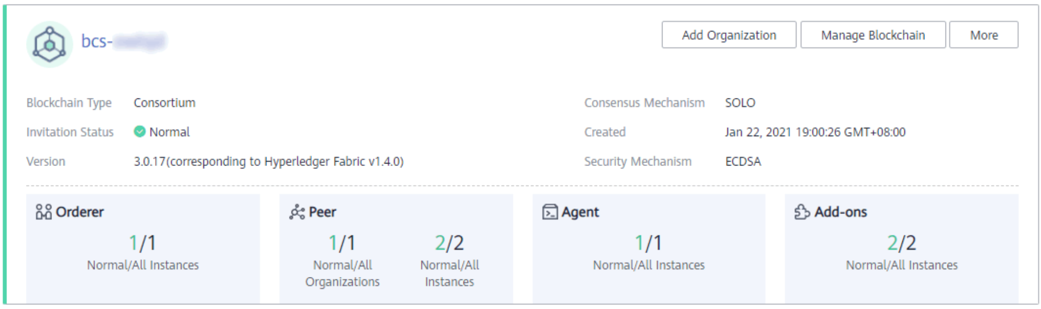


Table 2-8 Parameter description

Parameter	Description
Blockchain Type	Type of the blockchain, that is, **Consortium** or **Private**.
Edition	Edition of the BCS service, that is, **Professional** and **Enterprise**.
Consensus Mechanism	The name of the consensus mechanism, for example, **SOLO**. The following consensus mechanisms are supported: ● **SOLO**: Only one orderer is available, and therefore, fault tolerance is not supported. Solo is recommended for testing. ● **FBFT**: Requires 4 to 10 orderers for transaction ordering and tolerates faults at a maximum of (N - 1)/3 orderers, where N indicates the total number of orderers. For example, assume that there are 4 orderers. Transactions can be correctly ordered if no orderers or only 1 orderer experiences faults. ● **Kafka (CFT)**: Orders transactions using Kafka clusters. Multiple orderers can function as Kafka cluster clients to share loads, ensuring service reliability. Kafka enables higher horizontal scalability. ● **Raft (CFT)**: A CFT ordering service that tolerates faults at a maximum of (N – 1)/2 orderers, where N indicates the total number of orderers. For example, assume that there are 3 orderers, and then transactions can be correctly ordered if no orderer or only 1 orderer experiences faults.
Status	Status of the BCS service, which can be **Unknown**, **Normal**, **Abnormal**, **Creating**, **Upgrading**, **Adding peers**, **EIP abnormal**, **Deleting**, **Frozen**, **Hibernating**, or **Cluster frozen**.
Created	Time when the BCS service was created, for example, **2020/08/10 20:30:21 GMT+08:00**.
Orderer	Number of normal and abnormal orderer organizations.
Peer	Numbers of normal and abnormal peer organizations and instances.
Agent	Numbers of normal and abnormal agent organizations.

Parameter	Description
Add-ons	Number of add-ons. For example, **1/2** indicates that the total number of instances is 2 and 1 instance is abnormal.

Step 3 On the Service Management page, you can perform operations listed in **Table 2-9**.

Table 2-9 Operation list

Subject	Operation	Description
Organization	Adding an organization	1. In the service card, click Add Organization. Specify the name for **Peer Organization** and a quantity for **Peers**. 2. Click **Next**. **NOTE** ● Do not perform any operations on the service when an organization is being added. Otherwise, the service running may be affected. ● After you add an organization to an existing channel, update the endorsement policy of the channel before instantiating the chaincode. Otherwise, the instantiation may fail due to a certificate verification failure.
Service	Managing the blockchain	This operation is available only after an EIP is bound. In the service card, click Manage Blockchain. On the displayed **Blockchain Management** console, you can view, install, instantiate, update, and delete chaincodes.

<table>
<tr><th>Subject</th><th>Operation</th><th>Description</th></tr>
<tr><td></td><td>Upgrading the version</td><td>A BCS service can be upgraded to the latest version if Upgradable is displayed in the upper left corner of the card containing the created service. The operations are as follows:
1. Log in to the BCS console.
2. In the navigation pane, choose Service Management.
3. Choose More > Upgrade in the service card.
4. In the dialog box that is displayed, view the current service version or upgrade the BCS service to the latest version.
NOTE
● Services are unavailable during version upgrade. If you are a member of a consortium blockchain, your version upgrade will also affect other members in the same blockchain. Before upgrading the version, ensure that you have reached consensus with other members in the consortium and that their components will also be upgraded to the target version.
● Do not initiate version upgrade when the chaincode is being installed or instantiated.
● You can upgrade a BCS service from the version corresponding to Hyperledger Fabric v1.4 to the version corresponding to Hyperledger Fabric v2.0. If one member in a consortium blockchain has upgraded, all consortium members must also upgrade to the same version. Otherwise, transactions will fail.
– BCS v3.x.x corresponds to Hyperledger Fabric v1.4.0.
– BCS v4.x.x corresponds to Hyperledger Fabric v2.0.</td></tr>
<tr><td></td><td>Downloading the SDK configuration</td><td>Choose More > Download SDK Configuration in the service card. In the Download SDK Configuration dialogue box, set the parameters and click Download. The downloaded SDK configuration will be used for application development.</td></tr>
<tr><td></td><td>Resetting the management password</td><td>Choose More > Reset Management Password in the service card. By default, resetting this password will also reset the passwords for logging in to the Blockchain Management console and Trusted Computing Platform. If you do not want to reset these passwords together, change the passwords on the Blockchain Management console or Trusted Computing Platform separately.</td></tr>
<tr><td></td><td>Changing the blockchain network access address</td><td>Choose More > Change Access Address in the service card, select a new address, and click OK.</td></tr>
</table>

Subject	Operation	Description
	Hibernating a service	Choose **More** > **Hibernate** in the service card. **NOTE** Only services in the **Normal** state can be hibernated.
	Waking a service	Choose **More** > **Wake**. **NOTE** Only services in hibernation can be woken.
	Delete a service	Choose **More** > **Delete** in the service card.
Other Operations	Viewing failure records	In the upper left of the service cards on the **Service Management** page, the number of failure records is displayed. You can click the record quantity or to view the failure records. Click **View Details** to view details about a record.

Step 4 Click a service to view its details.

- Basic information

  On the **Basic Information** tab page, view the service details, blockchain organization information, CPU usage, physical memory usage, and add-ons.
- Monitoring data

  On the **Monitoring** tab page, view monitoring data about the service and the instances.
- Logs

  On the **Logs** tab, view the logs of the service.
- Downloading certificates

  In the **Blockchain Organizations** section on the **Basic Information** tab page, click to download the required certificates.

----End

2.8 Channel Management

Nodes communicate through channels. The channel management function enables you to create a channel or add peers to an existing channel.

Creating a Channel

Step 1 Log in to the BCS console.

Step 2 Choose **Channel Management** in the navigation pane on the left. Click **Create Channel** in the upper right corner of the page.

NOTE

- The maximum number of channels for each service is 2 for the professional edition, and 4 for the enterprise edition.
- Channels cannot be created for a service that is created by a tenant invited to a consortium blockchain.

Step 3 In the **Create Channel** dialog box, select a service, enter a channel name and description, and click **OK**.

----End

Adding a Peer

Step 1 After the channel is created, click **Add Peer** in the **Operation** column of the channel list.

Step 2 In the displayed **Add Peer** dialog box, select an organization, and specify the number of peers to be added to the channel.

Step 3 Click **OK**.

----End

Other Operations

Table 2-10 Other operations

Operation	Description
Querying channels	A channel list is displayed on the **Channel Management** page. You can view the information such as the channel name, name of the service for which the channel is used, and peers in the channel.
Viewing a peer	Click **View Peer** in the **Operation** column of the channel list to view peer information by organization, including the Membership Service Provider (MSP) ID, floating IP address (if bound), port number, peer name, domain of each peer, and whether the peer has been added to the channel.

2.9 Member Management

You can invite tenants to become blockchain consortium members, who can view invitations and topologies and delete invitations.

- To invite a tenant, click **Invite Tenant** in the upper right corner of the **Member Management** page. For details, see **Inviting a Tenant**.
- To view an invitation, click **View Invitation** in the **Operation** column on the **Member Management** page.
- To delete an invitation, click **Delete Invitation** in the **Operation** column on the **Member Management** page. After you delete an invitation, it is

withdrawn. This operation can be done only if the invited tenant has not accepted the invitation.

- To view the topology between consortium blockchain members, click **View Topology** in the **Operation** column on the **Member Management** page.

You can invite a tenant to join a channel to establish a consortium blockchain. Tenants cannot be invited to a private blockchain.

2.10 Notification Management

When another tenant invites you to join a consortium blockchain, you will receive an invitation notification. Then, you can view the invitation on the **Notification Management** page.

- To accept the invitation, click **View Details** in the **Operation** column of the notification list, select a BCS service and organization, and click **Accept**.
- To decline the invitation, click **View Details** in the **Operation** column of the notification list, and click **Decline**.
- To delete a notification, click **Delete Notification** in the **Operation** column of the notification list
- To postpone the processing of an invitation, click **View Details** in the **Operation** column of the notification list, and click **Process Later**.

NOTE

- If you have not created a BCS service, click **Create BCS Service** to create a service before selecting an organization. Otherwise, you cannot join the consortium.
- Notification statuses include:
  - **Unprocessed**: You have not processed the invitation notification. You can click **View Details** to accept or decline the invitation.
  - **Finished**: You have accepted the invitation to join the consortium blockchain.
  - **Canceled**: The inviting party has deleted the service before you accept the invitation. You cannot join the consortium blockchain.
  - **Declined**: You have declined the invitation to join the consortium blockchain.
  - **Quit**: You have accepted the invitation and joined the consortium blockchain but later quit the consortium.
  - **Dismissed**: The inviting party has deleted the service after you joined the consortium blockchain. As a result, the blockchain is dismissed.
  - **Frozen**: The inviting party's account is frozen.
  - **Upgraded**: A service in the consortium blockchain has been upgraded successfully after you join the blockchain.

2.11 Add-on Management

Add-ons allow you to extend the functionality of BCS services as required. On the **Add-on Management** page, you can install add-ons and upgrade, uninstall, and view details about the installed add-ons.

Table 2-11 lists the add-ons that are available for BCS services.

Table 2-11 Add-ons

Name	Description	Restrictions
tc3-taskserver	Provides trusted data sharing, multi-party confidential computing, and distributed identity management.	This add-on can be installed only if the BCS service meets all of the following conditions: ● Deployed in a CCE cluster ● Professional or platinum edition ● Uses ECDSA for the security mechanism ● v3.0.13 or later (corresponding to Fabric v1.4.0) or v4.0.3 (corresponding to Fabric v2.0)
baas-restapi	Supports access to the blockchain system by using RESTful APIs to manage DID, generate, apply, and issue verifiable credentials, and publish, authorize, share, and decode data.	This add-on can be installed only if the BCS service meets both of the following conditions: ● Deployed in a CCE cluster ● v3.0.16 or later (corresponding to Fabric v1.4.0) or v4.0.5 (corresponding to Fabric v2.0)

Installing the tc3-taskserver Add-on

Step 1 Log in to the BCS console.

Step 2 Choose **Add-on Management** in the navigation pane on the left.

Step 3 On the **Add-on Repository** tab page, click **Install** in the card of the tc3-taskserver add-on.

Step 4 Set the parameters by referring to **Table 2-12**.

Table 2-12 Parameters

Parameter	Description	Example Setting
Add-on	Add-on name.	tc3-taskserver
Version	Add-on version.	3.0.13
BCS Service	Select a BCS service.	bcs-6zbgus

Step 5 Click **Next**.

NOTE

- Do not perform any operations on the service when an add-on is being installed. Otherwise, the service may become abnormal.
- If you selected **No** for **Enable Support for RESTful API** during BCS service creation, you can enable support for RESTful APIs by installing an add-on.
- If you did not enable trusted computing during BCS service creation, you can still enable it by installing an add-on.

----End

Installing the baas-restapi Add-on

Step 1 Log in to the BCS console.

Step 2 Choose **Add-on Management** in the navigation pane on the left.

Step 3 On the **Add-on Repository** tab page, click **Install** in the card of the baas-restapi add-on.

Step 4 Set the parameters by referring to **Table 2-13**.

Table 2-13 Parameters

Parameter	Description	Example Setting
Add-on	Add-on name.	baas-restapi
Version	Add-on version.	3.0.13
BCS Service	Select a BCS service.	bcs-6zbgus
Enable DID API	Allows you to manage DID, generate, apply, issue verifiable credentials. Determine whether to enable the distributed identity APIs based on the service requirements.	-
Enable APIs for Trusted Data Exchange	Allows you to publish, authorize, share, and decode data. Determine whether to enable the trusted data exchange APIs based on the service requirements. **NOTE** This parameter is displayed only when **Enable DID API** is enabled.	-

Parameter	Description	Example Setting
Channel	Select a channel for installing chaincode. **NOTE** This parameter is displayed only when **Enable DID API** is enabled.	Select **channel**.

Step 5 Click **Next**.

NOTE

Do not perform any operations on the service when an add-on is being installed. Otherwise, the service may become abnormal.

----End

Add-on Instances

Step 1 Log in to the BCS console.

Step 2 Choose **Add-on Management** in the navigation pane on the left.

Step 3 View the add-ons on the **Add-on Instances** tab page.

You can perform the following operations on the add-ons as required:

- **tc3-taskserver**:
  - Click the add-on to view its details.
  - Click **Access Trusted Computing Platform** in the card of the **tc3-taskserver** add-on to go to the trusted computing platform. For details, see **Trusted Computing Platform**.
  - Click **Upgrade** to upgrade an add-on.
  - Click **Uninstall** to uninstall an add-on.
- **baas-restapi**:
  - Click the add-on to view its details.
    - You can click **Scale** next to **Normal/All Instances** to scale the number of instances in the range from 1 to 5.
    - Click **Modify** to enable or disable the APIs for DID and trusted data exchange.

      After you click **OK**, the service will be restarted and will be interrupted for a short period of time. Refresh the page later to continue to use the service.
  - Click **Upgrade** to upgrade an add-on.
  - Click **Uninstall** to uninstall an add-on.

----End

2.12 Contract Repository

A contract template is a smart contract that can implement certain functions. You can directly use the code provided by the templates or use the templates as a foundation for developing your own smart contracts.

In the Contract Management module on the console, you can view contract templates for various industries, download the ones you need, and manage your contract templates.

Downloading a Contract Template

Step 1 Log in to the BCS console.

Step 2 Choose **Contract Repository** in the navigation pane on the left.

Step 3 On the **Contract Repository** tab page, view contract templates for different industries.

Step 4 Click **View Details** to view details about a contract template, including the version, supported language, category, and interfaces.

Step 5 Click **Download** to download a contract template.

You can use the downloaded template files to install and instantiate chaincodes. For details, see **Chaincode Management**.

----End

3 FAQs

3.1 Service Usage

3.1.1 What Is Data Aging on Orderers and How Do I Enable It?

Description of Data Aging on Orderers

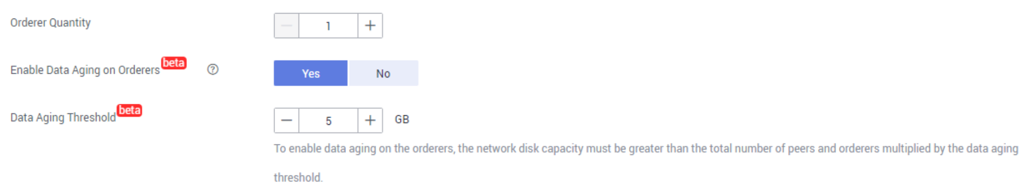
More and more ledger data is stored on orderers as the business grows. If the web disk capacity of orderers is not expanded in time, the orderers will experience performance deterioration and even stop working. In this case, services cannot run properly.

To implement data aging on orderers, a threshold must be set. After the size of the ledger data on orderers reaches the aging threshold, the earliest block will be removed every time a new block is generated. In this situation, the size of the ledger data on orderers always equals to the threshold. This prevents the impact of ledger data increase on the orderer performance and high costs of a large number of web disks.

The data aging function does not affect the service reliability. Data correctness can be ensured in the case of not more than 10 channels and high concurrency, meeting service requirements in most scenarios.

Method of Enabling Data Aging on Orderers

When deploying a BCS service, set **Consensus Mechanism** to **Kafka (CFT)** and **Enable Data Aging on Orderers** to **Yes**, and set the data aging threshold, as shown in the following figure. The data aging function of orderers is enabled after the service is deployed successfully.



3.1.2 How Do I Clear Residual Log Files After a BCS Service Is Uninstalled?

After a BCS service is uninstalled, log files are not automatically deleted from the cluster nodes. You are advised to manually delete the residual files to save your space.

Use the remote login tool to log in to each cluster node used by the service has been uninstalled, and check whether there are residual log files in the following paths:

```
/var/paas/sys/log/baas-agent
/var/paas/sys/log/baas-restapi
/var/paas/sys/log/baas-service
```

If residual log files exist, run the following command to delete them:

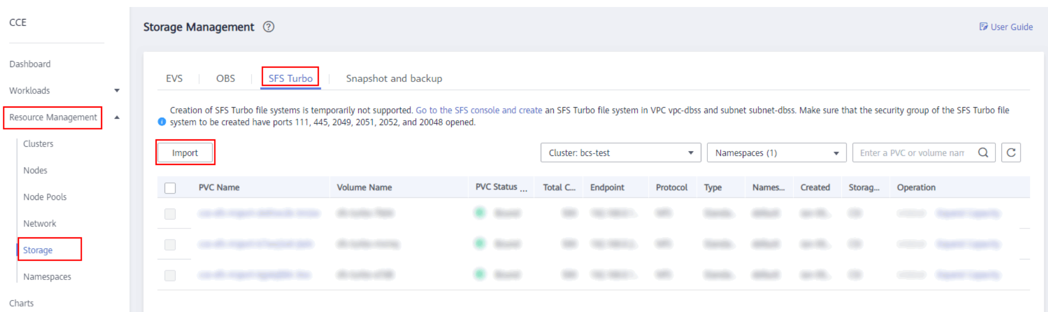
```
rm -rf /var/paas/sys/log/baas-agent /var/paas/sys/log/baas-restapi  /var/paas/sys/log/baas-service
```

3.1.3 How Can I Enable Automatic Backup and Restore Data of an SFS Turbo File System?

Enabling Automatic Backup

If you select **SFS Turbo** for **Volume Type** and **Create SFS Turbo File System** for the network storage of a peer organization when creating a BCS service, you are advised to enable the automatic backup function of SFS Turbo on the SFS console. If this function is enabled, the system automatically backs up file system data on the specified days. If file system data is deleted by mistake or contaminated, you can use the backup to restore the data, ensuring proper running of the BCS service.

After creating an SFS Turbo file system on the SFS console, import it on the CCE console before resuming BCS service creation. The following figure shows how to import an SFS Turbo file system.



NOTE

If you select **SFS Turbo** for **Volume Type** and **Automatically create SFS Turbo file system** for the network storage of a peer organization, the automatic backup function is enabled by default, and the data is backed up at 02:00 every day.

Restoring Data

1. Locate the BCS service on the **Service Management** page of the BCS console, and choose **More** > **Hibernate**.
2. Go to the SFS console, click the SFS Turbo file system, locate the backup generated at the desired time, click **Restore** in the **Operation** column, and click **Yes**.

   Wait until the data is restored. The file system is unavailable during data restore. After the restore is completed, the file system will become available again.
3. After the file system becomes available, go to the Service Management page of the BCS console, and choose **More** > **Wake** to wake up the service.

   NOTE

   - Data will be restored to the state at the backup time, and the modifications and new data added after the backup time will be lost. The file system is unavailable during data restore.
   - If a file system is deleted, its data cannot be restored from the backup.
   - Each SFS Turbo file system supports a maximum of 20 backups. If the data is backed up when 20 backups exist, the earliest backup will be deleted. For details, see the backup function description of the SFS service.

3.1.4 What Can I Do If I Can't Open the Blockchain Management Console?

Symptom

The Blockchain Management console cannot be opened.

Fault Locating

1

2

Solution

1. Check whether you are using the Internet Explorer to log in to the **Blockchain Management** console.

   If you use the Internet Explorer, you may fail to open the **Blockchain Management** login page and see a message indicating that the certificate is untrusted. In this case, you can click **here** to resolve the problem.
2. Check whether the BCS service status is abnormal.

   If the BCS service is in the **Abnormal**, **EIP abnormal**, **Frozen**, or **Unknown** state, the Blockchain Management console will become unavailable. For details, see **Abnormal Service Statuses**.

3.1.5 What Should I Do If My BCS Service Remains in the Creating State?

The possible cause is that the disk fails to be mounted.

If the DNS address in the Pod zone is incorrectly configured, the domain name cannot be resolved and the disk fails to be mounted. Run the following command to check the DNS address:

vi /etc/resolve.conf

3.1.6 What Should I Do If a Peer Restarts Frequently with the Error Message "PanicDB not exist"?

1. Go to the **/home/paas/evs/baas/{Service ID}/{Container ID}/** directory of the peer container and delete the **production** folder.
2. Restart the peer and agent containers, obtain the ledger again, and add the peer to the channel.

3.1.7 What Can I Do If the Block Height Is Inconsistent Between Peers Due to Gossip Exceptions?

1. Run the following command to check the block height of the peer and then compare it with that of other peers. If there is a difference, block retrieving may have stopped or delayed.

   peer channel getinfo –c *{Channel Name}*
2. Restart the peer and obtain the blocks again. If the fault persists, proceed to steps **3** to **5**.
3. Go to the peer container. In the **/etc/hyperledger/fabric/** directory, configure the following parameters in the **core.yaml** file to synchronize blocks from the orderer to the peer:
   ```
   useLeaderElection: false
   orgLeader: true
   ```
4. Run the following command to query the process ID of **peer node start**:

   ps –ef
5. Run the following command to restart the peer process:

   kill -9 {pid}

3.1.8 What Can I Do If the CPU Usage of a Blockchain Node Reaches 100%?

The user node may have been attacked by viruses. If this happens, perform the following operations:

- Use strong passwords that meet the requirements for all accounts (including system accounts and application accounts).
- Use security groups to control access over specific ports. For special service ports, use fixed source IP addresses, VPNs, or bastion hosts to establish your own O&M channel.
- Periodically back up data (VM internal backup, remote backup, and backup on and off the cloud) to protect data against encrypting ransomware attacks.

3.1.9 Why Can't I Log In to the Blockchain Management Console?

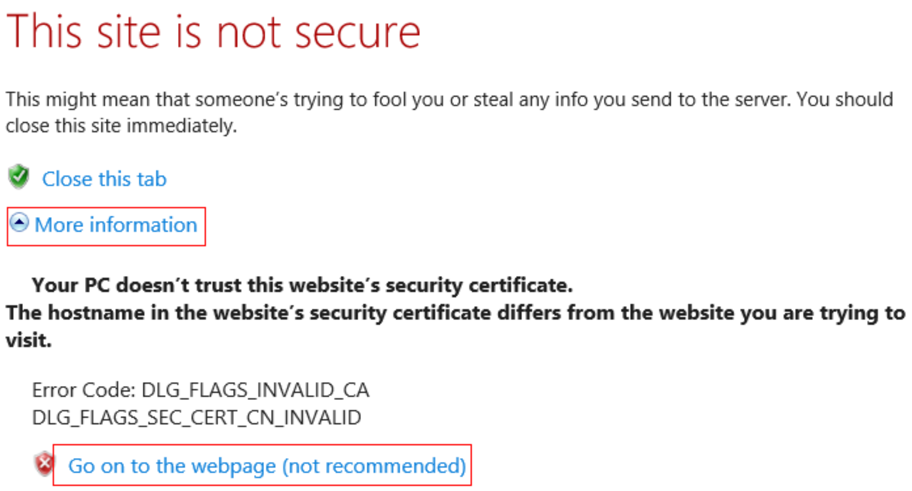
You may need to perform extra steps in your browser before you can be redirected to the Blockchain Management console.

- Internet Explorer

  NOTE

  These instructions are for reference only. The actual browser pages may vary depending on browser versions, but the operations are similar.

  a. Open Internet Explorer and enter the address of the Blockchain Management console in the address box.

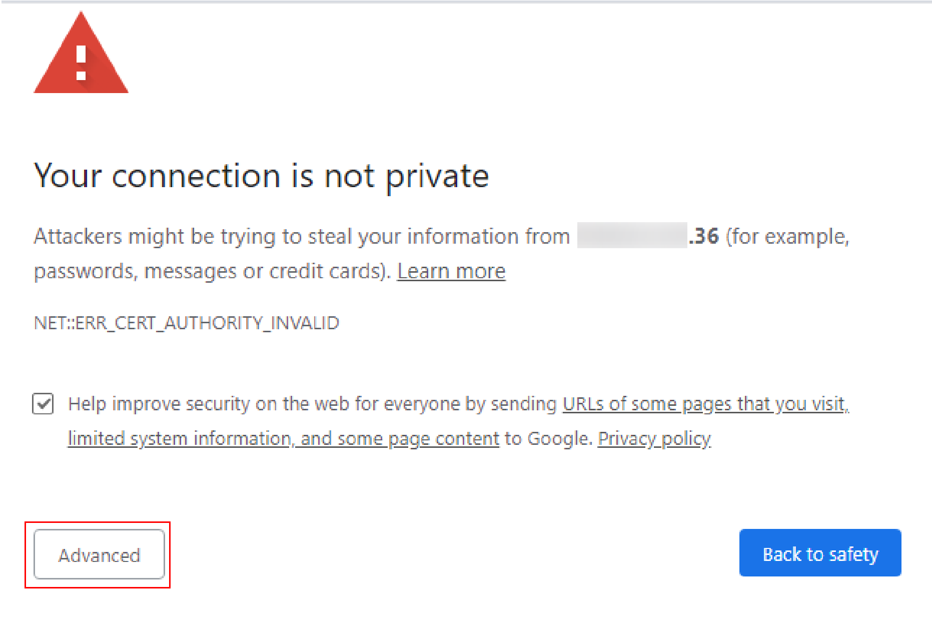
  b. Click **More information** > **Go on to the webpage** and the login page will be displayed.

- Google Chrome

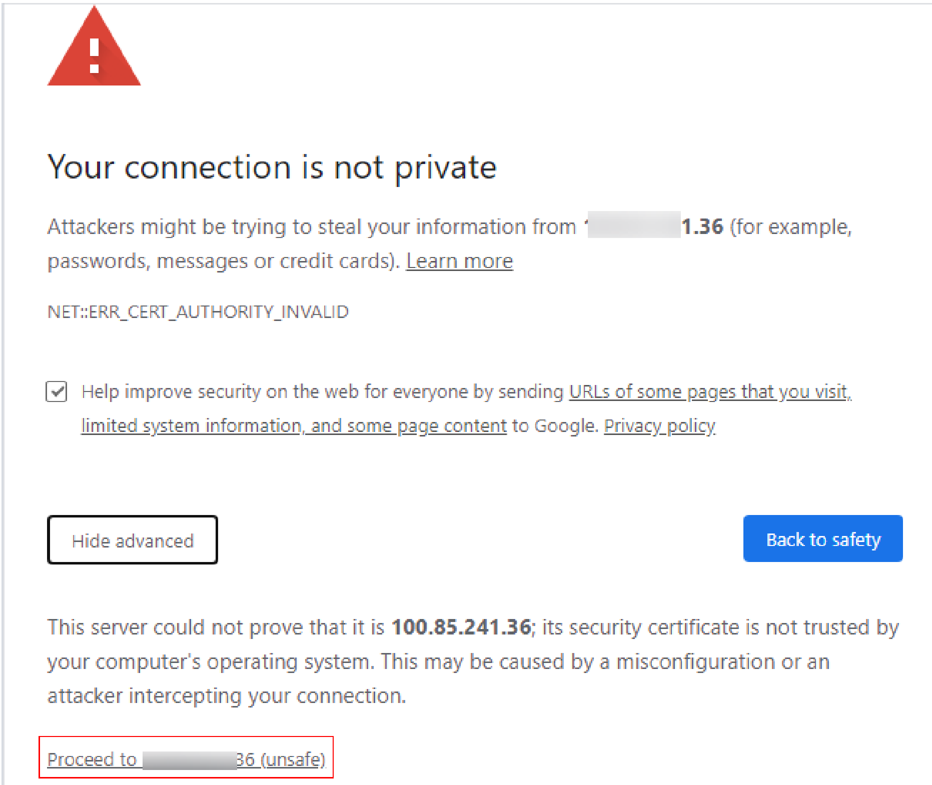
  NOTE

  These instructions are for reference only. The actual browser pages may vary depending on browser versions, but the operations are similar.

  a. Open Google Chrome and enter the address of the Blockchain Management console in the address box.



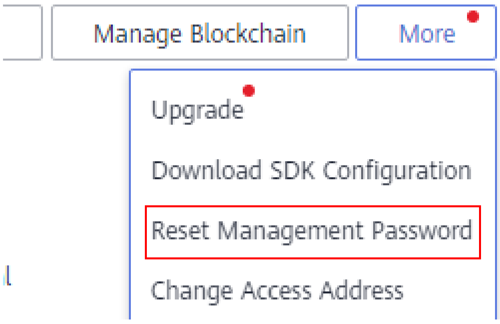
b. Click **Advanced**.



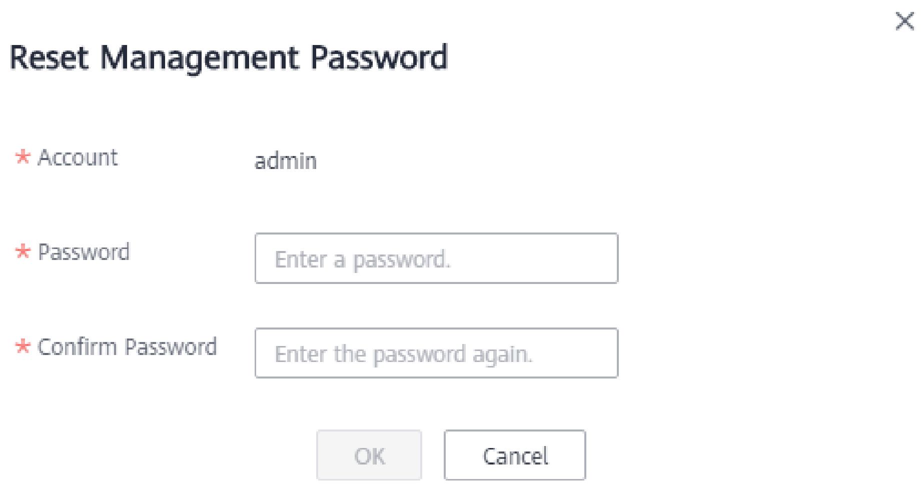
c. Proceed to the login page.

3.1.10 How Do I Reset the Management Password?

1. On the **Service Management** page, locate a service card and choose **More** > **Reset Management Password**.



2. Enter a new password and confirm the password.



3.1.11 BCS.4009100: System Error

Symptom

A system error message is displayed on the **Service Management** page.

Common Causes and Solutions

Scenario 1: The error occurs when you click any button on the **Service Management** page.

Possible cause 1: The RegionLB component of the ROMA service is abnormal.

Solution: Check the buttons on the AOM, CCE, and ServiceStage consoles. If similar errors occur, contact the RegionLB O&M personnel.

Possible cause 2: The BCS backend is abnormal and does not respond to requests.

Solution: Press F12, choose the **Network** tab, and check the request for which an error is reported. Provide the headers and preview information to the BCS O&M personnel.

Scenario 2: The error occurs only when you click some buttons on the **Service Management** page.

Solution: Press F12, choose the **Network** tab, and check the request for which an error is reported. Provide the headers and preview information to the BCS O&M personnel.

3.1.12 How Do I Change the Name of a BCS Service?

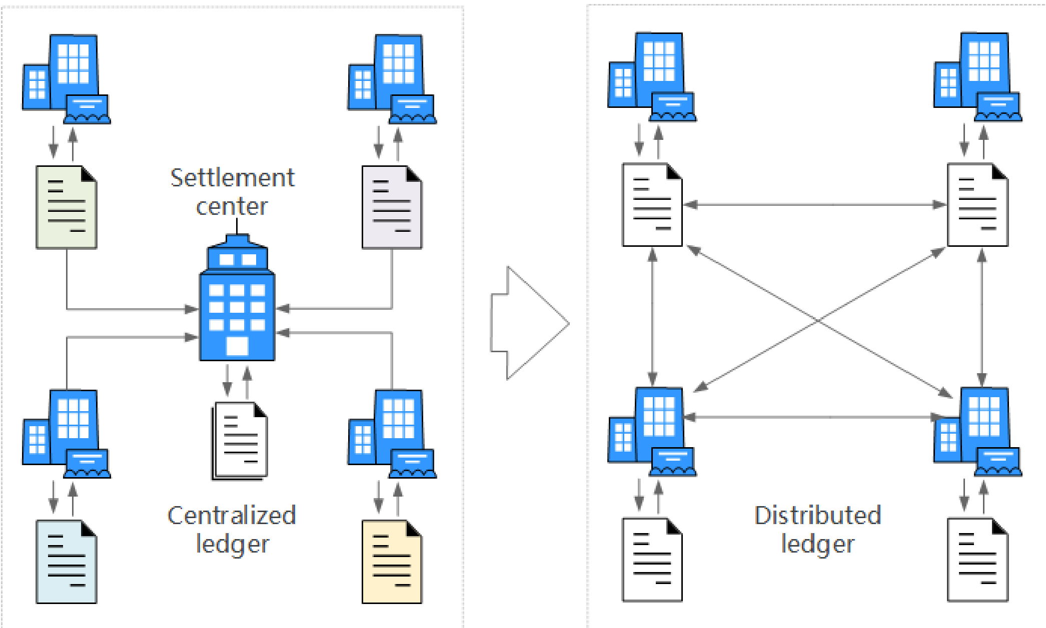
Currently, the name of a BCS service cannot be changed. To use a different name for your BCS service, you can only delete the BCS service and create another one with a new name.

3.2 Consultation

3.2.1 What Is Blockchain?

A blockchain is a decentralized and immutable distributed ledger, which does not require trust between participants to carry out transactions.

It can be regarded as a distributed database where data is almost impossible to change. Data is distributed to store and record and maintained by all blockchain participants.



Trust can be built among participants thanks to the blockchain technology. The participants in a blockchain consortium are connected to build a peer-to-peer value transfer network. That is why blockchain is recognized as a trust creator and cornerstone of the Internet of Value. It will offer great opportunities as well as huge challenges to the financial industry and even the whole society.

3.2.2 What Benefits Can Blockchain Bring?

- Improved efficiency: Transactions can be completed almost in real time.
- Lowered costs: The expenditure on the intermediary is cut.

- Reduced risks: Risks caused by tampering, fraud, and cybercrimes are reduced.
- Enhanced mutual trust: Shared, distributed transaction ledgers build trust among transaction participants.
- Transparent audit: Audit institutions can audit the immutable ledgers at any time.

3.2.3 How Do I Determine Whether a Blockchain Is Necessary?

To determine whether the blockchain technology is suitable for your project, answer the following questions in sequence:

- Need multiple parties share data?

  Will all business participants benefit from a complete and reliable record sharing system?
- Need multiple parties update data?

  Will data accuracy and update timeliness be enhanced if multiple participants can record and propagate concurrent transactions?
- Must data be verified?

  Can tamper-proof transactions in an untrusted environment improve the transaction throughput and reliability of partners?
- Can a central institution be removed?

  Is removal of a central institution helpful to reduce costs and transaction complexity?

If your answer is "Yes" for all of the above questions, then your project needs the blockchain technology.

3.2.4 What Underlying Framework Is Used for BCS?

BCS uses the HyperLedger open-source framework.

HyperLedger is a blockchain open source project hosted by the Linux Foundation to establish an underlying architecture for the distributed ledger platform oriented to multiple application scenarios. Hyperledger has sub-projects including Hyperledger Fabric, the most important one, and many other related projects derived on top of it. Developers from various sectors, such as finance, banking, IoT, supply chain, and manufacturing, have contributed towards this project seeking to build cross-field blockchain applications.

3.2.5 What Consensus Mechanisms Does BCS Support?

BCS provides the following consensus mechanisms to adapt to diverse scenarios:

- Solo (for testing)

  Solo requires only one node for transaction ordering. It does not support byzantine fault tolerance, but features quick startup and resource saving.
- Fast Byzantine Fault Tolerance (FBFT)

  FBFT is a consensus algorithm with high performance and availability. It requires at least four nodes for transaction ordering and tolerates byzantine faults at a maximum of (N – 1)/3 nodes, where N indicates the total number of blockchain nodes involved in transaction ordering. This algorithm is recommended in the production environment.

- Kafka (Crash Fault Tolerant)

  This high-speed consensus algorithm orders transactions based on Kafka/Zookeeper, tolerating crash faults on just under half of all nodes.

- Raft (CFT)

  A CFT ordering service that tolerates faults at a maximum of (N – 1)/2 orderers, where N indicates the total number of orderers. For example, assume that there are 3 orderers, and then transactions can be correctly ordered if no orderer or only 1 orderer experiences faults.

3.2.6 What Are the BCS Deployment Specifications?

BCS has the following deployment specifications:

- Only one BCS service can be deployed in a container cluster.
- Other specifications, such as the quantities of organizations, peers, channels, and orderers, vary with the editions. For details, see **Edition Specifications**.

3.2.7 How Do I Update a Chaincode If It Contains Bugs?

BCS supports chaincode upgrade. If you need to fix bugs in a chaincode, you can upload a new code package to update the chaincode.

3.2.8 Can All Blocks Be Saved As More and More Blocks Are Created?

The increasing number of transactions will lead to blockchain growth and require larger storage space. To ensure all the data is stored, you can:

- Expand the storage space.
- Back up data.

3.2.9 What Are the Specifications of VMs to Be Created for BCS?

Before deploying a BCS service, create virtual machine (VM) resources to run the service. VMs of the following specifications are recommended.

Table 3-1 Suggestions on VM creation

Your Business Phase	Consensus Algorithm	Recommended VM Specifications (Minimum)
POC test	Solo	1 VM with 4 vCPUs and 8 GB memory
POC test	FBFT	1 VM with 8 vCPUs and 16 GB memory
POC test	Kafka (CFT)	2 VMs, each with 4 vCPUs and 8 GB memory

Your Business Phase	Consensus Algorithm	Recommended VM Specifications (Minimum)
Commercial use	-	To be determined based on the service scale and number of users. You can contact the BCS O&M personnel to seek help.

3.2.10 When Do I Need to Hibernate or Wake a Service?

Scenario

You can hibernate a BCS service when it is not required temporarily and wake it when you need to use it. A service in hibernation is unavailable.

3.2.11 What Ledger Storage Modes Are Supported?

File database (GoLevelDB) and NoSQL (CouchDB) are available for ledger storage.

- File database: The Fabric native storage mode is used. Historical transaction data is stored in the blockchain, and status data is stored in the LevelDB.
- NoSQL: The CouchDB storage mode supported by the Fabric is used to store transaction data and status data. Each CouchDB database is a collection of independent documents. Each document maintains its own data and self-contained schema.

3.2.12 What Are the Differences Between Channel Isolation and Privacy Protection?

Channel isolation: A channel isolates the ledger data of a transaction from other transaction data in a consortium blockchain to ensure confidentiality. Each channel can be considered as a sub-blockchain and corresponds to a specific ledger. The ledger in a channel is invisible to other channels.

Privacy protection: Privacy is ensured within each channel because different members in a channel can have different access permissions. For example, member A has the permissions to access certain data, but member B, who does not have relevant permissions, cannot access the specified data.

In short, privacy protection isolates data for a member from other members in same channel, while channel isolation isolates data for all members in a channel from other channels.

3.2.13 How Is the Performance of BCS?

The following performance data is obtained from the pressure test carried out by using a 32 vCPUs | 64 GB ECS and two clients.

Table 3-2 BCS performance in different scenarios

Scenario	Performance	
Security Mechanism +Consensus Mechanism	**Maximum Concurrency**	**TPS (Consistency Maintained)**
ECDSA+Kafka	400	11,967
ECDSA+FBFT	50	6504
Chinese cryptographic algorithms+Kafka	300	9982
Chinese cryptographic algorithms+FBFT	50	5698

3.3 Abnormal Service Statuses

3.3.1 What Can I Do If the BCS Service Is in the Abnormal State?

Symptom

The BCS service is in the **Abnormal** state.

Fault Locating

Check item 1: Check whether the cluster, server, and storage resources on which the blockchain depends are normal.

Check item 2: Check whether the ECS specifications can meet the requirements.

Solution

- Check item 1: Check whether the cluster, server, and storage resources on which the blockchain depends are normal.
  a. Check the CCE cluster status.
     i. Log in to the CCE console, choose **Resource Management** > **Clusters**, and check the status of the cluster where the BCS service is deployed.

        If the cluster status is abnormal, locate the fault by following the CCE instructions: **How Do I Rectify the Fault When the Cluster Status Is Unavailable?**
     ii. On the CCE console, choose **Resource Management** > **Nodes**, and check the status of the node where the BCS service is deployed.

        If the node status is abnormal, locate the fault by following the CCE instructions: **What Should I Do If a Cluster Is Available But Some Nodes Are Unavailable?**

b. Check the ECS status.

   Log in to the ECS console and check the status of the ECS where the abnormal BCS service is deployed. ECS names usually take the following format: **Name of the cluster where the BCS service is deployed-A random number**.

   If the ECS is in the **Stopped** state, start the ECS, wait for about 5 minutes, and check its status again.

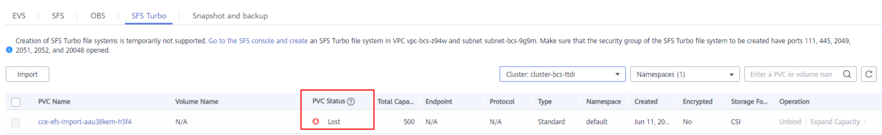
c. Check the storage status.

   i. Log in to the BCS console, go to the **Service Management** page, and click the abnormal service to view its volume type.

   ii. Log in to the CCE console, choose **Resource Management** > **Storage**. Click the **SFS Turbo** tab based on the volume type of the BCS service, and check the status of the file system in the CCE cluster where the abnormal BCS service is deployed.

   iii. If the SFS Turbo volume is in the **Lost** state, the state of the BCS service will be displayed as **Abnormal**.

   SFS Turbo:



   Solution:

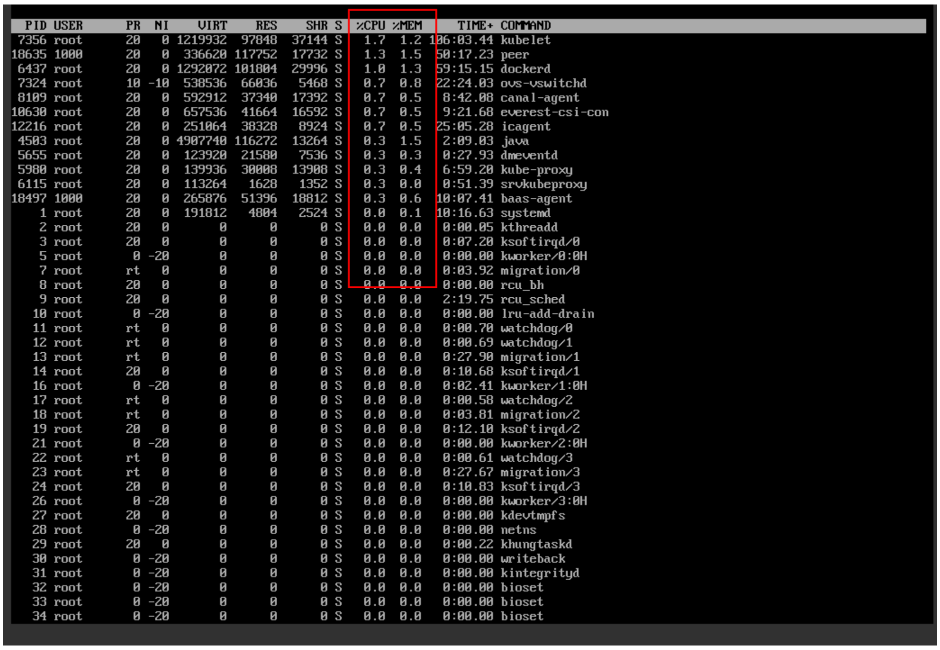
   Check whether the SFS Turbo file system exists or whether it is frozen, or contact the SFS O&M personnel.

- Check item 2: Check whether the ECS specifications can meet the requirements.

  a. Log in to the ECS where the BCS service is deployed.

     Log in to the ECS console, locate the target ECS in the ECS list, and click **Remote Login** in the **Operation** column. ECS names usually take the following format: **Name of the cluster where the BCS service is deployed-A random number**.

  b. Run the **top** command to check whether the resource usage of any application is too high.



- If the CPU or memory usage of the peer, orderer, and baas-agent containers exceeds 60% and continues to increase as the transaction quantity increases, the current ECS specifications cannot meet the transaction requirements. In this case, you need to expand the ECS specifications.
- If there are resources taking up 100% or even higher CPU or memory usage, contact the ECS O&M personnel to remove unnecessary resources.

3.3.2 What Can I Do If a BCS Service Is in the Unknown State?

Symptom

The BCS service is in the **Unknown** state.

Fault Locating

Check item 1: Check whether the cluster is hibernated.

Check item 2: Check whether the cluster exists.

Solution

- Check item 1: Check whether the cluster is hibernated.
  a. Log in to the BCS console, go to the **Service Management** page, and click the abnormal service. On the service details page, click **More** and view the name of the cluster used by the service.
  b. Log in to the CCE console, choose **Resource Management** > **Clusters**, and check the status of the target cluster.
  c. If the cluster is in the **Hibernating** state, the BCS service is in the **Unknown** state.

d. Wake up the cluster. The BCS service will become normal.

To wake up a cluster, choose **More** > **Wake** for the target cluster.

- Check item 2: Check whether the cluster exists.
  a. Log in to the BCS console and follow **a** to view the cluster name of the abnormal service.
  b. Log in to the CCE console, choose **Resource Management** > **Clusters**, and look for the target cluster.
  c. If the cluster where the BCS service is deployed does not exist, the BCS service status is displayed as **Unknown**. If the cluster is not manually deleted, contact the CCE O&M personnel.

3.3.3 What Can I Do If a BCS Service Is in the EIP abnormal State?

Symptom

The BCS service is in the **EIP abnormal** state.

Fault Locating

Check item: Check whether the EIP has been unbound or released.

1. On the BCS console, choose **Service Management**. In the card containing the target service, choose **More** > **Change Access Address** to view the EIP.
2. Go to the **Network Console**, locate the target EIP, and view its status.

Solution

1. If the EIP has been unbound, click **Bind** in the **Operation** column of the target EIP. Then, go back to the BCS console and refresh the **Service Management** page.
2. If the EIP has been released, it is not displayed in the EIP list. In this case, create a new EIP and bind it. For details, see **Assigning an EIP**. Then, go back to the BCS console. On the **Service Management** page, choose **More** > **Change Access Address** in the card containing the target service. Select the target EIP and click **OK**.

3.4 Abnormal Transactions

3.4.1 What Can I Do When Transaction Connections Fail or Time Out?

Symptom

Transaction connections fail or time out.

Fault Locating

Check item 1: Check whether the BCS service status is abnormal.

Check item 2: Check whether the Fabric SDK version used by the client matches the BCS service version.

Check item 3: Check whether the ledger of the peer is updated.

Check item 4: Check whether the DB file exists.

Check item 5: Check whether the BCS service uses CouchDB for ledger storage and has been upgraded or restarted.

Solution

- Check item 1: Check whether the BCS service status is abnormal.

  Log in to the BCS console and rectify the fault based on the service status by following instructions provided in **What Can I Do If the BCS Service Is in the Abnormal State?**

- Check item 2: Check whether the Fabric SDK version used by the client matches the BCS service version.

  Log in to the BCS console. In the navigation pane, choose **Service Management**. In the card containing the abnormal service, choose **More** > **View Details**.

  Record the value of **Version**. Check whether the Fabric SDK version used by the client is the same as the BCS service version. If the versions are inconsistent, transactions may fail or time out.

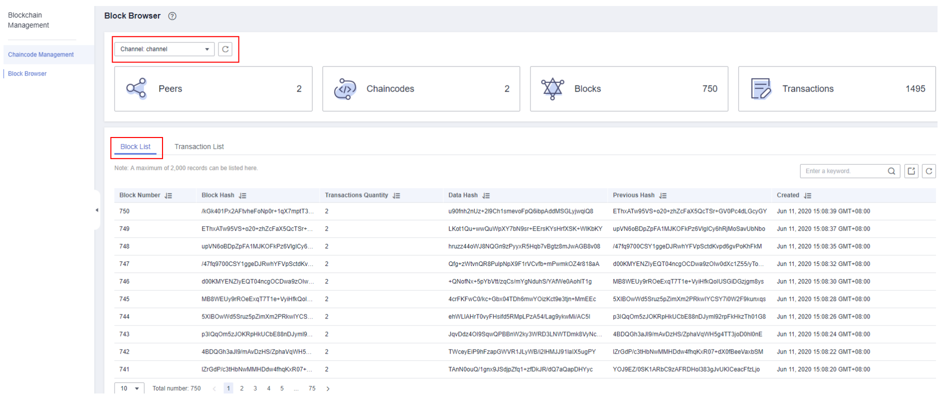
  Solution:

  Download the Fabric SDK of the version that corresponds to the Hyperledger Fabric version of the BCS service.

  Download link: **https://github.com/hyperledger/fabric**

- Check item 3: Check whether the ledger of the peer is updated.

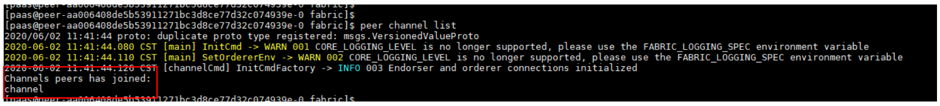
  a. Log in to the BCS console. In the navigation pane, choose **Service Management**. In the card containing the abnormal service, click **Manage Blockchain**. On the **Block Browser** page, select the abnormal channel, and view the block quantity on the **Block List** tab page.

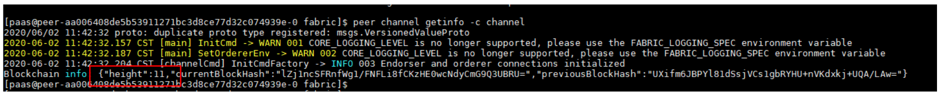
  b. Log in to the ECS where the BCS service is deployed, run the **docker ps|grep k8s_peer** command to check the peer containers, and record the ID of the container where transactions time out.

c. Run the **docker exec -it** *container id* **/bin/bash** command to enter the container.

d. Run the **peer channel list** command to query the channel to which the peer is added.



e. Run the **peer channel getinfo -c** *{Channel name}* command to check whether the ledger of the peer is updated.

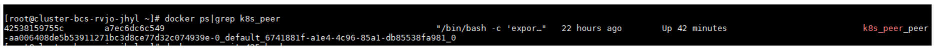


If the block quantity in the ledger of the peer is different from that displayed on the **Block Browser** page, query the block quantity again after 10 minutes. If the block quantity has not increased over the past 10 minutes, the ledger of the peer is not updated due to insufficient resources or high concurrency. As a result, transactions become abnormal.

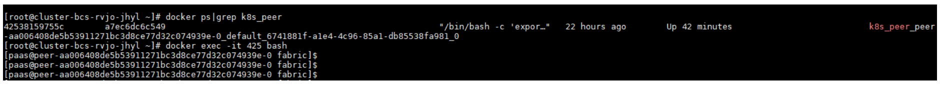
In this case, submit a service ticket for consultation.

- Check item 4: Check whether the DB file exists.

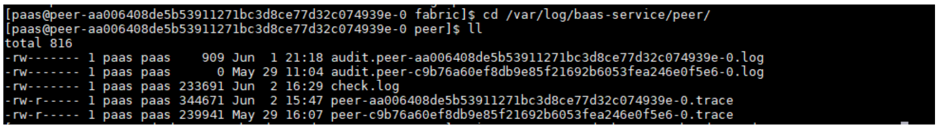
a. Log in to the ECS where the BCS service is deployed, run the **docker ps|grep k8s_peer** command to check the peer containers, and record the ID of the container where transactions are abnormal.



b. Run the **docker exec -it** *container id* **/bin/bash** command to enter the container.

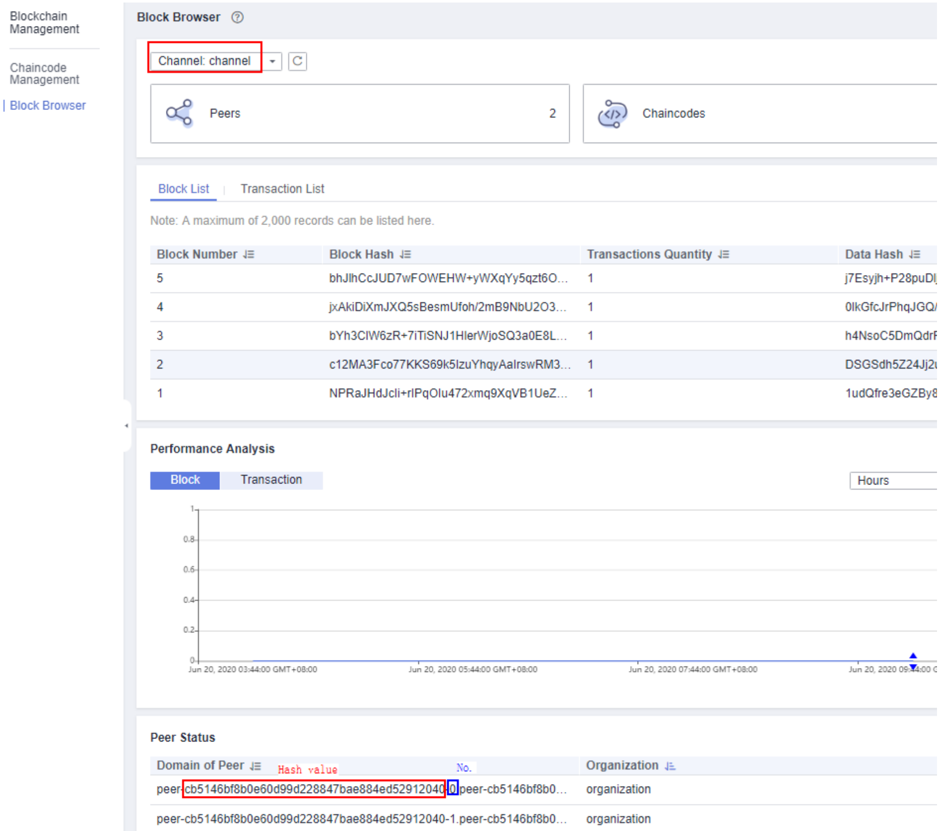


c. Run the **cd /var/log/baas-service/peer/** command to go to the directory where the logs of the peer are stored, and run the **ll** command to view all files.



d. Obtain the hash value and sequence number of the peer.

On the **Block Browser** page of the Blockchain Management console, view the domain name and sequence number of the peer in the **Domain of Peer** column in the **Peer Status** list.



e. The peer log file names take the following format: **peer-*{hash value}*-*{serial number}*.trace**. Run the **cat *{file name}*|grep -C 5** command to search for exception information.

If you see "Fail to recover DB: file does not exist", the DB file of the peer does not exist. As a result, transactions are abnormal.

In this case, submit a service ticket for consultation.

- Check item 5: Check whether the BCS service uses CouchDB for ledger storage and has been upgraded or restarted.

If a BCS service is of an earlier version and uses CouchDB for ledger storage, the status data will be lost after the service is restarted, because the status data is not stored in the SFS file system. In this case, CouchDB reloads the block data to generate the status data, and the BCS service is unavailable for a certain period of time.

It takes about 2 hours to synchronize data of 150,000 blocks. During data synchronization, port 7051 of the peer cannot be accessed.

Solution:

Upgrade the BCS service to the latest version, and this problem will not occur when you upgrade or restart the service.

NOTE

When a BCS service is upgraded to the latest version for the first time, the CouchDB container mounts the web disk and synchronizes status data. As a result, the BCS service is unavailable for a certain period of time. This duration increases linearly as the block quantity increases. It takes about 2 hours to synchronize data for every 150,000 blocks. The block quantity can be viewed on the **Block Browser** page of the Blockchain Management console.

3.4.2 What Can I Do If the Network Connection Is Terminated or Rejected During Blockchain Access?

Increase the node bandwidth or host specifications, or reduce the transaction concurrency.

3.5 Demo Problems

3.5.1 General Checks

Fault Locating

Check item 1: Check whether the service status is normal.

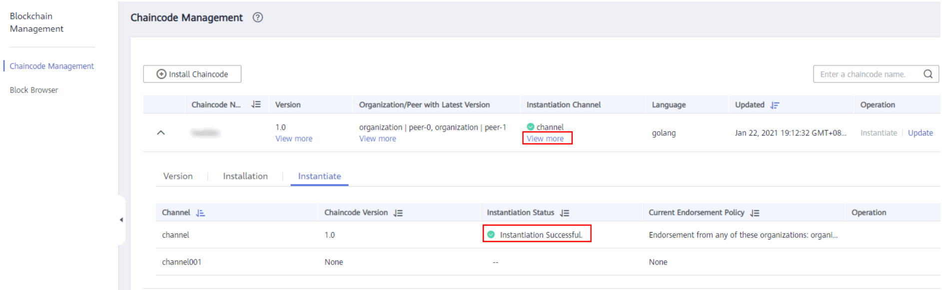
Check item 2: Check whether the chaincode is properly instantiated.

Check item 3: Check whether the demo has been modified.

Solution

- Check item 1: Check whether the service status is normal.

  Log in to the BCS console. In the navigation pane, choose **Service Management**. View the status of the target service. For details, see **Abnormal Service Statuses**.
- Check item 2: Check whether the chaincode is properly instantiated.
  a. Log in to the BCS console. In the navigation pane, choose **Service Management**. In the card containing the target service, click **Manage Blockchain**. Enter the password and verification code and click **Log In**. Then, go to the **Chaincode Management** page.
  b. Click **View more** in the **Instantiation Channel** column to view the instantiation status of the target chaincode.



- Check item 3: Check whether the demo has been modified.

  Follow the instructions in the next sections to check whether the specified demo has been modified.

4 Change History

Released On	Description
2021-01-15	This issue is the first official release.