

---

# Blockchain-based Service Network User Manual

Version 1.8.4

BSN Foundation

## CONTENTS

1	BSN Introduction .....	1
1.1	Brief Introduction .....	1
1.2	BSN Services .....	2
1.2.1	Permissioned Services.....	2
1.2.2	Permissionless Services .....	2
1.2.3	Interchain Services.....	3
1.3	Terminologies .....	3
2	Release Notes .....	5
3	Quick Start .....	9
3.1	Permissioned Blockchain.....	9
3.2	Permissionless Blockchain .....	10
3.3	Documentation.....	11
4	Registration and Activation .....	12
4.1	Registration.....	12
4.2	Login.....	14
4.3	Forgot Password .....	15
5	Permissioned Services .....	17
5.1	Overview.....	17
5.2	BSN Keys and Certificates Mechanism.....	18
5.2.1	BSN Keys and Certificates Mechanism.....	18
5.2.2	Locally generate the DApp access key pair .....	19
5.3	DApp Services Publication and Participation.....	20
5.3.1	Overview.....	20
5.3.2	DApp Services Publication .....	20
5.3.3	DApp Services Management .....	27
5.3.4	DApp Services Participation .....	30
5.4	Off-BSN System Access Guide .....	36
5.4.1	Overview .....	36
5.4.2	BSN Smart Contract Package Requirements .....	40
5.4.3	PCN Gateway Fabric API.....	46
5.4.4	PCN Gateway FISCO API.....	72
5.5	Development SDK and Examples .....	96
5.5.1	BSN Gateway SDK Example .....	96

5.5.2	Sample Smart Contract Packages.....	97
5.6	BSN Testnet Services .....	97
5.6.1	Overview .....	97
5.6.2	Permissioned DApp Service Publication .....	97
5.6.3	Interchain Services on BSN Testnet .....	99
6	Dedicated Node Services .....	100
6.1	Overview.....	100
6.2	Project Management .....	100
6.2.1	Create Projects .....	100
6.2.2	Edit Projects .....	103
6.2.3	Delete Projects .....	104
6.2.4	View Project Details .....	104
6.2.5	Unsubscribe Projects.....	109
6.2.6	Edit Authorized Account .....	110
6.2.7	Configuration Upgrade .....	110
6.3	Access Instructions .....	113
6.3.1	ConsenSys Quorum Access Instruction .....	113
6.3.2	Hyperledger Fabric Access Instruction .....	114
6.3.3	Hyperledger Besu Access Instruction .....	117
7	Permissionless Services .....	120
7.1	Overview.....	120
7.2	Select Plans.....	120
7.3	Create and Manage Projects.....	124
7.4	Off-BSN system Access Guide.....	126
7.4.1	Overview .....	126
7.4.2	Ethereum .....	127
7.4.3	EOS 128	
7.4.4	Tezos.....	128
7.4.5	dfuse-eos .....	129
7.4.6	Findora .....	129
7.4.7	Near130	
7.4.8	Klaytn.....	130
8	Interchain Services.....	132
8.1	Interchain Service Management .....	133

8.1.1	Open Interchain Services .....	133
8.1.2	View Interchain Services .....	134
8.1.3	Deactivation and Activation of Interchain Services.....	136
8.2	Interchain Services based on Poly Enterprise .....	137
8.2.1	Overview .....	137
8.2.2	Interchain Services based on Hyperledger Fabric.....	138
8.2.3	Interchain Services based on FISCO BCOS .....	139
8.2.4	Interchain Services based on Ethereum Ropsten .....	139
8.2.5	Interchain Services based on Neo Testnet .....	141
8.3	Interchain Services based on IRITA .....	143
8.3.1	Overview .....	143
8.3.2	Interchain Architecture based on IRITA.....	143
8.3.3	Interchain Services in BSN Testnet .....	144
8.3.4	Interchain Services based on Hyperledger Fabric.....	150
8.3.5	Interchain Services based on FISCO BCOS .....	152
9	Oracle Services .....	156
9.1	Oracle Service based on Chainlink .....	156
9.1.1	Overview .....	156
9.1.2	Invocation of the Chainlink oracle service in the BSN Testnet .....	157
9.1.3	Example of calling BSN IRITA Chainlink service across chains.....	158
10	IDE Services .....	166
10.1	Overview.....	166
10.2	Access Instructions .....	166
10.2.1	Service publication of permissioned chains.....	166
10.2.2	Service editing and upgrading of permissioned chains.....	170
10.2.3	Access to permissionless services.....	171
10.2.4	BSN Testnet Services.....	174
10.3	My IDE .....	174
11	DID Services .....	175
11.1	Overview.....	175
11.2	HTTP API.....	176
11.2.1	DID API.....	178
11.2.2	Issuer.....	182
11.2.3	Credential.....	186



11.2.4	Identity Hub .....	189
11.3	Response Code.....	196
11.4	SDK .....	200
11.4.1	DID 201 .....	
11.4.2	Issuer.....	205
11.4.3	Credential.....	210
11.4.4	Identity Hub .....	213
12	Account Management .....	220
13	Online Documentation .....	221
14	Contact Us.....	223

# Preface

Blockchain-based Service Network (BSN or Service network) is a worldwide infrastructure network that provides a one-stop-shop solution for blockchain and distributed ledger technology (DLT) applications (DApp). BSN is a complex system that involves programming, software development, resource and environment configurations, application deployment, gateway APIs, local SDK, key certificates, etc. To facilitate utilization, BSN International ([www.bsnbase.io](http://www.bsnbase.io)) has prepared this document for developers and users to learn how to use BSN. We hope that BSN will become the first choice for developers to develop and run their DApps.

BSN provides developers three types of services: Permissioned, Permissionless, and Interchain services.

Permissioned services are divided into two parts. The first part demonstrates how developers can deploy smart contracts to the selected public city nodes through the BSN portal; the second part describes how developers can connect their off-BSN systems to the corresponding smart contracts through the public city node gateway and conduct data transaction processing.

Permissionless services determine how developers can choose the appropriate public city nodes, plans, and public chain frameworks, to deploy and publish their DApps.

BSN's "Interchain Communications Hub" (ICH) integrates two interchain solutions based on relay chain mechanism: Poly Enterprise developed by Onchain and IRITA developed by Bianjie AI. It enables cross-chain interoperability between standard permissioned chains, open permissioned chains and public chains. We will continue to integrate more cross-chain protocols to achieve interoperability of all blockchains adapted to the BSN.

Please feel free to contact us if there are any further questions. Our contact information can be found in [Chapter 14. Contact Us](#). We strongly recommend users access the Online Documentation section to explore BSN technical details further.

# 1 BSN Introduction

## 1.1 Brief Introduction

The BSN design and concept as taken from the Internet, is a connected set of devices across data centers using the TCP/IP protocol. BSN is formed by the connection of the public city nodes using a set of blockchain operating environment protocols. Just like the Internet, BSN is also a cross-cloud, cross-portal, cross-framework, global infrastructure network.

With BSN, there are three types of participants: cloud service providers, blockchain framework providers, and portal operators.

Cloud service providers, through the installation of free BSN public city node software, can make their cloud service resources (computing power, storage, and bandwidth) accessible and sell through BSN to end-users.

Blockchain framework providers align with the BSN's framework adaptation standards and deploy them on BSN so developers can use it to develop and deploy applications. The Permissionless service only applies to the BSN international portal and international public city nodes.

Portal operators can easily and quickly build a Blockchain as a Service (BaaS) platform on their existing websites using BSN APIs. This allows them to provide BSN capabilities to their end users without users leaving their websites.

BSN is an open network that any cloud service provider, framework provider, or portal operator, that complies with BSN requirements and standards is free to use and stop using the service network at any time.

Similar to the Internet, most users of BSN are developers and technology companies. They can use any BSN portal to purchase cloud resources that charge based on transactions per second (TPS), storage quantity, and bandwidth from any public city node around the world. They select any pre-adapted framework to conveniently develop, deploy, and manage permissioned blockchain applications at a very low cost. Blockchain developers only need to deploy the application to one or more public city nodes on BSN so participants can connect to the application at no cost through any public city node gateway. All deployed applications share server resources in every public city node. For high-frequency applications, public city nodes can intelligently allocate a dedicated peer node with high processing capacity. For low-frequency applications, they share the same peer node. This resource-sharing mechanism allows BSN to reduce the resource cost to one-twentieth of the cost of traditional blockchain cloud services.

BSN is a blockchain infrastructure network. Just as households do not need to dig their own wells, but instead, enjoy the water supply services provided by public water plants in cities, BSN blockchain application publishers and participants do not need to buy physical servers or cloud resources to build their blockchain operating environment.

They use the public services provided by BSN and rent shared resources as needed, thus greatly reducing their costs. According to recent research, it takes about 20,000 USD per year for developers to build and deploy a traditional permissioned blockchain LAN-type environment. However, with BSN, the minimum cost to run such an application is as low as one dollar a day. Cost is a huge factor and will encourage a large number of small, medium, and micro enterprises and even individuals (including students) to innovate and start businesses through BSN. This will undoubtedly promote rapid development and popularization of blockchain technology. In general, the development from the closed architecture of the traditional blockchain to the resource-sharing architecture of BSN completely mimics the development process of the Internet, which gathered numerous isolated LANs in the early days to the global connectivity facilities we have today. We hope to make BSN the blockchain Internet.

## **1.2 BSN Services**

As mentioned above, BSN provides a one-stop-shop solution for developers to deploy, operate, and manage DApps. BSN provides three types of services: Permissioned, Permissionless, and Interchain services.

### **1.2.1 Permissioned Services**

BSN is continually adapting most of the mainstream permissioned blockchain frameworks. On the BSN portal users can deploy DApps on any public city nodes based on the type of selected framework and the number of peer nodes. The number of peer nodes per application can be up to 60 and can be distributed among public city nodes based on different cloud services. Users can easily complete the DApp deployment process by uploading smart contracts and configuring the corresponding parameters. This service mode allows developers to focus on business innovation, smart contract programming. All work related to environment construction, system maintenance, application deployment, node transmission, and network configuration is done by BSN.

The pricing strategy for the Permissioned service is based on three resource elements of each peer node of the published application. The three elements are TPS, storage, and data traffic. Among them, TPS and storage in the BSN portal are pre-paid, while data traffic will be charged based on actual usage. This pricing strategy is designed to minimize resource costs and provide users with the best services. Based on the data provided by the BSN portal, if a user deploys a three-peer Fabric DApp, and each peer node supports 10TPS and 10GB storage capacity, the monthly fee is only 20 USD.

The pricing strategy of BSN dedicated node services is based on the host configuration, hard disk, and data usage of the cloud platform selected for the service, where the host configuration and hard disk are prepaid, and the data usage fee is postpaid according to the actual amount incurred; the pricing strategy of interchain services is postpaid according to the actual number of cross-chain calls occurred; the permissioned services, oracle services and interchain services provided by the BSN Testnet services are all free of charge.

### **1.2.2 Permissionless Services**

The Permissionless service is only applicable to the BSN international portal

([www.bsnbase.io](http://www.bsnbase.io)) and international public city nodes. Compared with the complexity of the Permissioned service, Permissionless service has the virtue of simplicity. The Permissionless service mainly provides developers who develop public chain DApps, with unified access service covering numerous public chain nodes. Developers may choose different plans on the BSN portal, and can simultaneously deploy DApps and process transactions on all BSN adapted public chain nodes through the selected public city nodes.

We offer a free plan and different premium plans. The free plan includes up to 2,000 requests per day. There are 3 types of premium plans, priced at \$20, \$100, and \$500 per month. The premium plans include up to 20,000 requests, 125,000 requests, and 750,000 requests, respectively, per day. All requests can be assigned to any public chain freely.

Permissionless services only provide shared nodes and access environments and do not involve any business of the public chain itself. The gas fees incurred in publishing and running DApps on any public chain shall be borne by the developers themselves and have nothing to do with BSN.

### 1.2.3 Interchain Services

The vision of BSN is to become the Internet of blockchains. In the future, millions of DApps will be deployed and run on BSN. Both Permissioned and Permissionless DApps will be very easy to call and they can interact with each other just like applications currently do on the Internet. From this perspective, Interchain will become a very core part of the BSN technical architecture.

The BSN's "Interchain Communications Hub" (ICH) is now commercially available and integrates with Onchain's Poly Enterprise and Bianjie's IRITA cross-chain solutions. It supports cross-chaining between permissioned chains and cross-chaining between permissioned chains and ETH Ropsten testnet and NEO testnet.

A demo version of ICH is also live on the BSN Testnet, integrating two interchain solutions based on the relay chain mechanism: (1) Poly Enterprise developed by Onchain and (2) IRITA developed by Bianjie. We welcome all developers to try out and provide feedback and suggestions to us and we will continue to improve the interchain functionality.

BSN's "Interchain Communications Hub" (ICH) integrates two interchain solutions based on relay chain mechanism: Poly Enterprise developed by Onchain and IRITA developed by Bianjie. It enables cross-chain interoperability between standard permissioned chains, open permissioned chains and public chains. We will continue to integrate more cross-chain protocols to achieve interoperability of all blockchains adapted to the BSN.

## 1.3 Terminologies

- **Public city node (PCN):** This is the core element of BSN but the "node" part doesn't refer to the blockchain nodes and BSN isn't a blockchain. With BSN, each PCN is a virtual data center used to allocate a portion of resources from the cloud

service or data center on which it was deployed. An entire blockchain operating environment has been built within this resource pool and includes multiple blockchain frameworks, shared peer nodes, CA management, authority chain, PCN gateway, and PCN manager systems.

- **DApp:** This is a generic term for blockchain and distributed ledger technology application.
- **DApp Service or Service:** This is a DApp that is already deployed and in use on BSN that users can access with an invitation from the DApp publisher. The invitation allows them to directly join and use the service.
- **Service Publisher:** This is the individual or enterprise who published and deployed the DApp service on BSN and is responsible for granting access to users who apply to participate in the service.
- **Service Participant:** This is a user that uses the BSN DApp service via a BSN portal or the publisher's system. Also, the user's off-BSN system can connect to the DApp service via the PCN gateway to execute transactions and query data.
- **Off-BSN system:** A business IT system developed and managed by a DApp service publisher or a service participant outside BSN.

## 2 Release Notes

Release date	Version	Notes
2023/06/19	1.8.4	<ul style="list-style-type: none"><li>• The BSN International's website <a href="http://www.bsnbase.io">www.bsnbase.io</a> has taken down Permissionless Chain Casper.</li></ul>
2023/05/31	1.8.3	<ul style="list-style-type: none"><li>• The BSN International's website <a href="http://www.bsnbase.io">www.bsnbase.io</a> has taken down Permissionless Chain Polkadot.</li></ul>
2023/05/29	1.8.2	<ul style="list-style-type: none"><li>• The BSN International's website <a href="http://www.bsnbase.io">www.bsnbase.io</a> has taken down Permissionless Chain Oasis Network.</li></ul>
2022/04/24	1.8.1	<ul style="list-style-type: none"><li>• Optimized DID Services functions and interfaces;</li><li>• Added Unsubscribe function in Permissioned Services.</li></ul>
2022/01/23	1.8.0	<ul style="list-style-type: none"><li>• Iterative optimization and technical optimization of the BSN international website (<a href="http://www.bsnbase.io">www.bsnbase.io</a>) to enhance user experience;</li><li>• Added the cross-chain function of Hyperledger Fabric V2.3.2 framework by Poly Enterprise and IRITA in the Interchain Communications Hub;</li><li>• Integrated Hyperledger Fabric V2.3.2 to the BSN Testnet;</li><li>• Upgraded the version of Hyperledger Fabric from 2.2.0 to 2.3.2 in the Dedicated Node Services;</li><li>• Integrated Klaytn public chain and provide external services on the international website and international nodes;</li><li>• Fixed bugs and enhanced the stability of the system.</li></ul>
2021/10/31	1.7.0	<ul style="list-style-type: none"><li>• Iterative optimization and technical optimization of the BSN international website (<a href="http://www.bsnbase.io">www.bsnbase.io</a>) to enhance user experience;</li><li>• Added Hyperledger Fabric V2.3.2 framework to Permissioned Services;</li><li>• Added Configuration Upgrade function in the Dedicated Node Services;</li><li>• Upgraded IRITA in the Testnet to support Ethereum Ropsten network and Chainlink oracle services;</li><li>• Launched BSN DID Services to support credential issuance, authentication and authorized access management in a unified identity management system;</li><li>• Integrated Cypherium public chain and provide external services on the international website and international nodes;</li><li>• Fixed bugs and enhanced the stability of the system.</li></ul>
2021/07/31	1.6.0	<ul style="list-style-type: none"><li>• Iterative optimization and technical optimization of the BSN international website (<a href="http://www.bsnbase.io">www.bsnbase.io</a>) to enhance user experience.</li><li>• Launched BSN dedicated node services based on Hyperledger Fabric V2.2.0 and Hyperledger Besu V21.1.2 frameworks.</li><li>• Optimized and improved Interchain Communications Hub's protocols and service stability.</li></ul>

		<ul style="list-style-type: none"> <li>Updated the interface of the BSN Empowerment Platform to provide the DApp service management APIs of Open Permissioned Blockchains and BSN Testnet to BSN portals.</li> <li>Fixed some bugs and enhanced the stability of the system.</li> </ul>
2021/05/31	1.5.1	<ul style="list-style-type: none"> <li>Launched IDE Services, supports frameworks including Hyperledger Fabric, FISCO BCOS, Ethereum, Nervos and Algorand.</li> </ul>
2021/04/30	1.5.0	<ul style="list-style-type: none"> <li>Iterative optimization and technical optimization of the BSN international website (www.bsnbase.io) to enhance user experience.</li> <li>Launched BSN dedicated node services based on ConsenSys Quorum framework.</li> <li>Launched the commercial service of Interchain Communications Hub based on IRITA.</li> <li>Fixed some bugs and enhanced the stability of the system.</li> </ul>
2021/03/19	1.4.1	<ul style="list-style-type: none"> <li>Added public chain main net and test net nodes along with native API access services. Including: Casper, Findora and Near.</li> </ul>
2021/01/31	1.4.0	<ul style="list-style-type: none"> <li>Iterative optimization and technical optimization of the BSN international website (www.bsnbase.io) to enhance user experience.</li> <li>Launched the commercial service of Interchain Communications Hub based on Poly Enterprise.</li> <li>Fixed some bugs and enhanced the stability of the system.</li> </ul>
2020/11/30	1.3.1	<ul style="list-style-type: none"> <li>Added public chain main net and test net nodes along with native API access services. Including: BTY, Oasis and Polkadot.</li> </ul>
2020/10/31	1.3.0	<ul style="list-style-type: none"> <li>Optimized the BSN International website (www.bsnbase.io) to improve user experience.</li> <li>Launched BSN Permissioned Blockchain Testnet, providing developers with a free testing environment supporting: Hyperledger Fabric R1, FISCO BCOS K1 DApp Services publication</li> <li>-Interchain testing services</li> <li>Launched the BSN Interchain Communications Hub on BSN Testnet based on Poly Enterprise and IRITA.</li> <li>Added the BSN empowerment platform APIs to allow third-party portals to access BSN Permissionless Services.</li> <li>Added the TPD (Transactions Per Day) limit control function in the Permissionless Services</li> <li>Fixed some bugs and enhances the stability of the system.</li> </ul>
2020/9/24	1.2.1	<ul style="list-style-type: none"> <li>Updated the BSN Global website address to <a href="https://www.bsnbase.io">https://www.bsnbase.io</a>.</li> <li>Added public chain main net and test net nodes along with native API access services. Including: Algorand, ShareRing and Solana.</li> <li>Added Enable Key function in the public chain project.</li> </ul>

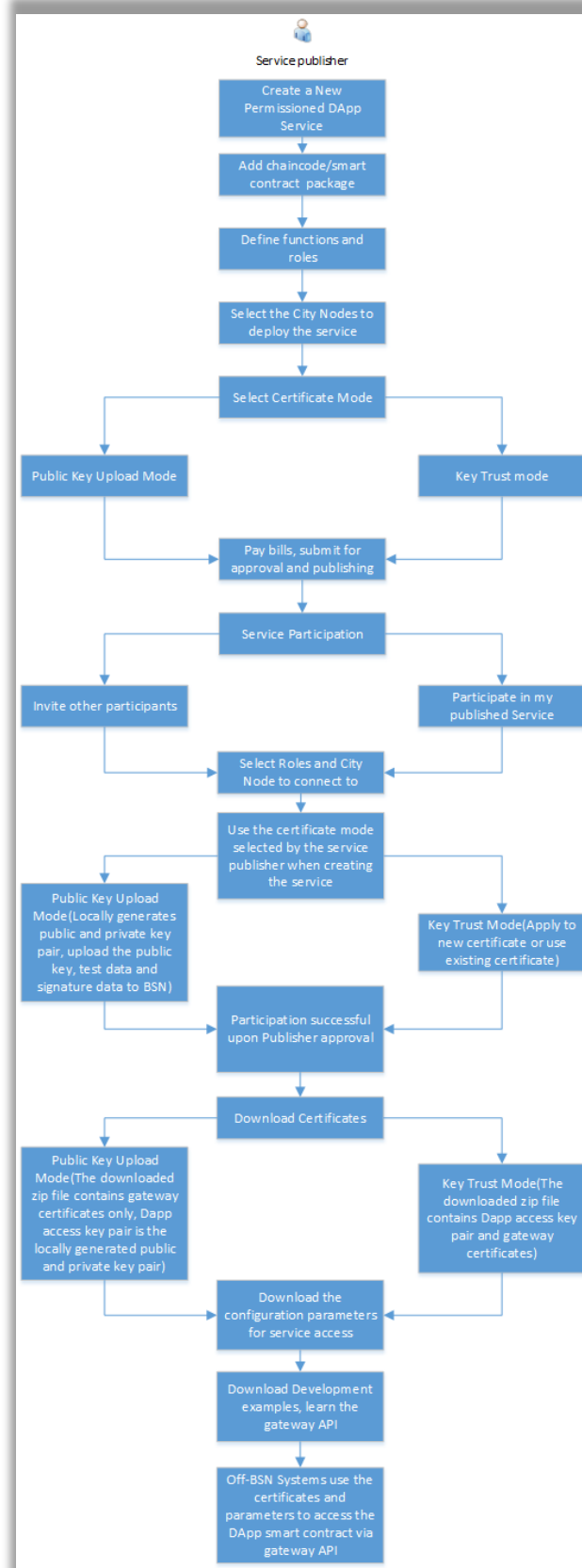


2020/8/10	1.2.0	<ul style="list-style-type: none"> <li>• Redesigned the user interface to provide better navigation and user experience.</li> <li>• Added public chain main net and test net nodes along with native API access services. Including: Nervos, Neo, ETH, Tezos, EOS, IRISnet, etc.</li> <li>• Added commercial functionality for Hyperledger Fabric and FISCO BCOS frameworks.</li> <li>• Updated FISCO BCOS framework to support SECP256 K1 encryption algorithm.</li> <li>• Added the following functionality to Permissioned services: recurring payment mechanism for service charge and data usage charge, service configuration upgrade.</li> <li>• Added Permissionless service plan purchase and upgrade.</li> <li>• Added "My Account" in User Center to make it easier for users to update credit card information, check bills, pay bills and download invoices (we process all credit card activities directly on Stripe).</li> <li>• Added Online Help Manual to provide developers and portal users easy-to-follow instructions.</li> <li>• Added PCN gateway SDK and all examples on Github: <a href="https://github.com/bsnda">https://github.com/bsnda</a>.</li> <li>• Fixed a few bugs to enhance the stability of the system.</li> </ul>
2020/4/25	1.1.0	<ul style="list-style-type: none"> <li>• The BSN global portal has officially launched. Beta testing will be held from April 25<sup>th</sup>, 2020 to June 25<sup>th</sup>, 2020.</li> <li>• Developers can deploy one three-peer DApp (service) at up to three public city nodes (PCNs) free of charge during beta testing.</li> <li>• There is a total of 10 available PCNs during beta testing. They are deployed on AWS, Microsoft Azure, Google Cloud, China Mobile Cloud, and Huawei Cloud.</li> <li>• During beta testing, there are two frameworks to choose from, Hyperledger Fabric V1.4.3 or FISCO BCOS V2.4.0.</li> <li>• Developers can choose "Key Trust Mode" or "Public-Key Upload Mode" to manage their service users' certificates and keys.</li> <li>• Basic information and chaincode/smart contracts in deployed services can be modified anytime. PCNs, however, cannot be changed once chosen.</li> <li>• Published services are private by default. Developers will need to apply for a public listing. After approval, they will be available on the App Store.</li> <li>• Developers will need to grant permissions to other users to participate in their services. The participants then follow the services' instructions to generate service access keys and user transaction keys by using either "Key Trust Mode" or "Public-Key Upload Mode".</li> <li>• The PCN gateway provides a set of user registration APIs for</li> </ul>

		<p>deployed services. Developers can register service users via these APIs from their off-BSN systems. Developers do not need to log in to the BSN global portal.</p> <ul style="list-style-type: none"><li>• The PCN gateway APIs support “Key Trust Mode” for both Fabric and FISCO BCOS. “Public Key Upload Mode” is only supported for Fabric.</li><li>• For more info on gateway APIs, please refer to the developer’s manual.</li></ul>
--	--	---

## 3 Quick Start

### 3.1 Permissioned Blockchain

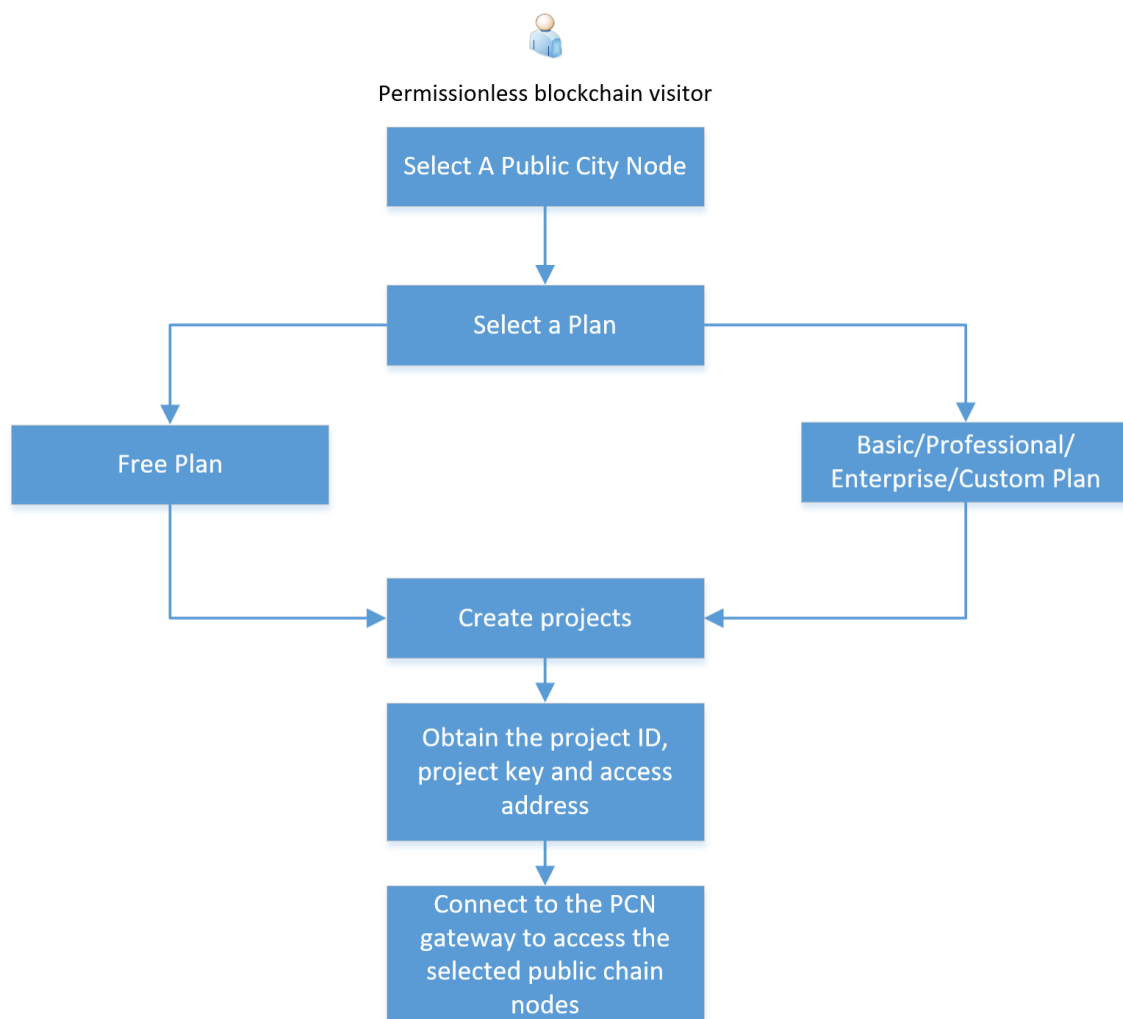


Permissioned DApp Service publishers can create DApp services in the BSN portal. To create the service, it is necessary to upload the smart contract/chaincode package, define the service functions and roles, select the public city nodes and select the participant certificate mode (including Key Trust mode and Uploaded Public Key mode). After that, publishers pay the bills and submit the service deployment request to the network operator for approval and publishing.

After the successful publication of the service, publishers can participate in their service or invite other users to participate in the service. To participate in the service, participants should select designated roles and the access public city node, then generate the certificates according to the certificate mode set by publishers. Participation will be successful after being approved by service publishers.

Once successfully participating in the service, participants can download the certificate, and use the certificate and service access configuration parameters to access the chaincodes/smart contracts through the gateway API.

## 3.2 Permissionless Blockchain



Permissionless services allow visitors to select public city nodes and plans to participate in a service. There are 2 types of plans, the free plan, and premium plans. Visitors can choose plans according to their business requirements. To connect to the public chain nodes, users can create projects to obtain project IDs, project keys, and access addresses to access the public chain services.

### **3.3 Documentation**

The direct users of the BSN portal are developers. As the environment and tools of the blockchain application's development, deployment, and operation, BSN is relatively complex in its overall operation. We strongly recommend that all developers start by examining the documentation and examples so that they will be able to master the use of BSN within a day or two.

For your convenience, all examples we've provided are available on Github. We hope that developers with serious interest can help us optimize and enrich the examples so that other developers are able to adapt and learn about blockchain development. Developers who share their samples, will receive small gifts and be invited to BSN's internal technical seminar.


For links to all documents and examples, please visit [Chapter 13. Online Documentation](#).

## 4 Registration and Activation

BSN requires its users to register and confirm their registration before they can access the network to carry out services and actions across the network. As a first-time user, follow these steps to register:

### 4.1 Registration

1. Click [here](http://www.bsnbase.io) to access the website at [www.bsnbase.io](http://www.bsnbase.io).
2. With the blockchain-based service network, you can access the system either as an Individual or a Corporate entity.
3. To register as an Individual, enter or select the following:
  - **Username** – Enter a preferred username
  - **Nationality** – Click the dropdown to select your country from the list of countries
  - **Name** – Enter your real name, different from the username
  - **Mobile Number** (Optional) – Enter your mobile number
  - **Email address** – Enter an email address you have access to
  - **Brief description of your programming experience** (Optional) – If you have some experience in programming, we would love to hear about it
  - Check the **I have read and agree to Terms of User and Privacy Policy** box
  - Click **Confirm** to finish the registration.



### Create an Account

Username \*

The username consists of 6-25 characters, including letters and numbers.

☒ Individual ☐ Corporate

Name \*

China

▼

Email Address \*

Mobile Number

Brief description of your Programming Experience

0/280

☐ I have read and agreed to [Terms of Use](#) and [Privacy Policy](#)

Confirm

[Go Back](#)

4. To register as a Corporate entity, enter or select the following:

- **Username** – Enter a preferred username
- **Nationality** – Click the dropdown to select your country from the list of countries
- **Enterprise Name** – Enter the legal name of your corporate body or company name
- **Detailed Address** – Enter a verifiable address of the company location
- **Contact Name** – Enter a contact name that represents the company
- **Mobile Number** (Optional) – Enter your corporate mobile number
- **Email address** – Enter a corporate email address that you have access to
- **Brief description of your programming experience** (Optional) –If you have some experience in programming, we would love to hear about it
- Check the **I have read and agree to Terms of User and Privacy Policy** box
- Click **Confirm** to finish the registration.

**Create an Account**

Username \*

☐ Individual ☒ Corporate

The username consists of 6-25 characters, including letters and numbers.

Enterprise name \*

China

▼

Detailed Address \*

0/160

Contact Name \*

Mobile Number

Email Address \*

Brief description of your Programming Experience

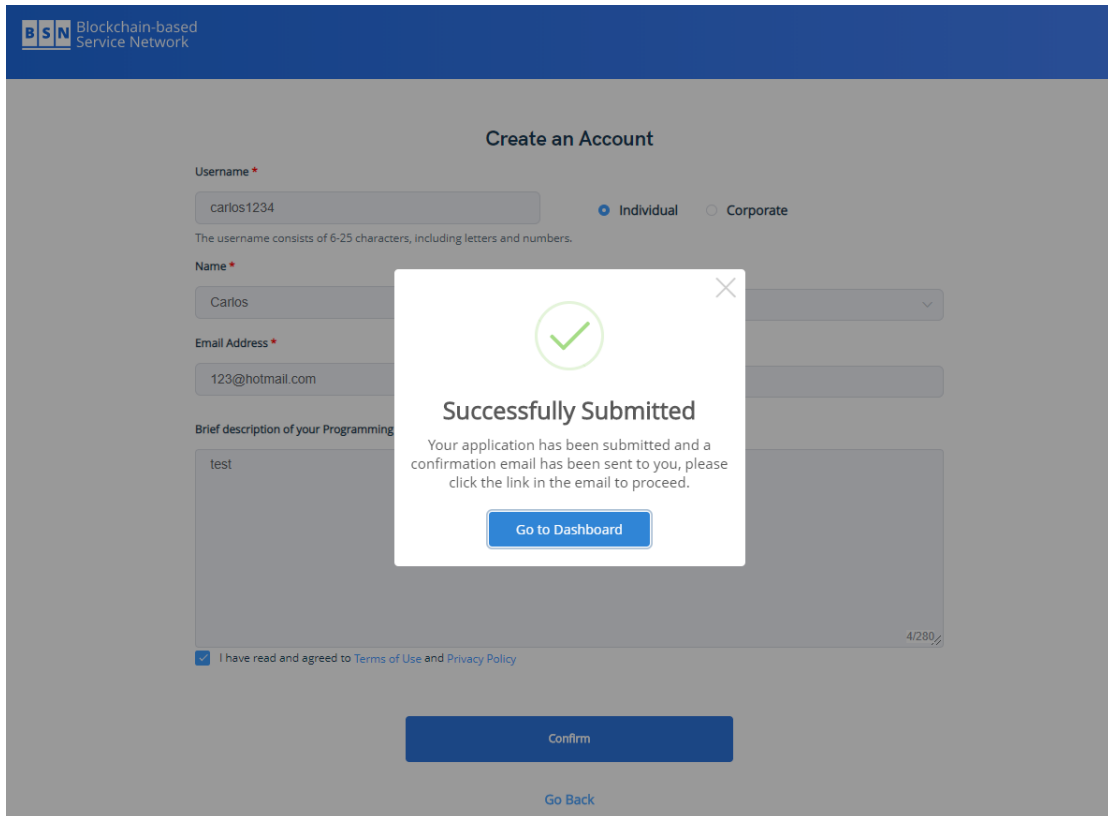
0/280

☐ I have read and agreed to [Terms of Use](#) and [Privacy Policy](#)

Confirm

[Go Back](#)

5. A confirmation dialog box will be displayed confirming your registration. Click **Go to Dashboard**.



The screenshot shows the 'Create an Account' page of the Blockchain-based Service Network (BSN). The page has a blue header with the BSN logo and name. The main content area is light gray. A white confirmation dialog box is centered on the screen, featuring a green checkmark icon and the text 'Successfully Submitted'. Below the text, it says 'Your application has been submitted and a confirmation email has been sent to you, please click the link in the email to proceed.' and includes a blue button labeled 'Go to Dashboard'. The background form is partially visible and dimmed. It includes fields for 'Username' (filled with 'carlos1234'), 'Name' (filled with 'Carlos'), 'Email Address' (filled with '123@hotmail.com'), and a 'Brief description of your Programming' (filled with 'test'). There are radio buttons for 'Individual' (selected) and 'Corporate'. At the bottom of the form, there is a checkbox for 'I have read and agreed to Terms of Use and Privacy Policy' and a 'Confirm' button. A 'Go Back' link is at the very bottom.

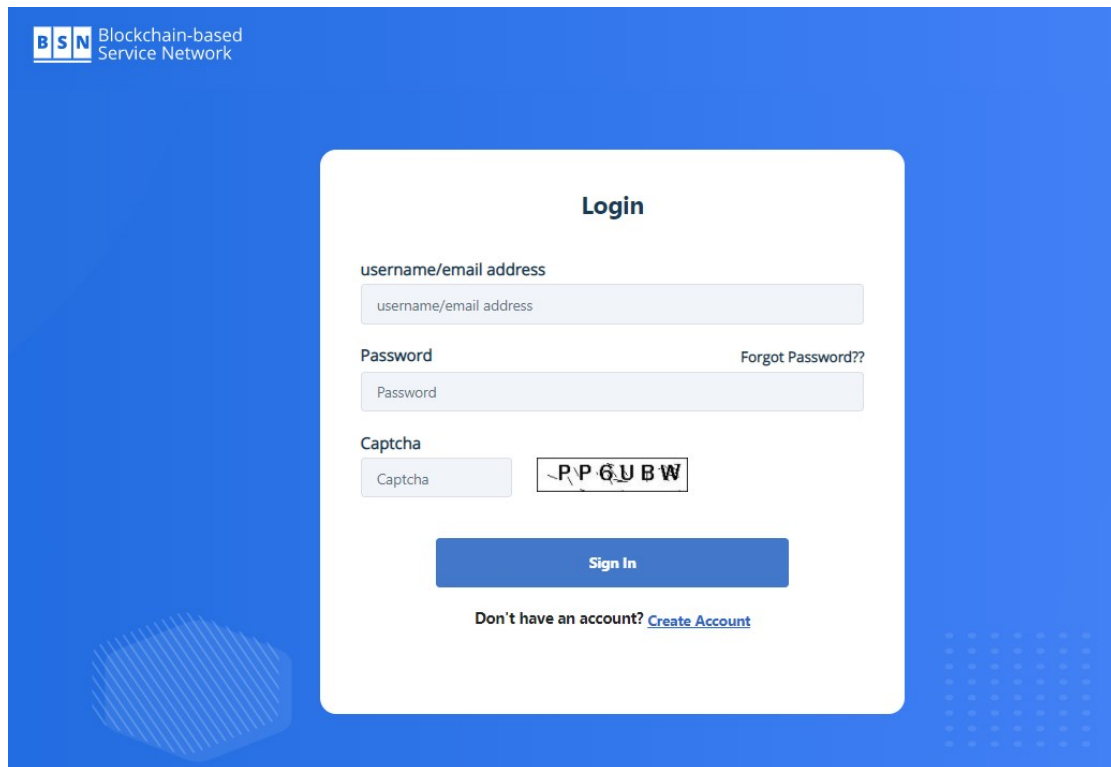
6. You will receive an email from BSN requesting that you confirm your registration.
7. Click on the link in the email to confirm your registration and enter your **Password** and **Password Confirmation**.
8. Click **Confirm** when done to return you to the login page.

## 4.2 Login

After you have successfully registered your account on BSN, you can login by following these steps:

1. Click on the [Login](#) link to access the login page.
2. On the login page, enter the **Username/Email**, **Password**, and the **Captcha Code**.
3. Click **Sign In** to access the **Home** page.





**BSN** Blockchain-based Service Network

### Login

username/email address

username/email address

Password [Forgot Password??](#)

Password

Captcha

Captcha

PP6UBW

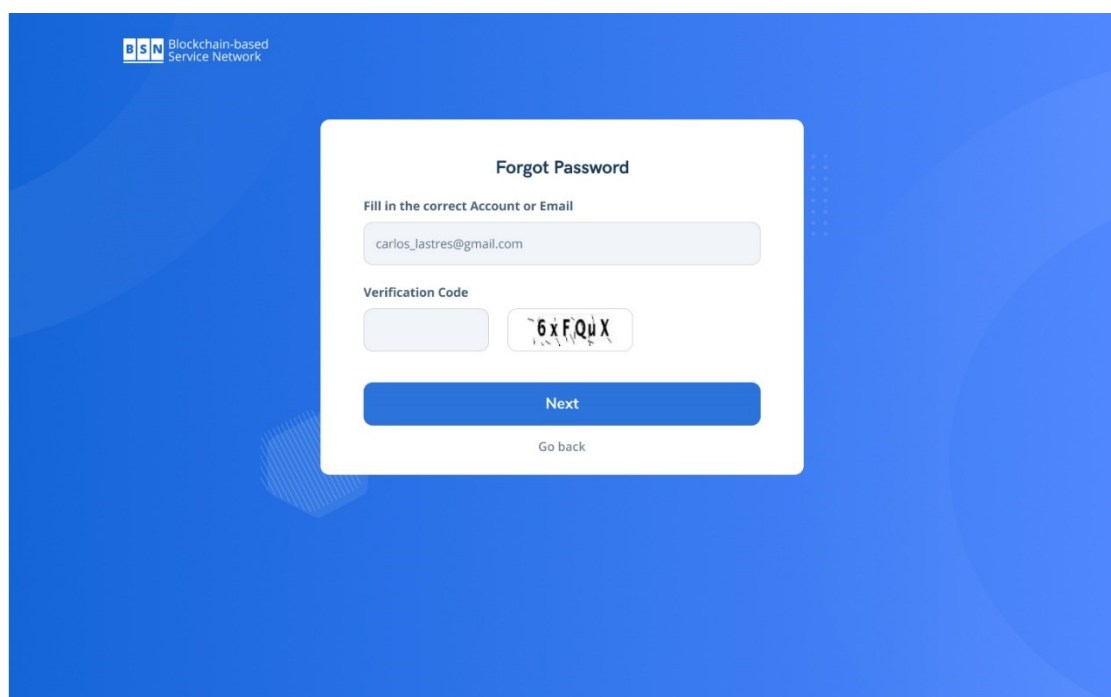
**Sign In**

Don't have an account? [Create Account](#)

## 4.3 Forgot Password

If at any time you have forgotten your password, you can follow these steps to retrieve it:

1. On the **Login** page, click the **Forgot Password** to open the forgot password page.
2. On the page, enter the **correct account or email**.
3. In the **verification code**, enter the displayed code. If you wish to generate another code, click on the code to generate another.



**BSN** Blockchain-based Service Network

### Forgot Password

Fill in the correct Account or Email

carlos\_lastres@gmail.com

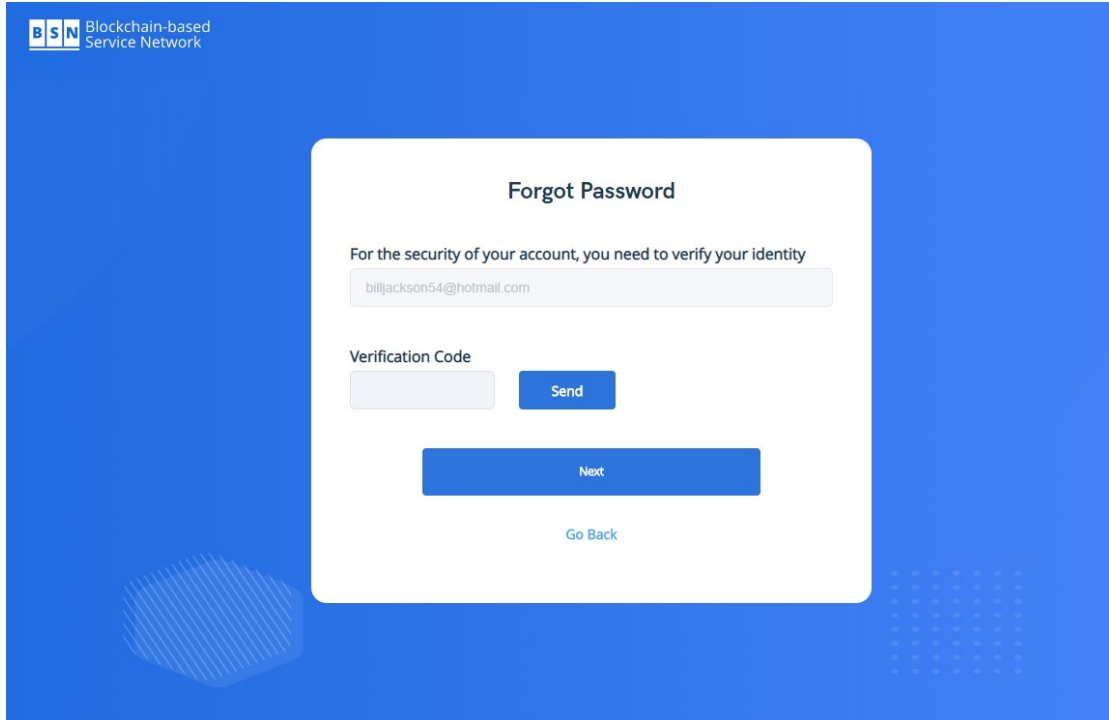
Verification Code

6x FQuX

**Next**

[Go back](#)

4. Click **Next** to view the **Authentication** page.
5. On the **Authentication** page, click the **Send** button to get verification code. This will generate a code that will be sent to your registered email address.



BSN Blockchain-based Service Network

### Forgot Password

For the security of your account, you need to verify your identity

billjackson54@hotmail.com

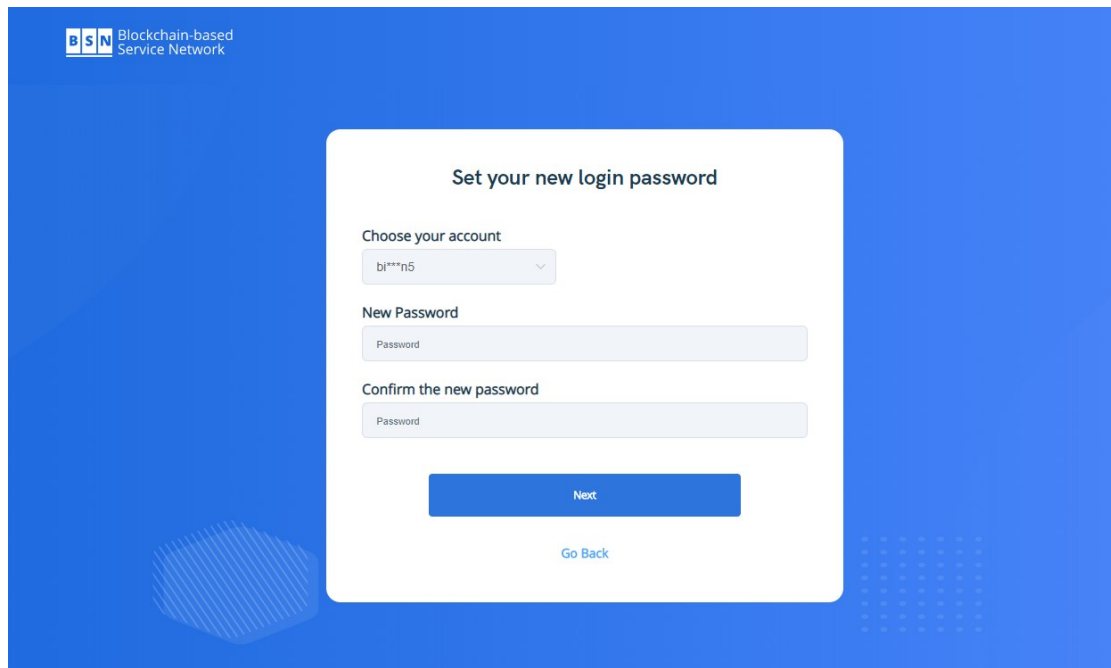
Verification Code

[Send](#)

[Next](#)

[Go Back](#)

6. Enter the code that was received in your mailbox and click **Next**.
7. On the reset login password page, enter your **New password** and **Confirm password**.
8. Click **Confirm** to change your password.

The screenshot shows a web interface for setting a new login password. At the top left, the BSN logo and text 'Blockchain-based Service Network' are visible. The main heading is 'Set your new login password'. Below this, there is a 'Choose your account' section with a dropdown menu showing 'bi\*\*\*n5'. The 'New Password' section has a text input field with a 'Password' label. The 'Confirm the new password' section also has a text input field with a 'Password' label. At the bottom, there is a blue 'Next' button and a 'Go Back' link.

## 5 Permissioned Services

### 5.1 Overview

The Permissioned service is one of the core services provided by BSN. Its goal is to make it easy for developers to publish decentralized applications (DApps) based on the framework of the permissioned blockchain on their selected public city nodes. Compared with the permissionless blockchain DApp, the permissioned blockchain DApp is more flexible in terms of architecture design, operation efficiency, and smart contract programming. It also has a larger space for innovation. However, from the perspective of development, because the developers need to build their underlying environments, and the environment for the public chain is readily available, the development, operation and maintenance of the permissioned chain DApp are relatively difficult. The developer's off-BSN system can access to DApp for data processing through the BSN public city node gateway.

Although BSN has greatly reduced the difficulty of permissioned blockchain DApp development, developers still need to have an in-depth understanding of the following three aspects which will be explained in detail in the following chapters.

1. **Keys and Certificates Mechanism:** the blockchain application itself is based on encryption algorithm technology, so the requirements of the keys and certificates are very high.
2. **DApp services publication and participation:** To build a permissioned blockchain DApp, the developer should firstly set up the chain and deploy the smart contracts. This part is entirely carried out on the BSN global portal ([www.bsnbase.io](http://www.bsnbase.io)), including the operations of smart contract upload, certificate mode selection, role's permissions setting, peer node configuration, public city node location, etc. Finally,

developers need to upload or download keys to facilitate the access from off-BSN system.

3. Off-BSN system access: This part contains a detailed description of the access parameter configuration, SDK usage, and the description of public city node gateway APIs to which the off-BSN systems connect. The API section includes all APIs of the currently permissioned blockchain frameworks that BSN has adapted.

## 5.2 BSN Keys and Certificates Mechanism

### 5.2.1 BSN Keys and Certificates Mechanism

Once a publisher deploys a permissioned DApp on BSN, the off-BSN systems of all participants (including the publisher) connects to the DApp via the PCN gateways to execute and record transactions based on the DApp's smart contracts. During this process, the participants need two key pairs to complete all steps: the *DApp Access Key Pair* and *User Transaction Key Pair*. When publishing and deploying a DApp on BSN, its publisher can choose from two modes to manage the DApp's keys and certificates: *Key Trust Mode* and *Public Key Upload Mode*. The key trust mode means that the two key pairs and related certificates will be generated and hosted by BSN when a participant joins the DApp. The participant can then download the private keys from the BSN portal, and use them to access BSN and sign transactions sent to the DApp from the off-BSN systems. The public key upload mode means that the two key pairs will be generated and stored locally on the participant's off-BSN system, and the public key is uploaded via the BSN portal or PCN gateway API, to BSN to generate the certificates. Once a mode is selected for the DApp, it cannot be changed. We strongly suggest all developers use the public key upload mode which is much more flexible and secure.

1. DApp Access Key Pair based on Key Trust Mode: DApp access key pair is used to generate the certificate to access the PCN gateway. If the DApp is on Key Trust Mode, the key pair can be generated on the BSN portal, and the private key can be downloaded. Please refer to the BSN Help Manual's service participation section.
2. User Transaction Key Pair based on Key Trust Mode: User transaction key pair is used to verify the requests and transactions sent to the DApp. If the DApp is on Key Trust mode, the key pair can be generated via the PCN gateway APIs by executing requests from the off-BSN systems. If the off-BSN systems have sub-users, it can even generate different key pairs for different sub-users. Refer to the API sections in this document for Hyperledger Fabric and FISCO BCOS frameworks to see how to generate the key pairs and use them to verify the transactions.
3. DApp Access Key Pair based on Public Key Upload Mode: In this mode, the DApp access key pair is generated and stored locally. The participant must upload the public key to BSN via the BSN portal to generate the access certificate to the PCN gateway. Please refer to section 5.2.2 below to see how to generate the key pair locally. Please refer to the "Public Key Upload" section of this document to learn how to upload the public key to BSN via the portal.
4. User Transaction Key Pair based on Public Key Upload Mode: In this mode, the

user transaction key pair is also generated and stored locally. Instead of using the BSN portal, the user transaction public key (one of the pair) is sent and registered on BSN via the PCN gateway certificate registration API. If the off-BSN systems have sub-users, they can also upload different public keys to generate different transaction certificates for different sub-users by using the API. Please refer to section 5.2.2 or the instructions inside the gateway SDK package about generating the key pair locally. Refer to the API sections in this document for registering the certificate via gateway APIs.

Please click the link to download the PCN Gateway SDK Package:

<https://github.com/BSNDA/PCNGateway-Go-SDK>

<https://github.com/BSNDA/PCNGateway-Java-SDK>

<https://github.com/BSNDA/PCNGateway-PY-SDK>

<https://github.com/BSNDA/PCNGateway-CSharp-SDK>

Currently, both permissioned frameworks Hyperledger Fabric and FISCO BCOS DApps support both Key Trust Mode and Public Key Upload Mode.

### 5.2.2 Locally generate the DApp access key pair

If the DApp service you participate in adopts Public Key Upload Mode for its application access key, you will need to generate the pair of public and private keys on the local client then save the private key locally and upload the public key to BSN via the portal.

It is recommended to use the latest version of OpenSSL to generate the keys. Please use the **prime256v1** cryptographic algorithm for Hyperledger Fabric and **secp256k1** for FISCO BCOS. The steps are as follows:

1. Preparation: Download the latest version of OpenSSL from <https://www.openssl.org/source/> and create a data.txt file in which some test phrases are entered, such as - Hello world.

2. Input "OpenSSL" in the terminal to show the open SSL command line.

```
OpenSSL>
```

3. Input the command - "ecparam -name **prime256v1** -genkey -out key.pem" to generate a private key file key.pem.

```
OpenSSL> ecparam -name prime256v1 -genkey -out key.pem
```

4. Input the command - "ec -in key.pem -pubout -out pub.pem" to generate a public key file pub.pem with the private key in the key.pem file.

```
OpenSSL> ec -in key.pem -pubout -out pub.pem
```

```
read EC key
```

```
writing EC key
```

5. Input the command - "dgst -sha256 -sign key.pem -out signature.bin data.txt" to sign the data.txt file with the private key in the key.pem file to generate the signature file: signature.bin.

```
OpenSSL> dgst -sha256 -sign key.pem -out signature.bin data.txt
```

6. Input the command - "dgst -verify pub.pem -sha256 -signature signature.bin data.txt". Use the public key in the pub.pem file to sign and verify the data.txt and signature.bin files.

```
OpenSSL> dgst -verify pub.pem -sha256 -signature signature.bin data.txt
```

```
Verified OK
```

7. If "Verified OK" is displayed, input the command - "base64 -in signature.bin -out signature64.txt" to convert the signature file signature.bin to base64 encoded signature64.txt.

```
OpenSSL> base64 -in signature.bin -out signature64.txt
```

8. Input the command - "pkcs8 -topk8 -inform PEM -in key.pem -outform PEM -nocrypt -out keypkcs8.pem" to convert the private key in the key.pem file to pkcs8 format.

```
OpenSSL> pkcs8 -topk8 -inform PEM -in key.pem -outform PEM -nocrypt -out keypkcs8.pem
```

9. Save the keypkcs8.pem file locally and copy all the contents of pub.pem, data.txt, and signature64.txt to the public key, test data, and signature data text boxes respectively on the Public Key Upload Mode page to verify the public key and submit it to BSN.

## 5.3 DApp Services Publication and Participation

### 5.3.1 Overview

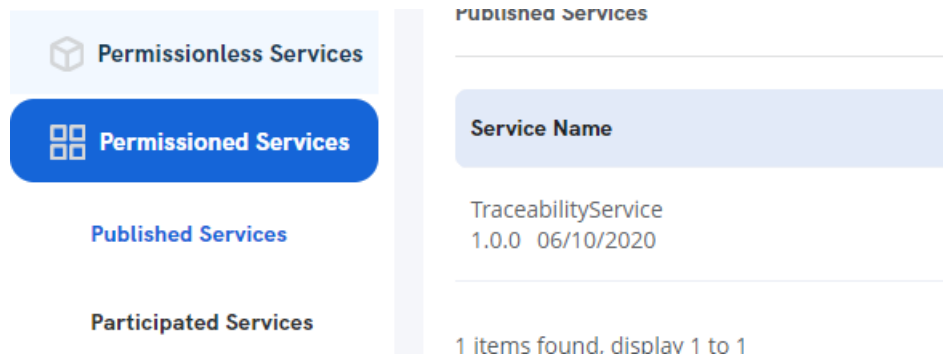
Permissioned DApp services refer to blockchain and DLT applications that are already deployed and operational on BSN. Users can use a BSN portal or the publisher's business system to apply to and join the service. Published services are private and cannot be browsed or searched by users through the BSN portal. DApp service participation must be initialized by the publishers' invitation links.


### 5.3.2 DApp Services Publication

#### 5.3.2.1 Create a New DApp Service

To create a new DApp service, follow these steps

1. In the **BSN** menu, click the **Permissioned Services** dropdown, in the list, click **Published Services** to open the **Published Services** page.



2. On the published services page, click the **Create Service** button.
3. In the **Basic Information** section enter or select the following:
  - **Service Name** – Enter an applicable name for the service to be provided
  - **Industry** – In the dropdown select from the various available service types
  - **Version** – The default version 1.0.0 is entered automatically. Unless necessary, leave it as is.
  - **Framework** – Select from **Fabric-1.4.3-secp256r1**, **FISCO-2.4-secp256k1** or **Fabric-2.3.2-secp256r1**
  - **Service Logo** – Click on the  icon to locate the image on your PC. Note that the image must be in png/jpg/jpeg format and should be exactly 160 x 160 pixels.

## Basic Information

* Service Name	<input type="text" value="Please enter Service Name"/>	* Industry	<input type="text" value="Supply Chain Management"/>
* Version	<input type="text" value="1.0.0"/>	* Framework	<input type="text" value="Fabric-1.4.3-secp256r1"/>
* Service Logo	<div style="border: 1px dashed gray; width: 100px; height: 100px; display: flex; align-items: center; justify-content: center; margin: 0 auto;">+</div>		

- **Service Introduction** – Enter a brief description of the service
  - **Service Description** – Enter a detailed description of the service including text and pictures where applicable
4. **Documents** – Documents can be added so that other users can easily understand your product. Click **Add** to display the **Add Document** dialog box. Click **Select** to locate the document on your PC.

## Documents &amp; Resources

Name	Type	Action
No Data		

Enter a **Name**, and choose a **Type** for the document. Click **Confirm** to add the

document.

5. In the **Contact Information** section, the login details of the user are automatically populated, including the **Contacts** and **Email**. If necessary, you can add a telephone number.

Contact Information

\* Contact Name  Mobile Number

\* Email

\* Publisher cannot publish company services in the name of an individual. If you publish company services, please register an account in the name of the company. Contact information must be true and valid.

[Next](#) [Cancel](#)

6. Click **Next** to continue.


### 5.3.2.2 Upload Chaincode Package

In the **Upload Chaincode Package** section, you can add your chaincode/smart contract package or use the preset chaincodes available in the system.

Upload Chaincode Package ⓘ

[+ Add Chaincode Package](#) [Use Preset Chaincode Package](#)

Chaincode Name	Version	Chaincode Package	Action
No Data			

1. To **Add Chaincode Package**, click on the button to display the **Add Chaincode Package** where you enter or select the following:
  - **Chaincode Name** – Enter a name for the chaincode
  - **Version** – Enter the chaincode version
  - **Chaincode Language** – Select from one the languages (Java, Golang or NodeJS)
  - **Initparam** – enter the initialization parameters and if multiple, separate it with commas
  - **Chaincode Package** – Click on the  icon to select the package file from the PC. Package files are to be in the .zip file format and the file name should only contain letters and numbers or underscores



Add Chaincode Package ?

\* Chaincode Name

\* Version

1.0.0


\* Chaincode Language

JAVA

Init Param

Split with commas.

\* Chaincode Package




Confirm

[Go Back](#)

2. To **Use Preset Chaincode Package**, click on the button to display the **Select preset chaincode package** option. In the list of packages, select one of the listed packages and click **Confirm** to add it.

Select preset chaincode package

<input type="checkbox"/>	Chaincode Name	Version	Download
<input type="checkbox"/>	bsnBaseCCEN	1.0.0	

1 items found, display 1 to 1

<

1

>

Confirm

[Go Back](#)

### 5.3.2.3 Define Service Functions and Roles

1. By selecting a **Preset Chaincode Package**, a set of automatic service functions are added to the service and each of the functions can be **Edited** or **Deleted**.

Define Service Functions ⓘ		+ Add Functions
SaveData	<a href="#">Edit</a> <a href="#">Delete</a>	
UpdateData	<a href="#">Edit</a> <a href="#">Delete</a>	
RemoveData	<a href="#">Edit</a> <a href="#">Delete</a>	
QueryData	<a href="#">Edit</a> <a href="#">Delete</a>	
Query historical data	<a href="#">Edit</a> <a href="#">Delete</a>	

2. If you wish to add more functions, click the **Add Functions** button to display the dialog box. In it, enter or select the following:

- **Function Name** – Enter a name for the function
- **Chaincode Name** – Select from the list of chain codes
- **Chaincode FUNC type** – Choose from **invoke**, **query** or **event**
- **Chaincode FUNC** – Enter a description of the function
- **Superior Functions** – Select a function from the list of functions in the system

Add Service Functions
×

\* Function Name

\* Chaincode Name

\* Chaincode FUNC Type
☒ invoke
☐ query
☐ event

\* Chaincode FUNC

\* Superior Functions

Confirm

[Go Back](#)

3. Click **Confirm** to add it to the functions.
4. When the **Use Preset Chaincode package** is selected, a system administrator role is automatically created with full access to the published service. To create another role, Click **Add Roles** to display the **Add Roles** function and enter or select the following:
- **Role Name** – Enter a role name
  - **Description** – Enter a description for the role
  - **Function Permissions** – Choose one or more from the DApp's existing functions, for example: **SaveData**, **UpdateData**, **RemoveData**, **QueryData**, and **Query historical data** from the **Preset Chaincode Package**.

**Add Roles**

\* Role Name [?](#)

Description [?](#)

\* Function Permissions [?](#)

☐ SaveData

☐ UpdateData

☐ RemoveData

☐ QueryData

☐ Query historical data

**Confirm** [Back](#)

5. When done, click **Confirm** to add the role.

#### 5.3.2.4 Select the Public City Nodes to deploy the service

Public city nodes are used by permissioned DApp publishers to deploy DApp's peers and smart contracts. Publishers can deploy all peers into one or more PCNs, so that all peers connect together to form the DApp. We strongly suggest not to deploy all peers onto one single PCN for data safety reasons. To add a public city node, follow these steps

1. In the **Select the City Nodes to deploy the service** section, click **Add City Nodes**.

Select the city nodes to deploy the service [?](#)

**Add City Nodes**

City Nodes	TPS	Capacity(GB)	Peers	TPS Price (USD/Month/Peer)	Capacity Price (USD/Month/Peer)	Data Usage (USD/GB)	Action
No Data							

Total: 0.00 USD

2. In the **Add City Nodes** window, enter or select:

- **Name** – Enter a name for the city code
- **Capacity (GB)** – 10 GB is allocated by default
- **Available TPS** – 10 is allocated by default
- **Cloud Provider** – All carriers are listed, however, if you prefer a particular carrier, click the dropdown and select that carrier

3. Click **Search** to list cloud providers.

- In the list of carriers, select more than one carrier for redundancy purposes. When done, click **Confirm**.

**Add City Nodes**

Name:  Cloud Provider:

Available TPS:  Capacity(GB):

<input type="checkbox"/>	Name	Available Peers	TPS Price (USD/Month/Peer)	Capacity Price (USD/Month/Peer)	Data Price (USD/GB)	Cloud Provider	Address
<input type="checkbox"/>	peer1.nodetest.bsnb...	13	6.73	0	0	AWS	123213
<input type="checkbox"/>	orgc	2	6.73	3.08	87.21	ecloud	test
<input type="checkbox"/>	orgb	1	6.73	3.08	0.11	ecloud	Beijing
<input type="checkbox"/>	js1	5	6.73	3.08	0.19	tencent	Beijing
<input type="checkbox"/>	orga	2	6.73	3.08	2.34	ecloud	Beijing

19 items found, display 1 to 5

The city nodes that have enough resources according to the TPS and storage configuration are displayed alongside their costs. The resource costs are different for each public city node.

### 5.3.2.5 Select Certificate Mode

There are two certificate modes, **Key Trust Mode** and **Public Key Upload Mode**. The key trust mode certificates are generated and hosted by BSN while the public key upload mode certificates are generated by developers, and the private key is stored locally and the public key is uploaded to BSN. It is recommended that all developers use the **Public Key Upload Mode**.

- To use the certificate mode, in the **Certificate Mode** section, click either **Key Trust Mode** or **Public Key Upload Mode**.

#### Certificate Mode ?

Participant's Certificate Mode: ☒ Key Trust Mode ☐ Public Key Upload Mode

- Click **Next** to continue.

### 5.3.2.6 Pay bills and submit for approval

In the bill detail section, the resource usage fees from the added city nodes are displayed alongside a monthly total payment fee. If the bill is satisfactory, click the **Confirm** button to proceed and make the payment. However, if you need to make changes to the bill, click **Back** and make changes in the **Add City Nodes** section.

Billing Summary

City Nodes	TPS	Capacity(GB)	Peers	Data Usage (USD/GB)	TPS Price (USD/Month/Peer)	Capacity Price (USD/Month/Peer)
orgc	10	10	1	87.21	6.73	3.08
orgb			1	0.11	6.73	3.08
js1			1	0.19	6.73	3.08

Total Amount:

☒ 112.59 USD Per Month

☐ 1228.53 USD Per Year (Discounts 122.55 USD)

Once the payment is successfully made, you will receive an email in your mailbox informing you that your BSN service has been submitted successfully and will be reviewed. You will be informed via email when the reviewed has finished.

Once the service has been approved, the service will be seen in the **Published Services** section.

Published Services					
Service Name	Framework	Participants	Status	Payment Status	Action
PolyCrossChain 1.0.1 02/03/2021	Fabric-1.4.3-secp256r1	1	Published	Paid	***

### 5.3.3 DApp Services Management

After the request for a service approval has been given, it will be listed in the Permissioned Services - **Published Services** section. For each listed service some Actions can be carried out. This includes **Invite Participants**, **Edit Basic Information**, **Service Upgrade**, **Configuration Upgrade** and **Details**.

#### 5.3.3.1 Invite Participants

After the service has been approved and the service is in use, you can invite other users of the blockchain network to participate in your service. To invite participants, follow these steps:

1. In the **BSN** menu, click the **Permissioned Services** dropdown and click **Published Services** to display the list of published services.
2. In the **Action** column, select the **Invite Participants** link to display the details to send to participants who intend to join the service.

Click **Copy** to copy the link details. This can be emailed to the participants who login with their BSN credentials to join or register with BSN first to use the service.

Invite Participants

✕

Please copy the following URL and send it by SMS or Email to the users you want to invite

```
https://global.bsnbase.com/g/home/PermissionedServices/ParticipatedServices/ServicesDetail?
types=8202DA05AB5E19006CEC8C1934B9F0BF&appCode=5C12A75B67A30B96B3402C968498CB57F709AE3A87C5D70D9550385F9D24
A4A8&appld=D89FB4771413A9E10A87C5EA40A7F4D3&auditState=DE0150CDBA08364042C9DF1C1E1306DD&appJoinAuditId=5C12A75
B67A30B96B3402C968498CB57F709AE3A87C5D70D9550385F9D24A4A8
```

Copy

Cancel

### 5.3.3.2 Edit Basic Information

After the service has been running and participants have joined, the publisher can edit basic information regarding the service including **Service Name**, **Industry**, **Framework**, **Version**, **Service Logo**, **Documents**, and **Contact Information**. To edit the basic information, follow these steps:

1. In the list of published services, locate the service to be edited. In the **Action** column of the service, select **Edit Basic Information** to display the editing page.
2. Add, edit or remove the basic detail of the service and click **Save** to store changes. If no changes were made click **Back** to return to the **Published Services** page.

### 5.3.3.3 Service Upgrade

After a service has been published, the publisher can use the **Service Upgrade** option to update the smart contracts and other functions. It will be reviewed again before it can be used. To edit the **Service Upgrade**, follow these steps:

1. In the list of published services, locate the service to be edited. In the **Action** column of the service, choose **Service Upgrade**.
2. In the **Basic Information** page, change the **Version Number**, which is mandatory and/or any other details in the **Basic Information** page. Click **Next** to upload the new smart contracts and set functions and roles as described before.

When done, click **Confirm**

### 5.3.3.4 Configuration Upgrade

In order to join the DApp services, the publisher should send out invitation links to the potential participant. The potential participant can then click on the link to the services' main page and apply for the service.

To upgrade the configuration, follow these steps:

1. Go to **Published Services** and select the enabled service on the list. Click **configuration upgrade** to enter the configuration upgrade list page as below:

Published Services / Configuration Upgrade

Configuration Upgrade

DApp number	Submit Time	Amount(USD)	Status	Action
Nothing to show here				

< 1 >

[Add](#)

- Click **Add** to create a configuration upgrade application form, and then click **Add city nodes** to add new city nodes:

Published Services / Create A New Service

Billing Summary

City Nodes	TPS	Capacity(GB)	Peers	Data Price (USD/GB)	TPS Price (USD/Month/Peer)	Capacity Price (USD/Month/Peer)
Sydney	10	10	1	0.21	29.48	0.10
Singapore			1	0.19	18.84	0.08
Paris			1	0.16	27.43	0.10

Total Amount: ☒ 78.55 USD Per Month ☐ 860.56 USD Per Year (Discounts 82.04 USD)

Note:   
 \* This time you need to pay 78.55USD, and subsequent periodic charges will be made from the payment method you choose.   
 \* This payment does not include the data usage fee. the data usage fee is calculated by the actual usage per week and deducted automatically.   
☐ Read and Agree to Terms of Service [the BSN service publishing agreement](#).

[Confirm](#) [Back](#) [Back to Published Services](#)

- Click **Submit** to submit the configuration upgrade application. When submitting, the system will prompt the publisher to pay the corresponding configuration upgrade fee. After the publisher confirms, the system generates the configuration upgrade bill and deducts money from the user's credit card. Whether the payment is successfully charged, or not, the configuration upgrade application will go through the review process. If the payment is successfully charged and the application is approved, the system will conduct a configuration upgrade process and complete the upgrade; if the charge fails, the bill will be kept for 72 hours and then expires. If the publisher still wants to upgrade the configuration, he/she needs to apply again.

Note: The fee paid when configuring the upgrade is the upgrade fee, which makes up the difference in the remaining payment period between the pre-upgrade configuration and the post-upgrade configuration of the billing cycle. After the upgrade is successful, future charges will be made according to the new configuration from the next period.

### 5.3.3.5 Details

The **View** option allows the publisher to view all the details of the published service including **Basic Information**, **Chaincode and Deployment**, **Roles**, **Review Records**, **Operating Status**, **Comments/Inquiries**, and **Historical Version**. To view these options, follow these steps:

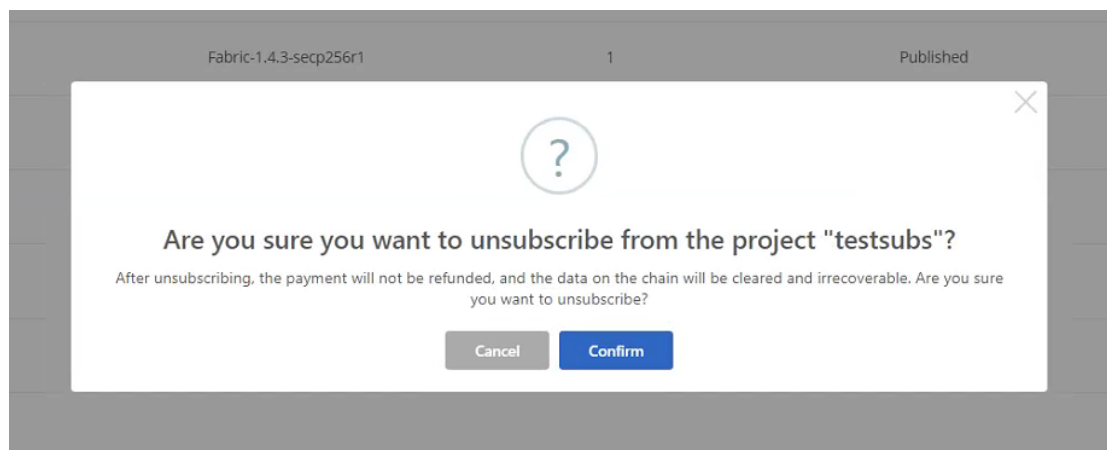
- In the list of published services, locate the service to be edited. In the **Action**

column of the service, click **Details** to display the view page tabs.

2. In the **Basic Information** tab, you can see all the details of the service that has been deployed including the **Service Name, Industry, Version, Framework, Service Logo, Service Introduction, Service Description, Documents, and Contact Information.**
3. In the **Chaincode and Deployment** tab, the information that can be viewed includes the **Chaincode Package, Service Functions, and City Nodes.**
4. In the **Roles** tab, the roles and their related functions are listed. To **View** a role, click on the **view** link for that role name.
5. In the **Review Records** tab, you will see all the requested approval and their status as well as time logs.
6. The **Operating Status** tab shows more information about the published service than any other tab. It shows the parameters of **City Nodes, number of transactions, Peer Information, Chaincodes, Blocks, and Logs** of how the activities took place.
7. The **Comments/Inquiries** tab shows the comments made on the published service that can be viewed by the publisher.
8. The **Historical Version** tab shows the history of the service including the **Service Name, Version, Industry, Service Introduction, and Action.**

#### 5.3.3.6 Service Unsubscribe

Users can unsubscribe published services. Click **Unsubscribe** in the published service list.



For users whose service resources are paid monthly, no refund will be generated when they unsubscribe; for users whose service resources are paid annually, refunds will be made at the point of time from the next month to the end of the billing cycle, and the refundable month will be cancelled when the refund is made, and the refund will be tallied according to the actual remaining months.

#### 5.3.4 DApp Services Participation

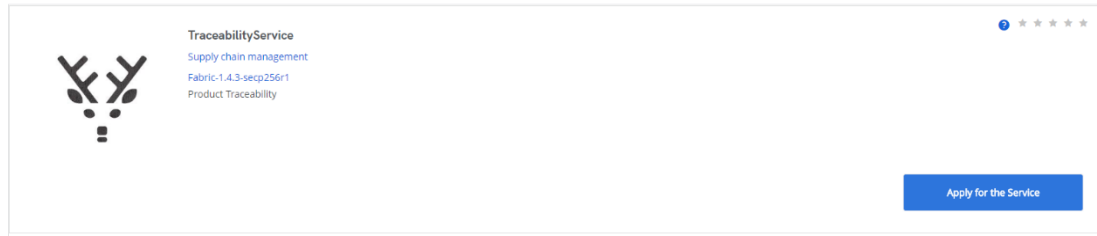
In order to join the DApp service, the publisher should send out invitation links to the potential participant. The potential participant can then click on the link to the services' main page and apply for the service.



### 5.3.4.1 Apply for a Service

To apply for a service, follow these steps:

1. Click the link that was shared. This will take you to the service information page.
2. In the service header, click **Apply for the Service**.



### 5.3.4.2 Select Roles and City Nodes

1. In the list of roles, select a role you want to use. You can click the **View** link in each of the roles to see more details about the role. You can select more than one role.

Choose a Service Role		
<input type="checkbox"/>	Name of Role	Action
<input type="checkbox"/>	Producer	<a href="#">Details</a>
<input type="checkbox"/>	Logistics	<a href="#">Details</a>
<input type="checkbox"/>	Retailer	<a href="#">Details</a>

2. In the Public **City nodes**, click **Add city nodes** to display the Public City Nodes the DApp is deployed on. You can select more than one node. The selected nodes' gateways are where the off-BSN systems connect to. Please select the public city node that is closest to you.

Add City Nodes

×

Cloud Provider

Please select

▼

Name

Name

Search

Reset

<input type="checkbox"/>	Name	Address	Cloud Provider
<input type="checkbox"/>	Sydney	Sydney, Commonwealth of Australia	AWS
<input type="checkbox"/>	Singapore	Singapore	Aliyun
<input type="checkbox"/>	Paris	Paris,French	AWS

3 items found, display 1 to 3

< 1 >

Confirm

Cancel

Click **Confirm** to view the nodes that were selected.

#### 5.3.4.3 Apply Certificate Mode

Depending on the settings of the service publisher, there are two certificate modes for service participation: Key Trust Mode and Public Key Upload Mode.

**Key Trust Mode:** Participants can select existing certificates on the city node or apply for a new certificate.

Set Password for The New Certificate

[Read Instruction](#)

\* Password

\* Confirm the new password

The certificate password cannot be recovered. Please keep it properly!

Confirm

Cancel

**Public Key Upload Mode:** Participants should upload the public key, test data and signature data. The generation of public and private keys can be viewed by clicking **Read Instruction**.

[Upload A New Certificate](#) [Read Instruction](#)

\* Public key

\* Enter test data

\* Signature data

Test

Confirm

Cancel

#### 5.3.4.4 Submit for approval

Click **Confirm** to join the service pending the **publisher's** approval.

Participated Services						
Participated Services(1)						
Service Name	Version	Platform Type	Publisher	Participation Time	Status	Action
TraceabilityService	1.0.0	Fabric-1.4.3-secp256r1	retailer	--	To be reviewed	***

#### 5.3.4.5 Approve the Request

As the publisher of a service, in the **Participation Management** list, the publisher can approve, deny or disable a participant from using the service. To perform any of these actions, follow these steps:

1. In the **Participation Management** section, locate the participant to review.

Service Participation Management					
Participants	<input type="text"/>	Service Name	<input type="text"/>	<a href="#">Query</a>	<a href="#">Reset</a>
Participants	Service Name	Version	Application Time	Status	Action
Bohan Shi	TraceabilityService	1.0.0	(UTC+8:00) 08/10/2020 01:03:04	To be reviewed	<a href="#">Approve</a> <a href="#">Details</a>
Bohan Shi	TraceabilityService	1.0.0	(UTC+8:00) 07/26/2020 16:15:33	Publisher not approved	<a href="#">Details</a>
olakunzo	TraceabilityService	1.0.0	(UTC+8:00) 07/05/2020 20:50:03	Confirmed	<a href="#">Edit Roles</a> <a href="#">Details</a>
Bohan Shi	TraceabilityService	1.0.0	(UTC+8:00) 07/01/2020 21:57:22	Publisher not approved	<a href="#">Details</a>
Bohan Shi	TraceabilityService	1.0.0	(UTC+8:00) 06/24/2020 13:36:49	Publisher not approved	<a href="#">Details</a>

2. For the participant to be reviewed, click the **Review** link in the **Action** column to view the participant details. In the **Review Result** section select either **Approved** or **Not Approved** and write a comment in the **Comment** box to give details.

### 3. Click **Confirm** or **Back** to return to the participant's list.

Approval Information

\* Approval Result: ☒ Approved ☐ Failed to Approve

\* Approval comments:

Submit for Approval
Go Back

4. If the participant is approved, a message will prompt showing that the service participation approval was successful.
5. After the approval has been given, the participant can view the service from their **Participated Services** page as well as add more **city nodes**.

#### 5.3.4.6 Download and renew a certificate

The BSN development team intends to build BSN into a most secure blockchain infrastructure network. The certificate and key mechanisms of BSN are complex. There are two kinds of key pairs used in generating certificates: DApp Access Key Pair and User Transaction Key Pair. For each, there are two modes, the Key Trust Mode and the Public Key Upload Mode. To work with certificates, follow these steps:

Key Trust Mode:

1. In the **My Certificates** menu, click **Key Trust Mode**. The certificate page will be displayed.

Key Trust Mode    Public Key Upload Mode

Service Name	TID	AppCode	Certificates	City Nodes	Password	Action
TraceabilityService	45202d9bb168477080d9ee5e02a41...	app0003202006101817263357932	USER0003202006101723379147576...	Paris	ab***23	<span style="color: #007bff;">⬇</span> Certificate update

2. To download the certificate, click the  icon. You will be required to enter the certificate password.

Fill in your certificate password
✕

\* Password

Confirm
Cancel

3. To update the certificate, click the **Certificate update** link. You will be requested to set a password for the certificate and confirm the password.

**Set Password for The New Certificate** ×

\* Password

\* Confirm Password

The certificate password cannot be recovered. Please keep it properly!

Confirm

Cancel

4. Click **Confirm** to update the certificate.

Public Key Upload Mode:

1. In the **My Certificates** menu, click **Public Key Upload Mode**. The certificate page will be displayed.

Key Trust Mode [Public Key Upload Mode](#)

Service Name	TID	AppCode	Certificates	City Nodes	Action
FlyingUnicorn	d77498a0967349a19454cb3a8d757893	app0003202007061026339883125	USER0003202006082324013815938_1	Sydney	<a href="#">Certificate update</a>

2. To update the certificate, the public key, test data and signature data need to be re-uploaded, and the update can only be completed after the test passes.

**Tips** ×

\* Public key

\* Enter test data

\* Signature data

View public/private key generation instructions [Read Instruction](#)

Test

Confirm

Cancel

3. The user only needs to upload the public key in the Public Key Upload Mode. The private key is kept locally by the user, so there is no need to download the certificate.

#### 5.3.4.7 Configuration parameters for service access

To view and download the configuration parameters, follow these steps:

1. In the **Permissioned Services** menu, click **Participated Services**.
2. In the list of services, click the **Detail** option in the **Action** column for the service.

Participated Services

Participated Services(1)

Service Name	Version	Framework	Publisher	Participated Date	Status	Action
PolyCrossChain	1.0.1	Fabric-1.4.3-secp256r1	billjackson5	02/03/2021	Confirmed	...

1 items found, display 1 to 1

[Settings](#)  
[Details](#)

3. Click the dropdown beside the **configuration parameters for service access** to view its configuration.

Configuration parameters for service access Download the configuration parameters

UserCode: USER0003202106251751437399976

APPCode: app0003202107141032301483410

Channel name: app0003202107141032301483410

Chaincode Name	Chaincode deployment name	Chaincode address	Function Name	FUNC
bsnBaseCCEN	cc_app0003202107141032301483...		Query historical data	getHistory

4. To download the **parameters for service access**, click **Download the configuration parameters** to begin the download.

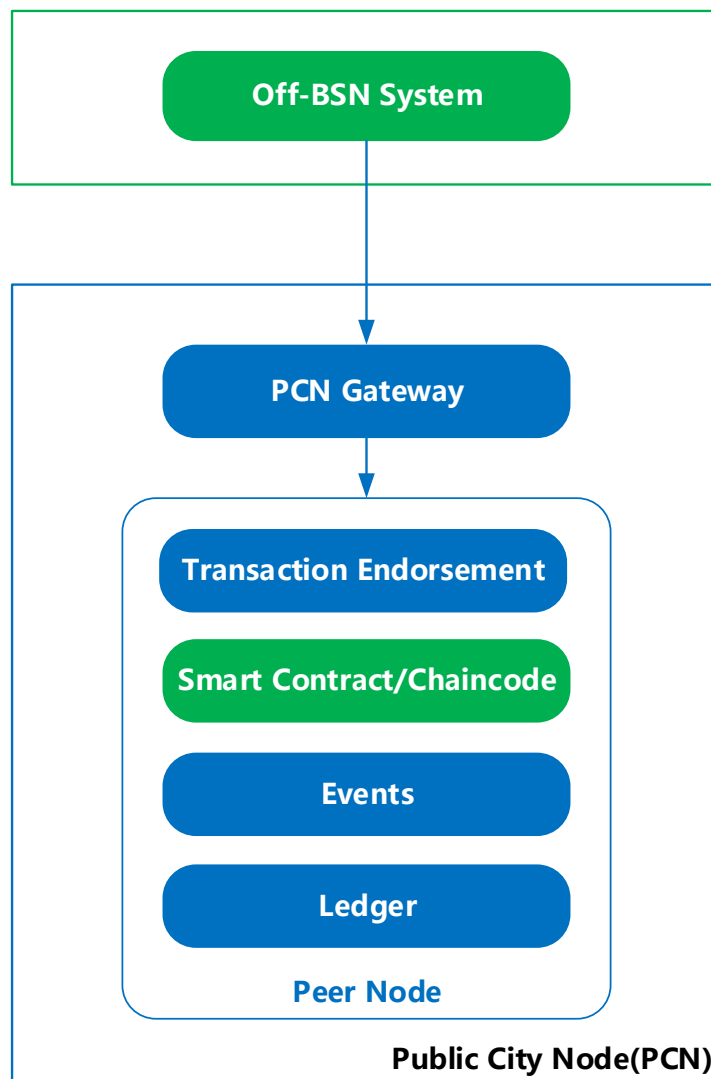
## 5.4 Off-BSN System Access Guide

### 5.4.1 Overview

Blockchain-based Service Network (hereinafter “Service Network” or “BSN”) is a cross-cloud, cross-portal, cross-framework global infrastructure network to deploy and operate all types of blockchain and distributed ledger technology (DLT) applications (DApp).

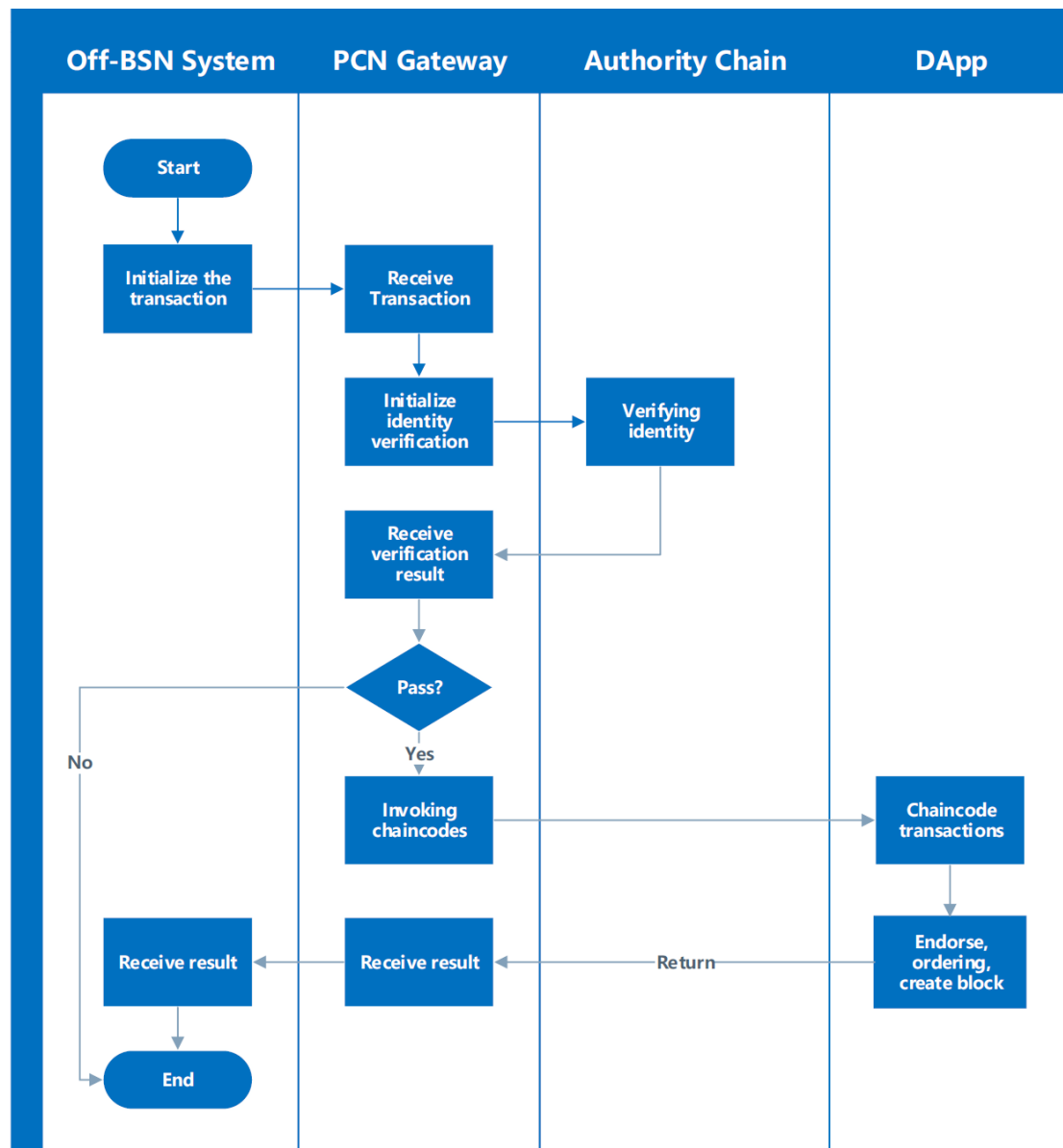
BSN aims to lower the cost of developing and deploying DApps by providing public blockchain resources and environment to developers, just like the internet. It can further reduce the costs associated with the development, deployment, operations, maintenance, and regulation of DApps and, thereby, accelerate the development and universal adaptation of blockchain and DLT technologies.

A complete DApp system based on BSN generally consists of two parts: the on-BSN DApp smart contracts and the off-BSN systems. The off-BSN systems use the BSN Public City Note (PCN) gateways to invoke the DApp smart contracts deployed on the PCN to carry out on-chain operations such as executing transactions, writing data chain, data queries, etc. The DApp service publishers and participants can deploy their off-BSN systems on any cloud services they choose and then connect to the BSN PCN gateways through the internet access DApp smart contracts and data.



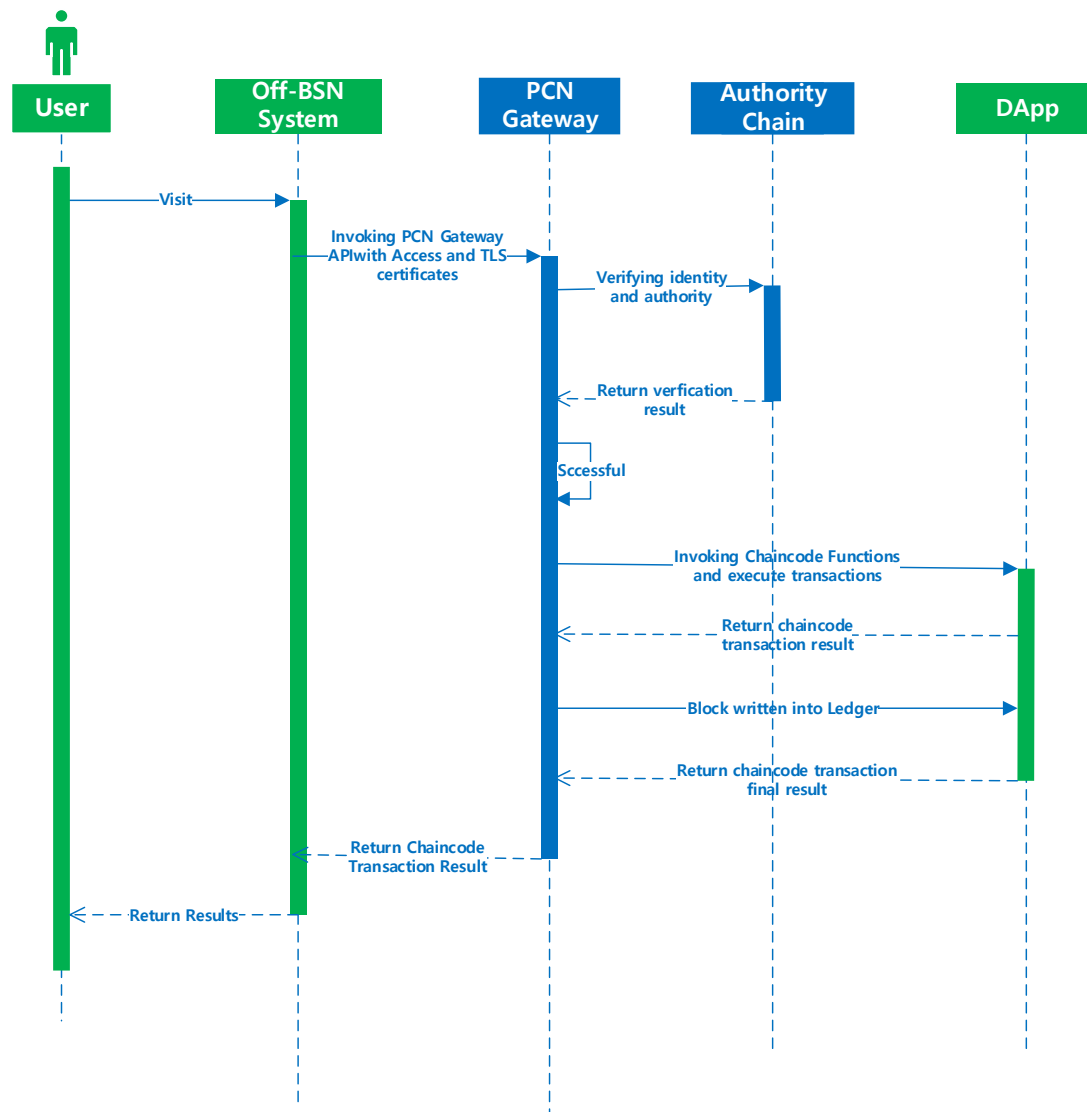
The BSN DApp service publishers and participants should have their off-BSN systems so that they can access the DApp smart contracts to execute transaction and query data via the PCN gateway APIs. The following are the charts to show the connecting flow and transaction sequences.

Off-BSN System Connection Flow:





Off-BSN System calling sequence:



## 5.4.2 BSN Smart Contract Package Requirements

A smart contract, also known as chaincode in Hyperledger Fabric, is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without a third party. These transactions are trackable and irreversible. A smart contract is invoked to automatically execute a transaction and operate ledger data. A DApp service on BSN can deploy multiple smart contracts. Each smart contract can contain multiple functions.

### 5.4.2.1 Hyperledger Fabric smart contract package requirements

Hyperledger Fabric (“Fabric”) chaincode can be compiled by multiple programming languages, including Golang, java, and node.js. Each chaincode program must implement a chaincode interface which usually consists of three basic functions: Init, Invoke, and Query.

- **Init:** This function is called during the chaincode instantiation and its purpose is to prepare the ledger for future requests. This function must be implemented in all chaincodes.
- **Invoke:** The Invoke function is called for all future requests from the off-BSN systems towards the DApps. Here all DApp custom functions or what the DApps can do (for example, to read data from the ledger, to write data in the ledger, to update data, to delete data) are defined. Simply put, Invoke can be understood as an entry point to the chaincode functions. The Invoke function also must be implemented in all chaincodes.
- **Query:** The Query function provides a method of querying ledger data. This function can only be used for query purposes and does not offer any operations of ledger data. The Query function is not required to be implemented in all chaincodes.

Note: Fabric 1.4 chaincode package cannot be directly used in Fabric 2.3.2, you need to modify the contract according to the latest chaincode dependencies with the corresponding language.

To realize the automatic deployment of DApp services and to improve deployment efficiency, the following Fabric chaincode packaging requirements have been issued with different programming languages.

#### 1. Golang

The main function must be at the same or higher level as all chaincodes in the project. The zipping path must be the same level folder where the main function is located, and the main function path is the src-based path.

Example: BsnBaseCC Package (the preset chaincode package)

```
BsnBaseCC
├─main.go
├─ChainCode/
├─models/
└─utils/
```

The package should be zipped under BsnBaseCC/ (package name is not required), and the main function path (reference path) is BsnBaseCC.

Example: FabricBaseChaincode chaincode package on github (preset chaincode package)

github.com

```
└─BSNDA
  └─FabricBaseChaincode
    └─chaincode
      └─go
        └─bsnBaseCC
          └─main.go
            └─ChainCode/
              └─models/
                └─utils/
```

It should be zipped under

github.com/BSNDA/FabricBaseChaincode/chaincode/go/bsnBaseCC/ (package name is not required), and the main function path (reference path) is  
github.com/BSNDA/FabricBaseChaincode/ chaincode/go/bsnBaseCC.

Description: main.go: the entry; ChainCode: chaincode; models: entities; utils: utilities.

Note: Below is the structure of the Fabric 2.3.2 preset chaincode package

chaincode-demo

```
└─main.go
└─chaincode/
└─vendor/
└─go.sum
└─go.mod
```

## 2. Java

gradle or maven-built projects, the projects must contain build.gradle or pom.xml files.

Example: BsnBaseCC package

BsnBaseCC

```
└─build.gradle
```

```
└─src
    └─main
        └─java
            └─com.example.javacc
                └─javacc.java
```

Package must be zipped under BsnBaseCC/, and there is no requirement for the name of .zip package.

Note: src/main/java: project directory; com.example.javacc: package name; javacc.java: chaincode information

### 3. Node.Js

package.json file must be built into the project's root directory. Package needs to be zipped under the directory of BsnBaseCC/. There is no requirement of the name of .zip package.

Example: BsnBaseCC package

BsnBaseCC

```
└─marbles_chaincode.js
└─package.json
```

Description: marbles\_chaincode.js: chaincodes

Note: when publishing DApp services in the BSN portal, chaincode packages should be created in the project's root directory using .zip format.

#### 5.4.2.2 Hyperledger Fabric preset smart contract package

A preset chaincode package (Golang) is provided to BSN developers which contains basic functions such as add, delete, edit, and query. New DApp developers can learn from this package about Fabric chaincode programming and further extend the functions if needed. The chaincode in this package supports data types such as string, integer, float point, and sets (map, list), etc.

Please click this link to download:

Fabric 1.4.3:

<https://github.com/BSNDA/FabricBaseChaincode>

Fabric 2.3.2:

<https://github.com/BSNDA/FabricBaseChaincode/tree/master/chaincode/go/bsnBaseChaincode>

DApp publishers can also select the preset chaincode package directly from the DApp publishing page on the BSN portal.

The Preset Chaincode package functions are as follows:

1. Add data (set)

Input parameter description:

baseKey: a unique primary key identifier of data

baseValue: stored data information

Example: {"baseKey": "str", "baseValue": "this is string"}

Of which, the baseKey cannot be a blank string and the baseValue can be any type of data. If the baseKey already exists, then directly return that it already exists and cannot be added; if it does not exist, then add data.

2. Update data (update)

Input parameter description:

baseKey: a unique primary key identifier of data

baseValue: stored data information

Example: {"baseKey": "str", "baseValue": "this is string"}

Of which, the baseKey cannot be a blank string and the baseValue can be any type of data. If the baseKey does not exist, then it cannot be updated; if it already exists, then update the data.

3. Delete data (delete)

Input parameter description

baseKey: a unique primary key identifier of data

Example: "str"

Of which, the baseKey value cannot be blank and must exist, else it cannot be deleted.

4. Get data (get)

Input parameter description

baseKey: a unique primary key identifier of data

Example: "str"

Of which, the baseKey value cannot be blank and must exist, else it cannot be retrieved.

5. Get historic data (getHistory)

Input parameter description

baseKey: a unique primary key identifier of data

Example: "str"

Of which, the baseKey value cannot be blank. Response results: transaction Id (txId), transaction time (txTime), whether to delete (isDelete) and transaction information (dataInfo).

We welcome developers to share their custom chaincodes as preset chaincode packages for the BSN and work with us to expand the blockchain application support capabilities of the BSN.

#### 5.4.2.3 FISCO BCOS smart contract package requirements

To realize automatic audit and deployment of FISCO BCOS (FISCO) DApp services and to improve efficiency, the following FISCO smart contract packaging requirements have been issued:

##### 1. Package Structure of the Solidity smart contract

All smart contracts must be stored in a single-level folder including smart contracts, libraries, and external contract interfaces. Import method of all contracts is import “./XXXX.sol”.

##### 2. Smart Contract deployment instruction document (deploy.md)

deploy.md is used to explain clearly how the smart contract is initialized and deployed. It consists of three main parts:

- Contract Description: to briefly describe the basic information of each contract.
- User Description: to describe the basic information of each transaction signing users during initialization and deployment.
- Contract initialization description: to describe the steps of smart contract initialization and deployment, so that BSN tech personnel can follow to complete the process.

##### 3. Contract uploading specifications

When uploading a chaincode package (smart contract package), fill in the chaincode name (contract name) that is consistent with the main contract class name and the main contract file name.

Example: BsnBaseGlobalContract chaincode package (preset chaincode package)

BsnBaseGlobalContract

└─BsnBaseGlobalContract.sol

└─Table.sol

Package must be zipped under BsnBaseGlobalContract/. The zipped package name is not required. If the main contract class name is BsnBaseGlobalContract, the main contract file name should be BsnBaseGlobalContract.sol, and the chaincode name (contract name) must be filled in as BsnBaseGlobalContract.

##### 4. BSN Adaptation for FISCO Solidity Version Descriptions

Currently, FISCO BCOS in the BSN only supports Solidity 0.4.25 and older versions.

#### 5.4.2.4 FISCO BCOS preset smart contract package

The FISCO Preset Smart Contract package is chosen from the Table.sol provided by the FISCO BCOS development team, and can provide developers with basic functions such as insert, remove, update, or query (using Solidity). New DApp developers can learn from this package

about FISCO smart contract programming and further extend the functions, if needed. The stored data types supported by this smart contract include int256(int), address, and string, of which string cannot exceed 16MB. To ensure on-chain performance, there is no analysis of duplicate base\_id and base\_key. This should be handled by the off-BSN system. It is recommended that each base\_id has only one corresponding base\_key and base\_value.

Please click this link to download:

<https://github.com/BSNDA/FISCOBaseContract>

The preset smart contract functions are as follows:

1. Insert data (insert)

Input parameter description

base\_id: the primary key identifier that requires inserting

base\_key: the key of the data to be inserted

base\_value: the value of the data to be inserted

Example: {"base\_id": "1", "base\_key": 1, "base\_value": "this is string"}

Of which, base\_id and base\_value cannot be blank strings and the base\_key is in int256 data type.

2. Update data (update)

Input parameter description

base\_id: the primary key identifier that requires updating

base\_key: the key of the data to be updated

base\_value: the value of the data to be updated

Example: {"base\_id": "1", "base\_key": "1", "base\_value": "this is string"}

Of which, base\_id and base\_value cannot be blank strings and the base\_key is in int256 data type. If the base\_id and base\_key do not exist, then they cannot be updated; if they already exist, then the data will be updated.

3. Remove data (remove)

Input parameter description

base\_id: the primary key identifier that requires removing

base\_key: the key of the data to be removed

Example: {"base\_id": "1", "base\_key": "1"}

Of which, the base\_id and base\_value cannot be blank and must exist, otherwise they cannot be removed.

4. Select data (select)

Input parameter description

base\_id: the value of the primary key identifier that requires being selected

Example: {"base\_id": "1"}

Of which, the base\_id cannot be blank and must exist, otherwise, it is not possible to select the corresponding data.

### 5.4.3 PCN Gateway Fabric API

A PCN gateway is deployed on each public city node (PCN) to receive off-BSN system requests signed and verified by DApp access keys. Then requests are routed to the corresponding Fabric-based DApp chaincodes. Invoking the PCN gateway is realized by sending HTTP requests to each PCN gateway service. The gateway is responsible for verifying user and application identities and then uses these identities and chaincode functions to process chaincode parameters and to send the chaincode transaction results back to the off-BSN systems.

#### 5.4.3.1 DApp Access Signature Algorithm

Whenever an off-BSN system sends requests to the PCN gateway, the HTTP request message should be signed with the participant's DApp access private key. When the PCN gateway receives the message with the digital signature, it will verify the authentication and message integrity with the corresponding hosted or uploaded DApp access public key. The gateway will only process the request message further after the verification is passed.

##### 1. Assemble signature string

Convert the request parameters into a joined string according to the order of the parameter table, of which the request parameter prioritizes joining UserCode and AppCode of the Header and the response parameter prioritizes joining code and msg. Then join the parameters in the Body according to the order of the parameter tables in the definition of APIs.

##### 2. Different type conversion formats

Type	Rule	Example	Result
String	No conversion	abc	Abc
Int/int64/long	Decimal conversion	-12	-12
Float	Decimal conversion; see notes for values after decimal point	1.23	1.23
Bool	Convert to "true" or "false"	true	True
Array	Join according to parameter sequence and type	{"abc", "xyz"}	Abcxyz
Map[key]value	Join key and value according to parameter sequence	{"a":1, "b":2}	a1b2
Object	Convert the attributes in the object one by one according to the document in the above-described format	{"name": "abc", "secret": "123456"}	abc123456

##### 3. Signature rules

- Getting the Hash value - The converted string to be signed is required to be computed with the SHA256 algorithm with UTF-8 encoding.
- Sign the Hash value - The hash value and private key should be encrypted with the ECDSA (secp256r1) algorithm. If signed with SHA256WithECDSA, which includes hash value computation, the first step is not necessary.
- Encoding the signature result to Base64.

##### 4. Example



Parameters:

```
{"header":{"userCode":"user01","appCode":"app01"},"mac":"","body":{"userId":"abc","list":["abc","xyz"]}}
```

Result: user01app01abcabcxyz

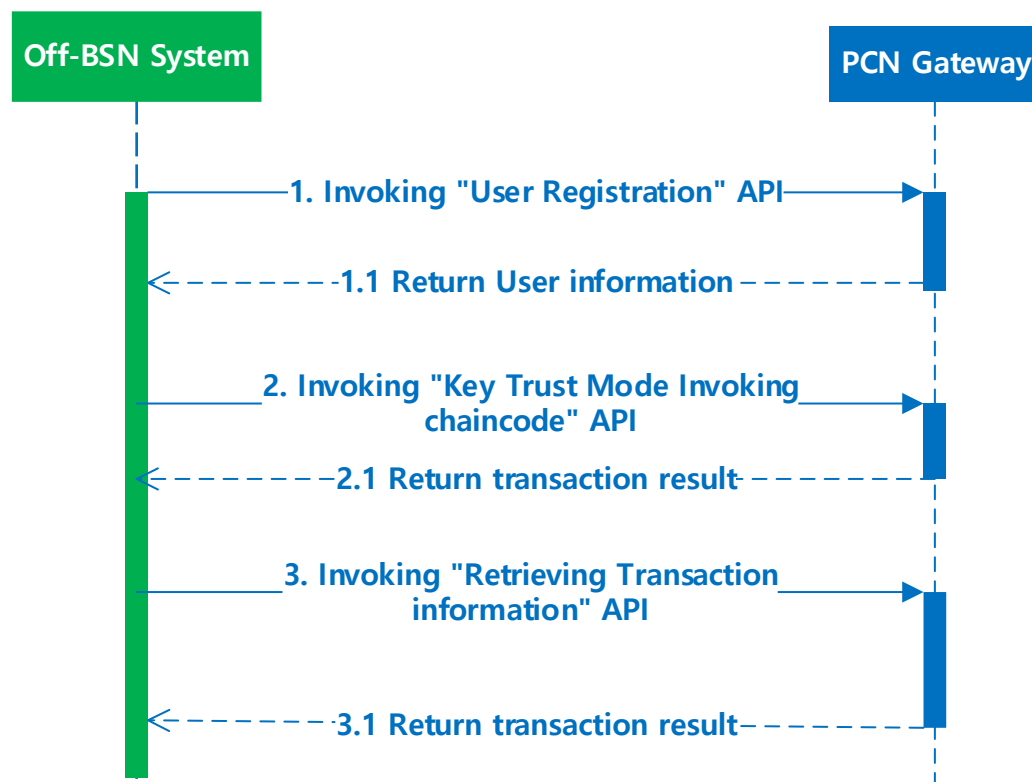
### 5.4.3.2 Keys and Certificate Modes

#### 1. Key Trust Mode

As described in chapter 5, DApp participants require two sets of key pairs to access the DApp: DApp access key pair and user transaction key pair. With key trust mode, the pairs are generated and hosted by BSN. The participants only need to download the private key (DApp access key) from the BSN portal.

- **DApp Access Key Pair:** After the participant has successfully joined the DApp, BSN will generate one key pair (private and public keys) that corresponds to the DApp's framework algorithms under the Key Trust Mode. The participant can download the private key from the "My Certificates" section of the BSN global portal and use it to sign the request message sent to the PCN gateway. The gateway will use the hosted public key from the generated key pair to validate the signature.
- **User Transaction Key Pair:** This is the identity of a participant used to invoke the chaincodes. Under the Key Trust Mode, after successfully joining a DApp, a participant's user transaction key pair will be created automatically by BSN by default. The participant's off-BSN system can use the participant's UserCode to invoke the certificate generated by the key pair. If the participant's off-BSN system has multiple sub-users, the off-BSN system can invoke the gateway's "User Registration API" to register the sub-users and generate separate user transaction key pair for each sub-user. The sub-users can use their UserCode to connect to the DApp to execute transactions.

Transaction process:

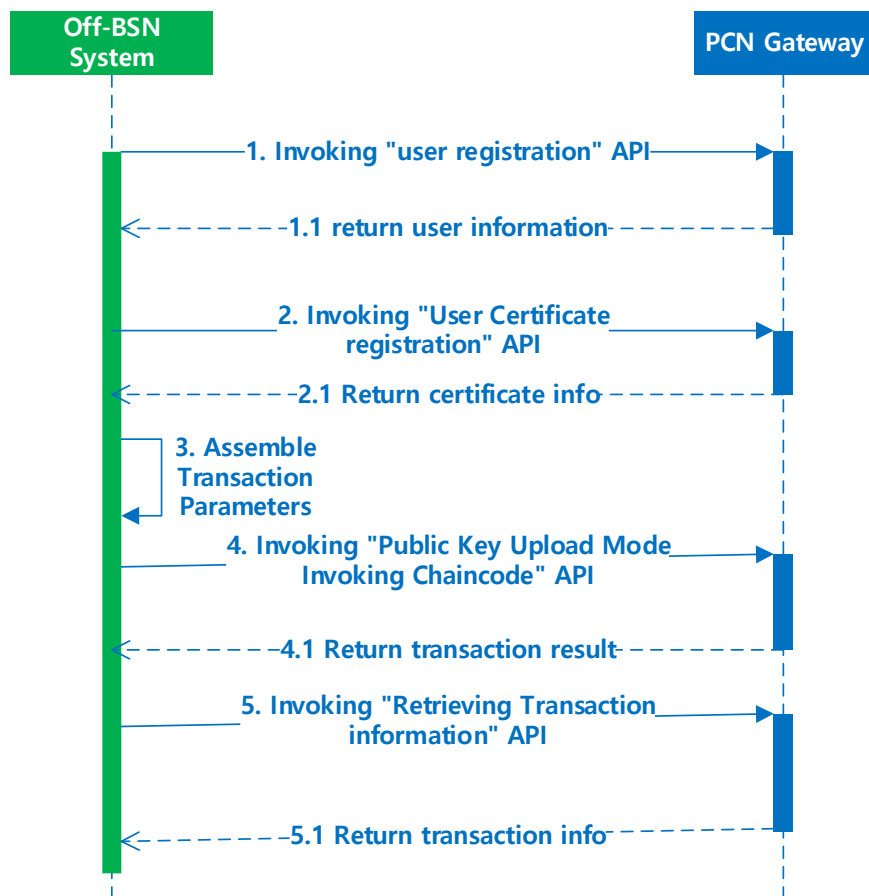


## 2. Public Key Upload Mode

As described in chapter 5, DApp participants require two sets of key pairs to fully access the DApp: DApp access key pair and user transaction key pair. With public-key upload mode, the key pairs are generated and stored locally by the participants. The participants only need to upload the public keys to BSN via the BSN portal or gateway APIs.

- **DApp Access Key Pair**: The DApp participant must generate the DApp access key pair locally according to the DApp framework algorithm after successfully joining the DApp. The participant stores the private key locally and uploads the public key to BSN via the BSN global portal. The participant's off-BSN system uses the private key to sign the transaction messages when invoking the PCN gateway. The PCN gateway will use the public key uploaded by the participant to verify the signature and validate the legality of the transaction.
- **User Transaction Key Pair**: This is the identity of a participant to invoke the chaincodes. Under the Key Trust Mode, the participant must generate the user transaction key pair locally and use the public key to generate the "public key registration application.", then from the participant's off-BSN system to submit the registration application to BSN by invoking the "Public Key Upload Mode user certification registration" API on the PCN gateway to receive the public key certificate. If the off-BSN system has sub-users, it should first invoke the "User Registration" API to register the sub-users before sending their public key registration applications.

Transaction process:



#### 5.4.3.3 Get DApp information API

Invoke this interface to get basic DApp information; this interface can be used with Public Key Upload Mode transactions.

1. Interface address:  
<https://PCNgatewayAddress/api/app/getAppInfo>
2. Call Method: POST
3. Signature Algorithm: Not Required
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Yes	
2	Body	body	Map	No	
3	Signature value	mac	String	Yes	
Header					
1	User unique ID	userCode	String	Yes	
2	DApp unique ID	appCode	String	Yes	
Body					
Example:					

```
{
  "header": {
    "userCode": "USER0001202004151958010871292",
    "appCode": "app0001202004161020152918451",
    "tId": "",
    "mac": "",
    "body": {}
  }
}
```

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: successful -1: failed
2	Response Message	msg	String	Y	
Body					
1	DApp name	appName	String	Y	
2	DApp type	appType	String	Y	
3	DApp encryption key type	caType	Int	Y	1: Key Trust Mode 2: Public Key Upload Mode
4	DApp algorithm Type	algorithmType	Int	Y	1: SM2 2: ECDSA (secp256r1)
5	City MSPID	mspId	String	Y	
6	DApp chain name	channelId	String	Y	Fabric corresponding channelId, fisco corresponding groupId
Example:					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac": "MEUCIQDE9zv0E/w4V/ILG6wUCFP08a7NDCAtX/loZOcCyY4gIQIgUTYWsFTA1KE88gE6452jKnnVBrhznGVOV2HPMCbNh8A=",   "body": {     "appName": "sdktest",     "appType": "fabric",     "caType": 2,     "algorithmType": 2,     "mspId": "OrgbNodeMSP",     "channelId": "app0001202004161020152918451"   } }</pre>					

### 5.4.3.4 User Registration API

In both Key Trust Mode and Public Key Upload Mode, when a user participated in a Fabric DApp wants to create a unique user transaction key certificate for a sub-user of the off-chain system, the off-BSN system should invoke the User Registration API to register the sub-users on the PCN first. A sub-user's username is name@appCode in the request parameters

#### 1. Interface address:

<https://PCNGatewayAddress/api/fabric/v1/user/register>

2. Call Method: POST
3. Signature Algorithm: required and refer to Section 5.4.3.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	signature value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	user name	name	String	Y	Length<20
2	user password	secret	String	N	For Key Trust Mode DApps, this can be blank; for public key upload mode DApp, if this is blank then return with a random password
3	extended properties	extendProperties	String	N	User extended properties, json format string
Example:					
<pre>{   "header": {     "userCode": "USER0001202101301022113596689",     "appCode": "app0001202101301022113596689",     "mac": "MEQCID3F4z2XpN4JsCU/gRO9l0Ziw1lOICx8eVgVWUVltvWyAiA1Y0uObgCV5tm1avSz9BscYr1aycmfBtFSlQ3o19OYEQ==",     "body": {       "name": "user20210927",       "secret": "123456",       "extendProperties": {         "key1": "value"       }     }   } }</pre>					

5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: successful -1: failed
2	Response Message	msg	String	Y	
Body					
1	user name	name	String	Y	Length<20
2	user password	secret	String	Y	For public key upload mode DApps, if the call parameter password is blank then return with a random password
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac": "MEUCIQC1fufMU8kRI1gMHIGqfWOH1iv2KIhS+H0dlIUdEuUrLQIgYJz98xp5w/KdVP6bJjHhV2pZPTe9Cn4xcOrPV4E7ZsA=",   "body": {     "name": "user01", </pre>					

```
"secret": "123456"
}
```

### 5.4.3.5 Invoke chaincode API in Key Trust Mode

For DApps in Key Trust Mode, when the off-BSN system invokes the chaincode functions via PCN gateway, it is required to insert the parameters in the request message. The gateway will return the response message from the chaincode.

- Interface address:

<https://PCNGatewayAddress/api/fabric/v1/node/reqChainCode>

This interface will directly return the response message without waiting for the generation of block. Please use “Get transaction information API” described in section 5.4.3.8 to check the status of a block generated based on transaction ID.

Note: After a user has successfully participated in a DApp service, this participant can view and download the DApp’s configuration parameters which are used for off-BSN systems to connect to this DApp’s chaincodes, including the PCN gateway address and Dapp access keys, as shown below:

The screenshot displays the 'Blockchain-based Service Network' dashboard. On the left is a navigation menu with options like Home, Permissionless Services, and My Certificates. The main content area shows a table of services. A red box highlights the 'Access Address' field in the top right, which contains the URL 'https://sydneynode.bsn.gateway.com:17602/api/node/reqChainCode'. Another red box highlights the 'Configuration parameters for service access' section, which lists fields: userCode (USER0003202006082324013815938), appCode (app0003202007061026339883125), tId (d77498a0967349a19454cb3a8d757893), and Channel name (app0003202007061026339883125). A third red box highlights a table of chaincodes, with columns for Chaincode Name, Chaincode deployment name, Chaincode address, Function Name, and FUNC. The 'FUNC' column lists methods: getHistory, get, delete, set, TestEvent, and update.

- Call Method: POST
- Signature Algorithm: required and refer to Section 5.4.3.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
3	User and DApp	tId	String	N	

	mapping ID				
Body					
1	user name	userName	String	N	
	random string	nonce	String	Y	Use 24 random byte array of the base64 encoding
1	chainCode	chainCode	String	Y	
2	function name	funcName	String	Y	
3	Call parameters	args	String[]	N	
4	Transient data	transientData	Map<string, string>	N	
Example:					
<pre>{   "header": {     "userCode": "USER0001202004161009309407413",     "appCode": "app0001202004161017141233920",     "tId": "",     "mac": "MEQCICJpE1jfeJKtw/ZboVuKSLy2RmmSdchrEVPGFJhm9IaIAiA/Qqs6RNz0ndSS4/AFSwBj7vC76Py1hXnqO5zMD9pNtA==",     "body": {       "userName": "",       "nonce": "lgH7Ozfv6npqg9D3pSbq9c6o+rAcpa5D",       "chainCode": "cc_app0001202004161017141233920_00",       "funcName": "set",       "args": [         {           "baseKey": "test2020048",           "baseValue": "this is string"         }       ],       "transientData": {}     }   } }</pre>					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	block information	blockInfo	blockInfo	N	If code is not 0, then leave blank
2	chaincode response result	ccRes	ccRes	N	If code is not 0, then leave blank
blockInfo					
1	Transaction ID	txId	String	Y	
2	Block HASH	blockHash	String	N	On synchronous mode returns Block HASH
3	status value	status	Int	Y	Refer to the detailed transaction status description in 5.4.3.17
ccRes					
1	chaincode response status	ccCode	Int	Y	200: Successful 500: Failed
2	chaincode response result	ccData	Str	N	Actual chaincode response result
Example					
<pre>{   "header": {</pre>					

```

"code": 0,
"msg": "Transaction Successful"
},
"mac":
"MEUCIQCbtO1AfYkoJ2hIlp8CfKK1iuhVEAYkPY8YFRAdvPJlAlgDjSqYgwlORJRyF6
KZPU/uC5Fx/DxXxu9VgKwU9+JhjU=",
"body": {
  "blockInfo": {
    "txId": "a144149150ee615a9d11c68485600f43dc2c3eb2a98d7b36de53a6b99e03c495",
    "blockHash": "",
    "status": 0
  },
  "ccRes": {
    "ccCode": 200,
    "ccData": "SUCCESS"
  }
}
}
}

```

#### 5.4.3.6 User certificate registration in Public Key Upload Mode

For DApps in Public Key Upload mode, after the participant registered the sub-users on the PCN by using “User Registration API” (section 5.4.3.4), he/she can use this interface to upload public key registration applications and receive the certificates (DApp access key pair certificates) for the sub-users. Invoking this interface from Key Trust Mode DApp will return an error message.

1. Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/user/enroll>
2. Call Method: POST
3. Signature algorithm: required and refer to Section 5.4.3.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	user name	name	String	Y	user name used at registration
2	user password	secret	String	Y	Password created at registration
3	Certificate Application file content	csrPem	string	Y	Use the ECDSA (secp256r1) algorithm to generate the certificate application



					file; the certificate CN is name@appCode
<p>Example:</p> <pre>{   "header": {     "userCode": "USER0001202004151958010871292",     "appCode": "app0001202004161020152918451",     "tId": "",     "mac": "MEQCICQaYMzs+edIQkft5hoaSO5dWqcrY7Q75FYwyJo/B4rAiAQ10aEpdNATsZYHVcJJ4TxVCgY8XdQBBlyTAOqUmSjkw==",     "body": {       "name": "user01",       "secret": "123456",       "csrPem": "-----BEGIN CERTIFICATE REQUEST-----\nMIHoMIGQAgEAMC4xLDAqBgNVBAMMI3VzZXIwMUBhcHAwMDAxMjAyMDA0MTYxMDIw\nMTUyOTE4NDUxMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEnguk1xunmuU1bnKB\nnam8QmeK6Geg/O6kL2D2ig85UMQTpG/sb9iYkdz8iC9SRnF9TvLiHuvJX2FGAOAQ\nnK1Vz8aAAMaOGCCqGSM49BAMCA0cAMEQCIE19Iin91KlfEvfFIbxfF14enFHhtvOU\nn5rK86huFiMMQAIbYXO4fJBq6eLGjaavR71O9fOvVZ5W7X+GQjIlQDuDgPQ==\n-----END CERTIFICATE REQUEST-----\n"} }</pre>					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: successful -1: failed
2	Response Message	msg	String	Y	
Body					
1	Certificate content	cert	String	Y	
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac": "MEUCIQCE0gg5VHWsZluNKA V2+xOJANGnCkw6f9J4+mFT1TWz/gIgf93jqzTzk0DU2lfMKnExcwVbgeI WMLvLmwKplCXNBA=",   "body": {     "cert": "-----BEGIN CERTIFICATE-----\nMIICvTCCAmSgAwIBAgIUcqn2HmCYmq/V2yKbnxuvC49KU00wCgYIKoZIzj0EAwIw\nTjELMAkGA1UEBhMCQ04xEDAOBgNVBAgTB0JlaWppbmxcDDAKBgNVBAAoTA0JTTjEP\nnMA0GA1UECxMGY2xpZW50MQ4wDAYDVQQDEwVic25jYTAqFw0yMDA0MjEwNTAzMDBA\nnGA8yMTAwMDMyMTEwMDQwMFowbDE8MA0GA1UECxMGY2xpZW50MA8GA1UECxMIb3Jn\nnYm5vZGUwDgYDVQQLEwdic25iYXNlMAoGA1UECxMDY29tMSwwKgYDVQQDDCNlc2Vy\nnMDFAYXBwMDAwMTIwMjAwNDE2MTAyMDE1MjIxODQ1MTBZMBMGByqGSM49AgEGCCqG\nnSM49AwEHA0IABJ4LpNcbp5rINW5ygWpVpEJniuhnoPzupC9g9ooPOVDEE6Rv7G/Ym\nnJHbs/IgvUkZxfU7y4h7ryV9hRgDgECtVc/Gjgf8wgfwwDgYDVR0PAQH/BAQDAgeA\nnMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFG28toKRbzJTFa6v/xIIYr6S9EvaMB8G\nnA1UdIwQYMBaAFACl4H+kIs8vn94ZYYpkrd+5ldMKMIGbBggqAwQFBgcIAQSBjnsi\nnYXR0cnMiOonsiaGYuQWZmaWxpYXRpb24iOiJvcmdibm9kZS5ic25iYXNlLmNvbSIs\nnImhmLkVucm9sbG1lbnRJRCI6InVzZXIwMUBhcHAwMDAxMjAyMDA0MTYxMDIwMTUy\nnOTE4NDUxliwiaGYuVHlwZSI6ImNsaWVudCIsInJvbGUuOiJjbGllbnQifX0wCgYI\nnKoZIzj0EAwIDRwAwRAIgLTlTps/DOHK8S3La7bnlChB+88b1Fko9bOAL36oAFPIC\nnIHQPCC30MoTHiId/X3fC5IxNukssmlMnEuDX73zRL55/\n-----"</pre>					

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

#### 5.4.3.7 Invoke chaincode in Public Key Upload Mode

For DApps in Public Key Upload mode, the participant needs to assemble the transaction message locally, and invoke this interface to initiate the transaction from the off-BSN system to the DApp's chaincode.

1. Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/node/trans>
2. Call Method: POST
3. Signature algorithm: required and refer to Section 5.4.3.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Transaction data	transData	String	Y	the transaction data should be encoded with base64

Example:

```
{
  "header": {
    "userCode": "USER0001202004151958010871292",
    "appCode": "app0001202004161020152918451",
    "tId": "",
    "mac": "MEUCIQCv8EZ2OqbSbI9xGGKX06Mquh+g+NhhbUoAJBbnemXdagIgNMF7W7ecu5uej9BpVx04qwJuVijbgcp3VYIcjDK0Z38=",
    "body": {
      "transData": "Cq0KCrsJCpcBCAMaCwi9gPr0BRD0o+Z2IhxhcHAwMDAxMjA yMDA0MTYxMDIwMTUyOTE4NDUxKkBJm2M2NTlZOTU4YzYzM4MTEwOTJiOGQ zNThkZDI2MTdmMWIxNGNiNjYxZGU2YjAyMmMxYTgyMjI0WU4YThjNDhkOi YSJBliY2NfYXBwMDAwMTIwMjAwNDE2MTAyMDE1MjkxODQ1MV8wMBKcC AqBCAoLT3JnYk5vZGVNU1AS8QctLS0tLUJFR0lOIENFUlRJRklDQVRFLS0tLS0K TUIJQ3ZUQ0NBbVNnQXdJQkFnSVVWanBGZTJFaERFaHJlOHBBVTTh4bk3dXhPb U13Q2dZSUtvWkl6ajBFQXdJdWpUakVMTUFR0ExVUVCaE1DUTA0eEVEQU9CZ 05WQkFnVEIwSmxhV3BwYm1jeEREQUtCZ05WQkFvVEEwSIRUakVQck1BMEdB MVVFQ3hNR1kyeHBaVzUwTVE0d0RBWURWUWFERXdWwWMyNWpZVEFnRnc weU1EQTBNGt3TkrNek1EQmEKR0E4eU1UQXdNRE15TVRFeE1EUXdNRm93Y kRFOE1BMEdBMBVVFQ3hNR1kyeHBaVzUwTUE4R0ExVUVDDeE1JYjNkbgpZbTV2 WkdVd0RnWURWUWFMRXdkWMyNWIZWE5sTUFvR0ExVUVDDeE1EWTI5dE1T d3dLZl1EIVFRFRERDTjBaWE4wCk1ESkFZWZlJ3TURBd01USXNakF3TkrFMk1U QXINREUxTWpreE9EUTFNVEJaTUJNR0J5cUdTTTQ5QWdFR0NDcUcKU000OUF 3RUhBMEIBQk5YZmFMVW1wMXIJSFVMMXVKEdwMDFQNHE5Zk81V2xFMF ZtallYQmVMejBhYlhqSU96NwpYb29KcGRUS1ZkUUJaZzYrZkVPWmhudm1vbUR XWjRpdTRhYWpnZjh3Z2Z3d0RnWURWUjBQOVFILOjBUURBZ2VBCk1Bd0dB M VVkrXQdFQ9i3UUNNQUF3SFFZRFZSME9CQlIFRkZZRDg5emtkVlIRbzZpUEh3d2R JejNaQ1lSck1COEcKQTFVZEI3UVINQmFBRkFjSTRIK2tJczh2bjk0WlZcGtyZCs1b GRNS01JR2JCZ2dxQXdRRkInY0lBUVNCam5zaQpZWFIwY25NaU9uc2lhR111UVda
```

```
bWFXeHBZWfJwYjI0aU9pSnZjbWRpYm05a1pTNWljMjVpWVhObExtTnZiU0lzCkl
taG1Ma1Z1Y205c2JHMWxiblJKUkNjNkluUmxjM1F3TWtCaGNIQXdnREF4TWpBe
U1EQTBNVFI4TURJd01UVXkKT1RFNE5EVXhJaXdpYUdZdVZlYHdaU0k2SW1Oc
2FXVnVkQ0lzSW5KdmJHVWlPaUpqYkdsbGJuUWlmWDB3Q2dZSQpLb1pJemowR
UF3SURSd0F3UkFJZ1ZZNi9jZ1NDTmpeNkxwTXVaZEQzVWYvWko5c3FSUVVT
R3hSQU9SeGZONThDCklFN0JHTDljOHRCcHJiVmpYTldtQmpObWhqeUE3N0I3S
W8rbUg1ZXp4R1B1Ci0tLS0tRU5EIENFUlRJRklDQVRFLS0tLS0KEhiQKmgB1Ibwb
gLAyoHXUNnjZSGOqBDheQMSbQprCmkIARlkEiJjY19hcHAwMDAxMjAyMDA0
MTYxMDIwMTUyOTE4NDUxXzAwGj8KA3NldAo4eyJiYXNlS2V5IjoidGVzdDIw
MjAwNDA0IiwiaWYmFzZVZhbHVlIjoidGhpcyBpcyBzdHJpbmcgIn0SRjBEAiB+mOUK
Y7fRjcZ1/qc96YP9GGod3UK56jJaWaE4o3J90QIgeirrjyzL6zQLN89tv3jDpI7vxKChk
GM9u8IEFiFEGYo=""}}
```

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	If code=0, can be null
Body					
1	block information	blockInfo	blockInfo	N	If code is not 0, then leave blank
2	chaincode response result	ccRes	ccRes	N	If code is not 0, then leave blank
blockInfo					
1	Transaction Id	txId	String	Y	
2	Block HASH	blockHash	String	N	On synchronous mode, returns Block HASH
3	status value	status	Int	Y	refer to detailed transaction status description in 5.4.3.17
ccRes					
1	chaincode response status	ccCode	Int	Y	200: successful 500: failed
2	chaincode response result	ccData	Str	N	actual chaincode response result
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac":   "MEQCICXNk400+Gkqqe2XgoaxdOoIvDQe4RfLtwXkxjC7ce8TAiBLVu6PjOqWueV B3t4h7REpNdcVf6L0qVzfdA1yovuc7g==",   "body": {     "blockInfo": {</pre>					

```

    "txId":
    "c3c6523958c3811192b8d358dd2617f1b14cb661de6b022c1a822269e8a8c48d",
    "blockHash": "",
    "status": 0
  },
  "ccRes": {
    "ccCode": 200,
    "ccData": "SUCCESS"
  }
}

```

#### 5.4.3.8 Get transaction information API

The off-BSN system can use this interface to get the transaction information based on transaction ID.

1. Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/node/getTransaction>
2. Call Method: POST
3. Signature algorithm: required and refer to Section 5.4.3.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	transactionId	txId	String	Y	
Example:					
<pre> {"header":{"userCode":"USER0001202004151958010871292","appCode":"app0001202004161020152918451","tId":""},"mac":"MEUCIQDIbcNI+C1iBbXWGW3qjhf80IRgCgvJuyxx0WXU2vn2TAIgZgA020L2aXBtrdLsYEkYPyiOJ9+AFrXOEwfuzy8B4bE=","body":{"txId":"c3c6523958c3811192b8d358dd2617f1b14cb661de6b022c1a822269e8a8c48d"}} </pre>					

5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response	msg	String	N	if code=0 then can

	message				be null
Body					
1	Block Hash	blockHash	String	Y	
2	Block Number	blockNumber	Long	Y	
3	Transaction status	status	Int	Y	refer to detailed transaction status description in 5.4.3.18
4	on-chain user name	createName	String	Y	
5	Timestamp Second	timeSpanSec	Int64	Y	“second” in the timestamp
6	Timestamp Nanosecond	timeSpanNsec	Int64	Y	“nanosecond” in the timestamp
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac":   "MEUCIQDUFw5pa4QJcEiQjYeLTl2L94HbsZbz7DArF+djgzWoTQIgU8u+dG6CcHw   BZjuf9PvhYdEFAa/ujwo8UAPbAmKxRq0=",   "body": {     "blockHash":     "ab9366cf63881228863c884527fceedabc9ad2e375aa0bcbf71f17f75c7d3ff5",     "blockNumber": 7,     "status": 0,     "createName": "test02@app0001202004161020152918451",     "timeSpanSec": 1587445821,     "timeSpanNsec": 249139700   } }</pre>					

### 5.4.3.9 Get transaction data API

This interface can be used by off-BSN systems to obtain transaction information based on the transaction ID and then returns the string of the transaction information by base64 encryption.

1. Interface address:

<https://PCNGatewayAddress/api/fabric/v1/node/getTransdata>

2. Call method: POST

3. Signature algorithm: required and refer to Section 5.4.3.1

4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					

1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Transaction Id	txId	String	Y	
Example:					
<pre>{   "header": {     "userCode": "USER0001202004151958010871292",     "appCode": "app0001202004161020152918451",     "txId": "",     "mac": "MEUCIQDIbcNl+C1iBbXWGW3qjhf80IRgCgvJuyxx0WXU2vn2TAIgZgA020L2aXBtrdLsYEkYPyiOJ9+AFrXOEwfuzy8B4bE=",     "body": {       "txId": "c3c6523958c3811192b8d358dd2617f1b14cb661de6b022c1a822269e8a8c48d"     }   } }</pre>					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	header	header	Map	Y	
2	body	body	Map	Y	
3	signature value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	Transaction Id	txId	String	Y	
2	Transaction data	transData	String	Y	String generated by base64 calculation
Example					
<pre>{   "header": {     "code": 0,     "msg": "success"   },   "mac": "MEUCIQDI63PUa4WjE01S4cdYy5spMRSYPLFzEvYGKHzsTSFxtAlgND/A/Cky9XDpHLNKQzOvgyf1nb6edVy3JQisBn7OuIM=",   "body": {     "txId": "b1b2ef26cff816dce49a40be3527092a2b0d43d244d57611bb2b95a05c063feb",     "transData": "CtYgCosgCrIKCpgBCAMa....."   } }</pre>					

### 5.4.3.10 Get block information API

After the data is uploaded to the chain, the off-BSN system can use this interface on the PCN

gateway to get the block information of the current transaction (body.blockInfo), the status (body.blockInfo.status), and transaction ID (body.blockInfo.txId). If the status value is 0, it signifies that the transaction has been successful and a block has been created. The block information can be queried according to the transaction ID.

1. Interface address:

<https://PCNGatewayAddress/api/fabric/v1/node/getBlockInfo>

2. Call Method: POST

3. Signature algorithm: required and refer to Section 5.4.3.1

4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Block number	blockNumber	Int64	N	Can't be null at the same time
2	Block HASH	blockHash	String	N	Can't be null at the same time
3	Transaction Id	txId	String	N	Can't be null at the same time
Example:					
<pre>{   "header": {     "userCode": "USER0001202004151958010871292",     "appCode": "app0001202004161020152918451",     "txId": ""   },   "mac": "MEUCIQCrGthrAvNaUsWEdnDxZkNXF4nCpXOXIFQdp1YYhGvugIgKvYql9Ex6RCcOhqt6coufNPH/QhtKYNeThWJ2rEL+4g=",   "body": {     "blockNumber": 6,     "blockHash": "",     "txId": ""   } }</pre>					

5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	header	header	Map	Y	
2	body	body	Map	Y	
3	signature value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	Block Hash	blockHash	String	Y	
2	Block Number	blockNumber	Long	Y	
3	Previous Block Hash	preBlockHash	String	Y	

4	Block Size	blockSize	Long	Y	byte
5	The number of transactions on current block	blockTxCount	Int	Y	
6	Transaction detail	transactions	[]Transaction Data	Y	Transaction Detail
TransactionData					
1	Transaction Id	txId	String		
2	Transaction Status	status	Int		refer to detailed transaction status description in 5.4.3.18
3	Transaction Provider	createName	String		
4	Transaction timestamp second	timeSpanSec	Int64		
5	Transaction timestamp nonasecond	timeSpanNsec	Int64		
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac":   "MEUCIQC8nfFnHw4sEYJmaSTT1xepxMGgomxyJtt0ysyGgPB0AwIgfuuiegdGEbBi/2wmF   Cco39wmik3isLWtvnN9ZmJDTdk=",   "body": {     "blockHash": "fc83c306677925efee540b4d7b7ca73e06f144cae34c706f1101d6b395ada2da",     "blockNumber": 6,     "preBlockHash":     "93c86551d812229274e144093cd4bd17dacb35bc6a01779930e11f43f886bf34",     "blockSize": 7020,     "blockTxCount": 1,     "transactions": [       {         "txId": "a8639f3a796267e048d475b00fe7646a4524f1c20d71880e19708821177b7bdb",         "status": 0,         "createName": "test02@app0001202004161020152918451",         "timeSpanSec": 1587271285,         "timeSpanNsec": 26436800       }     ]   } }</pre>					

#### 5.4.3.11 Get block data API

After the data is added to the chain, the off-BSN system will get the block information of the current transaction by calling this interface on the public city node gateway.

1. Interface address:



<https://PCNGatewayAddress/api/fabric/v1/node/getBlockData>

2. Call method: POST
3. Signature algorithm: required and refer to Section 5.4.3.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Block number	blockNumber	Int64	N	Can't be null at the same time
2	Block HASH	blockHash	String	N	Can't be null at the same time
3	Transaction Id	txId	String	N	Can't be null at the same time
Example:					
<pre>{   "header": {     "userCode": "USER0001202004151958010871292",     "appCode": "app0001202004161020152918451",     "txId": ""   },   "mac": "MEUCIQCrGthrAvNaUsWEdnDxZkNXF4nCpXOXIFQdp1YYhGvugIgKvYql9Ex6RCcOht6coufNPH/QhtKYNeThWJ2rEL+4g=",   "body": {     "blockNumber": 6,     "blockHash": "",     "txId": ""   } }</pre>					

#### Response parameters

No.	Field name	Field	Type	Required	Remarks
1	header	header	Map	Y	
2	body	body	Map	Y	
3	signature value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	Block Hash	blockHash	String	Y	
2	Block Number	blockNumber	Long	Y	
3	Previous Block Hash	preBlockHash	String	Y	
4	Block Data	blockData	String	Y	String generated by base64 calculation
Example					
<pre>{   "header": {     "code": 0, </pre>					

```

    "msg": "success"
  },
  "mac":
  "MEQCICAgU3G6o1Ky6UeYgqEgCee27TS2F8ScH+jaSj6w20OCAiB+/6z1a2jG5m4vvjz1ft
  2LQdIsaG2BAXqcwxmSFyEIzg==",
  "body": {
    "blockHash":
    "b8366a63ed32fddec720872d206802e670222f29d9a8a32983d26b59dbfd6971",
    "blockNumber": 3,
    "preBlockHash":
    "6dcc69799682e2fc7ffa950c56031b807c54b7a098b4fd69db9cf8c97518bcea",
    "blockData": "CkYIAxIgbcxpeZa....."
  }
}

```

#### 5.4.3.12 Get the latest ledger information API

Use this interface to get the latest ledger information, including block hash, previous block hash, and the current block height, etc.

1. Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/node/getLedgerInfo>
2. Call method: POST
3. Signature algorithm: required and refer to Section 5.4.3.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Example:					
<pre> {"header":{"userCode":"USER0001202004151958010871292","appCode":"app00012020 04161020152918451","tId":""},"mac":"MEQCID7Z3J2PiRDOx7JasRamBZRTAHXj1X AG1K/DUkzJEwuiAiBIY5p3H2kArE7OuYLogEqMHl15Xgj5Voi5zVPGhyU/+w==","b ody":{}} </pre>					

5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null

Body					
1	Block Hash	blockHash	String	Y	
2	Block Height	height	Long	Y	
3	Previous Block Hash	preBlockHash	String	Y	
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac":     "MEUCIQC4PhYTBNyt1rSeBeZTdOly42CxILVgK1b/RlieA33G1gIgeodoEa5Ou0X4uW     c/VGp0n6NKByhXIBbo22FME4xQ8aw=",   "body": {     "blockHash":       "ab9366cf63881228863c884527fceedfab9ad2e375aa0bcbf71f17f75c7d3ff5",     "height": 8,     "preBlockHash":       "fc83c306677925efee540b4d7b7ca73e06f144cae34c706f1101d6b395ada2da"   } }</pre>					

#### 5.4.3.13 Chaincode event registration API

Chaincode event in a DApp can trigger the off-BSN system to process further transactions. This interface is used to register the chaincode event to be monitored.

- Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/chainCode/event/register>
- Call method: POST
- Signature algorithm: required and refer to Section 5.4.3.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	ChainCode	chainCode	String	Y	
2	Chaincode event key	eventKey	String	Y	
3	Chaincode event notification URL	notifyUrl	String	Y	URL to receive the monitored chaincode event
4	Attached additional parameters	attachArgs	String	N	
Example:					

```
{
  "header": {
    "appCode": "CL20191107112252",
    "userCode": "lessing",
    "body": {
      "attachArgs": {
        "name": "TOM&age=20",
        "chainCode": "cc_bsn_test_00",
        "eventKey": "test01",
        "notifyUrl": "http://192.168.6.128:8080/api/event/notifyUrl",
        "mac": "MEUCIQCjzPr4KZVild2Vm5YgcunOXTh9mQK2QfWcRnYCK+jOzglgDW6oHca7/249M43p2ElwiMNBuejdWAnyW5OwiMqiWCQ="
      }
    }
  }
}
```

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: successful -1: failed
2	Response Message	msg	String	Y	
Body					
1	Event ID	eventId	String	Y	
Example					
<pre>{   "header": {     "code": 0,     "msg": "Event Registration Successful"   },   "body": {     "eventId": "bd3391deedbe44a7ad5b7f80ce59abfa",     "mac": "MEQCIEpLpj2R9mRL100vcMXs0X5rWfSjB/U7kMg+76GjEPNJAiBIUo/Eyj49uXTPrzRW0m4rJ0NQIkZnDMPbyalxojXwrA=="   } }</pre>					

### 5.4.3.14 Block event registration API

Block event in a DApp can trigger the off-BSN system to process further transactions. This interface is used to register the block event to be monitored.

#### 1. Interface address:

<https://PCNGatewayAddress/api/fabric/v1/chainCode/event/blockRegister>

#### 2. Call method: POST

#### 3. Signature algorithm: required and refer to Section 5.4.3.1

#### 4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Chaincode event notification URL	notifyUrl	String	Y	URL to receive the monitored block event
2	Attached additional parameters	attachArgs	String	N	
Example:					
<pre>{   "header": {     "userCode": "USER0001202007101641243516163",     "appCode": "app00"   } }</pre>					

```
01202101191411238426266","tId":"","mac":"MEUCIQClsjKy/ee1qaYrItzCO1b
Mfjs0g0kPu8+YOCjbk3rPRAIgSfeyYvfeoh8QciZPG4fZQepaiyh7PmmWjYzFSq
ylT/c=","body":{"chainCode":"","eventKey":"","notifyUrl":"http://192.168.6.78:5
8011/v1/fabric/test","attachArgs":"a=1"}}
```

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: successful -1: failed
2	Response message	msg	String	Y	
Body					
1	Event ID	eventId	String	Y	
Example					
<pre>{   "header": {     "code": 0,     "msg": "success"   },   "mac":   "MEUCIQC6PKsSqfkQGLrqi2vMpZzBP5beLhyP+fXVr8S5aqhaagIgaEtAnsuiub   ibYoYZzQ/8aGYErzm5rtU8Oj952OuHgCo=",   "body": {     "eventId": "002f0e1f0b0f4331ab541461547a38d6"   } }</pre>					

### 5.4.3.15 Chaincode and block event query API

Use this API to query the list of monitored chaincode and block events that have been registered.

- Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/chainCode/event/query>
- Call method: POST
- Signature algorithm: required and refer to Section 5.4.3.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Example:					
{"header":{"appCode":"CL20191107112252","userCode":"lessing"},"body":{"mac":"MEQCIAnJxvuKVe0u/bG0VYCjM3g3ctxTYIWkejYp462okNlcAiBcOTGvAkF7xErL2w1PiwgfFjlu3Sszgyfzym/pEwRGxA=="}}					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	[]body	Y	Event List
3	Signature Value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: Query successful -1: Query failed
2	Response Message	msg	String	Y	
body					
1	Event ID	eventId	String	Y	
2	Chaincode Event key	eventKey	String	N	Null if it's a block event
3	Chaincode Event Notification URL	notifyUrl	String	Y	
4	Attached additional parameters	attachArgs	String	N	
5	Creation Time	createTime	String	Y	
6	PCN ID	orgCode	String	Y	
7	user unique ID	userCode	String	Y	
8	DApp unique code	appCode	String	Y	
9	Chaincode ID	chainCode	String	N	Null if it's a block event
10	Event type	eventType	String	N	Returns "block" if it's a block event; Null if it's chaincode event
Example					
{ "header": { "code": 0, "msg": "Query Event Successful" }, "body": [ { "eventKey": "test001", "notifyUrl": "http://192.168.6.128:8080/api/event/notifyUrl", "attachArgs": "a=123\u0026b=456", "eventId": "945ee631d26140118963ad3104c81713", "createTime": "2019-11-18 14:22:59", "orgCode": "ORG1571365934172", "userCode": "lessing", "appCode": "CL20191107112252", "chainCode": "cc_bsn_test_00" }, { "eventKey": "test002", "notifyUrl": "http://192.168.6.128:8080/api/event/notifyUrl", "attachArgs": "hahahhahahahahah", "eventId": "346617a493d84c6d8512b8dddad87811", "createTime": "2019-11-18" }					

```
14:29:28", "orgCode": "ORG1571365934172", "userCode": "lessing",
"appCode": "CL20191107112252", "chainCode": "cc_bsn_test_00" },
{ "eventKey": "test01", "notifyUrl":
"http://192.168.6.128:8080/api/event/notifyUrl", "attachArgs":
"name=Zhangsan\u0026age=20", "eventId": "bd3391deedbe44a7ad5b7f80ce59abfa",
"createTime": "2019-11-19 10:52:15", "orgCode": "ORG1571365934172",
"userCode": "lessing", "appCode": "CL20191107112252", "chainCode":
"cc_bsn_test_00" } ], "mac":
"MEQCIEYXFma8dfBrjy/s9H5JAoFIrjROJBiw+7/daELUbF5eAiA7a6HvqqbOpv6vIkun
HGxCB1o5DoeuJFD0FM6kLoU34Q=="}
```

#### 5.4.3.16 Remove chaincode and block event API

This interface is used to remove a chaincode event's registration from the event list.

- Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/chainCode/event/remove>
- Call method: POST
- Signature algorithm: required and refer to Section 5.4.3.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Event ID	eventId	String	Y	
Example:					
{ "header": { "appCode": "CL20191107112252", "userCode": "lessing" }, "body": { "eventId": "bd3391deedbe44a7ad5b7f80ce59abfa" }, "mac": "MEQCIE3/CLG5LxZZN7En7LZvzthajw xHzpvDduXSsw4Tb1JFAiAXGJ4WVtyCKbtCasQGofCkge8NOgZDNPgJIdTCtCi2SQ=" }					

- Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: remove successful -1: remove failed
2	Response Message	msg	String	Y	
Example					
{ "header": { "code": 0, "msg": "Remove Event Successful" }, "body": null, "mac": "MEUCIQCaTFLiY7pPjkwcmSsLXOth7k9bQj9Sblq+1nMVjkFAAIgUsizFO+f1+dxU3/hPxjf/+na4qG6aQFftJIWgtMhlVI=" }					

### 5.4.3.17 Chaincode and block event notification API

This interface is implemented on the off-BSN system side. When the PCN gateway receives the notification of a triggered event, it uses this interface to notify the off-BSN system about the execution result.

After receiving the notification successfully, the off-BSN system returns a string containing “success”, otherwise, the gateway will send the notification again at 3, 12, 27, and 48 seconds respectively, for a total of five times.

1. Call method: POST
2. Signature algorithm: required and refer to Section 5.4.3.1
3. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
body					
1	Chaincode ID	chainCode	String	N	Null when the block event notification
2	PCN ID	orgCode	String	Y	
3	Registered Event key	eventKey	String	N	
4	Registered Event ID	eventId	String	Y	
5	Registered Event parameters	attachArgs	String	N	Additional parameters entered during registration
6	Monitored event key	eventName	String	N	The event name in the chaincode, null when the block event notification
7	Current Chaincode transaction Id	txId	String	N	Null when the block event notification
8	Monitored event value	payload	String	N	
9	Current Block Height	blockNumber	Long	Y	
10	Response random string	nonceStr	String	Y	Off-BSN system uses this value to judge if the notification is already received. This string remains the same at the repeated notifications.



11	Previous hash	previousHash	String	N	Null when chaincode event notification
Example:					
Chaincode event notification					
<pre>{   "header": {     "userCode": "lessing",     "appCode": "CL20191107112252",     "body": {       "chainCode": "cc_bsn_test_00",       "orgCode": "ORG1571365934172",       "eventKey": "test:\\S{32}",       "eventId": "2964a0f60b3e460f834618b3664af2da",       "attachArgs": "abc=123211",       "eventName": "test:12345678123456781234567812345678",       "txId": "32fc105681820fa556b8a460efc1e43a47daa864b959ea1753abb4640f2dce49",       "payload": "",       "blockNumber": 74,       "nonceStr": "522c8061b5e84837bad72ca08c6a353f",       "mac": "MEQCIDU4tROyJLtvD1b8TTbWWAICPuUbmdPAEUxwRRgVn7kIAiA58je5u/7xDuRPegeUWL3nB9mouUGQ6dGKJMmD7Jm08g=="     }   } }</pre>					
Block event notification					
<pre>{   "header": {     "userCode": "USER0001202007101641243516163",     "appCode": "app0001202101191411238426266",     "body": {       "orgCode": "ORG2020041114171692360",       "eventId": "8746bb9a1e854c9f8b3710f5a63f7c59",       "attachArgs": "a=1",       "previousHash": "022281f6089e3684501251775166b6b0afd18a176ec98a835cb5d09aff0d4950",       "blockNumber": 12,       "nonceStr": "79a7baa26c854caeb2e2e7abc0b7f07e",       "mac": "MEUCIQDiZrWf8fKG/3fuaVrsfTN3BKmLx+qnnEuuSaHfvIBbMQIgS+1qHKXeVR24WXwOGu3Nze/tLLziQ0LkXaueYu0ctM="     }   } }</pre>					

4. The payload parameter in the message is not passed in Fabric 1.4.3, please refer to <https://github.com/hyperledger/fabric/blob/v1.4.3/core/peer/deliverevents.go#L251>

For the specific code, if you need to use the payload parameter, you can get it as follows:

- 1) store the content to be passed through the event to the chain in the form of “key-value”.
- 2) pass the event name and the key splice in 1 as eventName, e.g.: eventkey\_key.
- 3) register the event with a regular registration, e.g. : eventkey\_[\s\S]\*.
- 4) after receiving the event, parse the key according to the event name and call a query to get the value.
- 5) if the amount of payload data to be passed is not large, it can be directly spliced to eventName, which has no length limitation.

#### 5.4.3.18 Transaction status description

Under both Key Trust Mode and Public Key Upload Mode, the description of the returned transaction status when the off-BSN system invokes the DApp chaincodes via PCN gateway APIs are shown as follows:

No.	Status Code	Remarks
	0	Successful
	-1	Block creation time out
	1	Submitted data empty
	2	Unusual response
	3	Error in the submitted information
	4	Error in the creator's signature
	5	Invalid “endorser” transaction

	6	Invalid transaction settings
	7	Unsupported transaction response
	8	Error in the transaction ID
	9	Duplicate transaction ID
	10	Failed endorsement
	13	Unknown transaction type
	14	Cannot locate target chaincode
	17	Expired chaincode
	18	Conflict in chaincode version
	254	Invalid transaction
	255	Invalid transaction for other reasons

#### 5.4.4 PCN Gateway FISCO API

A PCN gateway is deployed on each public city node (PCN) to receive off-BSN system requests signed and verified by DApp access keys, then used to route the requests to the corresponding FISCO BCOS-based DApp smart contracts. Invoking the PCN gateway is realized by sending HTTP requests to each PCN gateway service. The gateway is responsible for verifying user and application identities, and then uses these identities and smart contract functions to process smart contract parameters then sends the smart contract transaction results back to the off-BSN systems.

##### 5.4.4.1 DApp Access Signature Algorithm

Whenever an off-BSN system sends requests to the PCN gateway, the HTTP request message should be signed with the DApp participant's DApp access private key. When the PCN gateway receives the message with the digital signature, it will verify the authentication and message integrity with the corresponding hosted or uploaded DApp access public key. The gateway will only process the request message further after the verification is passed.

##### 1. Assemble signature string

Convert the request parameters into a joined string according to the order of the parameter table, of which, the call parameter prioritises joining UserCode and AppCode of the Header and the response parameter prioritises joining code and msg. Then join the parameters in the Body according to the order of the parameter tables in the definition of APIs.

##### 2. Different type conversion formats

Type	Rule	Example	Result
String	No conversion	abc	abc
Int/int64/long	Decimal conversion	-12	-12
Float	Decimal conversion; see notes for values after decimal point	1.23	1.23
Bool	Convert to "true" or "false"	true	true
Array	Join according to parameter sequence and type	{"abc","xyz"}	abcxyz
Map[key]value	Join key and value according to parameter sequence	{"a":1,"b":2}	a1b2
Object	Convert the attributes in the object one by one according to the document in the above-described format	{"name":"abc","secret":"123456"}	abc123456

### 3. Signature rules

#### 1. FISCO BCOS framework DApp using ECDSA (secp256k1) secret key algorithm

- Getting the Hash value: The converted string to be signed is required to be computed with SHA256 algorithm with UTF-8 encoding.
- Sign the Hash value: The hash value and private key should be encrypted with ECDSA (secp256k1) algorithm. In the processing of some programming languages (C#, Java), if signed with SHA256WithECDSA, which includes hash value computation, therefore, the first step is not necessary.
- Encoding the signature result to Base64.

#### 2. FISCO BCOS framework DApp using SM secret key algorithm

- Getting the Hash value: The converted string to be signed is required to be computed with SM3 algorithm with UTF-8 encoding.
- Sign the Hash value: The hash value and private key should be encrypted with SM2 algorithm.
- Encoding the signature result to Base64.

### 4. Example

Parameters:

```
{"header":{"userCode":"user01","appCode":"app01"},"mac":"","body":{"userId":"abc","list":["abc","xyz"]}}
```

Result: user01app01abcabcxyz

#### 5.4.4.2 Key and Certificate Modes

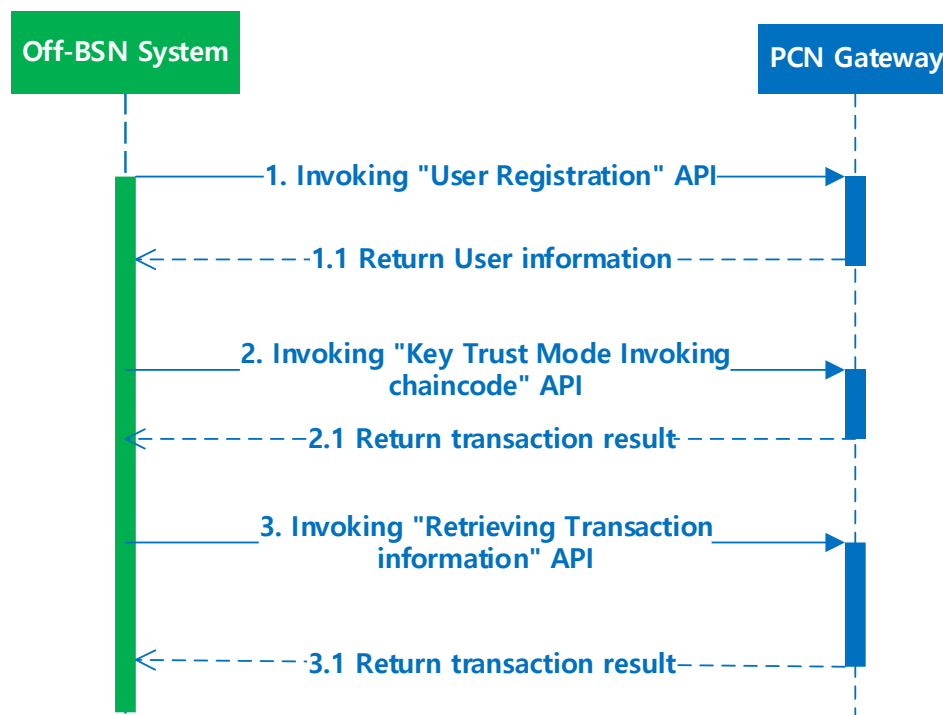
##### 1. Key Trust Mode

As described in chapter 5, DApp participants require two sets of key pairs to access the DApp: DApp access key pair and user transaction key pair. Under the key trust mode, the pairs are generated and hosted by BSN. The participants only need to download the private key (DApp access key) from the BSN portal.

**DApp Access Key Pair:** After the participant has successfully joined the DApp, BSN will generate one key pair (private and public keys) that corresponds to the DApp's framework algorithms under the Key Trust Mode. The participant can download the private key from "My Certificates" section of the BSN global portal and use it to sign the request message sent to the PCN gateway. The gateway will use the hosted public key from the generated key pair to validate the signature.

**User Transaction Key Pair:** This is the identity of a participant to invoke the chaincodes. Under the Key Trust Mode, after successfully joining the DApp, a participant's user transaction key pair will be created automatically by BSN by default. The participant's off-BSN system can use the participant's UserCode to invoke the certificate generated by the key pair. If the participant's off-BSN system has multiple sub-users, the off-BSN system can invoke the gateway's "User Registration API" to register the sub-users and generate a separated user transaction key pair for each sub-user. The sub-users can use their own UserCode to connect to the DApp to execute smart contract transactions.

Transaction process:

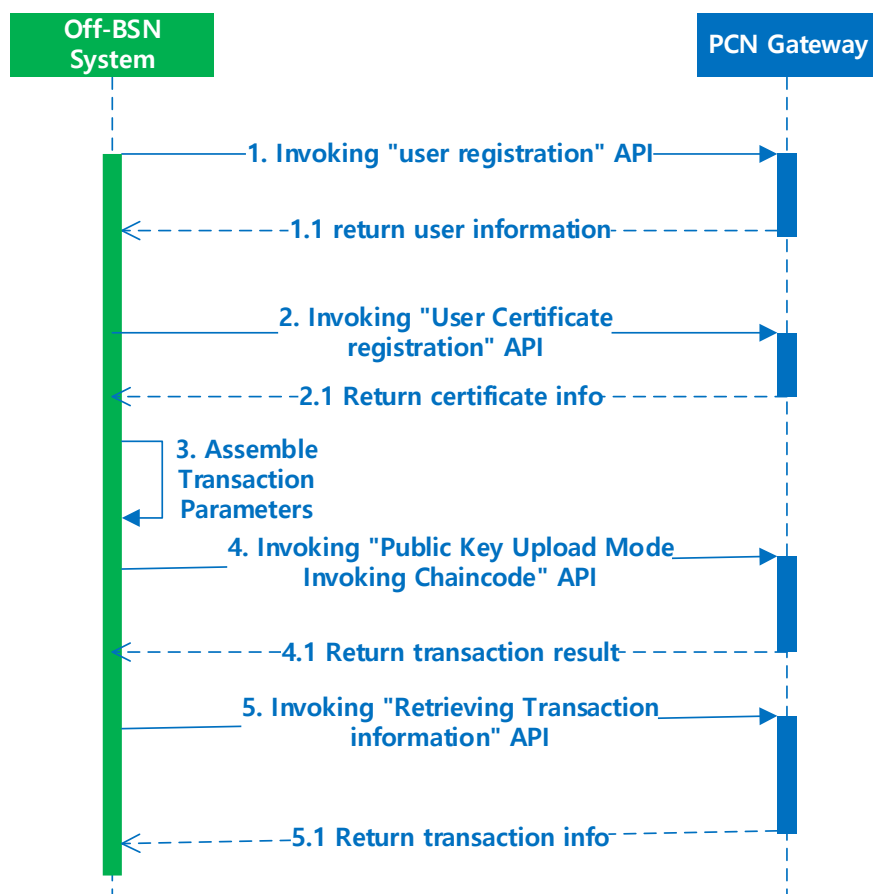


## 2. Public Key Upload Mode

As described in chapter 5, DApp participants require two sets of key pairs to fully access the DApp: DApp access key pair and user transaction key pair. With public-key upload mode, the key pairs are generated and stored locally by the participants. The participants only need to upload the public keys to BSN via the BSN portal or gateway APIs.

- **DApp Access Key Pair**: The DApp participant must generate the DApp access key pair locally according to the DApp framework algorithm after successfully joining the DApp. The participant stores the private key locally and uploads the public key to BSN via the BSN global portal. The participant's off-BSN system uses the private key to sign the transaction messages when invoking the PCN gateway. The PCN gateway will use the public key uploaded by the participant to verify the signature and validate the legality of the transaction.
- **User Transaction Key Pair**: This is the identity of a participant to invoke the chaincodes. Under the Key Trust Mode, the participant must generate the user transaction key pair locally and use the public key to generate the "public key registration application", then from the participant's off-BSN system to submit the registration application to BSN by invoking the "Public Key Upload Mode user certification registration" API on the PCN gateway to receive the public key certificate. If the off-BSN system has sub-users, it should first invoke the "User Registration" API to register the sub-users before sending their public key registration applications.

Transaction process:



#### 5.4.4.3 Get DApp information API

Invoke this interface to get basic DApp information; this interface can be used by transactions in Public Key Upload Mode.

- Interface address:  
<https://PCNgatewayAddress/api/app/getAppInfo>
- Call Method: POST
- Signature Algorithm: Not Required
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Yes	
2	Body	body	Map	No	
3	Signature value	mac	String	Yes	
Header					
1	User unique ID	userCode	String	Yes	
2	DApp unique ID	appCode	String	Yes	
Body					
Example:					
<pre>{   "header": {     "userCode": "USER0001202004151958010871292",     "appCode": "app0001202004161020152918451",     "tId": ""   },   "mac": "",   "body": {} }</pre>					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature value	mac	String	Y	
Header					
1	Response ID	code	int	Y	0: successful -1: failed
2	Response Message	msg	String	Y	
Body					
1	DApp name	appName	String	Y	
2	DApp type	appType	String	Y	
3	DApp encryption key type	caType	Int	Y	1: Key Trust Mode 2: Public Key Upload Mode
4	DApp algorithm Type	algorithmType	Int	Y	1: SM2 2: ECDSA(secp256r1)
5	City MSPID	mSPID	String	Y	
6	DApp chain name	channelId	String	Y	Fabric corresponding channelId, fisco corresponding groupId
Example:					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac": "MEUCIQDE9zv0E/w4V/ILG6wUCFP08a7NDCAtX/loZOcCyY4gIQIgUTYWsfTA1KE88gE6452jKnnVBrhznGVOV2HPMCbNh8A=",   "body": {     "appName": "sdktest",     "appType": "fabric",     "caType": 2,     "algorithmType": 2,     "mSPID": "OrgbNodeMSP",     "channelId": "app0001202004161020152918451"   } }</pre>					

## 5.4.4.4 User Registration API

After a participant has successfully joined in a FISCO BCOS (FISCO) DApp, his/her off-BSN system can invoke this interface to generate the user account and user address to execute smart contract transactions.

## 1. Interface address:

<https://PCNGatewayAddress/api/fiscobcos/v1/user/register>

## 2. Call Method: POST

## 3. Signature algorithm: required and refer to Section 5.4.4.1

## 4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
body					
1	user name	userId	String	Y	Registered user name
Example:					
<pre>{   "header": {"appCode": "CL1881038873220190902114314", "userCode": "newuser"},   "body": {     "userId": "abc"   },   "mac": "MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SfII9DIRvxcI3YVAiBt4XeNs+   EUjhBNSr3IjLRPZucsuGHxfjt9RiaNIQS8cA==" }</pre>					
signature value:					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
body					
1	User information	data	[]string	N	If code is not 0, then leave blank
data					
1	User ID	userId	String	Y	
2	User Address	userAddress	String	Y	
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac": "MEQCIEI5VKMyJUXls2Hf8TL0PXjZLT4/L2wyXoddgTnZdqRsAiBxEBMeCOZ8M97   OCRUAMZNMCL974vhzjOS/tk8/wbgbsA==",   "body": {     "userId": "100003",     "userAddress": "0x14647a48303b5e1c77934583883ebc327ba3b297"   } }</pre>					

```
} 
```

#### 5.4.4.5 Invoke Smart Contract API in Key Trust Mode

For the FISCO DApps in Key Trust mode, when the off-BSN system invokes the smart contract via PCN gateway, it is required to include the parameters in the request. The gateway will return the response message from the chaincode.

- Interface address:

<https://PCNGatewayAddress/api/fiscobcos/v1/node/reqChainCode>

Note: After a participant has successfully joined in a FISCO DApp service, the participant can view and download the DApp's configuration parameters which are used for off-BSN systems to connect to this DApp's smart contracts, including the PCN gateway address and Dapp access keys, as shown below:

- Call Method: POST
- Signature algorithm: required and refer to Section 5.4.4.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	User ID	userId	String	Y	Registered user ID via 7.3.1 API
2	Smart Contract Name	contractName	String	Y	
3	Function Name	funcName	String	Y	
4	Function Parameters	funcParam	string	N	convert array type to json string format



Example:
{       "header": {         "appCode": "cl0006202003181926573677572",         "userCode": "USER0006202003181951281835816"       },       "body": {         "contractName": "HelloWorld",         "userId": "100003",         "funcName": "set",         "funcParam": ["abc"]       },       "mac": "MEUCIQDTFe2Gerdf7YJrG1a1Yt99M0ZQ3T1lGpsXdNmFV7WuTgIgSkZ19abUhAJbMrJMBod8N7f26xhpQRuR4vNAfY7EEbs="     }

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	Invoke Type	constant	Bool	N	
2	Query information	queryInfo	String	N	If Constant is true, this field has value.
3	Transaction hash	txId	string	N	If Constant is false, this field has value and is valid.
4	Block HASH	blockHash	String	N	If Constant is false, this field has value and is valid.
5	Block Number	blockNumber	Int	N	If Constant is false, this field has value and is valid.
6	Gas Used	gasUsed	Int	N	If Constant is false, this field has value and is valid.
7	Transaction Status	status	String	N	If Constant is false, this field has value and is valid. 0x0 means transaction successful, status value refer to transaction receipt status in 7.3.9
8	From account	from	String	N	If Constant is false, this field has value and is valid.
9	To account	to	String	N	If Constant is false, this field has value and is valid.
10	Input	input	String	N	If Constant is false, this field has value and is valid.
11	Ouput	output	String	N	If Constant is false, this field has value and is valid.

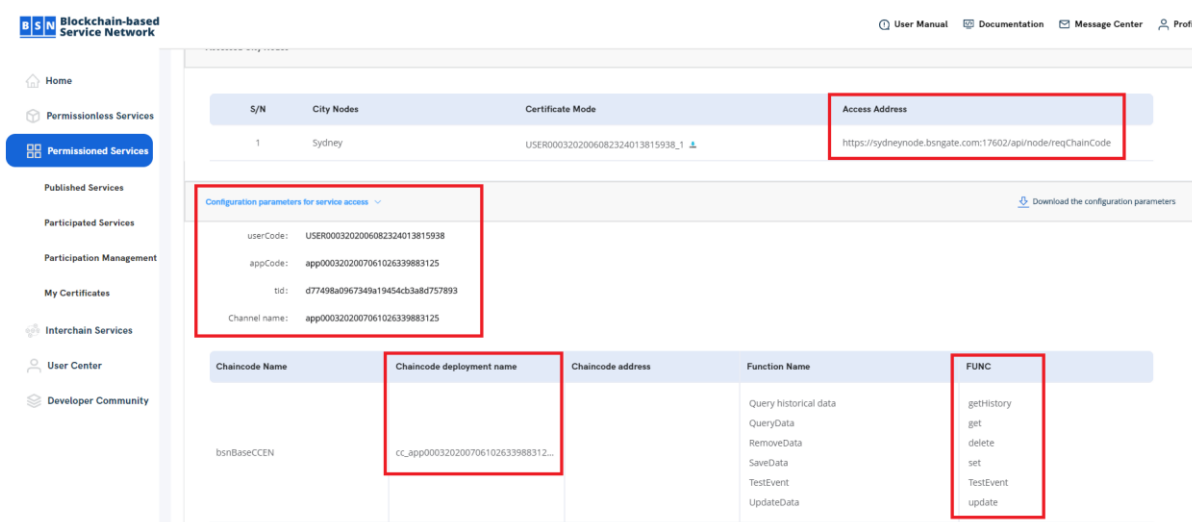
Example

#### 5.4.4.6 Invoke Smart Contract API Public Key Upload Mode

When the off-BSN system invokes the node gateway, it should follow the API descriptions to add the corresponding parameters. After invoking the node gateway, the node gateway returns the execution result of the smart contract. In the transaction of Public Key Upload mode, the private key of the transaction on the chain is generated and saved by the user. Then the client performs the assembly and signature of the data locally. The signed data is uploaded to the node gateway, which forwards the data to the corresponding blockchain node to initiate the transaction request. Data assembly in this pattern requires information such as the contract ABI, which is compiled when developing the contract, and the contract address, which is available on the application details page. In the SDK of the gateway, the assembly method of the data on the link has been implemented, which can be directly called.

1. Interface address:

<https://PCNGatewayAddress/api/fiscobcos/v1/node/trans>



Note: After a participant has successfully joined in a FISCO DApp service, the participant can view and download the DApp's configuration parameters which are used for off-BSN systems to connect to this DApp's smart contracts, including the PCN gateway address and Dapp access keys, as shown below:

2. Call Method: POST
3. Signature algorithm: required and refer to Section 5.4.4.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Smart Contract	contractName	String	Y	

[illegible]

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	Invoke Type	constant	Bool	N	
2	Query information	queryInfo	String	N	If Constant is true, this field has value.
3	Transaction hash	txId	string	N	If Constant is false, this field has value and is valid.
4	Block HASH	blockHash	String	N	If Constant is false, this field has value and is valid.
5	Block Number	blockNumber	Int	N	If Constant is false, this field has value and is valid.
6	Gas Used	gasUsed	Int	N	If Constant is false, this field has value and is valid.
7	Transaction Status	status	String	N	If Constant is false, this field has value

					and is valid. 0x0 means transaction successful, status value refers to transaction receipt status in 7.3.9
8	From account	from	String	N	If Constant is false, this field has value and is valid.
9	To account	to	String	N	If Constant is false, this field has value and is valid.
10	Input	input	String	N	If Constant is false, this field has value and is valid.
11	Output	output	String	N	If Constant is false, this field has value and is valid.
Example					

#### 5.4.4.7 Get Transaction Receipt API

After the smart contract executes one transaction, this interface can be used to get the transaction receipt information according to the transaction hash value.

- Interface address:  
<https://PCNGatewayAddress/api/fiscobcos/v1/node/getTxReceiptByTxHash>
- Call Method: POST
- Signature algorithm: required and refer to Section 5.4.4.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
body					
2	Transaction Hash	txHash	string	Y	
Example:					
{"header":{"appCode":"cl0006202003181926573677572","userCode":"USER0006202003181951281835816"},"body":{"txHash":"0x755f3e7833778f674e1b025f513f05722ba7248be43a3c9168b880847814021a"},"mac":"MEYCIQCe6sl9zqspys1bS6Ka9Q8O+pE7TEDWdsWj4UBSg6FM7AlhAJrud/EoxnURQcDc47iwTdh7OdxJEJPE+raK9UaHjNaJ"}					
signature value:					

- Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	

2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	Transaction Receipt Info	txId	string	N	If code is not 0, then leave blank
	Block HASH	blockHash			
	Block Number	blockNumber			
	Gas Used	gasUsed			
	From account	from			
	To account	to			
	Smart Contract Address	contractAddress			
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction successful"   },   "mac":   "MEUCIQCUIhmvH9a4HN/YITf4OWgTuHmhz6qMEO89I4effHdcIwIgStdeb/dVplhn3/FoCjeSc   VRyiEUhpkbze9bVmlgaXqs=",   "body": {     "blockHash":     "0x199eca276b60473dd65f8b36641684456694b419d89ef41b4953a9cdac848305",     "gasUsed": 2154887,     "blockNumber": 1,     "txId": "0x8ee0c68e222742b5b70878265d3fdbd3a8e0d549da42a298a4ae872ca4fbfd89",     "contractAddress": "0x20453db36c492fa49da9fab1b80db7fa5f46b01e",     "from": "0x08ac3132a6c7e6ca5a7fbaf0521bb8b6f370ed35",     "to": "0x00"   } }</pre>					

#### 5.4.4.8 Get Transaction information API

After the smart contract executes one transaction, this interface can be used to get the transaction detailed information according to the transaction hash value.

1. Interface address:

<https://PCNGatewayAddress/api/fiscobcos/v1/node/getTxinfoByTxHash>

2. Call Method: POST
3. Signature algorithm: required and refer to Section 5.4.4.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Transaction HASH	txHash	string	Y	
Example:					
{       "header": {"appCode": "cl0006202003181926573677572", "userCode": "USER0006202003181951281835816"},       "body": {"txHash": "0x755f3e7833778f674e1b025f513f05722ba7248be43a3c9168b880847814021a"},       "mac": "MEUCIQDDQudQBvHk15tlpeTDGkQA+LPRMTA2k9u7hCZAYVobvQIqNseUfaVw8d/LxooPPWyQSo2O4EUt6wmEISgtnTcUO7k="     }					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
	Transaction HASH	txId	String		
	Block HASH	blockHash	String		
	Block Number	blockNumber	Int		
	Gas Used	gasUsed	Int		
	From account	from	String		
	To account	to	String		
		value	Int		
		input	String		
Example					
{       "header": {         "code": 0,         "msg": "Transaction Successful"       },       "mac": "MEQCIBMqntmqQqZXkBBrLhmXEcuOqTG4YWvlfGJmebzEDbzcAiAKKHut9MBShqpSAEo8ts2MEQCIBMqntmqQqZXkBBrLhmXEcuOqTG4YWvlfGJmebzEDbzcAiAKKHut9MBShqpSAEo8ts2+OBIRmEEbedjihix5FZZvrw==",       "body": {         "blockHash": "0x199eca276b60473dd65f8b36641684456694b419d89ef41b4953a9cdac848305",					

```

"input":
"0x60806040523480156200001157600080fd5b506110016000806101000a81548173fffffffffffffffff
fffffffffffffffff021916908373fffffffffffffffffffffffffffffffff16021790555060008090549061
01000a900473fffffffffffffffffffffffffffffffff1673fffffffffffffffffffffffffffffffff1663c92a780
16040805190810160405280600681526020017f745f6261736500000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
1600087803b1580156200011c57600080fd5b505af115801562000131573d6000803e3d6000fd5b50
5050506040513d601f19601f8201168201806040525062000157919081019062000174565b506200
02f4565b60006200016c8251620002a3565b905092915050565b6000602082840312156200018757
600080fd5b600062000197848285016200015e565b91505092915050565b6000620001ad82620002
98565b808452620001c3816020860160208601620002ad565b620001ce81620002e3565b60208501
0191505092915050565b6000601382527f626173655f6b65792c626173655f76616c756500000000
00000000000000000000006020830152604082019050919050565b6000600782527f626173655f6964000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
0006060820190508181036000830152620002668184620001a0565b9050818103",
  "gasUsed": 100000000,
  "blockNumber": 1,
  "txId": "0x8ee0c68e222742b5b70878265d3fdbd3a8e0d549da42a298a4ae872ca4fbfd89",
  "from": "0x08ac3132a6c7e6ca5a7fbaf0521bb8b6f370ed35",
  "to": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "value": 0
}
}

```

#### 5.4.4.9 Get Block Information API

Corresponding block information can be queried according to block number or the block hash. The block number and block hash cannot simultaneously be blank. When neither is blank, the block number will be invoked in priority.

- Interface address:  
<https://PCNGatewayAddress/api/fiscobcos/v1/node/getBlockInfo>
- Call Method: POST
- Signature algorithm: required and refer to Section 5.4.4.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Block Height	blockNumber	string	N	When null, blockHash cannot be null
2	Block Hash	blockHash	String	N	When null, blockNumber

					cannot be null
Example:					
<pre>{   "header": {"appCode": "CL1881038873220190902114314", "userCode": "newuser"},   "body": {     {       "blockNumber": 22,       "blockHash": "0xf27ff42d4be65329a1e7b11365e190086d92f9836168d0379e92642786db7ade"     },     "mac": "MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SfII9DlRvxcI3YVAiBt4XeNs+E UjhBNSr3IjLRPZucsuGHxfjt9RiaNIQS8cA=="   } }</pre>					
signature value:					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
	Block HASH	blockHash	String	Y	
	Block Number	blockNumber	Int	Y	
	Parent Block HASH	parentBlockHash	String	Y	
	Block Size	blockSize	Int	Y	
	Block Time	blockTime	Int	Y	Timestamp in millisecond format
		author	String	Y	
	Transaction Information	transactions	[]Transaction Data	Y	
TransactionData					
	Transaction Id	txId	String	Y	
	Block HASH	blockHash	String	Y	
	Block Number	blockNumber	Int	Y	
	Gas Used	gasUsed	Int	Y	
		from	String	Y	
		to	String	Y	
		value	Int	Y	
		input	String	Y	
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction successful"   },   "mac":</pre>					





## 4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
body					
Example:					
{ "header": { "appCode": "cl0006202003181926573677572", "userCode": "USER0006202003181951281835816" }, "body": { }, "mac": "MEQCIHb2o7hb0apDukOQBxkZftETsizDBaftnHxO9A9ux5EtAiABuiFrVYPWT5FiU+Wd9HpXF/AJh0Yh2SXtL6h98m4eZw==" }					
signature value:					

## 5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response message	msg	String	N	if code=0 then can be null
Body					
1	Block Height	data	string	N	If code not 0, then leave blank
Example					
{ "header": { "code": 0, "msg": "Transaction Successful" }, "mac": "MEQCICtCOdv4ZL72M3WoA9nAei2P0/PpKjlgI0Y5qeuzg61uAiA9D3TcB/+b2RMuNwVq+X0vgiglHfM5NBhoTJPR0gCPMA==", "body": { "data": "4" } }					

## 5.4.4.11 Get Total Number of DApp Transactions API

This interface is used to get the total number of transactions in a DApp.

## 1. Interface address:

<https://PCNGatewayAddress/api/fiscobcos/v1/node/getTxCount>

2. Call Method: POST
3. Signature algorithm: required and refer to Section 5.4.4.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
body					
Example:					
{"header":{"appCode":"cl0006202003181926573677572","userCode":"USER0006202003181951281835816"},"body":{"mac":"MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SflI9DIRvxcI3YVAiBt4XeNs+EUjhBNSr3IjLRPZucsuGHxfjt9RiaNIQS8cA=="}}					
signature value:					

5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response Message	msg	String	N	if code=0 then can be null
Body					
1	Transaction Information	data	string	N	If code not 0, then leave blank
Example					
{ "header": { "code": 0, "msg": "Transaction Successful" }, "mac": "MEQCIGgXINn3B9d/hC/ow0IJvi5eKDj59QbZRFdrCqcUeNCgAiApI4jkwhTY33qevlRwsJ3veDBKXokvIiSe3ck7SKlXmg==", "body": { "data": {"txSum":5,"blockNumber":5,"txSumRaw":"0x5","blockNumberRaw":"0x5"} } }					

#### 5.4.4.12 Get Total Number of Block Transactions API

This interface is used to get the total number of transactions inside a block according the block number in a FISCO DApp. The block number cannot be null.

- Interface address:  
<https://PCNGatewayAddress/api/fiscobcos/v1/node/getTxCountByBlockNumber>
- Call Method: POST
- Signature algorithm: required and refer to Section 5.4.4.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
body					
2	Block number	blockNumber	string	Y	
Example:					
<pre>{   "header": {"appCode": "CL1881038873220190902114314", "userCode": "newuser"},   "body": {     "grouId": 1,     "blockNumber": 22,   },   "mac": "MEQCIBRhaM2szckWl9N9qcqnaYXOXGQw7SfII9DlRvxcI3YVAiBt4XeNs+EU jhBNSr3IjLRPZucsGHxfjt9RiaNIQS8cA==" }</pre>					

- Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: authentication successful -1: authentication failed
2	Response message	msg	String	N	if code=0 then can be null
Body					
1	Block total count of transactions info	data	string	N	If code not 0, then leave blank
data					
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction Successful"   },   "mac": "MEUCIQCMFbVhfH9X8pJ1mNI3YpzKIBcXCpfm2AniF/42ak9EwIgTWDEF+xW5l39ZDUnDSSSc8Zv8JlglEf9izp16eW/Rn4=", }</pre>					

```
"body": {
  "data": "1"
}
```

#### 5.4.4.13 Smart Contract Event Registration API

Smart contract event in a DApp can trigger the off-BSN system to process further transactions. This interface is used to register the smart contract event to be monitored.

1. Interface address:  
<https://PCNGatewayAddress/api/fiscobcos/v1/event/register>
2. Call method: POST
3. Signature algorithm: required and refer to Section 5.4.4.1
4. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
Header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Event Type	eventType	String	Y	1.Block generation event 2.Contract event
2	Contract address	contractAddress	String	N	EventType is 1 then can be null; EventType is 2 then EventType and contract Name cannot be null at the same time
3	Contract name	contractName	String	N	EventType is 1 then can be null; EventType is 2 then EventType and contractName cannot be null at the same time
4	Notification URL	notifyUrl	String	Y	
5	Attached parameters	attachArgs	String	N	
Example:					
<pre>{   "header": {     "userCode": "USER0001202006042321579692440",     "appCode": "app0001202006042323057101002",     "tId": ""   },   "mac": "MEUCIQCMP1ToZS5e8S94kYZ/8y5XfeyjRyUrPFpeIQMES3SGpQIgO8b6O8Kk/qpNTolvbNTwyAYNaw6HBI9OkAH8Rp23j8s=",   "body": {     "eventType": 1,     "contractAddress": "0x866aefc204b8f8fdc3e45b908fd43d76667d7f76",     "contractName": "BsnBaseContract k1",     "notifyUrl": "http://127.0.0.1:18080",     "attachArgs": "abc=123"   } }</pre>					

5. Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: successful -1: failed
2	Response Message	msg	String	Y	
Body					
1	Event ID	eventId	String	Y	Null when the code is not 0
Example					
<pre>{   "header": {     "code": 0,     "msg": "Transaction successful"   },   "mac":   "MEUCIQDYSTwYhh6EDHT5Z7ukcqXW9LMjZW6WPnr8Xt14RuH2A1gIwa5K7NK4/TThzs8   z6VfKpNNJU+dzAXeypFmfjkr88=",   "body": {     "eventId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"   } }</pre>					

#### 5.4.4.14 Smart Contract Event Query API

Use this API to query the list of monitored smart contract events that have been registered.

- Interface address:  
[https://PCNGatewayAddress/api/fiscobocs/v1 /event/query](https://PCNGatewayAddress/api/fiscobocs/v1/event/query)
- Call method: POST
- Signature algorithm: required and refer to Section 5.4.4.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Example:					
<pre>{"header": {"userCode": "USER0001202006042321579692440", "appCode": "app000120200604232 3057101002", "tId": ""}, "mac": "MEUCIQC2NTuUlsxQSWPpZwwhJK9zXEMaeYZC04Ar0P5Twy p5AQIgFvZrskasuLiYfOGxd1F9TCetWHIfENg8BCiYfNS1xGk=" }</pre>					

- Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
<b>Header</b>					
1	Response ID	code	int	Y	0: Query successful -1: Query failed
2	Response Message	msg	String	Y	
<b>body</b>					
1	Block generation event	blockEvent	[]blockEvent	Y	Null when the code is not 0
2	Contract event	contractEvent	[]contractEvent	Y	
<b>blockEvent</b>					
1	block generation event	eventId	string	Y	Null when the code is not 0
2	App code	appcode	String	Y	
3	User code	userCode	String	Y	
4	Notification URL	notifyUrl	String	Y	
5	Attachment parameters	attachArgs	String	N	
6	Create time	createTime	String	Y	UTCtime
<b>contractEvent</b>					
1	block generation event	eventId	string	Y	
2	App code	appcode	String	Y	
3	User code	userCode	String	Y	
4	Notification URL	notifyUrl	String	Y	
5	Attachment parameters	attachArgs	String	N	
6	Create time	createTime	String	Y	UTCtime
7	Contract address	contractAddress	String	Y	
<b>Example</b>					
<pre>{   "header": {     "code": 0,     "msg": "Transaction successful"   },   "mac":   "MEUCIQCQ/RjmlVklKZw6jcLKBPh1BwK4EIQE001vUAKPVq1HTgIgXUQ7Bn+y8D8xQxYUwtZOoh/bpteAPCUtKXZeAiN7cMU=",   "body": {     "blockEvent": [       {         "eventId": "ba537419953e4e219ceb0fe26ad5e125",         "appCode": "app0001202006042323057101002",         "userCode": "USER0001202006042321579692440",         "notifyUrl": "http://127.0.0.1:18080",         "attachArgs": "abc=123",         "createTime": "0001-01-01 00:00:00.000 +0000 UTC"       }     ]   } }</pre>					

```

    }
  ],
  "contractEvent": [
    {
      "eventId": "ba537419953e4e219ceb0fe26ad5e126",
      "appCode": "app0001202006042323057101002",
      "userCode": "USER0001202006042321579692440",
      "notifyUrl": "http://127.0.0.1:18080",
      "attachArgs": "abc=123",
      "createTime": "0001-01-01 00:00:00.000 +0000 UTC",
      "contractAddress": "0x866aefc204b8f8dc3e45b908fd43d76667d7f76"
    }
  ]
}

```

#### 5.4.4.15 Remove Smart Contract Event API

This interface is used to remove a smart contract event's registration information from the event list.

- Interface address:  
<https://PCNGatewayAddress/api/fabric/v1/chainCode/event/remove>
- Call method: POST
- Signature algorithm: required and refer to Section 5.4.4.1
- Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
Body					
1	Event ID	eventId	String	Y	
Example:					
<pre>{   "header": {     "appCode": "CL20191107112252",     "userCode": "lessing"   },   "body": {     "eventId": "bd3391dedbe44a7ad5b7f80ce59abfa"   },   "mac": "MEQCIE3/CLG5LxZZN7En7LZvzthajwxHzpvDduXSsw4Tb1JFAiAXGJ4WVtyCKbtCasQGofCkge8NOgZDNPgJIdTCtCi2SQ==" }</pre>					

- Response parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	Y	
3	Signature Value	mac	String	Y	
header					
1	Response ID	code	int	Y	0: remove successful



					-1: remove failed
2	Response Message	msg	String	Y	
Example					
{ "header": { "code": 0, "msg": "Remove Event Successful" }, "body": null, "mac": "MEUCIQCaTFLiY7pPjkwcmSsLXOth7k9bQj9Sblq+1nMVjkFAAIgUsizFO+f1+dxU3/hPxjf/+na4qG6aQFftJIWgtMhlVI=" }					

#### 5.4.4.16 Smart Contract Event Notification API

This interface is implemented on the off-BSN system side. When the PCN gateway receives the notification of a triggered event, it uses this interface to notify the off-BSN system about the execution result.

After receiving the notification successfully, the off-BSN system returns a string containing “success”, otherwise, the gateway will send the notification again at 3, 12, 27, and 48 seconds respectively, for a total of five times.

1. Call method: POST
2. Signature algorithm: required and refer to Section 5.4.4.1
3. Call parameters

No.	Field name	Field	Type	Required	Remarks
1	Header	header	Map	Y	
2	Body	body	Map	N	
3	Signature Value	mac	String	Y	
header					
1	user unique ID	userCode	String	Y	
2	DApp unique ID	appCode	String	Y	
body					
1	Registered Event ID	eventId	String	Y	
2	PCN ID	orgCode	String	Y	
3	Registered Event parameters	attachArgs	String	N	Additional parameters entered during registration
4	Response random string	nonceStr	String	Y	Off-BSN system uses this value to judge if the notification is already received. This string remains the same at the repeated notifications.
5	Event type	eventType	String	Y	
6	Event data	eventData	String	Y	
Example:					
{ "header": { "userCode": "USER0001202006042321579692440", "appCode": "app0001202006042323057101002" }, "body": { "eventId": "5b5b865f8dc94ae59d215cf26aa81d69", "orgCode": "ORG2020041114171692360", "appCode": "app0001202006042323057101002", "attachArgs": "abc=123", "nonc					

```
eStr":"52f080f27ff045eb87e21812d12cee40","eventType":1,"eventData":{"appId":"app000120
2006042323057101002\\","blockNumber\\":17,"eventType\\":1,"groupId\\":135}},"mac":"MEUCI
QD3Sp6xuI4DHy/GOB9z3nH6kQisEzfXvZ/Hn/mfZXIAOgIgYsISRfBKSJGt4FrmxETflfR4A8Ve
nCZHvxthMFUWRkc="}
```

#### 5.4.4.17 Transaction Receipt Status

Under Key Trust Mode, the description of the returned transaction status when the off-BSN system invokes the FISCO DApp smart contracts via PCN gateway APIs are shown as follows:

status(Decimal/ Hexadecimal)	message	Explanation
0(0x0)	None	No Error
1(0x1)	Unknown	Unknown Error
2(0x2)	BadRLP	Invalid RLP Error
3(0x3)	InvalidFormat	Invalid Format Error
4(0x4)	OutOfGasIntrinsic	The length of smart contract exceeds gas limit/smart contract invoking parameters exceed gas limit
5(0x5)	InvalidSignature	Invalid Signature Error
6(0x6)	InvalidNonce	Invalid nonce Error
7(0x7)	NotEnoughCash	Not enough cash Error
8(0x8)	OutOfGasBase	Parameters too long (RC version)
9(0x9)	BlockGasLimitReached	Gas limit reached Error
10(0xa)	BadInstruction	Bad Instruction Error
11(0xb)	BadJumpDestination	Bad Jump Destination Error
12(0xc)	OutOfGas	Out of gas to execute the smart contract/the length of smart contract exceeds the limit.
13(0xd)	OutOfStack	Out of Stack Error
14(0xe)	StackUnderflow	Stack Under Flow Error
15(0xf)	NonceCheckFail	Nonce check failed Error
16(0x10)	BlockLimitCheckFail	Block limit check failed Error
17(0x11)	FilterCheckFail	Filter check failed Error
18(0x12)	NoDeployPermission	No Deployment Permission Error
19(0x13)	NoCallPermission	Invalid call Error
20(0x14)	NoTxPermission	Invalid transaction Error
21(0x15)	PrecompiledError	Precompiled Error
22(0x16)	RevertInstruction	Revert Instruction Error
23(0x17)	InvalidZeroSignatureFormat	Invalid Signature Format
24(0x18)	AddressAlreadyUsed	Address Already Used Error
25(0x19)	PermissionDenied	Permission Denied
26(0x1a)	CallAddressError	Call Address does not exist Error

## 5.5 Development SDK and Examples

### 5.5.1 BSN Gateway SDK Example

Normally, if an off-BSN system wants to communicate with a permissioned DApp service on BSN, it has to call the public city nodes (PCN) gateway APIs. We provide a BSN Gateway SDK (Software Development Kit) which can help developers quickly implement an off-BSN system to call the PCN Gateway. Inside the SDK, we provide PCN gateway API encapsulation which you can use to implement the transaction querying, transaction interface calling,

generate public key and private key locally, register user certificate, generate certificate signature, encrypt and decrypt data, etc.

Download links:

<https://github.com/BSNDA/PCNGateway-Go-SDK>

<https://github.com/BSNDA/PCNGateway-Java-SDK>

<https://github.com/BSNDA/PCNGateway-PY-SDK>

<https://github.com/BSNDA/PCNGateway-CSharp-SDK>

## 5.5.2 Sample Smart Contract Packages

For your reference, the following is the sample source code of our preset chaincode/smart contract packages, including Golang, and solidity language examples.

### ➤ Fabric example

Download link:

<https://github.com/BSNDA/FabricBaseChaincode>

### ➤ FISCO BCOS example

Download link:

<https://github.com/BSNDA/FISCOBaseContract>

We invite experienced developers who are interested in BSN to work together to optimize the SDK and sample packages. If you'd like to participate, please contact us on GitHub.

## 5.6 BSN Testnet Services

### 5.6.1 Overview

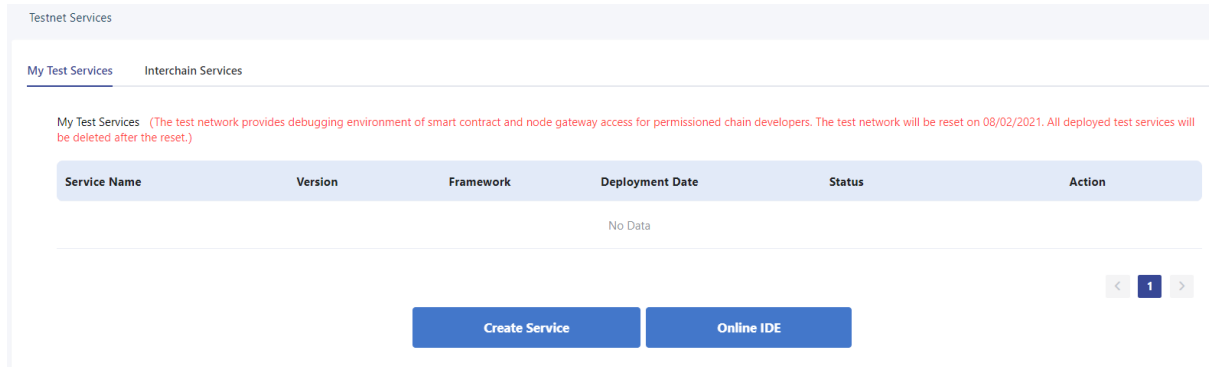
BSN Testnet is a free test environment for developers to test their permissioned DApp services. Developers can publish an unlimited number of permissioned DApp services on the testnet. Unlike the BSN production environment, it is not necessary to choose the public city nodes and configure the invocation authorities of smart contracts when publishing DApp services on the testnet. The Testnet supports Hyperledger Fabric and FISCO BCOS frameworks, and will continue to integrate all BSN-adapted permissioned frameworks. Like all testnets do, we will occasionally reset the Testnet and delete all smart contracts and ledger data. Therefore, please do not use the Testnet as a commercial or production environment. We welcome developers to try the service and provide us with feedback and suggestions as we continue to make improvements.

### 5.6.2 Permissioned DApp Service Publication

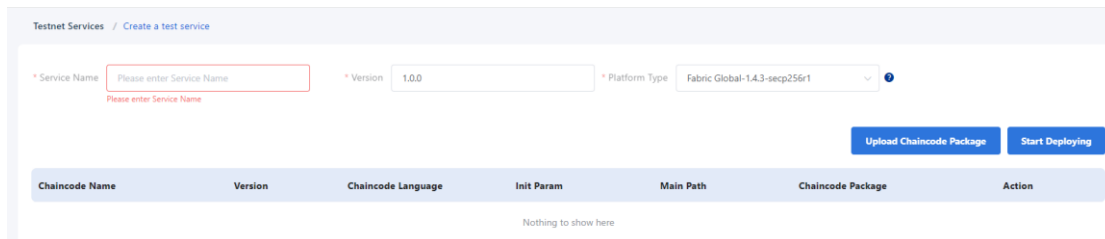
The steps to publish a permissioned DApp service for testing are as follows:

1. Create a new test service

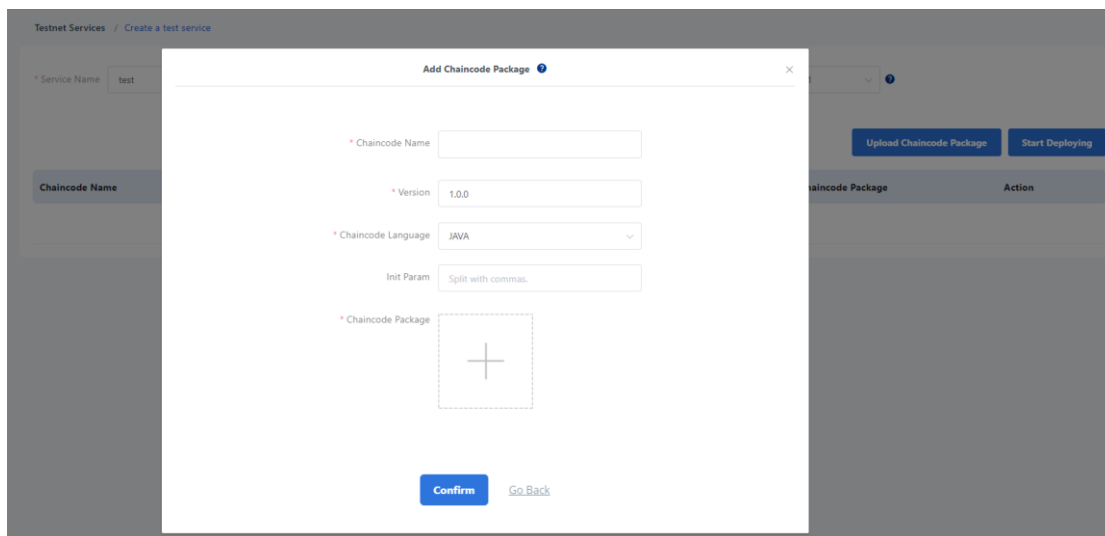
Go to the **Permissioned Services > Testnet Services** page to publish the service.



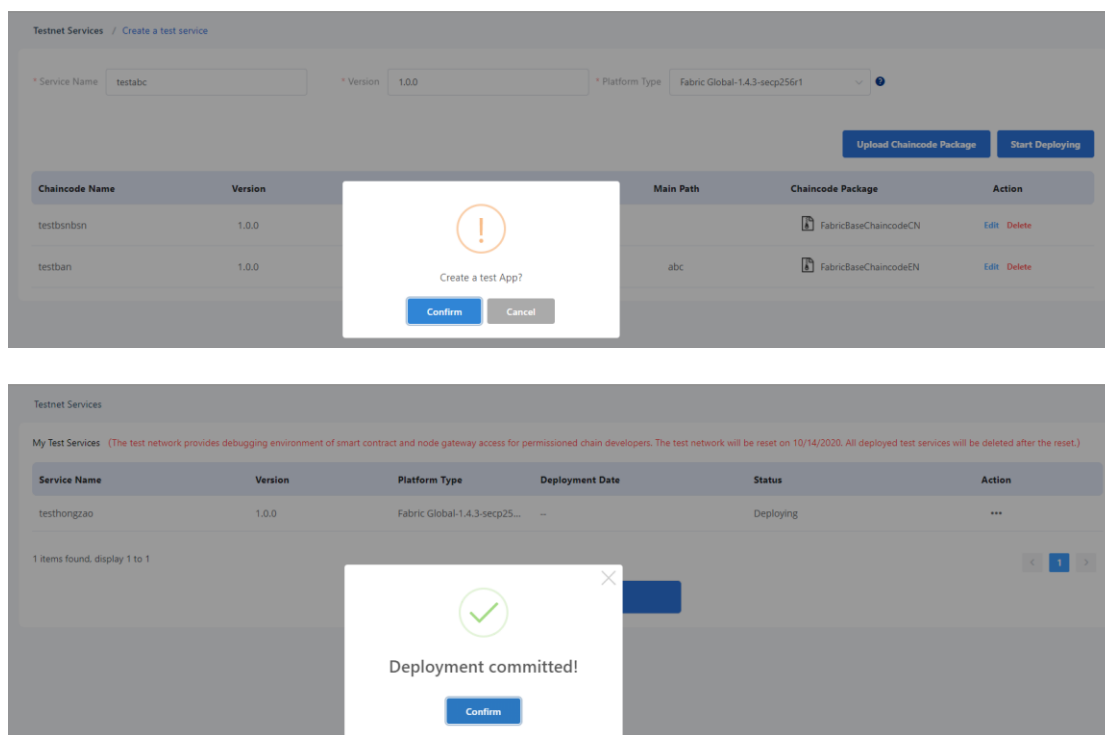
Click **Create a Test Service** and input the service name, version, and select a platform type.



Click **Upload Chaincode Package** to upload the chaincode or smart contract package. You can upload multiple chaincode/smart contract packages in a permitted DApp service. Input the information and click **Confirm** to upload the package.



2. Deploy the permitted DApp service:  
Click **Start Deploying** to deploy the service.



After successfully deploying the chaincode/smart contract, developers can call it from their off-BSN systems so that they can configure and debug the functions easily.

**Note:** To keep the resources stable, DevOps will periodically clean up the chaincode/smart contract packages and ledger data on the Testnet.

### 5.6.3 Interchain Services on BSN Testnet

A demo version of Interchain Communications Hub (ICH) is now live on the Testnet, integrating two interchain solutions based on the relay chain mechanism: (1) Poly Enterprise developed by Onchain Tech and (2) IRITA developed by Bianjie AI. We welcome developers to try out and provide feedback and suggestions, and we will continue to improve the functionality and expect to release a commercial version in 2021.

For detailed descriptions and examples of ICH services, please refer to chapter 8, "Interchain Services"

## 6 Dedicated Node Services

### 6.1 Overview

BSN dedicated node services apply BSN technologies including multi-layer framework adaptation, virtualized container, automated deployment and node gateway to provide users with "out-of-the-box" blockchain cloud services. Users can quickly create their own dedicated permissioned blockchain operating environment, configure node's CPU, memory, disk capacity and other parameters in the BSN portal; they can independently manage nodes, publish smart contracts, access node data and monitor blockchain operation status. The dedicated node does not restrict APIs of the framework, and all APIs can be called by developers after they access the dedicated node through the gateway.

Currently, dedicated node services allow users to build the permissioned chain services based on ConsenSys Quorum (an open source, free and enterprise-focused blockchain framework), Hyperledger Fabric, and Besu in the BSN public city node built on AWS cloud platform. The version of ConsenSys Quorum is v20.10.0, and its consensus mechanism supports Raft and IBFT mechanisms; the version of Hyperledger Fabric is v2.3.2, and its consensus mechanism is Raft; the version of Hyperledger Besu is v21.1.2, and its consensus mechanism is Clique and IBFT.

### 6.2 Project Management

#### 6.2.1 Create Projects

1. In the BSN menu, click the **Permissioned Service** dropdown, in the list, click **Dedicated Node Services** to open the page. The page lists the projects created by the user and shows the status information of each project.

Dedicated Node Services								
Project Name	Framework	Cloud Platform	Region	Payment Status	Payment Type	Deployment Time	Status	Action
test	ConsenSys Quorum-v2...	AWS	Hong Kong	Payment Successful	Annually	(UTC+8:00) 04/28/2021...	Running	<a href="#">Details</a> <a href="#">Unsubscribe</a> <a href="#">Edit Authorized Account</a>
dgds	ConsenSys Quorum-v2...	AWS	Hong Kong	Payment Successful	Monthly	(UTC+8:00) 04/28/2021...	Running	<a href="#">Details</a> <a href="#">Unsubscribe</a> <a href="#">Edit Authorized Account</a>
ABCCC	ConsenSys Quorum-v2...	AWS	Hong Kong	Unpaid	Annually	—	Not Deployed	<a href="#">Details</a> <a href="#">Pay</a>

3 items found, display 1 to 3

< 1 >

Create Project

2. Click **Create Project** button and jump to the information page. This page contains 4 sections: **Basic Information**, **Node Information**, **Gateway Information** and **Data Usage Information**.
  - 1) **Basic Information:** This section shows the basic information of the service.
    - When the framework is ConsenSys Quorum-v20.10.0, the following basic information will be displayed, including project name, framework, consensus mechanism (options including: Raft, IBFT), cloud platform and region.

Basic Information

\* Project Name

\* Framework

\* Consensus

\* Cloud Platform

Region

- When the framework is Hyperledger Fabric-v2.3.2, the following basic information will be displayed, including project name, framework, consensus mechanism (Raft), cloud platform, region, consortium name, and channel name.

Basic Information

\* Project Name

\* Framework

\* ConsensusType

\* consortium

\* channelId

\* Cloud Platform

Region

- When the framework is Hyperledger Besu-v21.1.2, the following basic information will be displayed, including project name, framework, consensus mechanism (Options including: Clique, IBFT), cloud platform and region.

Basic Information

\* Project Name

\* Framework

\* Consensus

\* Cloud Platform

Region

- 2) **Node Information:** The publisher can select the number of nodes and other resource information, including CPU, memory and data capacity. The price is automatically calculated based on the resources which publisher has selected.
- When the framework is ConsenSys Quorum-v20.10.0 or Hyperledger Besu-v21.1.2, the node information includes: Number of Nodes, Host Configuration, Data Capacity and Price.

## Node Information

\*Please select the number of nodes and resource information:

Number of Nodes	Host Configuration	Data Capacity	Price (USD/year)
1	2Core+4G	50G	1315.88

- When the framework is Hyperledger Fabric-v2.3.2, the node information includes: Number of Nodes, Number of Orderers, Host Configuration, Data Capacity and Price.

## Node Information

\*Please select the number of nodes and resource information:

Number of Nodes	Number of orderers	Host Configuration	Data Capacity	Price (USD/month)
3	3	4Core+8GB	50GB	747.42

- Gateway Information:** This section shows the information of the gateway node, and this node contains Nginx service and a blockchain browser. Publisher does not need to select resources.

## Gateway Information

Note: This node contains Nginx service and a blockchain browser.

Number of Nodes	Host Configuration	Data Capacity	Price (USD/year)
1	4Core+8G	50G	2717.85

- Data Usage Information:** This section shows the unit data price for inbound gateway traffic and outbound gateway traffic.

## Data Usage Information

Inbound Data Usage (USD/GB)	Outbound Data Usage (USD/GB)
0.00	0.01

[Next](#)
[Go Back](#)

- Click Next button to jump to **Charge Details** page. This page has 3 sections: **Resource Cost**, **Data Usage Information** and **Total Cost**.



Dedicated Node Services / Create Project

Resource Cost

Node Resource Cost Information

Number of Nodes	Host Configuration	Data Capacity	Price (USD/year)
1	2Core+4G	50G	1413.43

Gateway Cost Information:

Number of Nodes	Host Configuration	Data Capacity	Price (USD/year)
1	4Core+8G	50G	2717.85

Payment Amount ☐ \$378.71 (Pay by month) ☒ \$4131.28 (Pay by year) Discount of \$413.24

Data Usage Information

Inbound Data Usage (USD/GB)	Outbound Data Usage (USD/GB)
0.00	0.01

Total Cost

Total charges: \$4131.28

Note:

- \* This payment includes the total cost of \$4131.28 for node resource fee and gateway resource fee in the first year, click "OK" and the system will automatically deduct the fee from your account balance. Subsequent fees will be deducted automatically on a year basis.
- \* This payment does not include the node gateway data usage fee. The node gateway data usage charge is based on actual usage and will be deducted automatically on a weekly basis. Please ensure that there is sufficient balance in your account to avoid affecting the operation of the service.

Confirm Go Back

- Resource Cost:** Resource Cost section contains the cost of node resources and gateway resources. According to the resource cost information, the publisher can either pay by month or pay by year. A discount will be applied when paying annually.
    - When the framework is ConsenSys Quorum-v20.10.0 or Hyperledger Besu-v21.1.2, the node information section of the resource cost includes: number of nodes, host configuration, data capacity, and price.
    - When the framework is Hyperledger Fabric-v2.3.2, the node information section of the resource cost display includes: number of nodes, number of orderers, host configuration, data capacity, and price.
  - Data Usage Information:** This section shows the unit data price for inbound gateway data usage and outbound gateway data usage.
  - Total Cost:** The total charges that the publisher should pay for.
- After the publisher confirms the Charge details, click "**Confirm**" button to make payment. The payment will be deducted from the user's personal (or corporate) account. If the deduction fails, the bill will be kept for 72 hours before expiration. If you still want to open a dedicated node service, you can resubmit or recreate the project by editing the current project.

*Note: In terms of dedicated node services payment, developers can make payments for dedicated node services with the status of "not deployed" and pending payment, payment failed, and "running" but in arrears. The payment will be debited from the user's personal (or corporate) account. After the payment is successful, the developer should wait for the deployment of the dedicated node.*

## 6.2.2 Edit Projects

- Dedicated node services with the status of "not deployed" and billing invalid, pending payment, payment failed, and "deployment failed" and fully refunded can be edited. In the edit page, developer can edit the basic information and node information.

- Once edited the information, developer can jump to the Charge Details page to pay the bill. After the payment is successfully made, developer can then wait for the deployment of the dedicated node.

### 6.2.3 Delete Projects

Dedicated node services that are in the status of "not deployed" with expired billing and "deployment failed" with full refund can be deleted.

### 6.2.4 View Project Details

When the dedicated node has been deployed, the developer can view the detailed information of the project. Click **Details** button in **Action** column to jump to the project details page. There are 3 sections in this page: **Basic Information**, **Resource Information** and **Deployment Information**.

- When the framework is ConsenSys Quorum-v20.10.0, the page is shown as below:

Dedicated Node Services / Details

Basic Information

Project Name: Qmtest

Framework: ConsenSys Quorum-v20.10.0

Cloud Platform: aws

Payment Status: Paid

ConsensusType: raft

Region: Paris

Created Date: (UTC+8:00) 01/19/2022 17:44:14

Resource Information

Node Resource and Cost Information

Number of Nodes	Host Configuration	Data Capacity	Price (USD/month)	Price (USD/year)
1	2Core+4GB	50GB	109.65	1196.48

Gateway Cost Information:

Number of Nodes	Host Configuration	Data Capacity	Price (USD/month)	Price (USD/year)
1	2Core+4GB	50GB	109.65	1196.48

Data Usage Information

Inbound Data Usage (USD/GB)	Outbound Data Usage (USD/GB)
0	0.01

Deployment Information

Deployment node list

Authorized Username: c16PR7x3FuqLxPINRI

Authorized Password: \*\*\*\*\*

Type	Peer Name	Status	Deployment Time	Action
Peer Node	node1	Running	(UTC+8:00) 01/19/2022 17:49:44	<a href="#">Details</a>
Gateway Services	Browser	Running	(UTC+8:00) 01/19/2022 17:49:45	<a href="#">Details</a> <a href="#">Open URL</a>

Back

- Basic Information:** Project Name, Framework, Consensus, Cloud Platform, Region, Payment Status and Created Date.
- Resource Information:** Node resource and cost information, Gateway Cost Information and Data Usage Information.

- 3) **Deployment Information:** The developer can view node information and browser information. Clicking on the "**Details**" button corresponding to the peer node, developer can view the information of Access and Credentials, Transaction Manager cluster, and Default Wallet.

**Access and Credentials**

RPC Endpoint	<a href="https://bsnl7xt7eab.bsngate.com:19602/node1">https://bsnl7xt7eab.bsngate.com:19602/node1</a>	<a href="#">Copy</a>
Transaction Manager (TM) Endpoint	<a href="https://bsnl7xt7eab.bsngate.com:19602/tm1">https://bsnl7xt7eab.bsngate.com:19602/tm1</a>	<a href="#">Copy</a>

**Transaction Manager Cluster**

Public Key	*****	<a href="#">Copy</a>
Private Key	*****	<a href="#">Copy</a>

**Default Wallet**

Address	0x3ab477BFcf7c5861e9B805EC93542B6F4bf057D0	<a href="#">Copy</a>
Public Key	*****	<a href="#">Copy</a>
Private Key	*****	<a href="#">Copy</a>

- 4) By clicking on the "**Details**" button corresponding to gateway services, the developer can obtain the URL address of the blockchain browser.

**Access and Credentials**

URL:	<a href="https://bsnl7xt7eab.bsngate.com:19602/explorer">https://bsnl7xt7eab.bsngate.com:19602/explorer</a>	<a href="#">Copy</a>
------	---	----------------------

2. When the framework is Hyperledger Fabric-2.3.2, the page is shown as below:

Dedicated Node Services / Details

### Basic Information

Project Name: hf232

Framework: Hyperledger Fabric-v2.3.2

consortium: test

Cloud Platform: aws

Payment Status: Paid

ConsensusType: raft

channelId: hf

Region: Beijing

Created Date: (UTC+8:00) 01/19/2022 17:20:33

### Resource Information

Node Resource and Cost Information

Number of Nodes	Number of orderers	Host Configuration	Data Capacity	Price (USD/month)	Price (USD/year)
3	3	2Core+4GB	50GB	280.98	3066.32

Gateway Cost Information:

Number of Nodes	Host Configuration	Data Capacity	Price (USD/month)	Price (USD/year)
1	2Core+4GB	50GB	93.66	1022.1

Data Usage Information

Inbound Data Usage (USD/GB)	Outbound Data Usage (USD/GB)
0	0.2

### Deployment Information

Deployment node list

Access to Peer node, Orderer and CA certificate service requires downloading CA certificate and configuration information; the account/password for accessing the gateway service browser is: admin/adminpw

Click to download CA certificate and configuration information

Type	Peer Name	Status	Deployment Time	Action
Peer Node	peer1.org1.test.com	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a>
Peer Node	peer3.org1.test.com	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a>
Peer Node	peer2.org1.test.com	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a>
Gateway Services	Browser	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a> <a href="#">Open URL</a>
Orderer	orderer1.orderer.test.com	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a>
Orderer	orderer3.orderer.test.com	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a>
Orderer	orderer2.orderer.test.com	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a>
CA Certificate Service	ca.org1.test.com	Running	(UTC+8:00) 01/19/2022 17:25:15	<a href="#">Details</a>

Back

- Basic Information:** Project Name, Framework, Consensus, Cloud Platform, Region, Consortium Name, Channel Name, Payment Status and Created Date.
- Resource Information:** Node resource and cost information, Gateway Cost Information and Data Usage Information.
- Deployment Information:** Click on the "Click to download CA certificate information" button to download the certificate. Clicking on the "Details" button corresponding to the peer node, developer can view the information of Access and Credentials.

**Access and Credentials**

RPC Endpoint

grpcs://18.167.69.153:1051

Copy

- 4) Clicking on the "**Details**" button corresponding to the browser, developer can obtain the URL address of the blockchain explorer.

**Access and Credentials**

URL:

http://71.131.227.181:18080

Copy

Close

3. When the framework is Hyperledger Besu-v21.1.2, the page is shown as below:

Dedicated Node Services / Details

### Basic Information

Project Name: TestBesu  
 Framework: Hyperledger Besu -v21.1.2  
 Cloud Platform: AWS  
 Payment Status: Paid  
 Consensus: Clique  
 Region: Hong Kong  
 Created Date: (UTC+8:00) 07/22/2021 14:26:06

### Resource Information

Node Resource and Cost Information

Number of Nodes	Host Configuration	Data Capacity	Price (USD/month)	Price (USD/year)
4	4Core+8GB	50GB	996.56	10392.4

Gateway Cost Information:

Number of Nodes	Host Configuration	Data Capacity	Price (USD/month)	Price (USD/year)
1	4Core+8GB	50GB	249.14	2598.1

Data Usage Information

Inbound Data Usage (USD/GB)	Outbound Data Usage (USD/GB)
0	0.01

### Deployment Information

Deployment node list

Authorized Username: VVOIgrhtgInDSqfTR Authorized Password: \*\*\*\*\*

Type	Node Name	Status	Deployment Time	Action
Peer Node	node2	Running	(UTC+8:00) 07/22/2021 14:29:02	<a href="#">Details</a>
Peer Node	node1	Running	(UTC+8:00) 07/22/2021 14:29:02	<a href="#">Details</a>
Peer Node	node4	Running	(UTC+8:00) 07/22/2021 14:29:02	<a href="#">Details</a>
Peer Node	node3	Running	(UTC+8:00) 07/22/2021 14:29:02	<a href="#">Details</a>
Gateway Services	Browser	Running	(UTC+8:00) 07/22/2021 14:29:02	<a href="#">Details</a> <a href="#">Open URL</a>

- 1) Basic Information:** Project Name, Framework, Consensus, Cloud Platform, Region, Payment Status and Created Date.
- 2) Resource Information:** Node resource and cost information, Gateway Cost Information and Data Usage Information.
- 3) Deployment Information:** The developer can view node information and browser information. Clicking on the "**Details**" button corresponding to the peer node, developer can view the information of Access and Credentials, Transaction Manager cluster, and node information.

### Access and Credentials

RPC Endpoint	https://bsnH8WhcljA.bsngate.com:19602/node2	Copy
Transaction Manager (TM) Endpoint	https://bsnH8WhcljA.bsngate.com:19602/tm2	Copy

### Transaction Manager Cluster

Public Key	*****	Copy
Private Key	*****	Copy

### Node Information

Address	0x4A253dC4fC1f404D5301ae4861ec8a19CA1F9bBC	Copy
Public Key	*****	Copy
Private Key	*****	Copy

- 4) By clicking on the **"Details"** button corresponding to gateway services, the developer can obtain the URL address of the blockchain explorer.

### Access and Credentials

URL:	https://bsnH8WhcljA.bsngate.com:19602	Copy
------	---------------------------------------	------

### 6.2.5 Unsubscribe Projects

For the dedicated node service in **Running** status, the publisher can unsubscribe that project:

The screenshot shows a table with columns: Cloud Platform, Region, Payment Status, Payment Type, Deployment Time, Status, and Action. Two rows are visible, both with 'Running' status. In the 'Action' column, there are links for 'Details', 'Unsubscribe', and 'Edit Authorized Account'. The 'Unsubscribe' link is highlighted with a red box. A modal dialog is open in the foreground, asking for confirmation to unsubscribe from the project 'test'.

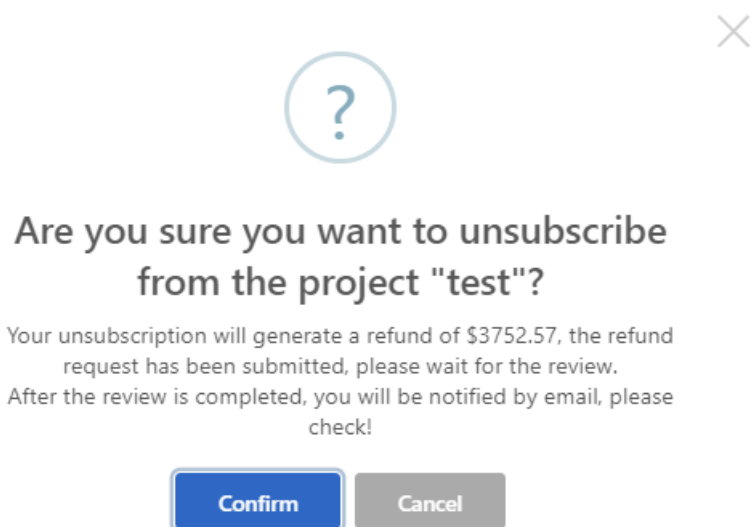
**Are you sure you want to unsubscribe from the project "test"?**

Your unsubscription will generate a refund of \$3752.57, the refund request has been submitted, please wait for the review. After the review is completed, you will be notified by email, please check!

Buttons: Confirm, Cancel

For users who pay monthly for node and gateway resources, no refund will be generated when unsubscribing; for users who pay annually for node and gateway resources, refunds will be

made at the point of time from the next month to the end of the billing cycle when unsubscribing. The discount policy for annual payment will be cancelled and the refund will be calculated by actual refundable months.



### 6.2.6 Edit Authorized Account

Authorized account is mainly used for the verification of connecting nodes or blockchain browsers to increase network security. Only the dedicated node with successful payment and running can edit the authorized account. Click "Edit Authorized Account" button in the dedicated node service list and jump to the page of editing the authorized account. Enter the new username, new password, confirm the new password, and click the "Confirm" button to edit the authorization account.

Edit Authorized Account
×

Current Authorized Username: 6y9BbiAMog594e3of1

New Authorized Username:

New Authorized Password:

Confirm Authorized Password:

Cancel Confirm

### 6.2.7 Configuration Upgrade

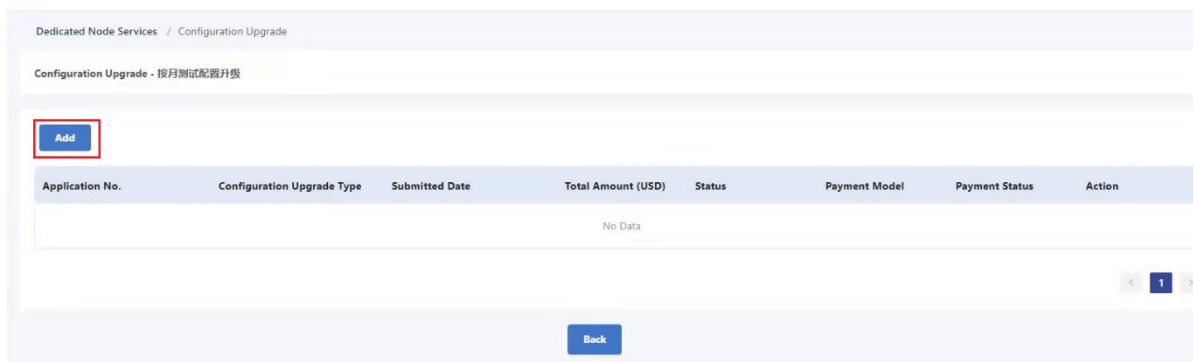
Publishers can upgrade the node information or resource information of the service by using the “**Configuration Upgrade**” function. They need to pay the corresponding resource upgrade fee when upgrading the configuration.



The configuration upgrade cannot delete the original node and downgrade the configuration of the original node. It can only upgrade on the basis of the current configuration.

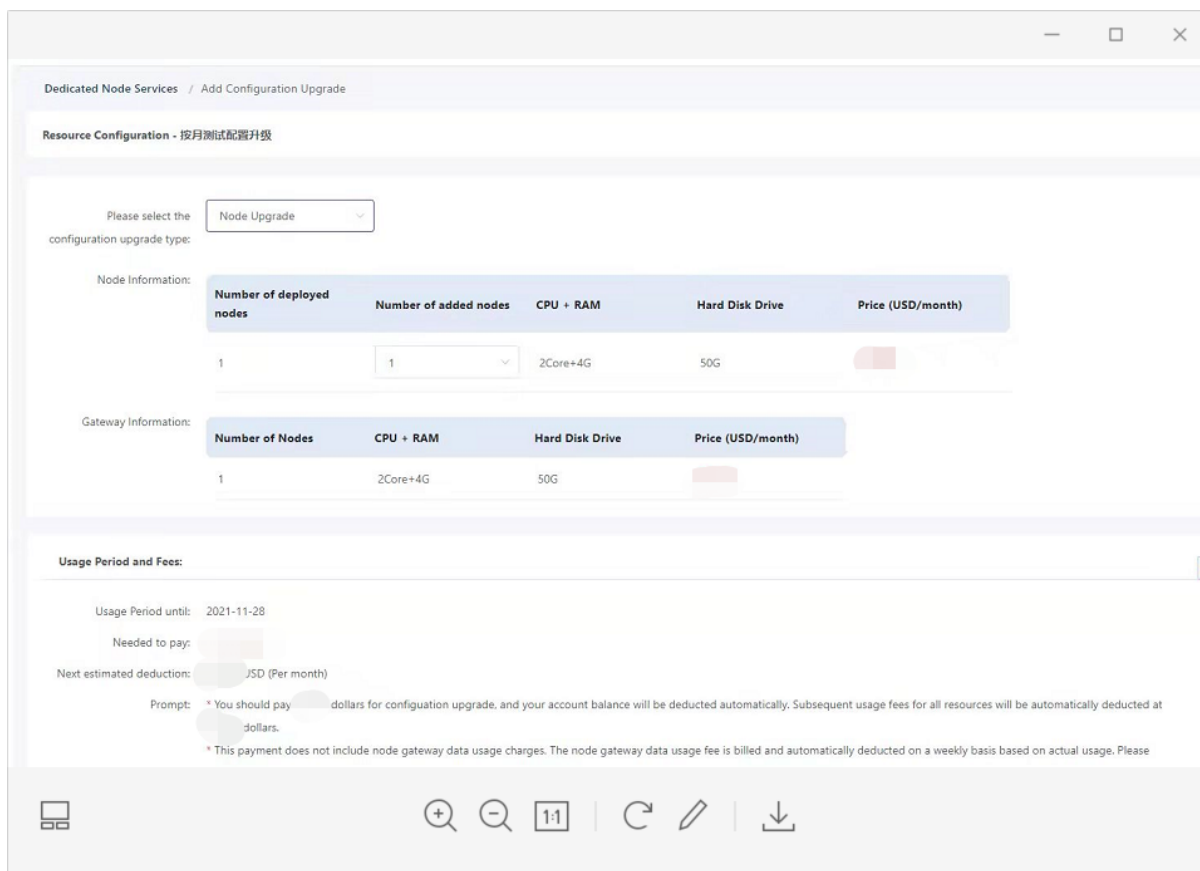
The operation steps of the configuration upgrade are as follows:

Go to “**Permissioned Services**” -> “**Dedicated Node Services**”, select the running service, click “**Configuration Upgrade**” to enter the configuration upgrade list page as below:



In the configuration upgrade list page, click “**Add**” to enter the configuration upgrade application page. The service configuration upgrade type is divided into node upgrade and resource upgrade.

When the publisher is selecting the configuration type as node upgrade, he or she can select the number of new nodes to be added:



When the publisher is selecting the configuration type as resource upgrade, he or she can select the CPU, memory, and hard disk drive to be added in the node information and gateway information:

The screenshot shows the 'Dedicated Node Services / Add Configuration Upgrade' interface. The 'Resource Configuration - 按月测试配置升级' section is active. A dropdown menu shows 'Resource Upgrade' selected. Below, there are two tables for configuration details.

**Node Information:**

Number of deployed nodes	Deployed CPU - RAM	Deployed hard disk drive	CPU - RAM	Hard Disk Drive	Price (USD/month)
1	2Core+4G	50G	2Core4G	50G	

**Gateway Information:**

Number of deployed nodes	Deployed CPU - RAM	Deployed hard disk drive	CPU - RAM	Hard Disk Drive	Price (USD/month)
1	2Core+4G	50G	2Core4G	50G	

**Usage Period and Fees:**

Usage Period until: 2021-11-28  
 Needed to pay: [redacted]  
 Next estimated deduction: [redacted] (Per month)  
 Prompt: \* You should pay [redacted] dollars for configuration upgrade, and your account balance will be deducted automatically. Subsequent usage fees for all resources will be automatically deducted at [redacted] dollars.  
 \* This payment does not include node gateway data usage charges. The node gateway data usage fee is billed and automatically deducted on a weekly basis based on actual usage. Please ensure that there is sufficient balance in your account to avoid affecting the normal operation of the service.

Buttons: Confirm, Back

Click “**Confirm**” to submit the application of configuration upgrade. The system will ask for the payment of the corresponding configuration upgrade:

The screenshot shows the same 'Add Configuration Upgrade' interface, but with a modal dialog box titled 'Confirm the payment' overlaid. The dialog contains a warning icon and the text: 'The configuration upgrade fee is [redacted]. Are you sure to pay?'. There are 'Confirm' and 'Cancel' buttons at the bottom of the dialog.

The background interface is dimmed, showing the same configuration details as the previous screenshot.

After the publisher clicks “**Confirm**”, the system generates a configuration upgrade bill and debits the publisher's account. If the deduction is successful, the system will upgrade the configuration. If the deduction fails, the bill will be retained for 72 hours before expiration, if you still want to upgrade the configuration, you need to re-apply.

*Note: The fee paid for the configuration upgrade is the upgrade fee, which is to make up the difference between the pre-upgrade configuration and the post-upgrade configuration in the billing cycle. After a successful upgrade, the next deduction cycle will be debited according to the cost of new configuration.*

## 6.3 Access Instructions

Once the project is created, the system will allocate the access information to the project corresponding to the framework. The user will verify the access authorization when accessing the dedicated node and can access the blockchain after passing the verification.

### 6.3.1 ConsenSys Quorum Access Instruction

#### 1. Use GoQuorum Client to interact with nodes

Example:

```
$ ./geth attach
https://VuF0h0y7pLwtvDqjuW:y2sYuiIciR6JFtHbmC@bsnmu7d0gNn.bsngate.com:19602/node1

INFO [05-08|16:05:52.534] Running with private transaction manager disabled -
quorum private transactions will not be supported

Welcome to the Geth JavaScript console!

instance: Geth/v1.9.7-stable-af752518(quorum-v20.10.0)/linux-amd64/go1.13.15
coinbase: 0xf957d0ae8a1c1b2cdcea0acb8fb0a2a750abadaa
at block: 109 (Sat May 08 2021 16:05:48 GMT+0800 (CST))
datadir: /root/quorum/data
modules: admin:1.0 debug:1.0 eth:1.0 istanbul:1.0 miner:1.0 net:1.0
personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> web3.eth.blockNumber

11
```

#### 2. Use JSON-RPC API to interact with nodes

- Geth JSON-RPC documentation:

<https://github.com/ethereum/wiki/wiki/JSON-RPC>

- Quorum API documentation:

<https://docs.goquorum.consensus.net/en/latest/Reference/APIs/PrivacyAPI/>

The API can be called by using CURL and Postman.

Example:

```
$ curl -H "Content-Type: application/json" -d
'{"jsonrpc": "2.0", "method": "eth_blockNumber", "params": [], "id": 2}'
https://VuF0h0y7pLwtvDqjuW:y2sYuiIciR6JFtHbmC@bsnmu7d0gNn.bsngate.com:19602/node1
{"jsonrpc": "2.0", "id": 2, "result": "0x10"}
```

### 3. Use web3.js to interact with nodes

- Web3.js class library:

<https://github.com/ChainSafe/web3.js>

Example:

```
const Web3 = require("web3");

const web3 = new Web3(
  new
  Web3.providers.HttpProvider("https://VuF0h0y7pLwtvDqjuW:y2sYuiIciR6JFtHbmC@bsn
mu7d0gNn.bsngate.com:19602/node1")
);

web3.eth.getBlockNumber().then(console.log);
```

## 6.3.2 Hyperledger Fabric Access Instruction

### 1. Use fabric-tools to interact with nodes

Example:

- Start cli:

```
➤ # docker-compose-cli.yaml file:
➤ $ cat docker-compose-cli.yaml
➤ version: '2'
```

```
➤ services:
➤ cli:
➤   container_name: fabric_peercli
➤   image: hyperledger/fabric-tools:2.3.2
➤   restart: always
➤   tty: true
➤   stdin_open: true
➤   environment:
➤     - FABRIC_LOGGING_SPEC=DEBUG
➤     - CORE_PEER_TLS_ENABLED=true
➤     - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
➤     - CORE_PEER_ADDRESS=peer1.org1.example.com:1051
➤     - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/fabric/msp
➤     - CORE_PEER_LOCALMSPID=Org1MSP
➤     -
➤     ORDERER_CA=/etc/hyperledger/fabric/orderer/tlsca/tlsca.orderer.example.com
➤     -cert.pem
➤     - ORDERER_ADDRESS=orderer1.orderer.example.com:7050
➤   working_dir: /etc/hyperledger/fabric
➤   command: /bin/bash
➤   volumes:
➤     - /var/run/:/host/var/run/
➤     - ./certs/ordererOrganizations/orderer.example.com:/etc/hyperledger/fabric
➤       /orderer
➤     - ./certs/peerOrganizations/org1.example.com/users/Admin@org1.example.com/
➤       msp:/etc/hyperledger/fabric/msp
➤     - ./certs/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/
➤       tls:/etc/hyperledger/fabric/tls
➤     - ./sharedfiles/chaincode:/etc/hyperledger/fabric/src
➤   extra_hosts:
```

```
➤ - "orderer1.orderer.example.com:161.189.69.75"
➤ - "peer1.org1.example.com:161.189.69.75"
```

➤ Access to containers for chaincode deployment and invocation

```
➤ #Chaincode deployment
➤ // package
➤ peer lifecycle chaincode package basic414.tar.gz --path ./asset-transfer-
  basic/chaincode-javascript/ --lang node --label basic414
➤ // install
➤ peer lifecycle chaincode install basic414.tar.gz
➤ // queryinstalled
➤ peer lifecycle chaincode queryinstalled
➤ // approveformyorg
➤ peer lifecycle chaincode approveformyorg --name basic414 --package-id
  basic414:16bf72ced8451fc6fd94bd139de1532adfdd190af075c2e84a87513915a97365
  -o $ORDERER_ADDRESS --tls --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE
  --cafile $ORDERER_CA --version 1.0 --channelID netchannel --sequence 1 --
  connTimeout 30s
➤ // queryapproved
➤ peer lifecycle chaincode queryapproved --channelID netchannel -n basic414
➤ // checkcommitreadiness
➤ peer lifecycle chaincode checkcommitreadiness --channelID netchannel --
  name basic414 --version 1.0 --sequence 1 --output json
➤ // commit
➤ peer lifecycle chaincode commit -o $ORDERER_ADDRESS --cafile $ORDERER_CA -
  -channelID netchannel --name basic414 --version 1.0 --sequence 1 --
  peerAddresses $CORE_PEER_ADDRESS --tls --tlsRootCertFiles
  $CORE_PEER_TLS_ROOTCERT_FILE
➤ // querycommitted
➤ peer lifecycle chaincode querycommitted -o $ORDERER_ADDRESS --channelID
  netchannel --tls --tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE --cafile
  $ORDERER_CA
➤
➤ #Invoke the chaincode
➤ // InitLedger
```

```

➤ peer chaincode invoke -o $ORDERER_ADDRESS --tls --cafile $ORDERER_CA -C
  netchannel -n basic414 --peerAddresses $CORE_PEER_ADDRESS --
  tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE -c '{"Args":["InitLedger"]}'

➤ // GetAllAssets

➤ peer chaincode query -C netchannel -n basic414 -c
  '{"Args":["GetAllAssets"]}'

➤ // CreateAsset

➤ peer chaincode invoke -o $ORDERER_ADDRESS --tls --cafile $ORDERER_CA -C
  netchannel -n basic414 --peerAddresses $CORE_PEER_ADDRESS --
  tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE -c
  '{"Args":["CreateAsset","asset7","white", "15", "zx1", "800"]}'

➤ //UpdateAsset

➤ peer chaincode invoke -o $ORDERER_ADDRESS --tls --cafile $ORDERER_CA -C
  netchannel -n basic414 --peerAddresses $CORE_PEER_ADDRESS --
  tlsRootCertFiles $CORE_PEER_TLS_ROOTCERT_FILE -c
  '{"Args":["UpdateAsset","asset7","zx1", "1218", "zx1", "1218"]}'

➤ //ReadAsset

➤ peer chaincode query -C netchannel -n basic414 -c
  '{"Args":["ReadAsset","asset7"]}'

```

### 6.3.3 Hyperledger Besu Access Instruction

1. Interact with node by Geth console.

Example:

```

[root@localhost ~]# geth attach https://admin:123456@bsn91000001.bsngate.com:1
9602/node1

INFO [07-
15|13:41:24.391] Running with private transaction manager disabled - quorum pr
ivate transactions will not be supported

WARNING: call to admin.getNodeInfo() failed, unable to determine consensus mec
hanism

Welcome to the Geth JavaScript console!

instance: besu/v21.1.2/linux-x86_64/oracle_openjdk-java-11
coinbase: 0xfd0fdb96a3326c61756711d314bea0149088c3d9
at block: 195 (Thu Jul 15 2021 13:41:24 GMT+0800 (CST))

modules: eea:1.0 eth:1.0 ibft:1.0 net:1.0 perm:1.0 priv:1.0 web3:1.0

```

```
> web3.eth.blockNumber  
  
202  
  
>
```

## 2. Interact with nodes by JSON-RPC.

```
[Geth JSON-RPC document](https://github.com/ethereum/wiki/wiki/JSON-RPC):  
  
[Quorum API document](https://besu.hyperledger.org/en/stable/Reference/API-Methods/): You can use curl and Postman to call the API.
```

- Example:

```
```shell  
[root@localhost ~]# curl -H "Content-Type: application/json" -d \  
> '{"jsonrpc":"2.0","method":"eth_blockNumber","params":[],"id":2}' \  
> https://admin:123456@bsn91000001.bsngate.com:19602/node1  
  
{  
  "jsonrpc" : "2.0",  
  "id" : 2,  
  "result" : "0x365"  
}  
```
```

## 3. Interact with node by Web3.js.

```
- [web3.js class base](https://github.com/ChainSafe/web3.js)  
  
- Example:
```



```
```shell

const Web3 = require("web3");

const web3 = new Web3(
  new Web3.providers.HttpProvider("https://VuF0h0y7pLwtvDqjuW:y2sYuiIciR6JFtHb
mC@bsnmu7d0gNn.bsngate.com:19602/node1")
);

web3.eth.getBlockNumber().then(console.log);

```
```

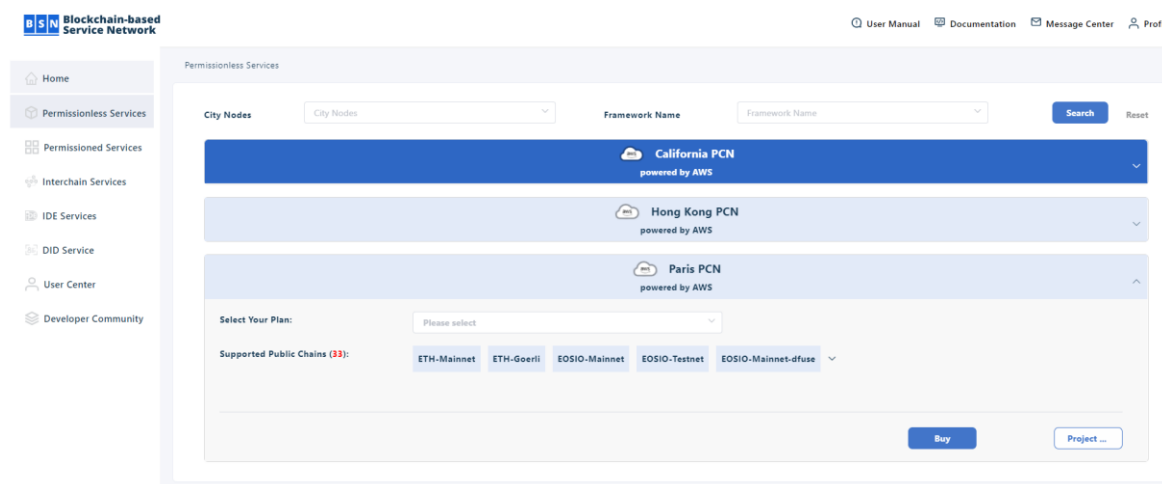
## 7 Permissionless Services

### 7.1 Overview

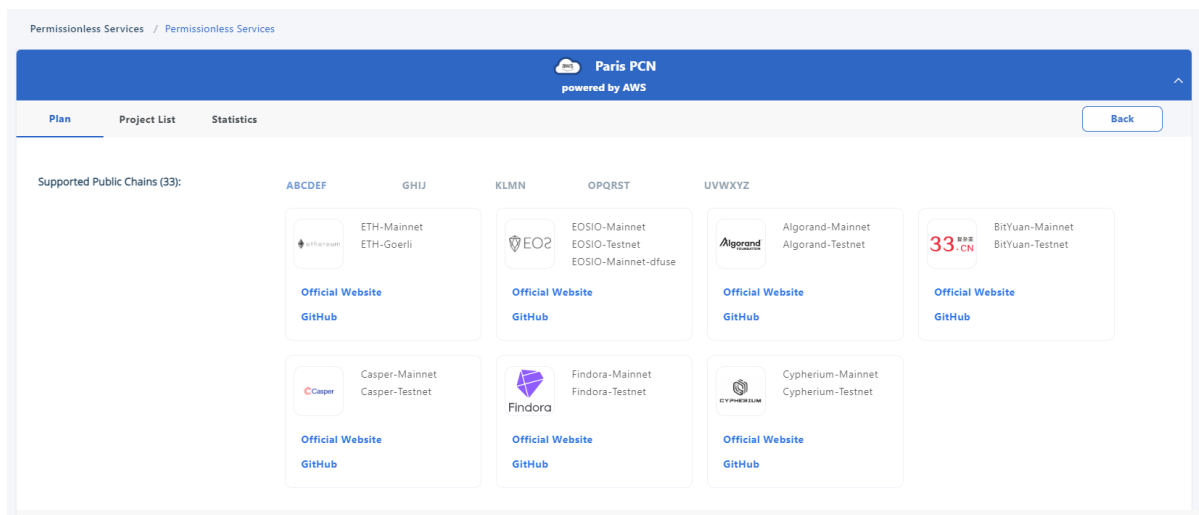
The Permissionless service allows the participant to select a public city node to access a plan that can be a free plan or a premium plan. When this is done, the participant can create a project, obtain the project ID, key and access parameters which can be used to access selected public chain node gateway. With the Permissionless service, the default plan is free for participants, however, it has limited daily requests and projects. BSN has created several other plans that can be upgraded to, for a certain fee, paid on a monthly basis.

### 7.2 Select Plans

On the page of Permissionless services, users can select different city nodes to participate in Permissionless services. The nodes in blue at the top of the list represent those activated for the free plan or premium plans on that city node. The nodes in grey at the bottom represent no plans are purchased or used on that city node.

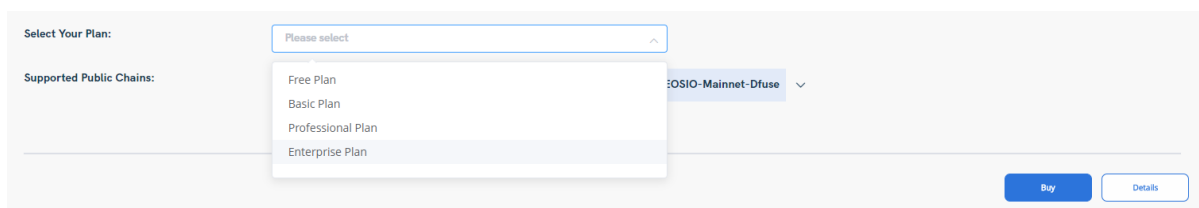


When you click and expand the public city node, you can see all public chain frameworks supported by the city node. Users can decide whether to choose this city node as the access entrance according to their needs. The public chain frameworks supported by different city nodes may be different. In general, we recommend that developers choose a city node that is close to them, so that the access speed will be relatively fast.

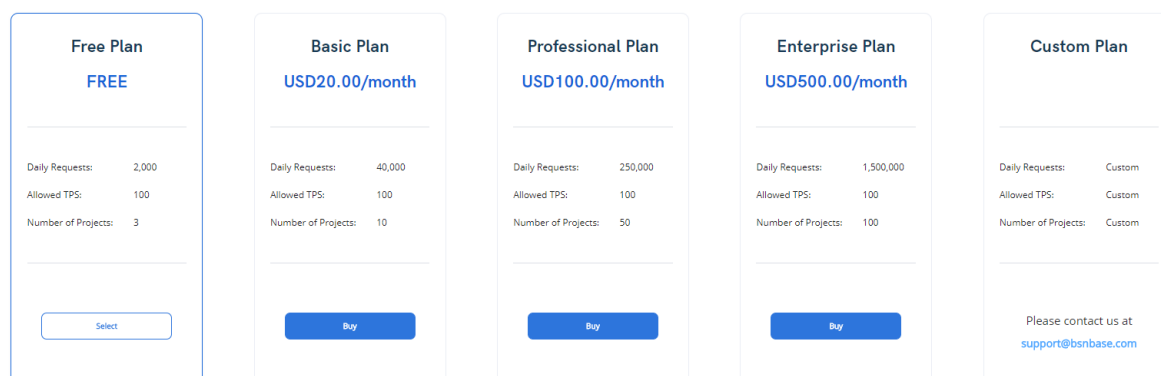


By default, participants on the Permissionless service have a **free** plan that is free to use up to 2000 daily requests, allowed TPS of 100 and maximum of 3 projects. However, a participant can upgrade to a higher plan available on the platform. To select plans, follow these steps

1. On the Permissionless page, click **Buy** in the **Select Your Plan** section.



2. In the **Details** page, locate the **Select or Update your plan** section and click **Buy** on the appropriate plan.



3. In the **Are you sure you want to buy package** window, click the project agreement and click **Confirm**.



Are you sure to select Basic Plan?

☒ Read and agree to [BSN public chain project protocol/agreement](#).

Confirm

4. In the **Select Payment Method** page, select the appropriate payment method and click **Next Step to be redirected to Stripe**.

**The BSN portal never records and stores any credit card information.**

Checkout

×

Overview

Basic plan

20.00USD/month

Daily Requests:

40000


Allowed TPS:







100

Number of Projects:

10

Payment Method

 Pay by Credit Card



| Description | Quantity | Price    |
|-------------|----------|----------|
| Basic plan  | 1        | 20.00USD |

Total: 20.00USD

Confirm

[Go back](#)

You will be directed to Stripe. We never store credit card information

- On the **Stripe Payment** page, click **Pay** to display the **Receipt** and **Invoice**.

## Invoice from RED DATE (HONG KONG) TECHNOLOGY LIMITED

Billed to billjackson5  
Invoice #841DA2D2-0001

\$20.00 USD due Aug 13, 2020



Card number

MM / YY CVC

Pay invoice

| DESCRIPTION                         | QTY | PRICE   | TOTAL              |
|-------------------------------------|-----|---------|--------------------|
| "Basic Plan" Permissionless service | 1   | \$20.00 | \$20.00            |
| <a href="#">↓ PDF</a>               |     |         | Amount due \$20.00 |

If you have any questions, contact RED DATE (HONG KONG) TECHNOLOGY LIMITED at [support@bsnbase.com](mailto:support@bsnbase.com) or call +86 10 8646 2811.

### 7.3 Create and Manage Projects

With the Permissionless service, projects can be created in a much simpler way when compared with Permissioned service as plans are embedded into the project, making it easier for participants to manage. To create and manage projects follow these steps:

1. In the Permissionless Service page, click **Create Project** in the development plan section.

2. In the **Create a new project** window, enter the **Project Name**, select the Public **Chain to access** from the dropdown list, input the **Daily Requests** number if needed. Then click **Create Project**. The Daily Requests number is optional, and it is used to control the TPD (transactions per day) for this project.

This will automatically create the project and list it in the **Project Information** tab.

After a project has been created it can be managed using available tools for the project. To manage a project, follow these steps:

1. Locate the project to be managed, click **Upgrade** to display the **Plans** page.

2. Select the appropriate plan to **Upgrade** to and click **confirm** to display the payment page.
3. To enable the project key, in the Permissionless Service page, click **Project list** to display the list of projects. In **Action**, click **Enable Key** to enable the project key. Then the information page on enabling the key will be displayed. Click **Confirm**.

| Project Name | Public Chain | Daily Requests | Project ID   | Project Key | Access Address   | Action   |
|--------------|--------------|----------------|--|-------------|--|--|
| ETHmain      | ETH-Mainnet  | 200            | 5b14244913ebe275d770edcb1502cc6e6e7c7dc3f670aa865f5068828e49923c |             | http://192.168.1.187:8080/api/5b14244913ebe275d770edcb1502cc6e6e7c7dc3f670aa865f5068828e49923c/ETH-Mainnet/rpc | <div> <div>Enable Key</div> <div>Delete</div> </div> |

4. To update a project key, click **Update Key**. Then the information page on updating the key will be displayed. Click **Confirm**.

| Project Name | Public Chain | Daily Requests | Project ID   | Project Key  | Access Address   | Action  |
|--------------|--------------|----------------|--|--|--|---|
| ETHmain      | ETH-Mainnet  | 200            | 5b14244913ebe275d770edcb1502cc6e6e7c7dc3f670aa865f5068828e49923c | aaf3e633160e09b6064a27b0c8ae44a3fbb10299e115959bbae838bc19fbbc23 | http://192.168.1.187:8080/api/5b14244913ebe275d770edcb1502cc6e6e7c7dc3f670aa865f5068828e49923c/ETH-Mainnet/rpc | <div> <div>Update Key</div> <div>Disable Key</div> <div>Delete</div> </div> |

1 items found, display 1 to 1

5. To delete a project, click **Delete**. A confirmation message will be displayed asking if you wanted to delete the project. Click **Confirm** to delete it.



Are you sure you want to delete?

Confirm

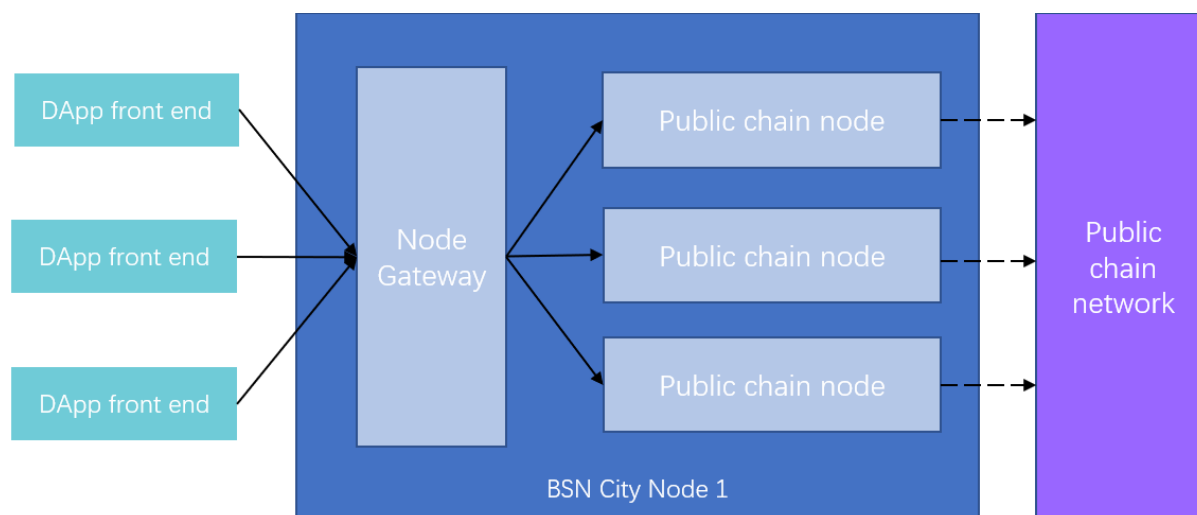
Cancel

## 7.4 Off-BSN system Access Guide

### 7.4.1 Overview

BSN provides shared or dedicated public chain nodes for public chain application developers. Developers can quickly access all public chain networks by accessing the gateway of the public city node.





After developers select the public chain framework (netcode) in the BSN portal to create the public chain project, they will get the gateway's domain name address (url), project number (id), project key (key), public chain supportive protocol {protocol} and public chain gateway API address.

The developer accessing the PCN gateway via HTTP should concatenate the request address in "https://{url}/api/{id}/{netcode}/{protocol}/{subUrl}" format. If project key is enabled, "x-API-key:{key}" should be added to the request header. If the public chain nodes provide multiple components, they should add {subUrl}; If the Nervos CKB has an Indexer component service in addition to the RPC service, "{subUrl}" should fill the indexer value, {subUrl} is optional.

The developer accessing the node gateway via WebSocket, should concatenate the Key and SubUrl to the path address of the target machine and concatenate to the format of {url}/api/{id}/{key}/{netcode}/{subUrl}. If the project key is not enabled, then the {key} filed should be null. If there is no subUrl, this field can be null. That is, developers can think of the content after/API as the method name of a target machine.

### 7.4.2 Ethereum

Ethereum is a global, open-source platform for decentralized applications. On Ethereum, you can write code that controls digital value, runs exactly as programmed, and is accessible anywhere in the world.

For more resources, please visit: <https://ethereum.org/en/developers/>

The BSN public city node gateway is adapted to the Ethereum JSON RPC API, so developers can initiate transaction requests to the node gateway via HTTP JS-RPC. For detailed docking instructions please visit: <https://eth.wiki/json-rpc/API>

The following table shows additional error code definitions for public city node gateways:

| Error code | Transaction error code | Error code description     |
|------------|------------------------|----------------------------|
| 500        | -32099                 | Service internal exception |
| 503        |                        |                            |
| 429        | -32098                 | TPS, TPD current limit     |

|     |        |                                  |
|-----|--------|----------------------------------|
| 401 | -32097 | Authentication permission failed |
|-----|--------|----------------------------------|

### 7.4.3 EOS

EOSIO is a blockchain platform designed for the real world. Built for both public and private use cases, EOSIO is customizable to suit a wide range of business needs across industries with rich role-based security permissions, industry-leading speeds and secure application processing.

For more resources, please visit: <https://developers.eos.io>

The BSN city node gateway is adapted to EOSIO's JSON RPC API, so developers can initiate transaction requests to the node gateway via HTTP JSON-RPC. For detailed docking instructions please visit:

[https://developers.eos.io/manuals/eos/latest/nodeos/plugins/chain\\_api\\_plugin/api-reference/index#operation/get\\_block](https://developers.eos.io/manuals/eos/latest/nodeos/plugins/chain_api_plugin/api-reference/index#operation/get_block)

The following table shows additional error code definitions for public city node gateway:

| Error code | Transaction error code | Error code description           |
|------------|------------------------|----------------------------------|
| 401        | 3090000                | Authentication permission failed |
| 429        | 3210000                | TPS, TPD current limit           |
| 500        | 3100000                | Service internal exception       |
| 503        |                        | Service Unavailable              |

### 7.4.4 Tezos

Tezos is an open-source platform for assets and applications backed by a global community of validators, researchers, and builders. Tezos is designed to provide the safety and code correctness required for assets and other high value use cases. Its native smart contract language, Michelson, facilitates formal verification, a methodology commonly used in mission-critical environments such as the aerospace, nuclear, and semiconductor industries.

For more resources, please visit: <https://developers.tezos.com>

The BSN city node gateway is adapted to the Tezos JSON RPC API, so developers can initiate transaction requests to the node gateway via HTTP JSON-RPC. For detailed docking instructions please visit:

<https://tezos.gitlab.io/api/rpc.html>

The following table shows additional error code definitions for public city node gateway:

| Error code | Transaction error code | Error code description           |
|------------|------------------------|----------------------------------|
| 401        | 3090000                | Authentication permission failed |
| 429        | 3210000                | TPS, TPD current limit           |
| 500        | 3100000                | Service internal exception       |
| 503        |                        | Service Unavailable              |

### 7.4.5 dfuse-eos

dfuse is a massively scalable open-source platform for searching and processing blockchain data. It provides real-time, historical and fork-aware search engine (dfuse Search), transaction push guarantees (dfuse Push Guarantee), transaction lifecycle (dfuse Lifecycle), historical state services (dfuse State), and many more blockchain building blocks. dfuse empowers developers with capabilities to build modern blockchain applications with fast, fluid interfaces that deliver exceptional user experiences.

For more resources, please visit: <https://docs.dfuse.io>

The current dfuse EOS mainnet access on BSN is available through dfuse Community Edition hosted by EOS Nation. Try out the dfuse API features on the [GraphiQL playground] (<https://eos.dfuse.eosnation.io/graphiql>).

If your needs exceed the Community Edition limits, please contact dfuse to set up an [Enterprise plan] ([https://dfuse.io/zh/pricing/?utm\\_source=BSN](https://dfuse.io/zh/pricing/?utm_source=BSN)).

The BSN PCN gateway is equipped with dfuse's JSON RPC API and GraphQL, so developers can issue EOSIO transaction requests to the node gateway via HTTP JSONrpc or GraphQL.

For detailed instructions, please visit the link: <https://docs.dfuse.io/reference/eosio>

The following table shows additional error code definitions for public city node gateway:

| Error code | Transaction error code | Error code description           |
|------------|------------------------|----------------------------------|
| 401        | 3090000                | Authentication permission failed |
| 429        | 3210000                | TPS, TPD current limit           |
| 500        | 3100000                | Service internal exception       |
| 503        |                        | Service Unavailable              |

### 7.4.6 Findora

Findora's mission is to build a decentralized financial network for issuing confidential assets and smart contracts. Findora has created a system that achieves privacy-preserving transparency. Its flexible technology can also be used by institutions to replace their current infrastructure or deploy in the cloud – all interoperable with the public Findora network.

For more resources, please visit: <https://findora.org/>

The BSN city node gateway is adapted to the Findora REST API, so developers can initiate transaction requests to the node gateway via HTTP REST. For detailed docking instructions please visit:

<https://api.findora.org/>

The following table shows additional error code definitions for public city node gateway:

| Error code | Transaction error code | Error code description |
|------------|------------------------|------------------------|
|------------|------------------------|------------------------|

|     |         |                                  |
|-----|---------|----------------------------------|
| 401 | 3090000 | Authentication permission failed |
| 429 | 3210000 | TPS, TPD current limit           |
| 500 | 3100000 | Service internal exception       |
| 503 |         | Service Unavailable              |

#### 7.4.7 Near

NEAR is a Proof-of-Stake Layer-1 public blockchain platform built with usability and developer accessibility in mind. With a novel sharding mechanism called Nightshade, NEAR can scale limitlessly and offers familiar user experiences just like the web today.

For more resources, please visit: <https://near.org/>

The BSN city node gateway is adapted to the Near JSON RPC API, so developers can initiate transaction requests to the node gateway via HTTP JSON-RPC. For detailed docking instructions please visit:

<https://docs.near.org>

The following table shows additional error code definitions for public city node gateway:

| Error code | Transaction error code | Error code description           |
|------------|------------------------|----------------------------------|
| 401        | 3090000                | Authentication permission failed |
| 429        | 3210000                | TPS, TPD current limit           |
| 500        | 3100000                | Service internal exception       |
| 503        |                        | Service Unavailable              |

#### 7.4.8 Klaytn

Klaytn is a Global Public Blockchain Platform created to realise the vision of a new internet world. It is designed to be a practical blockchain with the aim of being a technology that anyone can adopt in their daily lives. Klaytn follows a consortium structure where leading global companies of various industries actively participate and operate together. The key features include fast response time, high efficiency, and a stable transaction fee structure, which makes the platform ideal for operating large-scale user services. It also adopts a Hybrid Blockchain Platform format that combines public and private blockchains to meet the needs of services and businesses that target a large user base.

For more resources, please visit: <https://www.klaytn.com/>

The BSN city node gateway is adapted to the Klaytn JSON RPC API, so developers can initiate transaction requests to the node gateway via HTTP JSON-RPC. For detailed docking instructions please visit:

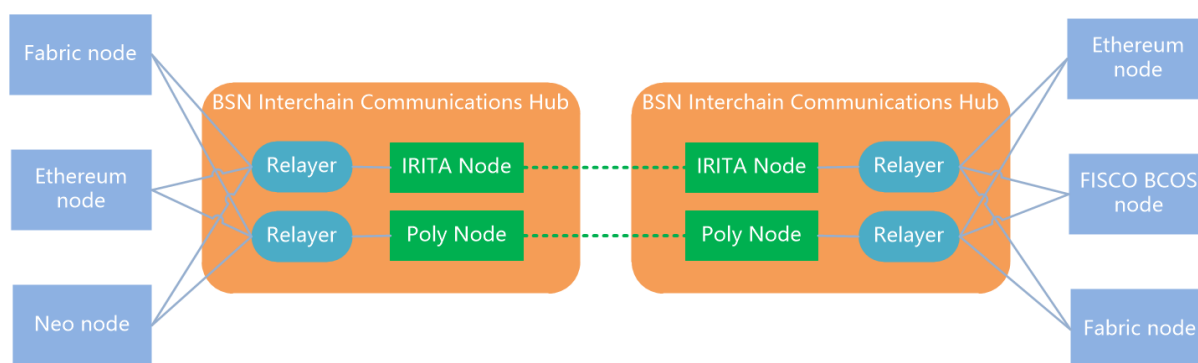
<https://refs.klaytnapi.com/en/node/latest>

The following table shows additional error code definitions for public city node gateway:

| Error code | Transaction error code | Error code description           |
|------------|------------------------|----------------------------------|
| 401        | 3090000                | Authentication permission failed |
| 429        | 3210000                | TPS, TPD current limit           |
| 500        | 3100000                | Service internal exception       |
| 503        |                        | Service Unavailable              |

## 8 Interchain Services

A cross-chain mechanism is the interoperability between two or more relatively independent blockchains, and it enables the swap and transfer of data, asset and information. On the BSN, every blockchain maintains its own transactions, consensus, and ledgers, carrying business data and information of different DApps. The cross-chain mechanism realizes data sharing and business collaboration among blockchains, and to break the silos between chains, allows data to flow securely and reliably across multiple chains. The main functions of the cross-chain system include: cross-chain registration management mechanism, cross-chain contract functions, cross-chain transaction verification, cross-chain message routing protocol, cross-chain transaction atomicity guarantee, etc.



The BSN Interchain Communications Hub (ICH) adopts the cross-chain protocol of heterogeneous chains and the design of double-layer structure, using relay chains as cross-chain coordinators, multiple heterogeneous chains as cross-chain transaction executors, and acts as a relay of cross-chain data. By solving validity, security, and transactional issues of cross-chain data, a secure, easy-to-use and efficient cross-chain system is implemented:

- Supports both isomorphic and heterogeneous chains.
- Supports any information to cross the chains.
- Very easy to access. Application chains do not need to do custom development adaptation, just deploy one smart contract per chain.
- Transactional support, supporting not only scenarios with the need for ultimate consistency of transactions, but also scenarios with the need for strong consistency of transactions, with support for any transaction, and scalable to any number of chains.
- Cross-chain protocols are secure and reliable, based on cryptography and consensus algorithms, and each application chain can verify the legitimacy of cross-chain transactions on its own, thus ensuring the security of cross-chain interactions.

The BSN's "Interchain Communications Hub" (ICH) is now commercially available and integrates with Onchain's Poly Enterprise and Bianjie's IRITA cross-chain solutions. It enables cross-chain interoperability between standard permissioned chains, open permissioned chains and public chains.

A demo version of ICH is also live on the BSN Testnet, integrating two interchain solutions based on the relay chain mechanism: Poly Enterprise developed by Onchain and IRITA developed by Bianjie AI. We welcome all developers to try out and provide feedback and suggestions to us and we will continue to improve the interchain functionality.

## 8.1 Interchain Service Management

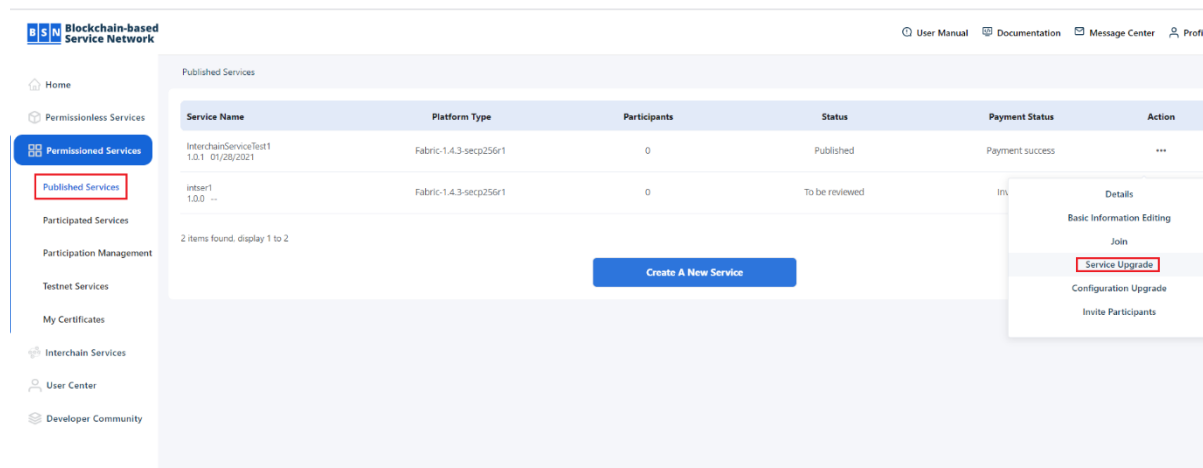
### 8.1.1 Open Interchain Services

There are two ways to open Interchain Services: permissioned DApp service publishers can either open it when upgrading their services, or they can open Interchain Services separately.

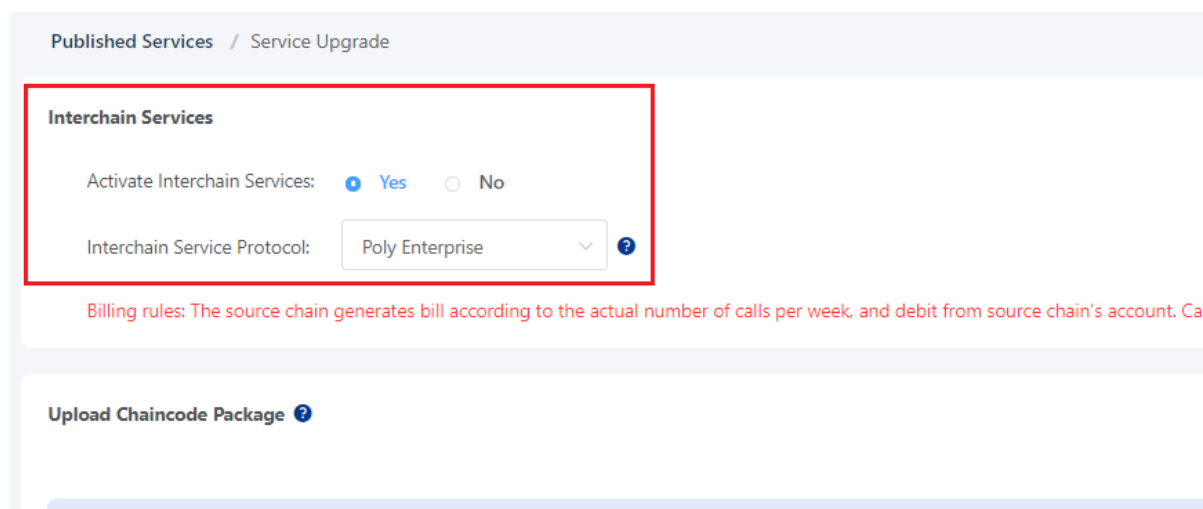
1) Open the Interchain Service when upgrading the permissioned service.

For published permissioned services, publishers can open Interchain Services through the **Service Upgrade** function:

On the home page, click **Permissioned Services** -> **Published Services**, click **Service Upgrade** in the **Action** column to enter the service upgrade page.



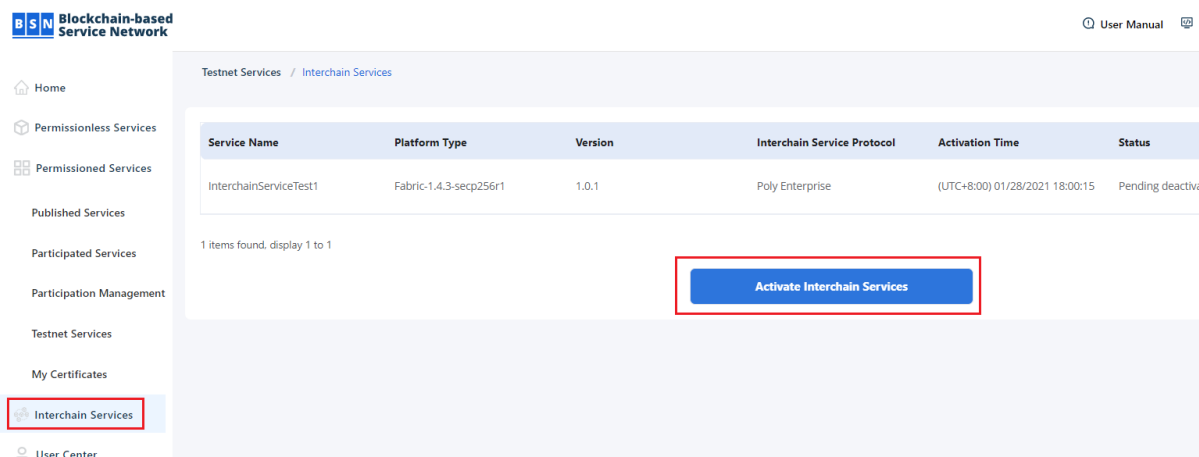
In the Interchain Services section, select **Yes** to activate Interchain Services, and choose the Interchain Service Protocol. Then, click **Confirm** to submit the service upgrade. After the system review and approval, the Interchain Service is successfully opened.



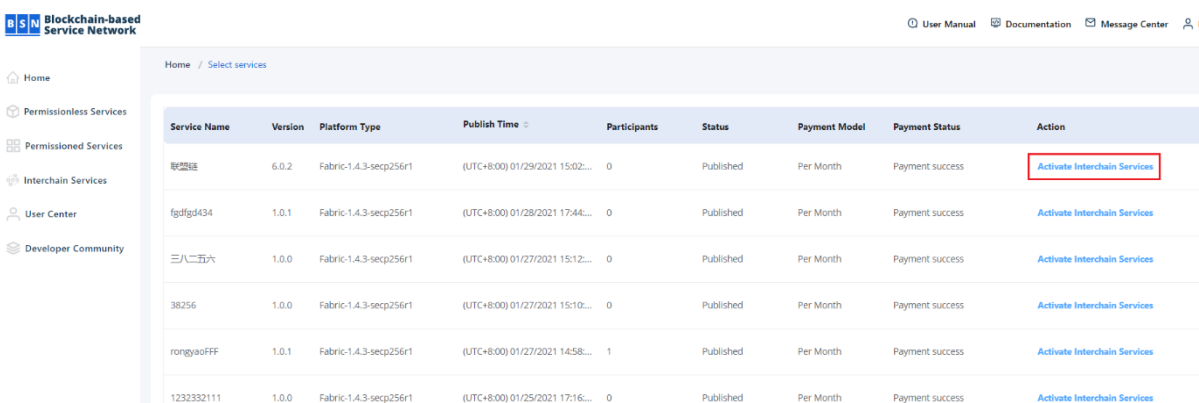
Note: If you open Interchain Services only, you don't need to upload new chaincode package; after opening the service, when calling across the chain, both source chain and target chain need to communicate off the BSN about cross-chain parameters, methods and specifications.

## 2) Directly open the service in Interchain Services

On the home page, click **Interchain Services**



Click **Activate Interchain Services** button to enter **Select services** page, click **Activate Interchain Services** in the **Action** column.



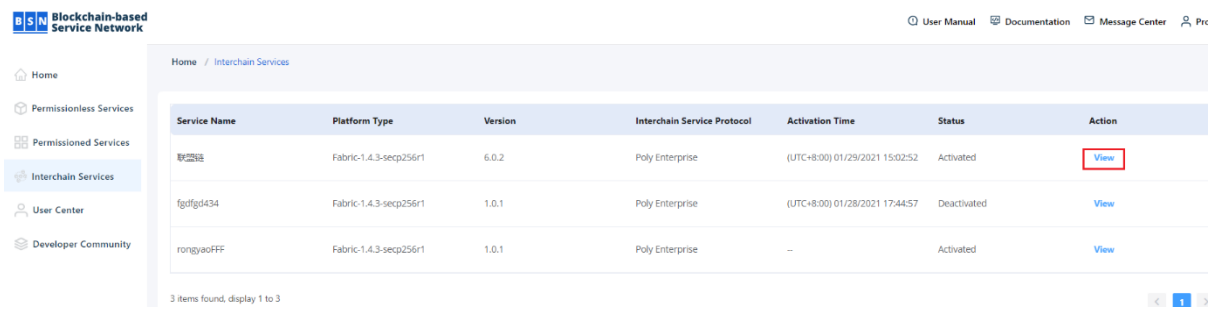
The following steps can refer to **Open the Interchain Service when upgrading the permissioned service**.

Note: For activated interchain services, users cannot change the interchain service protocols. The protocol can only be changed by re-opening the interchain services.

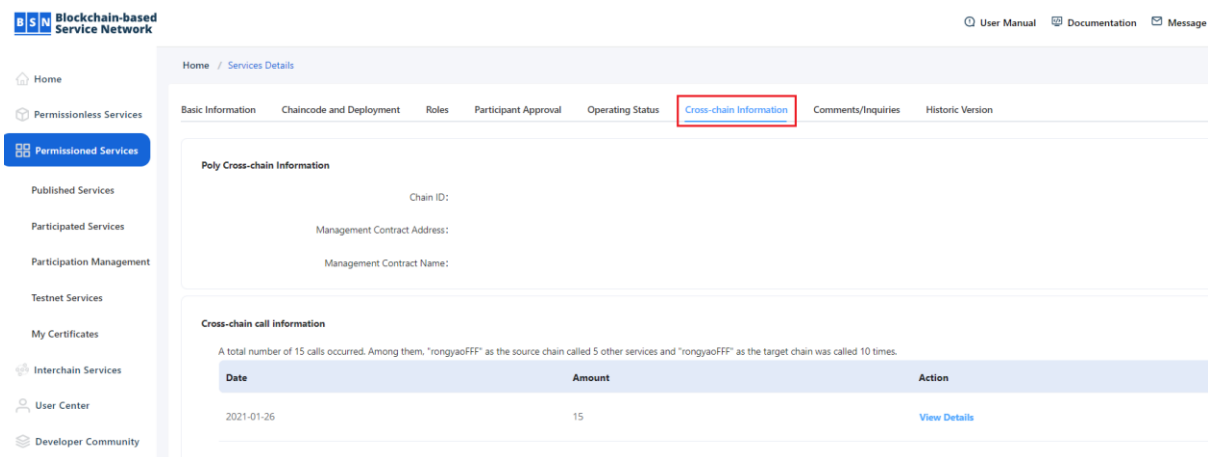
### 8.1.2 View Interchain Services

On the home page, click **Interchain Services**, users can find the service list of their activated interchain services.

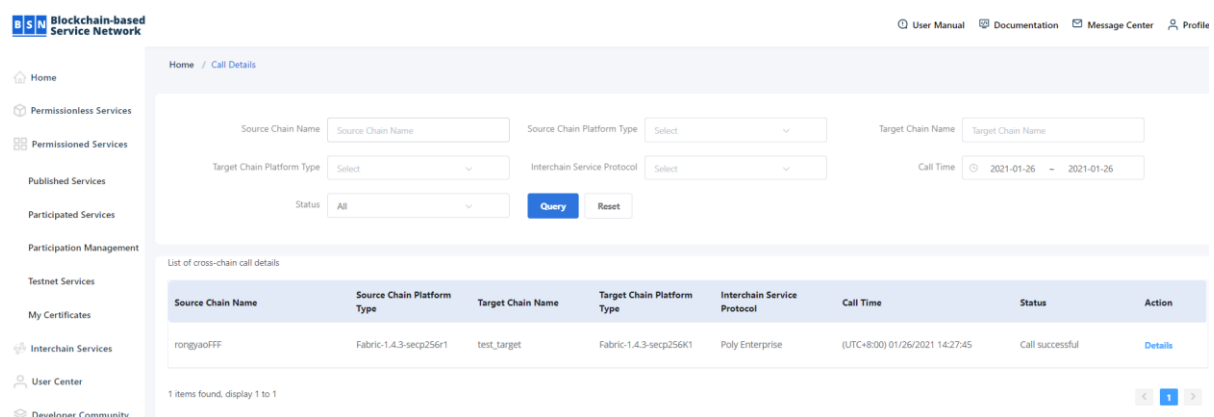




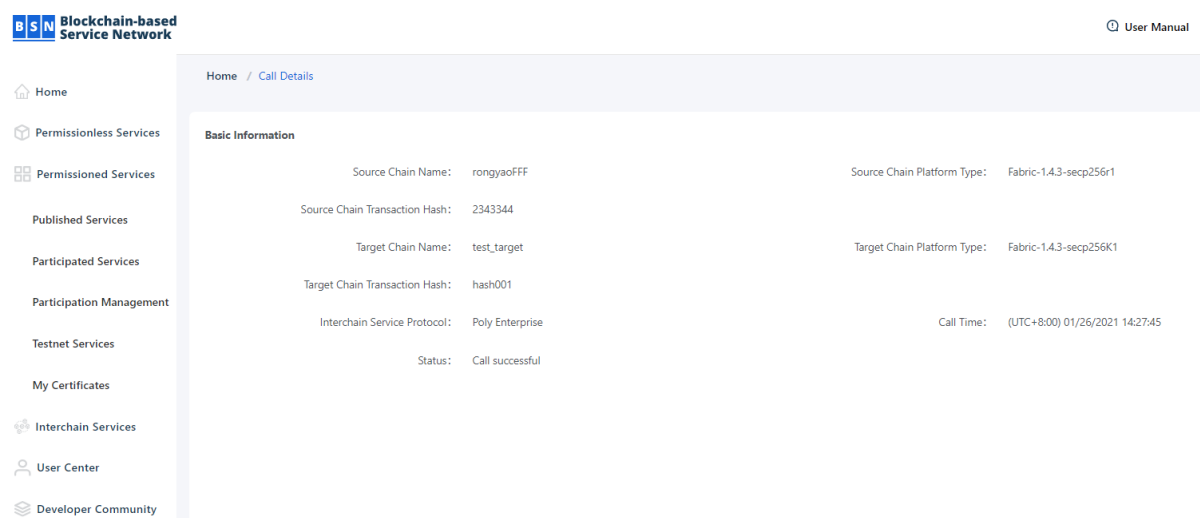
Select the service to be checked, click **View** in the **Action** column, select **Cross-chain Information**, users can check the chain ID, management contract address, management contract name and cross-chain information.



On the Cross-chain Information page, click **Details** button to jump to **Call Details** page. Select the parameter and click **Query** to retrieve the detailed cross-chain call information.



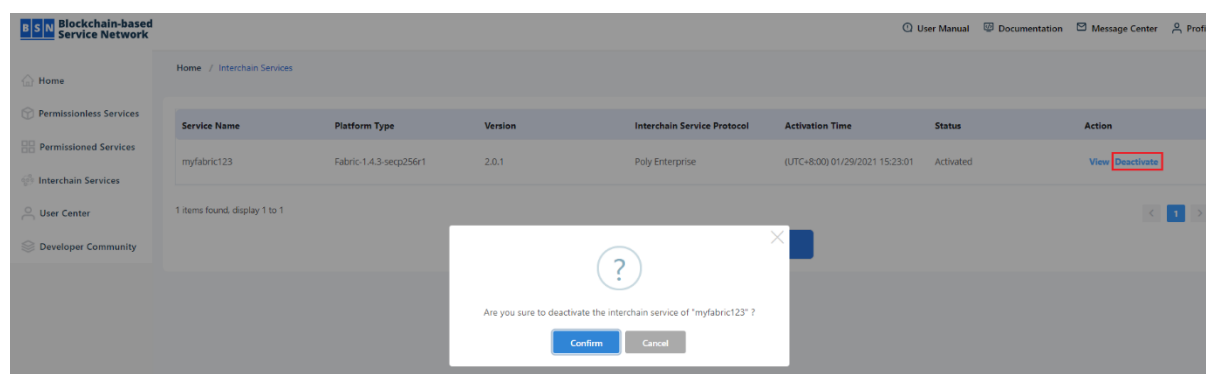
Go to **List of cross-chain call details** section, click **Details** button in **Action** column to enter the Basic Information page, you can view the basic information of the cross-chain call details, as shown in the figure:



### 8.1.3 Deactivation and Activation of Interchain Services

#### 1) Deactivation of Interchain Services

On the home page, click **Interchain Services**, users can see a list of their activated interchain services. Select the service which needs to be deactivated and click **Deactivate** button in **Action** column.

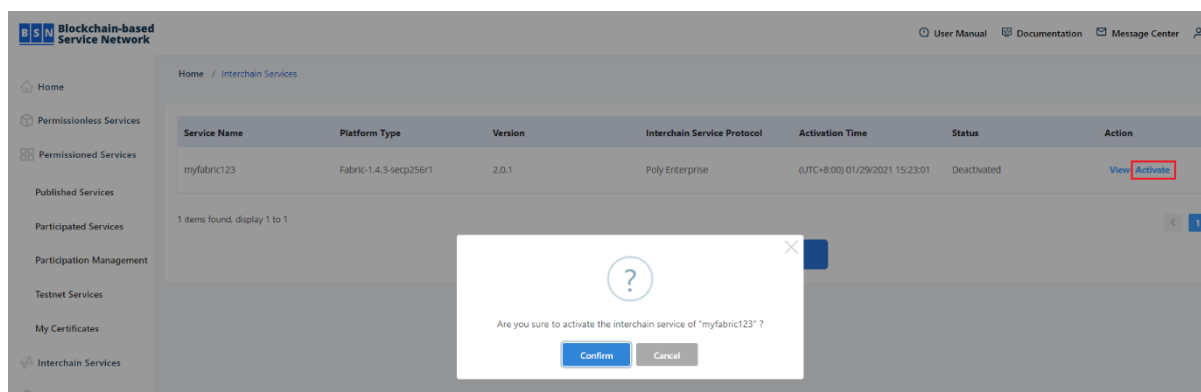


Click **Confirm** in the pop-up message to deactivate the interchain service.

Note: It takes a few minutes to deactivate the interchain service, please be patient.

#### 2) Activation of Interchain Services

On the home page, click **Interchain Services**, users can see a list of their activated interchain services. Select the service which needs to be activated and click **Activate** button in **Action** column.



Click **Confirm** in the pop-up message to activate the interchain service.

## 8.2 Interchain Services based on Poly Enterprise

### 8.2.1 Overview

A complete cross-chain transaction requires application contracts for multiple chains. For example, there is an application contract on the Ethereum Ropsten and a FISCO BCOS application contract on the BSN. These two contracts can interact across chains through the cross-chain protocol to ensure the correctness of the information. The cross-chain contract includes a management contract and an application contract. The management contract implements the core logic of the cross-chain protocol, developed by the BSN development team and is deployed in each chain; the application contract needs to be implemented by blockchain application publishers according to the cross-chain protocol and deployed in the blockchain network.

**Management contracts include the following implementations.**

#### 1. ETH and FISCO BCOS

- EthCrossChainManager: contains logic of management.
- EthCrossChainData: used to save and manipulate data.
- EthCrossChainManagerProxy: used to implement logical contract upgrades.

#### 2. Neo

- CCMC: contains the logic of management.

#### 3. Fabric

- CCMC: contains the logic of management.

#### 4. BSN Testnet Cross-chain management contract address

The following table shows the framework names, chain IDs, and cross-chain contract names or addresses for Poly Enterprise-based cross-chain services.

| Testnet | Framework | Chain ID | Cross-chain contract names/addresses | Application Example Contract |
|---------|-----------|----------|--------------------------------------|------------------------------|
| China   | Fabric    | 88       | ccm                                  | myhellopoly                  |

|               |            |    |  |  |
|---------------|------------|----|--|--|
|               | FISCO BCOS | 98 | 0x8f866dE652d34308De82E7D<br>aF504D1af4B4b05E9 | 0x2e98f68147887288f1eb2eb<br>d065ccc46be9bc4f9 |
| International | Fabric     | 89 | ccm  | myhellopoly                                    |
|               | FISCO BCOS | 99 | 0xaF92fAe702C24CF5B214645<br>AdFE25821b5664667 | 0xd8e0013aa9b41bb946ae1a<br>848b5665c17951200  |
| Ropsten       | Ethereum   | 2  | 0x1a9C1FE6cba599598d7F451<br>50C2FD16F7338e2b0 | 0x7210c828d9455C5319f50d2<br>06C9EdD603CE1F999 |
| Testnet       | Neo        | 4  | 0x10b6edbb6e44188d0ff390654<br>42081b13bbd109b | 0x73090f73056cfc40895799c<br>2a061da7904d8b53d |

### The application cross-chain contains the following functions:

1. Outbound: The source chain's application contract initiates a cross-chain transaction request and transfers this request from the source chain to the target chain. The user can call a self-defined method in the source chain's application contract which calls the 'crossChain' method of the management contract. This will send the cross-chain data through events.
2. Inbound: The target chain application contract receives the cross-chain transaction request. This request information sent from the source chain is passed to the target chain application contract. The cross-chain management contract receives and verifies the cross-chain information. The cross-chain protocol requires the target chain application contract and function name to be included in the cross-chain information. Then the management contract invokes the specified method for the specified contract address and passes the information to the target chain application contract.

## 8.2.2 Interchain Services based on Hyperledger Fabric

### 8.2.2.1 Application Contract Development Guide in BSN production environment

The development of Fabric application contract is based on its own business scenario. The main implementation includes two parts: if the source chain initiates a cross-chain transaction, its application contract needs to get outbound to access the target chain; if the target chain receives a cross-chain transaction, its application contract needs to get inbound. Fabric's chain ID and cross-chain management contract's name are automatically assigned and generated through the BSN operations and maintenance system when users open interchain services, and can be viewed in the BSN portal.

An example of a specific cross-chain transaction call can be found in [7.2.2.3 Demo Contract Example](#).

### 8.2.2.2 Application Contract Development Guide in BSN Testnet

Fabric's chain ID in the BSN China Testnet is 88 and in the BSN International Testnet is 89. This chain ID is registered in Poly Enterprise, not the channel ID corresponding to Fabric itself. The name of Fabric cross-chain contract is ccm.

An example of a specific cross-chain transaction call can be found in [7.2.2.3 Demo Contract Example](#).

### 8.2.2.3 Demo Contract Example

BSN production environment and BSN Testnet:

<https://github.com/BSNDA/ICH/tree/main/sample/polychain/fabric-contract/online/hellopoly>

## 8.2.3 Interchain Services based on FISCO BCOS

### 8.2.3.1 Application Contract Development Guide in BSN production environment

The development of FISCO BCOS application contract is based on its own business scenario. The main implementation includes two parts: if the source chain initiates a cross-chain transaction, its application contract needs to get outbound to access the target chain; if the target chain receives a cross-chain transaction, its application contract needs to get inbound.

An example of a specific cross-chain transaction call can be found in [7.2.3.3 Demo Contract Example](#).

### 8.2.3.2 Application Contract Development Guide in BSN Testnet

FISCO's chain ID in the BSN China Testnet is 98 and in the BSN International Testnet is 99. This chain ID is registered in Poly Enterprise, not the group ID corresponding to FISCO itself.

The application contract example in BSN test network is the same as the production environment, please visit 7.2.3.1 Application Contract Development Guide in BSN Production Environment for details.

An example of a specific cross-chain transaction call can be found in [7.2.3.3 Demo Contract Example](#).

### 8.2.3.3 Demo Contract Example

BSN Production Environment and Testnet:

[https://github.com/BSNDA/ICH/tree/main/sample/polychain/fisco\\_contracts/hellopoly](https://github.com/BSNDA/ICH/tree/main/sample/polychain/fisco_contracts/hellopoly)

## 8.2.4 Interchain Services based on Ethereum Ropsten

### Application Contract Development Guide

The development of the ETH application contract is based on its own business scenario. The main implementation includes two parts: if the source chain initiates a cross-chain transaction, its application contract needs to get outbound to access the target chain; if the target chain receives a cross-chain transaction, its application contract needs to get inbound. ETH's chain ID in the BSN Testnet is 2. This chain ID is registered in Poly Enterprise and the configuration is applicable to both BSN Production Environment and Testnet.

Below is an example of a source chain initiating a cross-chain transaction call:

```
/**
 * @dev: Implements a cross-chain call by invoking the "say" method
 * @param _toChainId: The chain ID corresponding to the target chain being called in the Poly network
```

```

* @param _functionName: The method of the called contract

* @param _somethingWoW: Parameters passed across the chain

* @return bool

**/

function say(uint64 _toChainId, string memory _functionName, string memory
_somethingWoW) public returns (bool){

    // Get the cross-chain management contract interface

    IEthCrossChainManagerProxy eccmp =
    IEthCrossChainManagerProxy(managerProxyContract);

    // Get the cross-chain management contract address

    address eccmAddr = eccmp.getEthCrossChainManager();

    // Get the cross-chain management contract object

    IEthCrossChainManager eccm = IEthCrossChainManager(eccmAddr);

    // Get the target chain application contract address

    bytes memory toProxyHash = proxyHashMap[_toChainId];

    // Call across the chain

    require(eccm.crossChain(_toChainId, toProxyHash, bytes(_functionName),
bytes(_somethingWoW)), "EthCrossChainManager crossChain executed error!");

    emit Say(_toChainId, toProxyHash, bytes(_somethingWoW)); return true;
}

```

Below is an example of a target chain call when receiving a cross-chain transaction:

```

/**

* @param _somethingWoW: Parameters passed across the chain

* @param _fromContractAddr: The address of the application contract being invoked

* @param _toChainId: Contract framework chainId being called

```

```

* @return bool

**/

function hear(bytes _somethingWoW, bytes _fromContractAddr, uint64 _toChainId) public
returns (bool){

    hearSomething = _somethingWoW;

    emit Hear(_somethingWoW, _fromContractAddr);

    return true;

}

```

### Demo Contract Example

GitHub: [https://github.com/BSNDA/ICH/tree/main/sample/polychain/eth\\_contracts/hellopoly](https://github.com/BSNDA/ICH/tree/main/sample/polychain/eth_contracts/hellopoly)

## 8.2.5 Interchain Services based on Neo Testnet

### Application Contract Development Guide

The development of Neo application contract is based on its own business scenario. The main implementation includes two parts: if the source chain initiates a cross-chain transaction, its application contract needs to get outbound to access the target chain; if the target chain receives a cross-chain transaction, its application contract needs to get inbound. Neo's chain ID in the BSN Testnet is 4. This chain ID is registered in Poly Enterprise the configuration is applicable to both BSN Production Environment and Testnet.

Below is an example of a source chain initiating a cross-chain transaction call:

```

/// <summary>

    /// This method is used to make cross-chain calls to other target chains (this method is
    self-defining)

    /// </summary>

    /// <param name="toChainId">The chain ID of the target chain in the Poly
    network</param>

    /// <param name="msg">The cross-chain information that the target chain needs to
    pass to apply the contract</param>

    /// <returns></returns>

    [DisplayName("say")]

    public static bool Say(BigInteger toChainId, string method ,byte[] msg)

```

```

{
    // Get the application contract on the target chain
    var toProxyHash = HelloPoly.GetProxyHash(toChainId);

    // Get the CCMC contract address
    var ccmcScriptHash = HelloPoly.GetProxyHash(neoChainID);

    // Call across the chains
    bool success = (bool)((DynCall)ccmcScriptHash.ToDelegate())("CrossChain", new
object[] { toChainId, toProxyHash, method, msg });

    HelloPoly.Notify(success, "[HelloPoly]-Say: Failed to call CCMC.");

    // Event notification
    HelloPoly.SayEvent(toChainId, toProxyHash);

    return true;
}

```

Below is an example of a target chain call when receiving a cross-chain transaction:

```

/// <summary>
    /// This method is used to make cross-chain calls to other target chains (this method is
self-defining)
    /// </summary>

    /// <param name="fromChainId">The chain ID of the source chain in a Poly
network</param>

    /// <param name="toChainId">The chain ID of the target chain in the Poly
network</param>

    /// <param name="msg">Receive a cross-chain message sent by the source
chain</param>

    /// <param name="callingScriptHash">Callback script hash</param>

    /// <returns></returns>

    [DisplayName("hear")]
    public static bool Hear(byte[] inputBytes, byte[] fromProxyContract, BigInteger
fromChainId, byte[] callingScriptHash)

    //commit into ledger

```



```

Storage.Put(fromProxyContract, inputBytes);

// Event notification

HearEvent(fromChainId, fromProxyContract, inputBytes);

return true;

}

```

### Demo Contract Example

GitHub: <https://github.com/BSNDA/ICH/tree/main/sample/polychain/neo-contract>

## 8.3 Interchain Services based on IRITA

### 8.3.1 Overview

BSN interchain services based on IRITA is a cross-chain network and is a part of the ICH. In the interchain communication process, developers initiate an interchain service call by their application contracts, and after receiving this call, the relay of that application chain will initiate a cross-chain request to the ICH. After receiving the request, the service provider transfers the request to the target chain, acquires the transaction result and returns it to the ICH. Finally, the relay returns the transaction result to the application chain by calling the cross-chain contract.

The application contract, cross-chain contract, relay, ICH, service provider and services are working together to complete the entire process of interchain services.

| Testnet       | Framework  | Cross-chain contract names/addresses        |
|---------------|------------|---|
| China         | Fabric     | cc_cross                                    |
|               | FISCO BCOS | 0x836d5cc8f65a1e2e6a9224f36060aa2606f0ae58  |
| International | Fabric     | cc_cross                                    |
|               | FISCO BCOS | 0x0be396b885eea99171ebdfefb5214d3582ae2eb7a |

### 8.3.2 Interchain Architecture based on IRITA

#### 1. Cross-chain Contract

##### a. Service Core

As the core of the entire cross-chain contract, Service Core is responsible for receiving a cross-chain request from the application contract, the request result from the relay, and return the cross-chain call result, including the following functions:

- **Call Service:** Function to make a cross-chain request. It is invoked by the application contract by passing the interchain service name, input parameters, callback contract info, etc. A successful invocation will return a unique request ID.
- **Set Response:** Return the cross-chain invocation result called by relay and write the result

to the application chain. If an application contract method needs to be called back during service invocation, it needs to be called in this function.

- **Get Response:** Query the cross-chain invocation result by request ID.

## **b. Target Core**

TargetCore is the entrance of the target contract on the target chain, responsible for receiving cross-chain requests from the cross-chain to the target chain, redirecting requests to the target contract, and generating events containing the response message after getting the response, including the following functions:

- **Call Service:** Call the function of the target contract, called by the provider, pass in the requestID of the cross-chain request, target contract address, call parameters, and return the call result after successful call.

## **2. Application Contract**

Application contract is developed by the developer for the interchain services. In an application contract, the developer can invoke the call service function of the cross-chain contract to make a cross chain call. A callback function in the application contract also needs to be provided to get the call result, otherwise developers will have to call the Get Response function in an application contract to query the service invocation results.

## **3. Relayer**

Relayer is a service that listens for the cross-chain request submitted to ICH by a source chain. It is responsible for listening the call service function event, and submitting to the ICH.

## **4. Service Provider**

Service provider, connects the HUB to the destination chain. It is a service that listens for the cross-chain request from the ICH, invokes the destination chain, and returns the transaction results to the ICH.

## **5. Interchain Service**

The interchain service is developed by the service provider. It could be the smart contract deployed on one blockchain for an application on another blockchain to call across the chain. Support from the service provider is required to turn a service into an interchain service.

## **6. Interchain Communications Hub**

As the core component of BSN's interchain service, the ICH is responsible for receiving cross chain requests submitted by the source chain Relayer, verifying the transaction and initiating the cross-chain transaction with the Service Provider of the destination chain. After the transaction is complete, it is also responsible for obtaining the transaction result from the Service Provider, verifying and returning the result to the source chain Relayer.

### **8.3.3 Interchain Services in BSN Testnet**

In the BSN Testnet, we provide cross-chain calls between Fabric, FISCO BCOS and ETH (Ropsten) , as well as enable the function of calling Chainlink services.

Before calling the service, users need to develop an application contract or use the sample contract provided by the BSN to invoke it. The following are the parameters of the interchain service deployed in the Testnet, according to which developers can initiate cross-chain calls.

#### **ChainID of the Testnet**

| Testnet       | Framework  | Chain ID          |
|---------------|------------|-------------------|
| China         | Fabric     | 10227431719070001 |
|               | FISCO BCOS | 10227431719070002 |
|               | ETH        | 20227431719070003 |
|               | ChainLink  | 10227431719070003 |
| International | Fabric     | 20227431719070001 |
|               | FISCO BCOS | 20227431719070002 |
|               | ETH        | 20227431719070003 |
|               | ChainLink  | 10227431719070003 |

### Cross-chain contract

| Testnet       | Framework  | Cross-chain management contract name/address | Cross-chain proxy contract                 | Target chain proxy contract                 |
|---------------|------------|--|--|---|
| China         | Fabric     | cc_cross                                     | —  | cc_target                                   |
|               | FISCO BCOS | 0x2ce23b61ea3d6797b246ca0f60b3def089787152   | 0x2a3edc16a9525e3ae038509493adeb51257d388d | 0x8f070608d8ea72fa7eef426656bb12ff79f1fed2  |
|               | ETH        | 0x7182cBfEA08d00Bf11689cfeA0cceE9b4290cFA0   | —  | 0x7CA61d368B9b088FD5669C411E1eBF44cbEE8B9A  |
| International | Fabric     | cc_cross                                     | —  | cc_target                                   |
|               | FISCO BCOS | 0x81d1194138fb53a7d8e295c7ae21dc7e21a5f551   | 0x830ec4a69c662a59b4a3df021355be01f4ffe2ed | 0x15a4daab593214a277f32f82093eb9bb9efbfcbb0 |
|               | ETH        | 0x7182cBfEA08d00Bf11689cfeA0cceE9b4290cFA0   | —  | 0x7CA61d368B9b088FD5669C411E1eBF44cbEE8B9A  |

### 8.3.3.1 Interchain Application Contract based on Fabric

#### Interchain Application Contracts Development Guide

##### 1. Preparation:

Run the code below to obtain the BSN interchain consuming contract help package (ICH.git). Currently, only the GO language version is supported but more versions will be added later.

```
cd $GOPATH
```

```
mkdir -p src/github.com/BSNDA && cd src/github.com/BSNDA
```

```
git clone https://github.com/BSNDA/ICH.git
```

##### 2. Initiate the interchain service request

After creating the Fabric chain code struct and invoke function, import below package:

```
import (  
    "github.com/BSNDA/ICH/sample/irita/consumers/fabric/crosschaincode"  
)
```

3. Call the `crosschaincode.CallService` method in the `invoke` function using the parameters of the method as follows:

- `stub`: `shim.ChaincodeStubInterface`
- `serviceName`: the interchain service name to invoke, `nft` for `ETH`, and `bcos-store` for `FISCO BCOS`
- `input`: the input object for interchain service
- `callbackCC`: callback chaincode name
- `callbackFcn`: callback chaincode function name
- `timeout`: timeout

The input parameter varies according to the interchain service and type passed in. In the `ETH` service, the input structure is as follows:

```
type Input struct {  
    ABIEncoded string `json:"abi_encoded,omitempty"`  
    To          string `json:"to"`  
    AmountToMint string `json:"amount_to_mint"`  
    MetaID      string `json:"meta_id"`  
    SetPrice    string `json:"set_price"`  
    IsForSale   bool  `json:"is_for_sale"`  
}
```

In the `FISCO BCOS` service, the input structure is as follows:

```
type BcosInput struct {  
    Value string `json:"value"`  
}
```

A unique request ID will be returned after successful invocation. Keep this value and use it to determine the cross-chain results in the callback function.

4. Implement the callback interface:

After the cross chaincode receives the service response from Fabric Relayer, the callback method name and callback chaincode name passed in will be called to return the service response. The first parameter of the call returns a JSON-formatted string. Below is the service response structure:

```
type ServiceResponse struct {  
    RequestId string `json:"requestID,omitempty"`  
    ErrMsg    string `json:"errMsg,omitempty"`  
    Output    string `json:"output,omitempty"`  
    IcRequestId string `json:"icRequestID,omitempty"`  
}
```

Method `crosschaincode.GetCallBackInfo()` could be called to Serialize the value. The `requestID` is unique and can be used to conduct the business processing. Output a JSON-

formatted string which is the return value of the crosschain response. Below is the input data structure:

```
type InputData struct {  
    Header interface{} `json:"header"`  
    Body interface{}    `json:"body"`  
}
```

Parameter “Body” is the output object for the service. In ETH service, the input structure is as follows:

```
type Output struct {  
    NftID string `json:"nft_id"`  
}
```

In FISCO BCOS service, the output structure is as follows:

```
type BcosOutput struct {  
    Key string `json:"key"`  
}
```

## 5. Package the chaincode

ICH.git, which is imported by the chaincode, needs to be packaged with the chaincode together by govendor. If you haven't installed govendor, you can install it as below:

Install govendor:

```
go get -u -v github.com/kardianos/govendor
```

Execute in the main method directory:

```
govendor init
```

```
govendor add -tree github.com/BSNDA/ICH/sample/irita/consumers/fabric/crosschaincode
```

After execution, the vendor folder will be generated. For the last step, compress the project and vendor folder together, then upload and deploy it in BSN portal.

## Application Contract Example

GitHub: <https://github.com/BSNDA/ICH/tree/main/sample/irita/consumers/fabric/chaincode>

### 8.3.3.2 Interchain Application Contract based on FISCO BCOS

The interchain consuming contracts based on iService with Solidity are applicable for the EVM compatible application blockchain platform like Ethereum and FISCO BCOS.

## Application Contracts Development Guide

IService Client: For convenience, the contract named iService Client is built to encapsulate the interaction with the iService Consumer Proxy and handle logistics including event triggering, request validation and status maintaining, which helps improve development efficiency.

The iService Client source code can be found in the [Example Contract](#).

### 1. Import iService Client

To use the iService Client, import the corresponding contract path. For example:

```
import ServiceClient.sol
```

*Note: You can directly use the iService Client code as a part of the consuming contract.*

## 2. Inherit iService Client

```
contract <consuming-contract-name> is iServiceClient {  
  
}
```

## 3. Initiate the interchain service request

- **Set the iService Consumer Proxy (i.e. iService Core Extension) contract address.**

The contract address in BSN Testnet is

**0xcdad7e31edb24fc5f7c1aaf9edb8b8640d2fe3ca**

This can be performed by invoking the `setIServiceConsumerProxy` method inherited from the `iService Client`. A constructor taking the `iService Consumer Proxy` address can be used. E.g.

```
constructor(  
    address _IServiceConsumerProxy  
)  
public  
{  
    setIServiceConsumerProxy(_IServiceConsumer);  
}
```

- **Implement the callback interface**

When the `iService Consumer Proxy` receives the service response, the method implementing the callback interface will be called to paginate the response to the corresponding consuming contract.

The below is the callback interface signature.

```
function callback(  
    bytes32 _requestID,  
    string calldata _output  
)
```

- **Initiate iService invocation**

The `iService` request can be sent by the `sendIServiceRequest` function in the `iService Client`.

```
bytes32 memory requestID = sendIServiceRequest(  
    serviceName,  
    requestInput,  
    timeout,  
    address(this),  
    this.callback.selector  
);
```

*Note: Developers need to retrieve information related to the service from the iService Market Ex deployed on the application chain for the interchain service invocation, such as the service name and schemas of the input and output.*

#### 4. NFT Service Consuming Contract Example

The NFT service is provided by the NFT contract on the Ethereum Ropsten to create NFT assets.

The definition of the NFT service is as follows:

- Service name: nft
- Service Input JSON Schema:

```
{
  "type": "object",
  "properties": {
    "to": {
      "description": "address to which the NFT will be minted",
      "type": "string"
    },
    "amount_to_mint": {
      "description": "amount of the NFT to be minted",
      "type": "string"
    },
    "meta_id": {
      "description": "meta id",
      "type": "string"
    },
    "set_price": {
      "description": "price in Ethereum Wei",
      "type": "string"
    },
    "is_for_sale": {
      "description": "whether or not the minted NFT is for sale",
      "type": "boolean"
    }
  }
}
```

- Service Output JSON Schema:

```
{
  "type": "object",
  "properties": {
    "nft_id": {
      "description": "id of the minted NFT",

```

```
        "type": "string"
    }
}
}
```

Developers can develop contracts on application chains to implement to mint NFT assets across chains.

### Application Contract Example

GitHub:

<https://github.com/BSNDA/ICH/tree/main/sample/irita/consumers/fiscobcos/NFTServiceConsumer>

## 8.3.4 Interchain Services based on Hyperledger Fabric

### Application Contract Development Guide

#### 1. Preparation:

Run the code below to obtain the BSN interchain consuming contract help package (ICH.git). Currently, only the GO language version is supported but more versions will be added later.

```
cd $GOPATH
mkdir -p src/github.com/BSNDA && cd src/github.com/BSNDA
git clone https://github.com/BSNDA/ICH.git
```

#### 2. Initiate the interchain service request

After creating the Fabric chain code struct and invoke function, import below package:

```
import (
    "github.com/BSNDA/ICH/sample/irita/consumers/fabric/crosschaincode"
)
```

Call the crosschaincode.CallService method in the invoke function using the parameters of the method as follows:

- stub: shim.ChaincodeStubInterface
- serviceName: the interchain service name of permissioned chain is “cross\_service”
- input: the input object for interchain service
- callbackCC: callback chaincode name
- callbackFcn: callback chaincode function name
- timeout: timeout

The input parameter varies according to the interchain service and type passed in. In the Fabric service, the input structure is as follows:

```
type Input struct {
    ChainId    uint64 `json:"chainId"`
    ChainCode  string `json:"chainCode"`
    FunType    string `json:"funType"`
    Args       []string `json:"args"`
}
```

In the FISCO BCOS service, the input structure is as follows:



```
type BcosInput struct {  
    OptType    string `json:"optType"`  
    ChainID    uint64 `json:"chainId"`  
    ContractAddress string `json:"contractAddress"`  
    CallData    string `json:"callData"`  
}
```

A unique request ID will be returned after successful invocation. Keep this value and use it to determine the cross-chain results in the callback function.

### 3. Implement the callback interface:

After the cross-chain contract receives the service response from Fabric Relayer, the callback method name and callback chaincode name passed in will be called to return the service response. The first parameter of the call returns a JSON-formatted string. Below is the service response structure:

```
type ServiceResponse struct {  
    RequestId string `json:"requestID,omitempty"`  
    ErrMsg    string `json:"errMsg,omitempty"`  
    Output    string `json:"output,omitempty"`  
    IcRequestId string `json:"icRequestID,omitempty"`  
}
```

Method `crosschaincode.GetCallBackInfo()` could be called to serialize the value. The `requestID` is unique and can be used to conduct the business processing. Output is a JSON-formatted string which is the returned value of the cross-chain response. Below is the input data structure:

```
type InputData struct {  
    Header interface{} `json:"header"`  
    Body interface{} `json:"body"`  
}
```

Parameter “Body” is the output object for the service. In Fabric service, the input structure is as follows:

```
type Output struct {  
    TxValidationCode int32 `json:"txValidationCode"`  
    ChaincodeStatus int32 `json:"chaincodeStatus"`  
    TxId            string `json:"txId"`  
    Payload         string `json:"payload"`  
}
```

In FISCO BCOS service, the output structure is as follows:

```
type BcosOutput struct {  
    Result string `json:"result,omitempty"`  
    Status bool  `json:"status,omitempty"`  
    TxHash string `json:"tx_hash,omitempty"`  
}
```

```
}
```

#### 4. Pack the chaincode

ICH.git, which is imported by the chaincode, needs to be packaged with the chaincode together by govendor. If you haven't installed govendor, you can install it as below:

Install govendor:

```
go get -u -v github.com/kardianos/govendor
```

Execute in the main method directory:

```
govendor init
```

```
govendor add -tree github.com/BSNDA/ICH/sample/irita/consumers/fabric/crosschaincode
```

After the execution, the vendor folder will be generated. For the last step, compress the project and vendor folder together, then upload and deploy it in the BSN portal.

#### Application Contract Example

GitHub:

<https://github.com/BSNDA/ICH/tree/bsnirita/sample/irita/consumers/fabric/chaincode>

### 8.3.5 Interchain Services based on FISCO BCOS

#### Application Contract Development Guide

iService Client: For convenience, the contract named iService Client is built to encapsulate the interaction with the iService Consumer Proxy and handle logistics including event triggering, request validation and status maintaining, which helps improve development efficiency.

The iService Client source code can be found in the [Example Contract](#).

#### 1. Import iService Client

To use the iService Client, import the corresponding contract path. For example:

```
import ServiceClient.sol
```

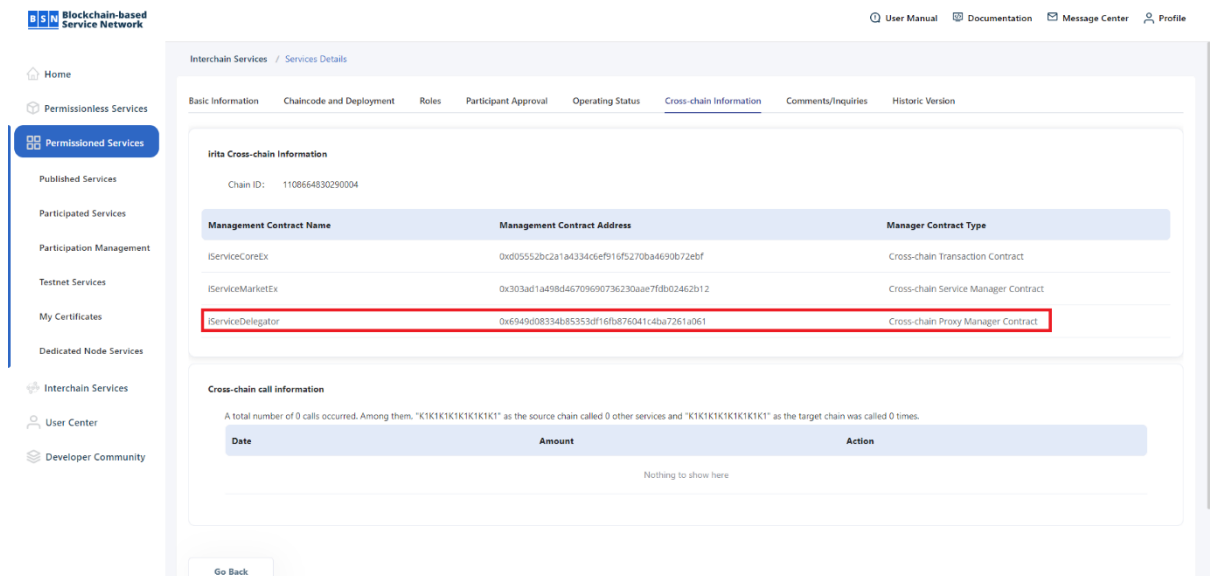
*Note: You can directly use the iService Client code as a part of the consuming contract.*

#### 2. Inherit iService Client

```
contract <consuming-contract-name> is iServiceClient {  
  
}
```

#### 3. Initiate the interchain service request

- Set up the iService Consumer Proxy, that is, after the successful deployment of the cross-chain contract in the portal, find the iServiceDelegator cross-chain proxy contract address in the cross-chain information page.



This is done by calling `setIServiceConsumerProxy(address _iServiceConsumerProxy)`, a method inherited from `iService Client`, or by passing in the contract constructor. E.g.

```
constructor(
    address _iServiceConsumerProxy
)
public
{
    setIServiceConsumerProxy(_iServiceConsumer);
}
```

- Implement the callback interface

When the `iService Consumer Proxy` receives the service response, the method implementing the callback interface will be called to paginate the response to the corresponding consuming contract.

Below is the callback interface:

```
function callback(
    bytes32 _requestID,
    string calldata _output
)
```

- Initiate `iService` invocation

The `iService` request can be sent by the `sendIServiceRequest` function in the `iService Client`.

```
bytes32 memory requestID = sendIServiceRequest(
    serviceName,
    requestInput,
```

```
    timeout,  
  
    address(this),  
  
    this.callback.selector  
  
);
```

*Note: Developers need to retrieve information related to the service from the iService Market Ex deployed on the application chain for the interchain service invocation, such as the service name and schemas of the input and output.*

#### 4. Fabric cross-chain call example

- Service name: cross\_service
- Service Input JSON Schema:

```
{  
  ChainId    uint64  `json:"chainId"`  
  ChainCode  string  `json:"chainCode"`  
  FunType    string  `json:"funType"`  
  Args       []string `json:"args"`  
  
}
```

- Service Output JSON Schema:

```
{  
  TxValidationCode int32 `json:"txValidationCode"`  
  ChaincodeStatus  int32 `json:"chaincodeStatus"`  
  TxId             string `json:"txId"`  
  Payload          string `json:"payload"`  
  
}
```

#### 5. FISCO BCOS cross-chain call example

The definition of FISCO BCOS service is as follows:

- Service name: cross\_service
- Service Input JSON Schema:

```
{  
  OptType    string `json:"optType"`  
  ChainID    uint64 `json:"chainId"`  
  ContractAddress string `json:"contractAddress"`  
  CallData    string `json:"callData"`  
  
}
```

- Service Output JSON Schema:

```
{  
  Result string `json:"result,omitempty"`  
  Status bool  `json:"status,omitempty"`  
}
```

```
TxHash string `json:"tx_hash,omitempty"`  
}
```

### **Application Contract Example**

GitHub: <https://github.com/BSNDA/ICH/tree/bsn-irita/sample/irita/consumers/fiscobcos>

## 9 Oracle Services

Blockchain is a deterministic and closed environment. Transactions generated by smart contracts in the blockchain also need deterministic results, so the host virtual machine (VM) of smart contracts is prohibited to include uncertain external calls when building transactions. This processing mechanism leads to the inability to actively access external data within the blockchain. To solve this problem, we integrate the oracle service. The oracle service is a bridge between the blockchain and the real world, connecting the blockchain to the real world by feeding real-world data to the blockchain and providing smart contracts with connectivity to the external world.

The main functions of BSNs services include: management mechanism of oracle management contract, oracle application contract function, and oracle off-BSN service.

BSN oracle services will integrate centralized and decentralized mechanisms for blockchain to acquire off-chain data, and developers can choose different forms of oracle services to meet their demands for off-chain data access.

At present, BSN oracle services have been launched on the BSN Testnet, which integrates the Chainlink oracle service supported by IRITA Hub built by IRISnet's core development team Bianjie. We will continue to improve the functions and integrate more oracle services.

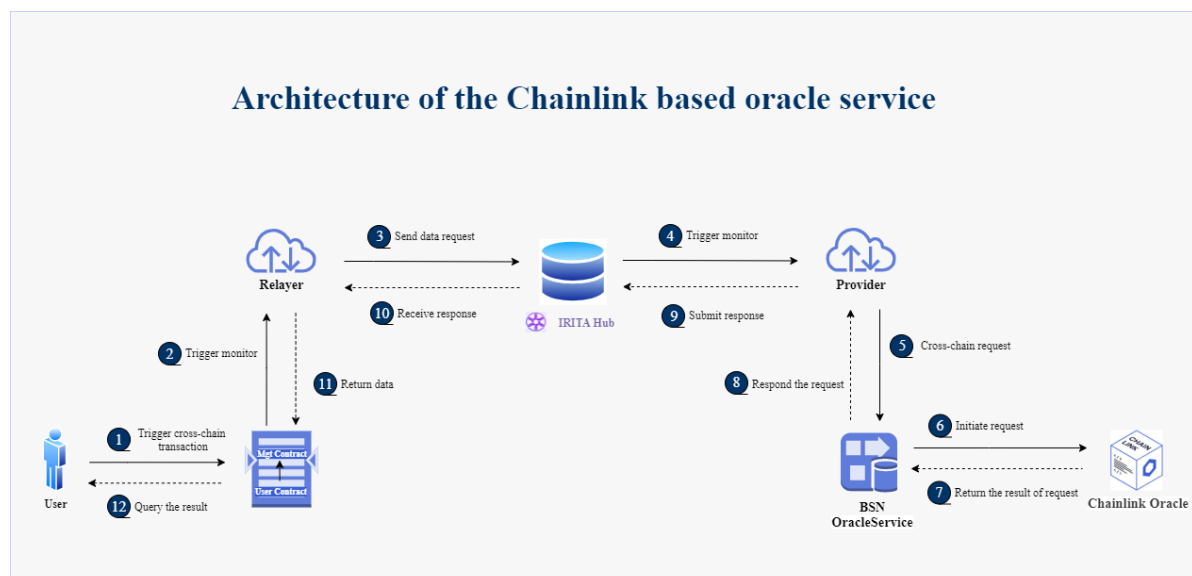
### 9.1 Oracle Service based on Chainlink

#### 9.1.1 Overview

Chainlink is a decentralized oracle network. This network is an integrated platform that provides developers with tools and data to help them develop various types of smart contracts on any platform. Chainlink is both an oracle and a flexible framework that provides developers with a secure and reliable oracle solution.

In the BSN Testnet, users can use the IRITA-based cross-chain architecture to invoke the Chainlink oracle services, and the services currently available for trial in the Testnet is `eth_usdt_price` service.

| Testnet                 | Framework | Chain ID          |
|-------------------------|-----------|-------------------|
| BSN China/International | IRITA     | 10227431719070003 |



The business process is as follows:

- (1) Appchain in the BSN triggers cross-chain transaction request, and the user contract invokes management contract.
- (2) Relayer triggers monitor event.
- (3) Relayer sends the data request to IRITA Hub in the Interchain Communications Hub.
- (4) Provider triggers monitor event.
- (5) Provider sends cross-chain request to BSN Oracle Service.
- (6) BSN Oracle Service initiates a request to Chainlink Oracle.
- (7) Chainlink Oracle returns the request result to BSN Oracle Service.
- (8) Provider receives the response to the cross-chain request from the corresponding chain/service.
- (9) Provider submits the received response to IRITA Hub.
- (10) Relayer receives the response from IRITA Hub.
- (11) Data is sent back to the BSN appchain that originally initiated the request.
- (12) The user queries the result of the request.

### 9.1.2 Invocation of the Chainlink oracle service in the BSN Testnet

Due to the nature of IRITA cross-chain model, the user initiates a call to the Chainlink service by invoking the cross-chain contract in the BSN permissioned services by the contract deployed in the Testnet. After processing by Relayer, BSN Hub and other components, the deployed Chainlink service provider issues a call to Chainlink after receiving an event from a cross-chain contract and returns the result of the call via the BSN Hub.

In the BSN, calling the Chainlink service through the IRITA interchain hub is just like calling the cross-chain service, the `CallService` method is called in the user's application contract.

Currently, the Testnet provides `eth_usdt_price_test` service. The service call is the same as the cross-chain call, please refer to chapter 8.3 "[Interchain Services based on IRITA](#)".

When calling the Chainlink service, the target contract address is:

```
iaa15s9sulrnmctzluc42g7lkxh92ardkc9xccxsy9
```

The parameters are as follows:

```
{ "service_name": "eth-usdt-price", "providers": ["iaa16eu2jvgpa5ek9mn2tn985jlm89e6dch92qwj09"], "input": "{ \"header\": {}, \"body\": {} }", "timeout": 100, "service_fee_cap": "1000000upoint" }
```

The format of the returned parameters is as follows:

```
{ "last": "1000" }
```

Note: Since the name of the called Chainlink oracle service is already identified in the parameters of the cross-chain service call, the input parameter to call the Chainlink service is empty.

The specific example contracts can be found in the chapter 8.3.4 "Interchain Services based on Hyperledger Fabric" and chapter 8.3.5 "Interchain Services based on FISCO BCOS" for the description of Fabric and the FISCO BCOS framework for invoking cross-chain contracts.

### 9.1.3 Example of calling BSN IRITA Chainlink service across chains

1. write and deploy the application contract to the `appchain`

- `endpointInfo` is a fixed-format json string, where `endpoint_type` is written as `service`.
- `callDataJsonStr` is also a fixed-format json string, which is the call parameter of the chainlink service.

```
//SPDX-License-Identifier: SimPL-2.0

pragma solidity ^0.6.0;

/**
 * @title iService interface
 */
interface iServiceInterface {
    /**
     * @dev Send cross chain request
     * @param _endpointInfo information of endpoint
     * @param _method Target method name
     */
}
```



```

* @param _callData Target method callData
* @param _callbackAddress Callback contract address
* @param _callbackFunction Callback function selector
* @return requestID Request id
*/

function sendRequest(
string calldata _endpointInfo,
string calldata _method,
bytes calldata _callData,
address _callbackAddress,
bytes4 _callbackFunction
) external returns (bytes32 requestID);
/**
* @dev Set the response of the specified service request
* @param _requestID Request id
* @param _errMsg Error message of the service invocation
* @param _output Response output
* @return True on success, false otherwise
*/

function setResponse(
bytes32 _requestID,
string calldata _errMsg,
string calldata _output
) external returns (bool);
}
/*
* @title Contract for the iService core extension client
*/

contract iServiceClient {
iServiceInterface iServiceCore; // iService Core contract address

// mapping the request id to Request
mapping(bytes32 => Request) requests;

```

```
// request

struct Request {

    address callbackAddress; // callback contract address

    bytes4 callbackFunction; // callback function selector

    bool sent; // request sent

    bool responded; // request responded

}

/*

 * @dev Event triggered when the iService request is sent

 * @param _requestID Request id

 */

event RequestSent(bytes32 _requestID);

/*

 * @dev Make sure that the given request is valid

 * @param _requestID Request id

 */

modifier validRequest(bytes32 _requestID) {

    require(

        requests[_requestID].sent,

        "iServiceClient: request does not exist"

    );

    require(

        !requests[_requestID].responded,

        "iServiceClient: request has been responded"

    );

    _;

}

/**

 * @dev Send cross chain request

 * @param _endpointInfo information of endpoint

 * @param _method Target method name

 * @param _callData Target method callData
```

```
* @param _callbackAddress Callback contract address

* @param _callbackFunction Callback function selector

* @return requestID Request id

*/

function sendServiceRequest(

string memory _endpointInfo,

string memory _method,

bytes memory _callData,

address _callbackAddress,

bytes4 _callbackFunction

) internal returns (bytes32 requestID) {

requestID = iServiceCore.sendRequest(

_endpointInfo,

_method,

_callData,

address(this),

this.onResponse.selector

);

Request memory request = Request(

_callbackAddress,

_callbackFunction,

true,

false

);

requests[requestID] = request;

emit RequestSent(requestID);

return requestID;

}

/*

* @dev Callback function

* @param _requestID Request id

* @param _output Response output
```

```

*/

function onResponse(bytes32 _requestID, string calldata _output)

external

validRequest(_requestID)

{

address cbAddr = requests[_requestID].callbackAddress;

bytes4 cbFunc = requests[_requestID].callbackFunction;

(bool success, ) = cbAddr.call(

abi.encodeWithSelector(cbFunc, _requestID, _output)

);

if (!success) revert();

}

/**

* @dev Set the iService core contract address

* @param _iServiceCore Address of the iService core contract

*/

function setIServiceCore(address _iServiceCore) internal {

require(

_iServiceCore != address(0),

"IServiceClient: iService core address can not be zero"

);

iServiceCore = iServiceInterface(_iServiceCore);

}

}

/*

* @title Contract for inter-chain NFT minting powered by iService

* The service is supported by price service

*/

contract ServiceConsumer is IServiceClient {

string private endpointInfo =

'{"dest_chain_id":"10227431719070003","dest_chain_type":"opb","endpoint_address":"iaa15s9sulnm

ctzluc42g7lkxh92ardkc9xccsy9","endpoint_type":"service"}';

```

```
event Hello(bytes32 _requestID, string _helloMsg);

string public result;

/*
 * @notice Constructor
 * @param _iServiceContract Address of the iService contract
 * @param _defaultTimeout Default service request timeout
 */

constructor(address _iServiceCore) public {
    setIServiceCore(_iServiceCore);
}

/*
 * @notice Start workflow for minting nft
 * @param _method method name
 * @param _hello arguments
 */

function helloWorld() external {
    string
    memory callDataJsonStr = '{"service_name":"eth-usdt-
    price","providers":["16eu2jvgpa5ek9mn2tn985jlm89e6dch92qwj09"],"input":'
    '{"header":{"body":
    {}},'',"timeout":100,"service_fee_cap":"1000000upoint"}';

    bytes memory callData;

    callData = bytes(callDataJsonStr);

    sendServiceRequest(
        endpointInfo,
        "service",
        callData,
        address(this),
        this.onHello.selector
    );
}

/*
```

```

* @notice NFT service callback function

* @param _requestID Request id

* @param _output NFT service response output

*/

function onHello(bytes32 _requestID, string calldata _output)

external

validRequest(_requestID)

{

requests[_requestID].responded = true;

result = _output;

}

}

```

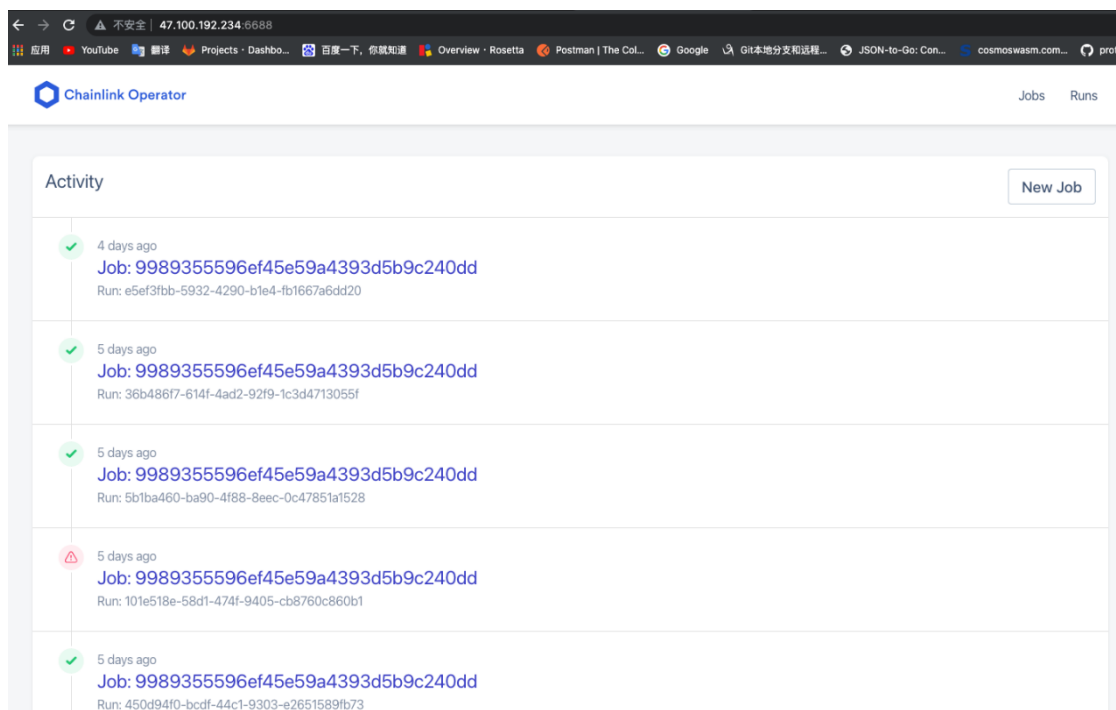
## 2. Service call

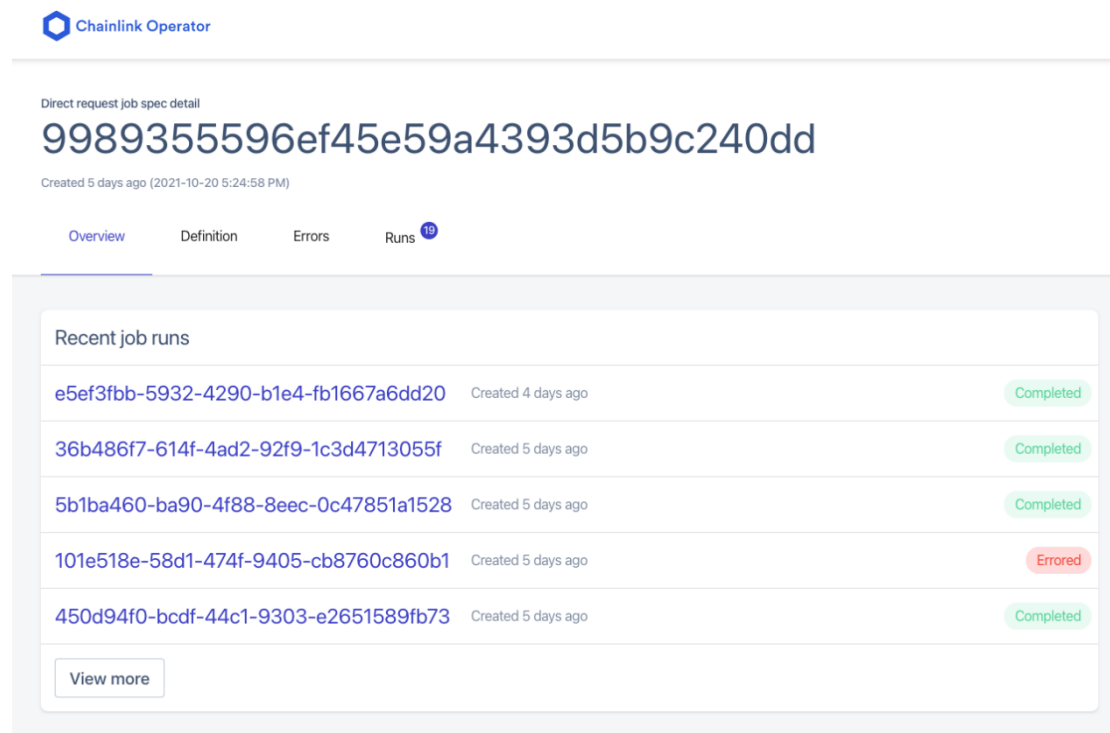
Execute the helloworld method in the application contract.

## 3. View the result

Open your browser and login to the Chainlink Operator Dashboard to view the result of the oracle service invocation in the interface.

- Link: <http://47.100.192.234:6688/>
- Account number: test@gmail.com
- Password: chainlinkPASSWORD123



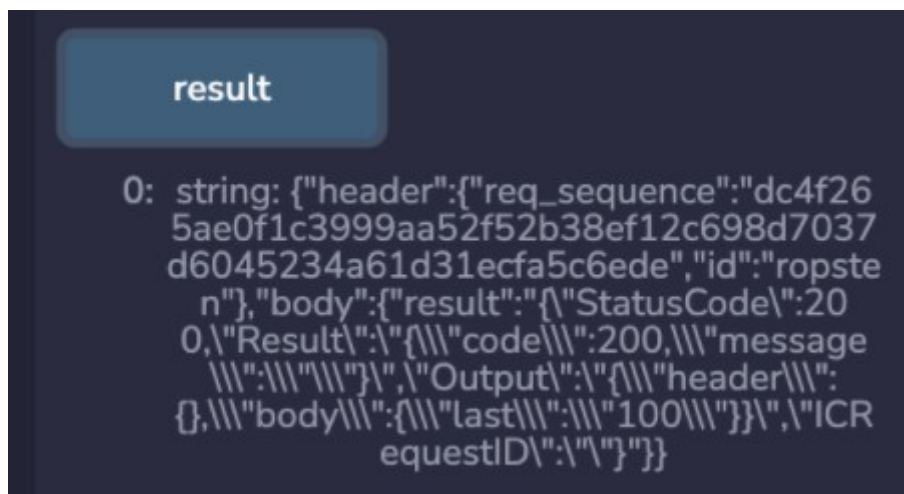


The screenshot shows the Chainlink Operator interface. At the top, the Chainlink Operator logo is visible. Below it, the text "Direct request job spec detail" is displayed. The main heading is a long alphanumeric string: "9989355596ef45e59a4393d5b9c240dd". Below this, it says "Created 5 days ago (2021-10-20 5:24:58 PM)". There are four tabs: "Overview", "Definition", "Errors", and "Runs" (which is active and has a badge with the number 19). Below the tabs is a section titled "Recent job runs" containing a table of job runs.

| Job ID                               | Created            | Status    |
|--------------------------------------|--------------------|-----------|
| e5ef3fbb-5932-4290-b1e4-fb1667a6dd20 | Created 4 days ago | Completed |
| 36b486f7-614f-4ad2-92f9-1c3d4713055f | Created 5 days ago | Completed |
| 5b1ba460-ba90-4f88-8eec-0c47851a1528 | Created 5 days ago | Completed |
| 101e518e-58d1-474f-9405-cb8760c860b1 | Created 5 days ago | Errored   |
| 450d94f0-bcdf-44c1-9303-e2651589fb73 | Created 5 days ago | Completed |

Below the table is a "View more" button.

The result returned by the cross-chain call can be viewed through the result method of the application contract.



The screenshot shows a terminal window with a dark background. A blue box with the word "result" is at the top. Below it, the output of a cross-chain call is displayed as a JSON string.

```
0: string: {"header":{"req_sequence":"dc4f265ae0f1c3999aa52f52b38ef12c698d7037d6045234a61d31ecfa5c6ede","id":"ropste n"},"body":{"result":{"StatusCode\\":200,\\\"Result\\\":{\\\"code\\\":200,\\\"message\\\":\\\"\\\"\\\"}\\\"},\\\"Output\\\":{\\\"header\\\":{\\\"body\\\":{\\\"last\\\":\\\"100\\\"}}\\\"},\\\"ICR requestID\\\":\\\"\\\"}}}
```

## 10 IDE Services

### 10.1 Overview

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. It enhances the development experience and efficiency by integrating tools such as compiler, editor, interpreter, debugger, project manager, etc.

BSN IDE Services customize and integrate the third-party IDEs corresponding to the frameworks adapted in BSN to form a development tool suite. When users create, publish and upgrade DApp services in the BSN portal, if the framework used in the current DApp service has already integrated with the IDE, they can create and modify smart contracts by calling the IDE website with one click from the DApp service page. After finishing programming, developers can debug the code and deploy the smart contract in the BSN production environment or BSN Testnet. Developers do not need to install their own development tools and set up debugging environment, all edited smart contract packages can be synchronized and saved in the BSN portal and IDE.

Currently, the BSN International portal provides IDE services for permissioned frameworks including Hyperledger Fabric and FISCO BCOS, as well as public chains such as Ethereum,.

This iteration of IDE services is only available on the web. In the future, BSN will continue to integrate more frameworks and launch the BSN IDE client version.

### 10.2 Access Instructions

Users can access to the IDE from BSN portal, or directly log in to the IDE web page with BSN username and password.

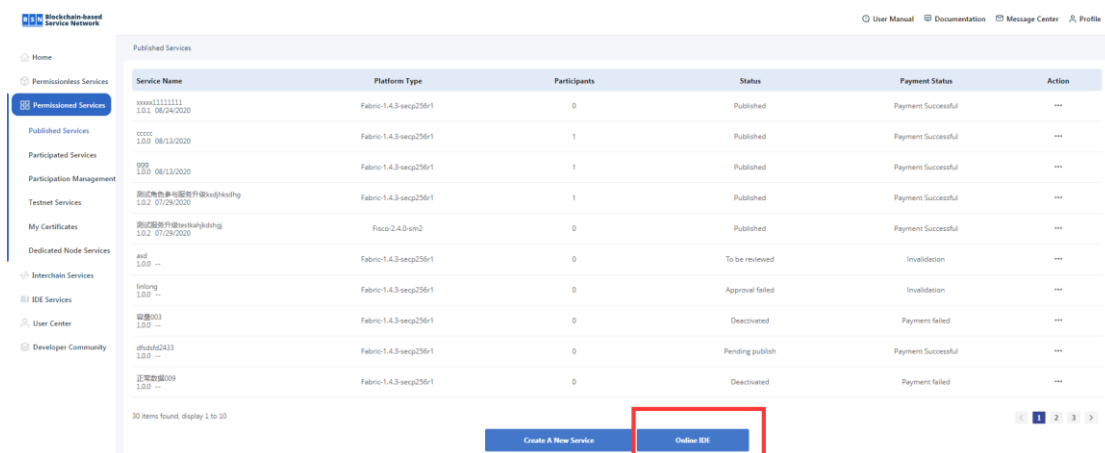
If users jump from BSN portal to the IDE, the chaincode package will be automatically synchronized, and after IDE finishes the operation of chaincode package and jumps back to the portal, the chaincode package will be automatically synchronized to the corresponding service in the portal.

BSN's IDE is mainly applicable to the following services: service publication of permissioned chains, service editing and upgrading of permissioned chains, permissionless services and BSN testnet services.

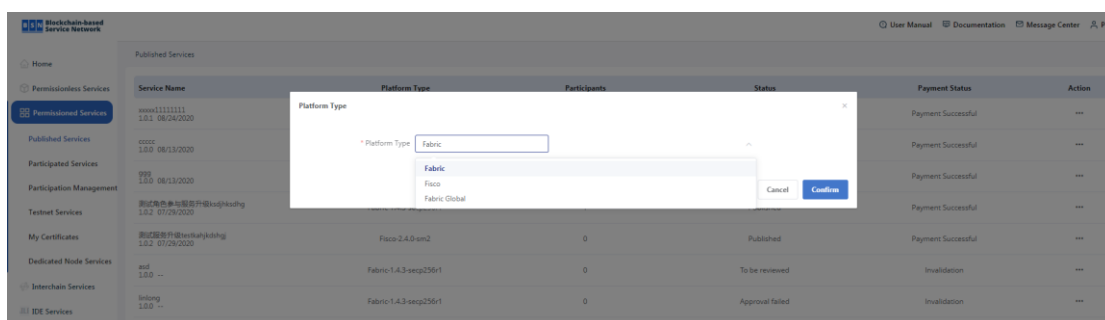
#### 10.2.1 Service publication of permissioned chains

1. Log in to BSN portal, go to **【Permissioned Services】** -> **【Published Services】** , select “Online IDE”;





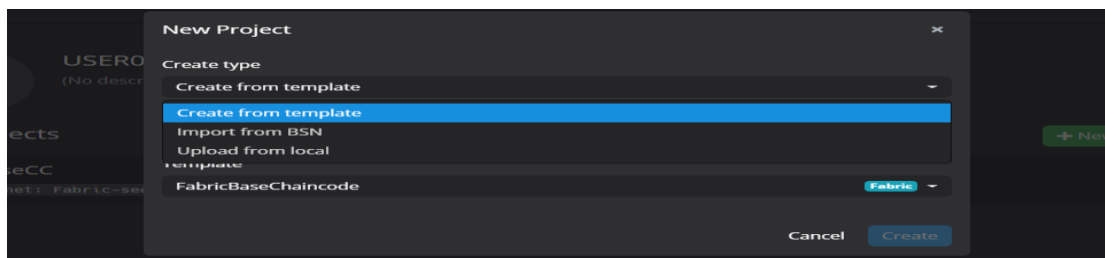
- Choose the platform type, click “Confirm” button to jump to the IDE web page;



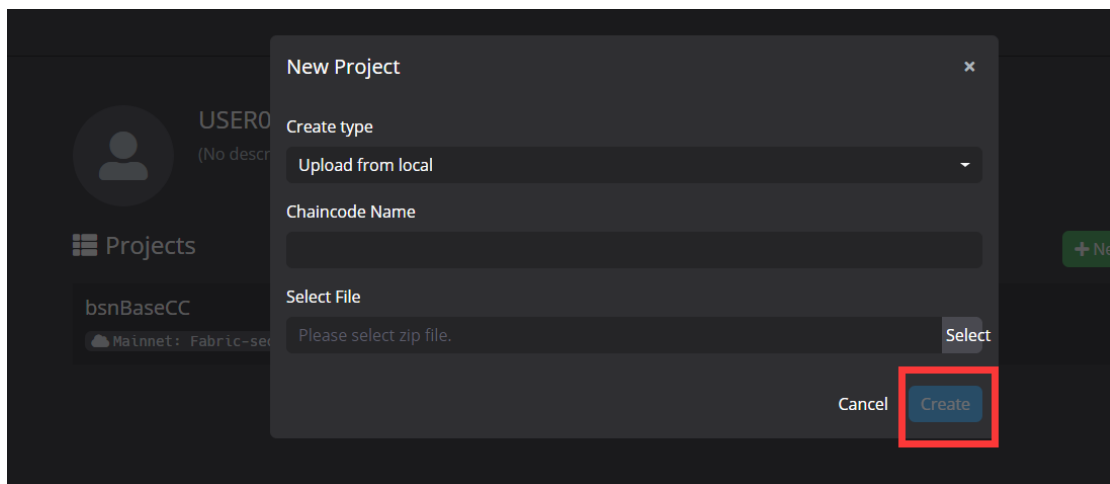
- Create, edit and deploy the chaincode package in the IDE, and select “Create a new service”;
  - Create chaincode package:  
Go to IDE, and click “New Project” button



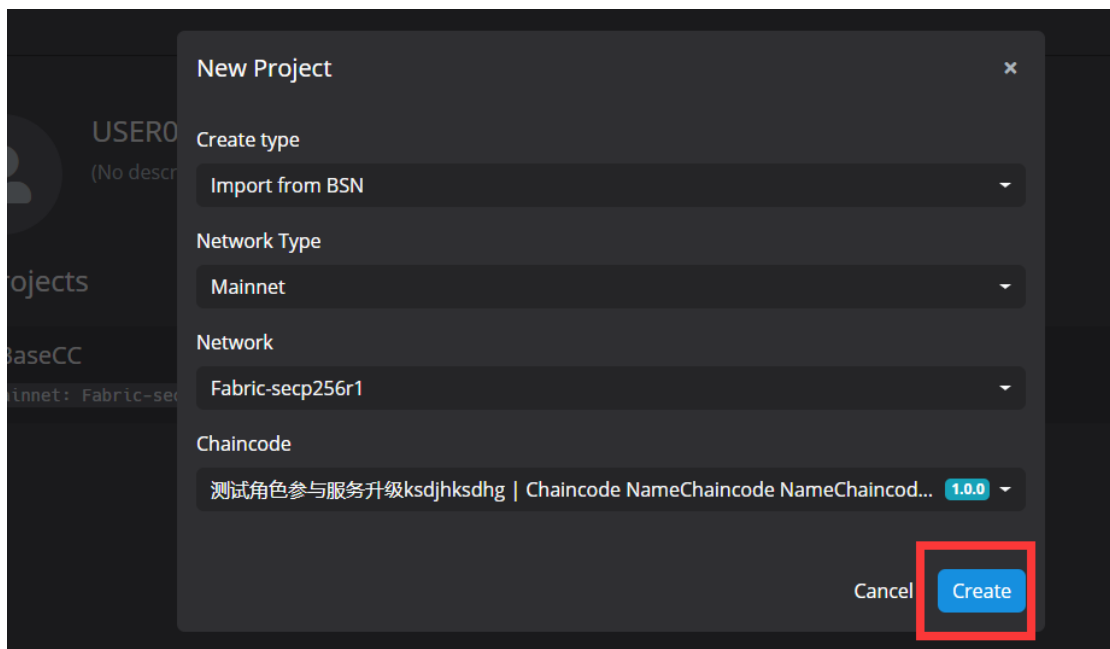
When creating the chaincode package, the IDE supports to import the chaincode package from BSN portal, or developer can create from template or upload from local disk drive.



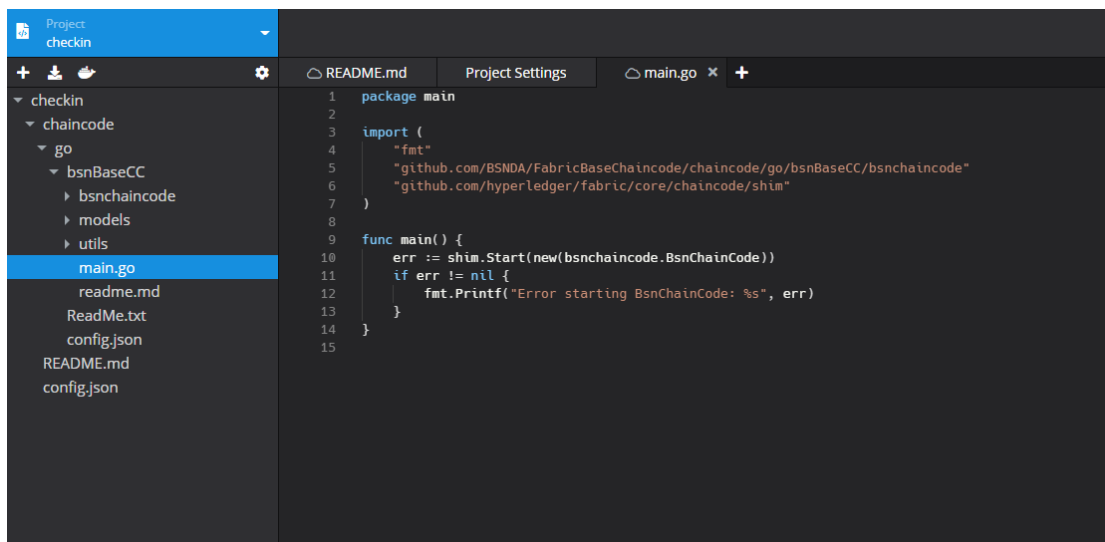
If “Create Type” is selected as “Upload from local”, developer should input chaincode name, upload the chaincode package, and click “Create” button. Note that the file in the chaincode package cannot contain Chinese characters.



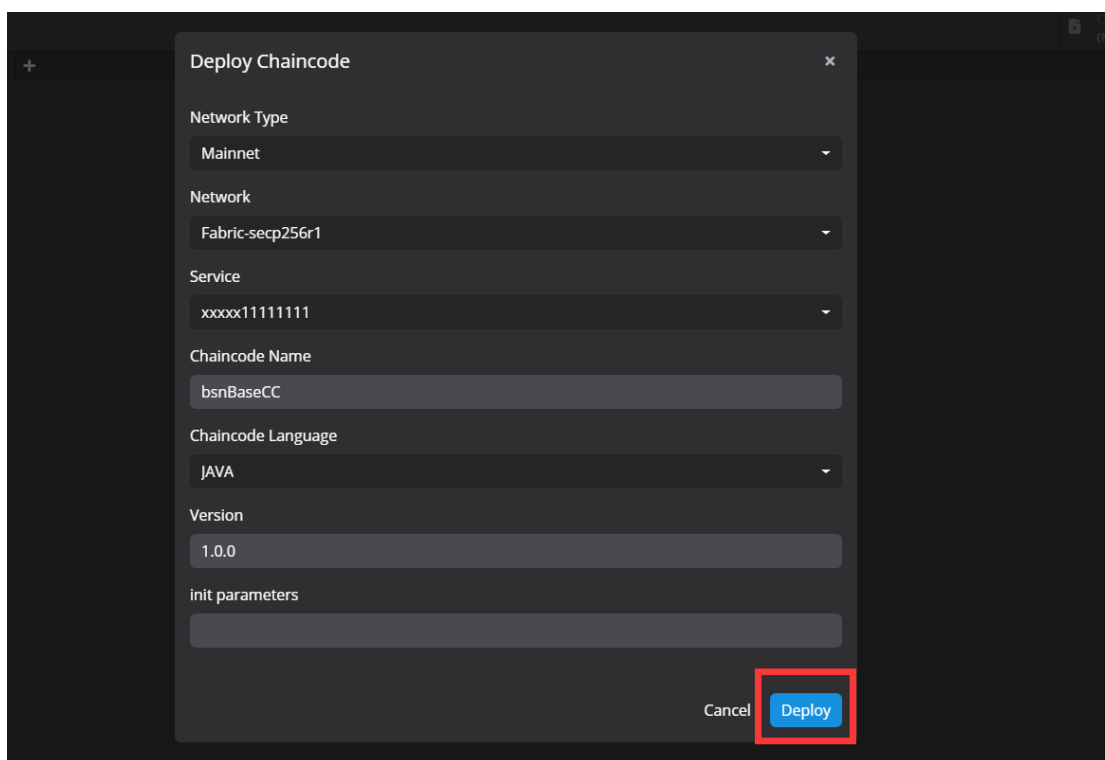
If “Create Type” is selected as “Import from BSN”, developer should select the network type, framework and chaincode package, and then click “Create” button to finish creating the project.



- Edit chaincode package:  
Click and expand the chaincode package in the IDE, and edit the chaincode in the editing page.



- Deploy chaincode package:  
On the editing page, click on “Deploy” button and go to the “Deploy Chaincode” page. Complete the form and click on “Deploy” button to deploy the chaincode package. Note that the chaincode name cannot contain Chinese characters.

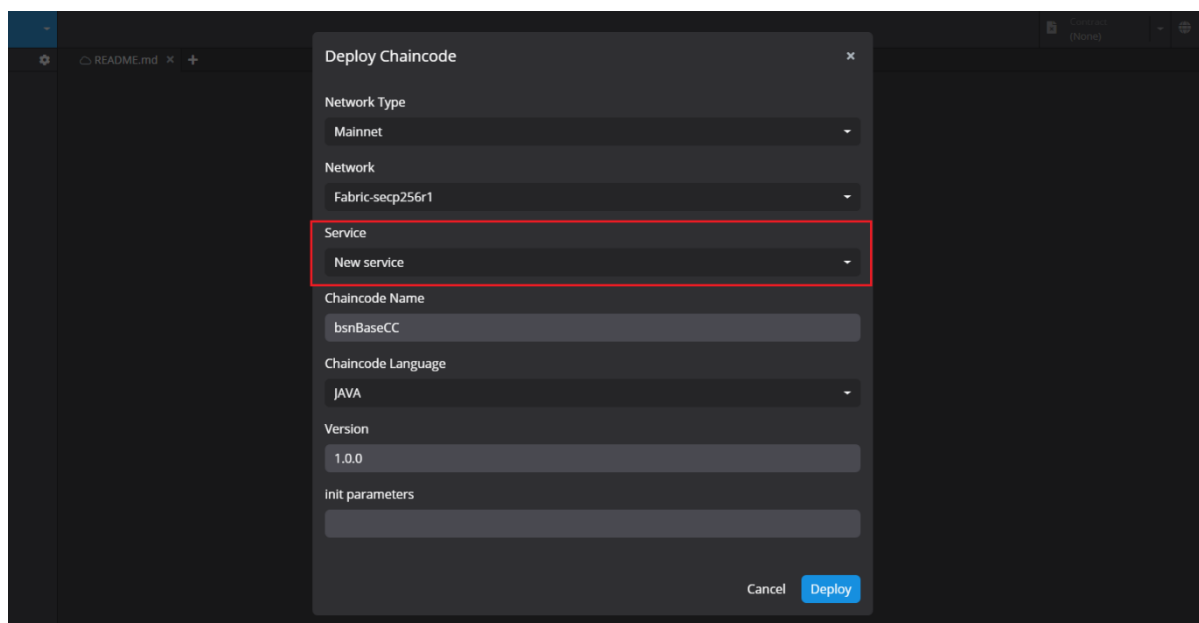


- Create a new service  
On the chaincode deployment page, select “New service” in the Service part, it will navigate to BSN portal, Create a New Service page.

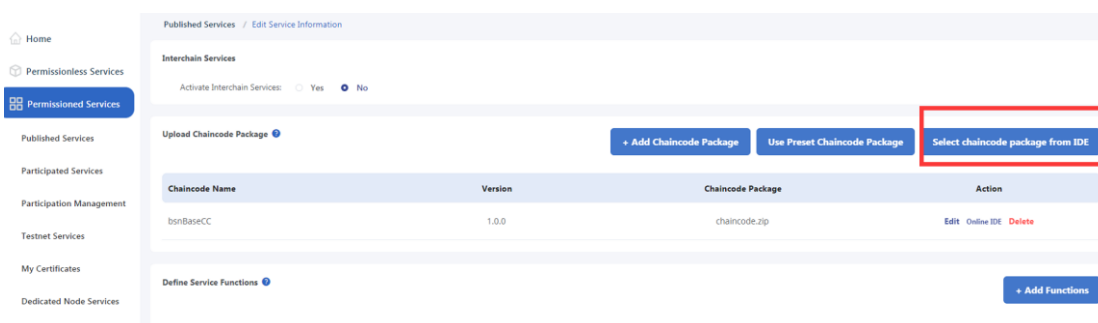




3. Edit and deploy the chaincode package in the IDE;  
The steps of editing and deploying the chaincode is the same as they are in 9.2.1.
4. Select the service to edit or upgrade;  
On “Chaincode deployment” page, select the service which needs to be upgraded, and jump back to the BSN portal Service Upgrade page.



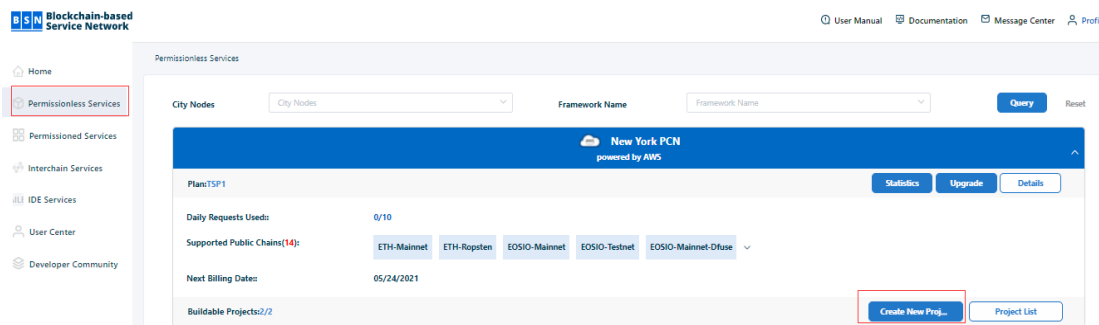
5. Navigate to the BSN portal and continue the following service upgrade process. In the “Upload chaincode package” section, click on “Select chaincode package from IDE” button to select the chaincode package from the IDE and replace the current one.



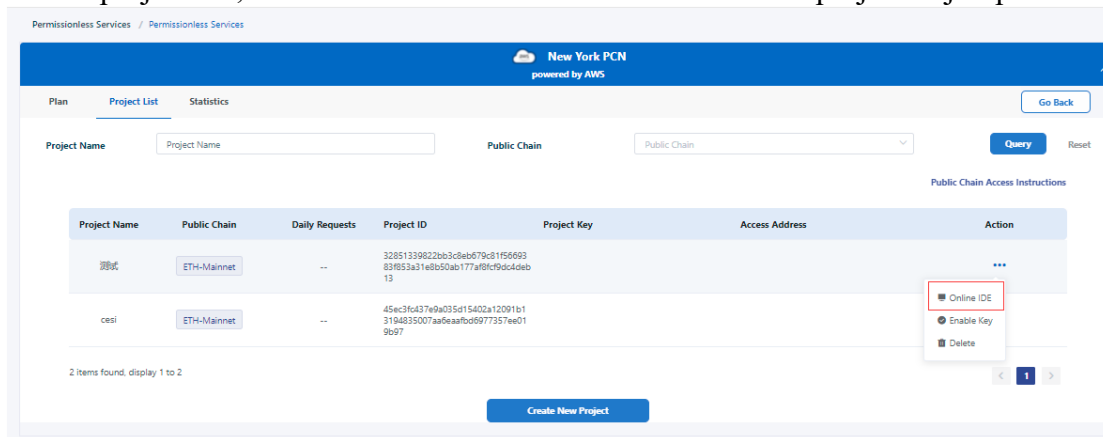
### 10.2.3 Access to permissionless services

Developers can access to the IDE to create, edit and deploy the chaincode package after creating the project in the BSN permissionless services.

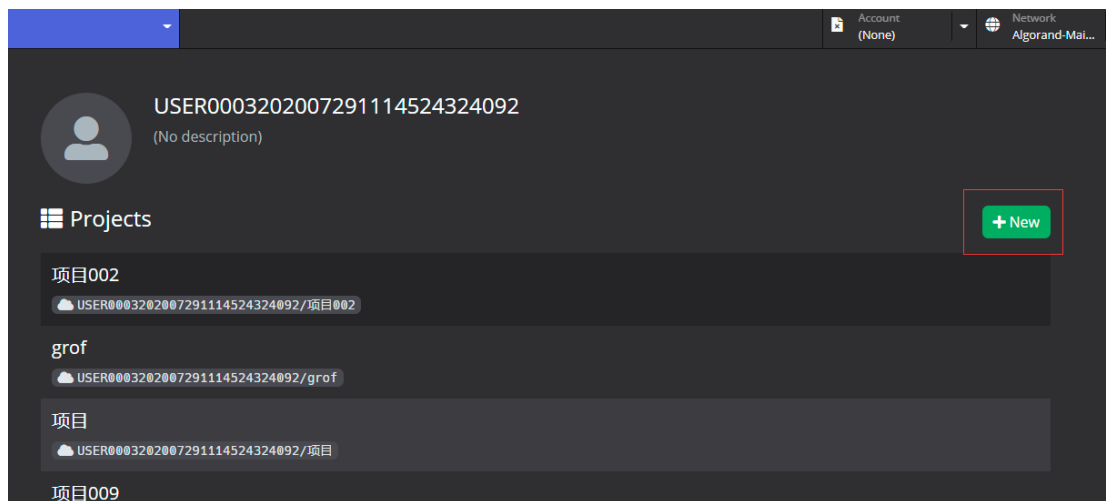
1. Log in to the BSN portal, “Permissionless Services”, select the public city node and buy a plan and create a project (for example on Ethereum-Mainnet);



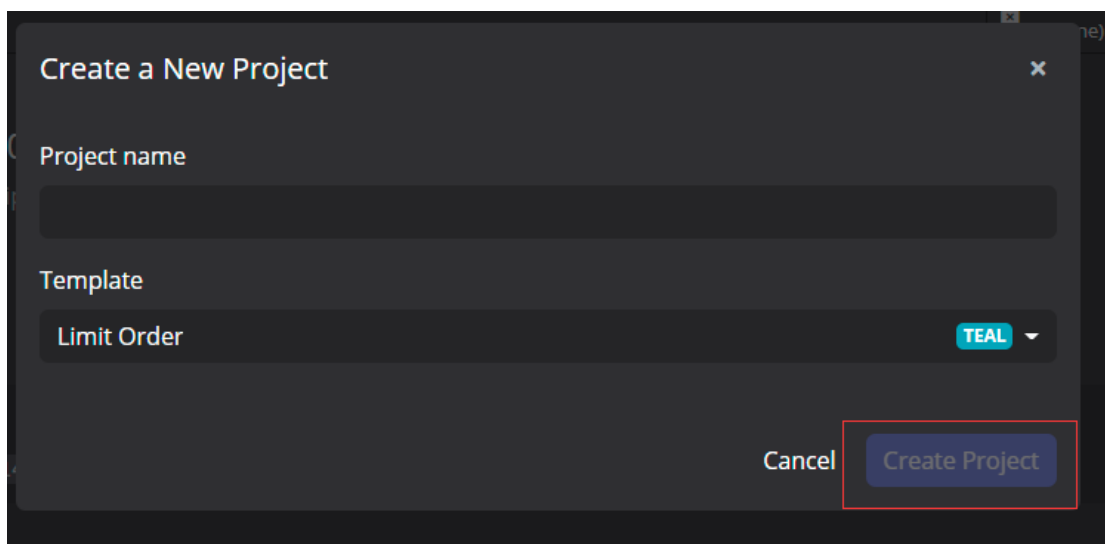
- On the project list, click on “Online IDE” button in the created project to jump to the IDE;



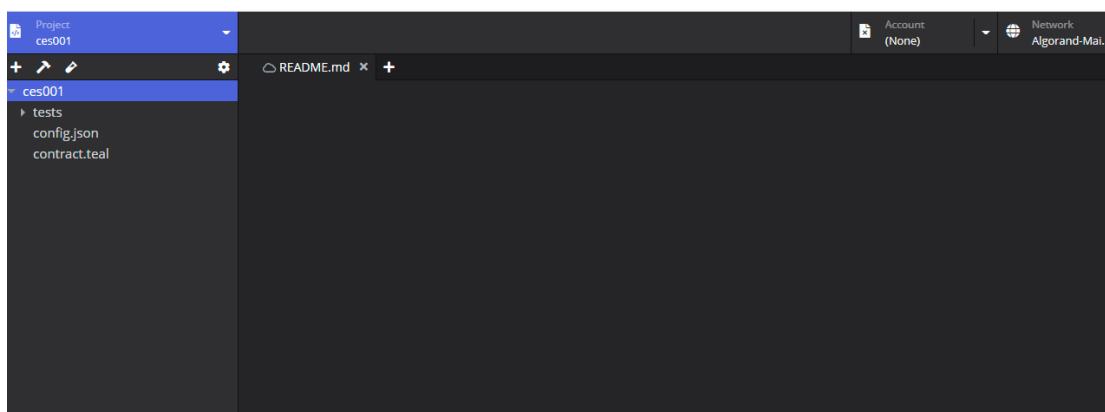
- Create, edit, test and deploy the chaincode package in the IDE;
  - Chaincode package creation:  
Go to IDE, and click “New” button.



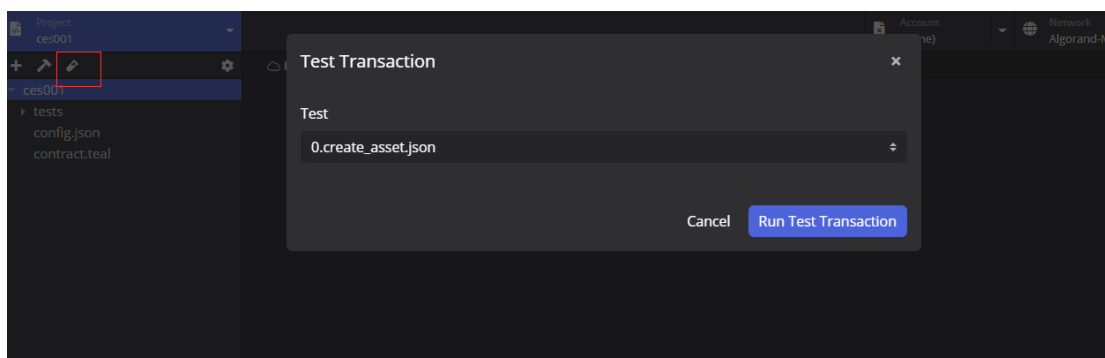
On “Create a New Project” page, input the project name, select the template and click “Create Project” button to create the chaincode package.



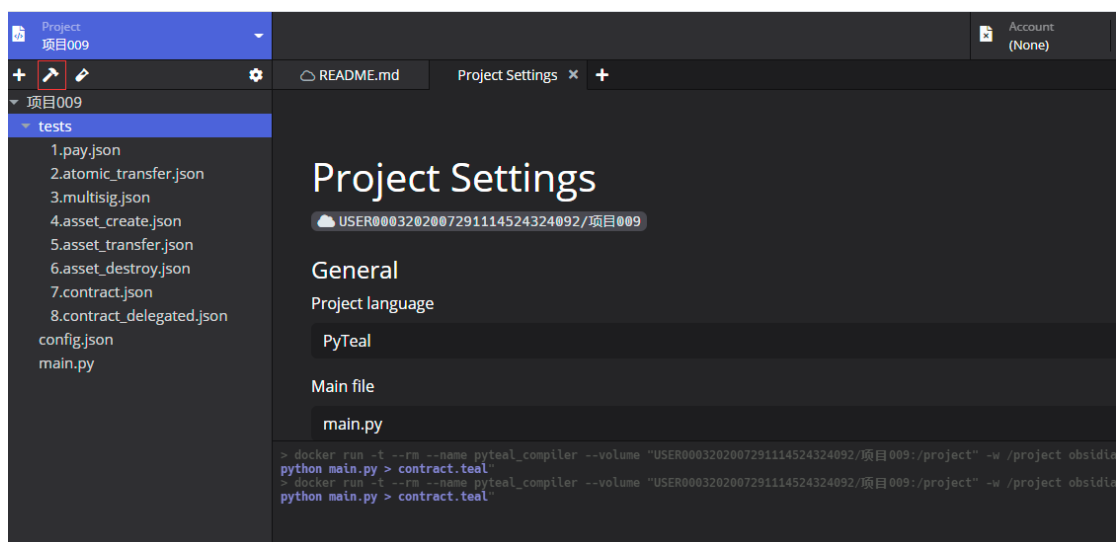
- Edit chaincode package:  
Click and expand the chaincode package in the IDE, and edit the chaincode in the editing page.



- Test chaincode package:  
In the editing page, click on “Run Test Transaction” to test the chaincode package.



- Deploy chaincode package:  
In the editing page, after passing the test, click on “Deploy” button to complete the deployment of the chaincode package.



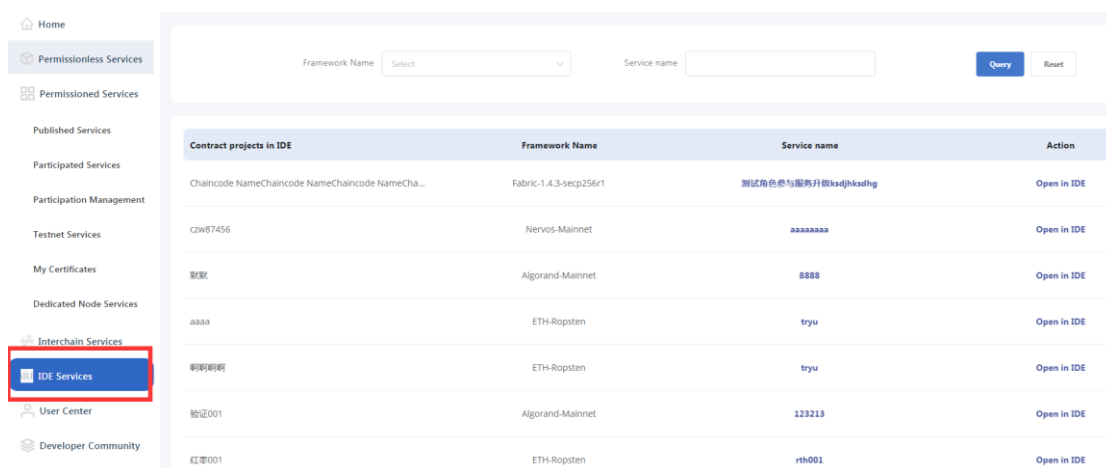
4. Chaincode package deployment. Developers can select the access information of the project in the BSN portal.

### 10.2.4 BSN Testnet Services

BSN testnet services has integrated the IDE, and the specific steps are consistent with the production environment.

## 10.3 My IDE

Login to the BSN portal and go to home page, click **【IDE Services】**, and enter the service inquiry page.



BSN International portal supports developers to view the edited chaincode packages in IDE service query page, developers can query by framework name or service name.

If a chaincode package is used in multiple services or projects, it will be associated with multiple services or projects in the service name column, click the service name to jump to the service to view the service/project details; click "Open in IDE" to jump to the IDE for chaincode package editing.



## 11 DID Services

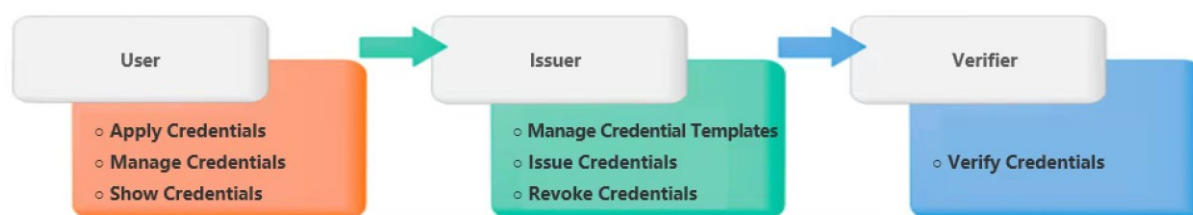
### 11.1 Overview

With blockchain technology as the cornerstone and W3C DID as the specification, BSN DID Services achieve decentralized on-chain mapping of real entity, thus achieving the ability to provide digital identity and digital credential interaction for individuals/organizations.

#### Roles

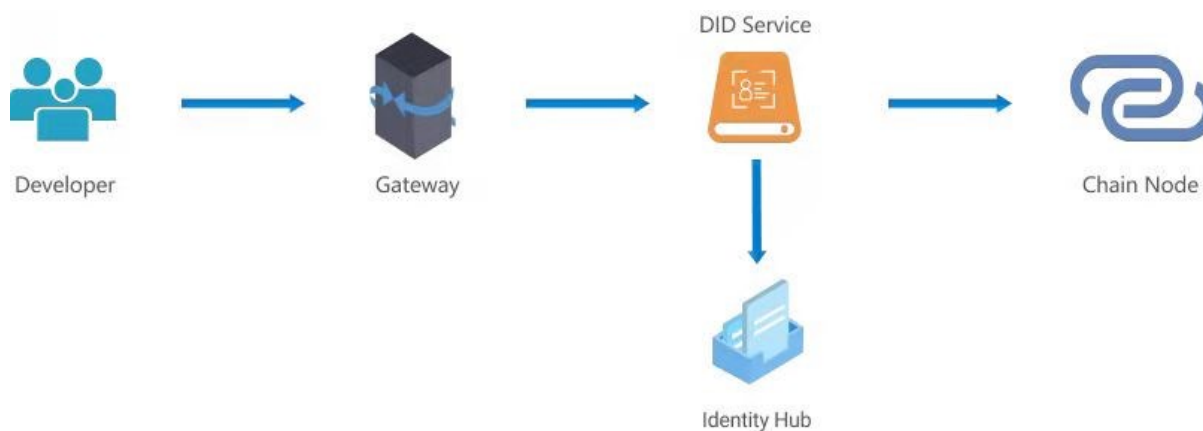
In the DID ecosystem, there are three roles: User, Issuer and Verifier.

- **User:** Any individual/organization/entity that has a digital identity on the chain. Any entity object can create and manage its DID through the developer's project.
- **Issuer:** The individual or organization that can issue the digital credentials. For example, if a university can issue a digital diploma to a student, then the university is an issuer.
- **Verifier:** Also known as a business party, is an individual or organization that uses digital credentials. After being authorized by the user, the verifier can verify the identity of the user or their digital credentials. For example, when a company hires someone, it needs to verify his college diploma, then the company is a verifier.



#### Components

The DID system consists of three components: SDK, Service and Smart Contract. The SDK can be integrated in the developer's project; Service handles the business logic and connects the private data storage area (Identity Hub) to the chain node; the smart contract is deployed on the chain, and the methods in the contract are called by the Service.



#### Functions and features

- Deployed on the BSN, the DID Service builds a decentralized digital identity management system, which facilitates autonomous participation and affirmative collaboration among users, issuers and verifiers.
- Provide a unified decentralized digital identity management, including identifier creation, update and verification functions.
- Provide mechanisms for issuance, authorization, verification and revocation of user data credentials.
- Provide a private data storage area (Identity Hub) where credentials are fully controlled by users, and is stored and transmitted in encryption.
- Provide unified access to API services and SDKs, integrate object encapsulation, signature, verification and other methods for easy docking by developers.

## 11.2 HTTP API

### 1. Access parameters

| No. | Parameter | Description | Value                         |
|-----|-----------|-------------|-------------------------------|
| 1   | url       | Gateway URL | https://did.bsngate.com:18602 |
| 2   | projectId | Project ID  | 8320935187                    |
| 3   | token     | Project Key | 3wxYHXwAm57grc9JUr2zrPHt9HC   |

### 2. Public parameters

| Public request header parameters |              |                 |                             |                     |
|----------------------------------|--------------|-----------------|-----------------------------|---------------------|
| No.                              | Parameter    |                 | Value                       |                     |
| 1                                | token        |                 | 3wxYHXwAm57grc9JUr2zrPHt9HC |                     |
| 2                                | projectId    |                 | 8320935187                  |                     |
| 3                                | Content-Type |                 | application/json            |                     |
| Public request parameters        |              |                 |                             |                     |
| No.                              | Parameter    | Type            | Required                    | Description         |
| 1                                |              | RequestParam<T> | Y                           | Public parameter    |
| RequestParam                     |              |                 |                             |                     |
| 1                                | projectId    | String          | Y                           | Project ID          |
| 2                                | did          | String          | Y                           | DID                 |
| 3                                | data         | T               | Y                           | Any type of data    |
| 4                                | sign         | String          | Y                           | Secp256k1 signature |
| Public Response Parameter        |              |                 |                             |                     |
| No.                              | Parameter    | Type            | Description                 |                     |
| 1                                |              | ResultData<T>   | Public response data        |                     |
| ResultData                       |              |                 |                             |                     |
| 1                                | code         | Integer         | Returned code               |                     |

|   |      |        |                  |
|---|------|--------|------------------|
| 2 | msg  | String | Message          |
| 3 | data | T      | Any type of data |

### 3. DID Creation

The process of creating DID generates public and private key pair. In order to avoid the transmission of private keys, BSN DID Services do not provide the process of creating DIDs. Developers can generate them locally as described below, or they can use or refer to the SDK to complete the generation.

- 1) Generate two public and private key pairs through the elliptic curve algorithm Secp256k1.
- 2) Save the private key and specify the primary and recovery public keys to assemble the Base DID Document, the contents of which are shown below:

```
{
  "@context": "https://w3id.org/did/v1",
  "authentication":
  {
    "type": "Secp256k1",
    "publicKey":
    "28986472722394106073871327423452879123214061743224210681401278929598807211
    14000127450753032422192379586544768083674234896333734351022988066996849973
    5858"
  }
  "recovery":
  {
    "type": "Secp256k1",
    "publicKey":
    "92519711680429159415515746419877215039845427616418520648539645411813788327
    46959340151297908312616596971625573967556676367696067937171601766581709843
    378481"
  }
}
```

- 3) The DID identifier is generated by the base58(ripemd160(sha256(<Base DID Document>))) algorithm in the following example format:

did:bsn:3wxYHXwAm57grc9JUr2zrPHt9HC

- 4) Assemble the contents of the DID Document, as the following example:

```
{
  "did": "did:bsn:3wxYHXwAm57grc9JUr2zrPHt9HC",
  "version": 1,
  "created": "2021-05-20T16:02:20Z",
  "updated": "2021-05-20T16:02:20Z",
  "authentication":
  {
    "type": "Secp256k1",
    "publicKey":
    "28986472722394106073871327423452879123214061743224210681401278929598807211
    14000127450753032422192379586544768083674234896333734351022988066996849973
    5858"
  }
}
```

```
5858"
}
"recovery":
{
  "type": "Secp256k1",
  "publicKey":
"92519711680429159415515746419877215039845427616418520648539645411813788327
46959340151297908312616596971625573967556676367696067937171601766581709843
378481"
}
}
```

5) Use the primary private key to sign the DID Document content with Secp256k1, and finally form the DID Document with the signature attribute, as following:

```
{
  "did": "did:bsn:3wxYHXwAm57grc9JUzrPHt9HC",
  "version": 1,
  "created": "2021-05-20T16:02:20Z",
  "updated": "2021-05-20T16:02:20Z",
  "authentication":
  {
    "type": "Secp256k1",
    "publicKey":
"28986472722394106073871327423452879123214061743224210681401278929598807211
14000127450753032422192379586544768083674234896333734351022988066996849973
5858"
  }
  "recovery":
  {
    "type": "Secp256k1",
    "publicKey":
"92519711680429159415515746419877215039845427616418520648539645411813788327
46959340151297908312616596971625573967556676367696067937171601766581709843
378481"
  }
  "proof":
  {
    "type": "Secp256k1",
    "creator": "did:bsn:3wxYHXwAm57grc9JUzrPHt9HC",
    "signatureValue":
"zD5nt+P/Ga/CRG2hJU/SMRXy210CLdvATsxQdPxTEy9Mc9Y0OSFpE3Yu5k2+OjQKV
Otu5of9VFbgO3Zljw/vQxs="
  }
}
```

### 11.2.1 DID API

The creation of DID is done offline, so the following API is used to upload the DID to the chain and query the DID Document information on the chain.

#### 11.2.1.1 Verify DID Document

|                   |                |
|-------------------|----------------|
| Interface Address | /did/verifyDoc |
|-------------------|----------------|

|                              |                |  |  |  |
|------------------------------|----------------|--|--|--|
| Description                  |                | Verify the content format and signature value of the offline generated DID Document. |  |  |
| Interface request parameter  |                |  |  |  |
| No.                          | Parameter      | Type   | Required                                       | Description  |
| 1                            |                | VerifyDocumentReq  | Y  | Wrapper class  |
| VerifyDocumentReq            |                |  |  |  |
| 1                            | didDoc         | DidDocument  | Y  | DID Document   |
| DidDocument                  |                |  |  |  |
| 1                            | did            | String   | Y  | DID  |
| 2                            | version        | String   | Y  | Version  |
| 3                            | created        | String   | Y  | Created date   |
| 4                            | updated        | String   | Y  | Updated date   |
| 5                            | authentication | PublicKey  | Y  | Primary public key                                   |
| 6                            | recovery       | PublicKey  | Y  | Recovery public key                                  |
| 7                            | proof          | Proof  | Y  | Signature  |
| PublicKey                    |                |  |  |  |
| 1                            | publicKey      | String   | Y  | Public key   |
| 2                            | type           | String   | Y  | Algorithm type                                       |
| Proof                        |                |  |  |  |
| 1                            | creator        | String   | Y  | DIDs involved in the calculation of signature values |
| 2                            | type           | String   | Y  | Algorithm type                                       |
| 3                            | signatureValue | String   | Y  | Signature value                                      |
| Interface response parameter |                |  |  |  |
| No.                          | Parameter      | Type   | Description                                    |  |
| 1                            |                | Boolean  | Return true if success, return false if failed |  |

### 11.2.1.2 Add DID Document to the chain

|                             |           |  |          |               |
|-----------------------------|-----------|--|----------|---------------|
| Interface Address           |           | /did/putDoc  |          |               |
| Description                 |           | The DID Document is stored in the chain. The verification will be executed internally, so if you want to upload the DID Document to the chain, you can directly call this interface. |          |               |
| Interface request parameter |           |  |          |               |
| No.                         | Parameter | Type   | Required | Description   |
| 1                           |           | DidDocSotreReq   | Y        | Wrapper class |
| DidDocSotreReq              |           |  |          |               |
| 1                           | didDoc    | Document   | Y        | DID Document  |
| Document                    |           |  |          |               |
| 1                           | did       | String   | Y        | DID           |
| 2                           | version   | String   | Y        | Version       |
| 3                           | created   | String   | Y        | Created date  |
| 4                           | updated   | String   | Y        | Updated date  |

| 5                                   | authentication | PublicKey | Y  | Primary public key                                     |
|-------------------------------------|----------------|-----------|--|--|
| 6                                   | recovery       | PublicKey | Y  | Recovery public key                                    |
| 7                                   | proof          | Proof     | Y  | Signature  |
| <b>PublicKey</b>                    |                |           |  |  |
| 1                                   | publicKey      | String    | Y  | Public key   |
| 2                                   | type           | String    | Y  | Algorithm type   |
| <b>Proof</b>                        |                |           |  |  |
| 1                                   | creator        | String    | Y  | DID involved in the calculation of the Signature value |
| 2                                   | type           | String    | Y  | Algorithm type   |
| 3                                   | signatureValue | String    | Y  | Signature value  |
| <b>Interface response parameter</b> |                |           |  |  |
| No.                                 | Parameter      | Type      | Description                                    |  |
| 1                                   |                | Boolean   | Return true if success, return false if failed |  |

### 11.2.1.3 Get DID Document

| Interface Address            |                | /did/getDoc   |  |               |
|------------------------------|----------------|---|--|---------------|
| Description                  |                | The information in the DID Document is a record and description of the DID, so anyone can query the corresponding DID Document on the chain by the DID. It can be used to verify the DID and obtain the DID public key. |  |               |
| Interface request parameter  |                |   |  |               |
| No.                          | Parameter      | Type  | Required   | Description   |
| 1                            |                | DidDocumentReq  | Y  | Wrapper class |
| DidDocumentReq               |                |   |  |               |
| 1                            | did            | String  | Y  | DID           |
| Interface response parameter |                |   |  |               |
| No.                          | Parameter      | Type  | Description  |               |
| 1                            |                | DidDocument   | DID Document   |               |
| DidDocument                  |                |   |  |               |
| 1                            | did            | String  | DID  |               |
| 2                            | version        | String  | Version  |               |
| 3                            | created        | String  | Created date   |               |
| 4                            | updated        | String  | Updated date   |               |
| 5                            | authentication | PublicKey   | Primary public key                                     |               |
| 6                            | recovery       | PublicKey   | Recovery public key                                    |               |
| 7                            | proof          | Proof   | Signature  |               |
| PublicKey                    |                |   |  |               |
| 1                            | publicKey      | String  | Public key   |               |
| 2                            | type           | String  | Algorithm type   |               |
| Proof                        |                |   |  |               |
| 1                            | creator        | String  | DID involved in the calculation of the Signature value |               |

|   |                |        |                 |
|---|----------------|--------|-----------------|
| 2 | type           | String | Algorithm type  |
| 3 | signatureValue | String | Signature value |

#### 11.2.1.4 Verify DID Signature

| Interface Address            |           | /did/verifyDidSign  |  |                     |
|------------------------------|-----------|---|--|---------------------|
| Description                  |           | Verify the signature value of the DID to ensure the authenticity and validity of the current DID. |  |                     |
|                              |           |   |  |                     |
| Interface request parameter  |           |   |  |                     |
| No.                          | Parameter | Type  | Required                                       | Description         |
| 1                            |           | VerifyDidReq  | Y  | Wrapper class       |
| VerifyDidReq                 |           |   |  |                     |
| 1                            | did       | String  | Y  | DID                 |
| 2                            | didSign   | String  | Y  | DID signature value |
| Interface response parameter |           |   |  |                     |
| No.                          | Parameter | Type  | Description                                    |                     |
| 1                            |           | Boolean   | Return true if success, return false if failed |                     |

#### 11.2.1.5 Update Key

| Interface Address           |                | /did/resetDidAuth  |          |   |
|-----------------------------|----------------|--|----------|---|
| Description                 |                | The generation of the new authentication public-private key pair from the recovery public-private key information is done by the DID SDK. The interface receives new DID Document content from the user for on-chain update. |          |   |
| Interface request parameter |                |  |          |   |
| No.                         | Parameter      | Type   | Required | Description   |
| 1                           |                | RestDocAuth  | Y        | Wrapper class   |
| RestDocAuth                 |                |  |          |   |
| 1                           | didDoc         | Document   | Y        | DID Document  |
| 2                           | authPubKeySign | String   | Y        | The recovery private key performs k1 signature on the recovery public key |
| Document                    |                |  |          |   |
| 1                           | did            | String   | Y        | DID   |
| 2                           | version        | String   | Y        | Version   |
| 3                           | created        | String   | Y        | Created date  |
| 4                           | updated        | String   | Y        | Updated date  |
| 5                           | authentication | PublicKey  | N        | Primary public key  |
| 6                           | recovery       | PublicKey  | Y        | Recovery public key   |
| 7                           | proof          | Proof  | Y        | Signature   |
| PublicKey                   |                |  |          |   |
| 1                           | publicKey      | String   | Y        | Public key  |
| 2                           | type           | String   | Y        | Algorithm type  |
| Proof                       |                |  |          |   |

| 1                            | creator        | String  | Y   | DID involved in the calculation of the Signature value |
|------------------------------|----------------|---------|---|--|
| 2                            | type           | String  | Y   | Algorithm type   |
| 3                            | signatureValue | String  | Y   | Signature value  |
| Interface response parameter |                |         |   |  |
| No.                          | Parameter      | Type    | Description                               |  |
| 1                            |                | KeyInfo | New authentication public key information |  |
| PublicKey                    |                |         |   |  |
| 1                            | publicKey      | String  | Public key                                |  |
| 2                            | type           | String  | Algorithm type                            |  |

### 11.2.2 Issuer

The issuer and user are two roles, and the following APIs are the pre-constraints for issuing credentials. The process of changing the DID user to the issuer does not change the DID Identifier or DID Document, but only the status.

To issue a credential, a DID user needs to register as an issuer and then define a template for registering the credential they want to issue. The credential template will be stored on the chain and everyone can query.

#### 11.2.2.1 Issuer Registration

| Interface Address              |               | /did/registerAuthIssuer   |  |               |
|--------------------------------|---------------|---|--|---------------|
| Description                    |               | The DID user becomes the issuer and the issuer information is uploaded if the registration is successful. |  |               |
| Interface request parameter    |               |   |  |               |
| No.                            | Parameter     | Type  | Required                                       | Description   |
| 1                              |               | RegisterAuthorityIssuerWrapper  | Y  | Wrapper class |
| RegisterAuthorityIssuerWrapper |               |   |  |               |
| 1                              | did           | String  | Y  | DID           |
| 2                              | name          | String  | Y  | Issuer’s name |
| 3                              | publicKeySign | String  | Y  | Public key    |
| Interface response parameter   |               |   |  |               |
| No.                            | Parameter     | Type  | Description                                    |               |
| 1                              |               | Boolean   | Return true if success, return false if failed |               |

#### 11.2.2.2 Query Issuer

|                             |           |   |          |             |
|-----------------------------|-----------|---|----------|-------------|
| Interface Address           |           | /did/queryAuthIssuerList  |          |             |
| Description                 |           | Check whether you are the issuer by DID and identify which type of credentials can be issued by name. |          |             |
| Interface request parameter |           |   |          |             |
| No.                         | Parameter | Type  | Required | Description |



| 1                            |           | AuthIssuerListWrapper | Y                  | Wrapper class     |
|------------------------------|-----------|-----------------------|--------------------|-------------------|
| AuthIssuerListWrapper        |           |                       |                    |                   |
| 1                            | did       | String                | Y                  | DID               |
| 2                            | page      | Integer               | Y                  | Page              |
| 3                            | size      | Integer               | Y                  | Number of entries |
| Interface response parameter |           |                       |                    |                   |
| No.                          | Parameter | Type                  | Description        |                   |
| 1                            |           | AuthorityIssuer       | Issuer Information |                   |
| AuthorityIssuer              |           |                       |                    |                   |
| 1                            | did       | String                | DID                |                   |
| 2                            | name      | String                | Issuer's name      |                   |

### 11.2.2.3 Register Credential Template

| Interface Address            |                | /did/registerCpt  |             |  |
|------------------------------|----------------|---|-------------|--|
| Description                  |                | The issuer customizes the credential template and can agree on which attribute values must be provided by the applicant. For example, in the template of college diploma, you can agree that "name" and "student number" are mandatory information. |             |  |
| Interface request parameter  |                |   |             |  |
| No.                          | Parameter      | Type  | Required    | Description  |
| 1                            |                | RegisterCptWrapper  | Y           | Wrapper class  |
| RegisterCptWrapper           |                |   |             |  |
| 1                            | did            | String  | Y           | DID  |
| 2                            | cptJsonSchema  | Map<String, JsonSchema>   | Y           | JsonSchema information for MapType                     |
| 3                            | title          | String  | Y           | Credential template title                              |
| 4                            | description    | String  | Y           | Credential template description                        |
| 5                            | type           | String  | Y           | Credential type, fill in proof                         |
| 6                            | proof          | Proof   | Y           | Signature  |
| 7                            | create         | String  | Y           | Created date   |
| 8                            | update         | String  | Y           | Updated date   |
| Proof                        |                |   |             |  |
| 1                            | creator        | String  | Y           | DID involved in the calculation of the Signature value |
| 2                            | type           | String  | Y           | Algorithm type   |
| 3                            | signatureValue | String  | Y           | Signature value  |
| JsonSchema                   |                |   |             |  |
| 1                            | type           | String  | Y           | Field type   |
| 2                            | description    | String  | Y           | Field description                                      |
| 3                            | required       | boolean   | Y           | Whether is required to fill                            |
| Interface response parameter |                |   |             |  |
| No.                          | Parameter      | Type  | Description |  |

|                    |            |             |                                 |
|--------------------|------------|-------------|---------------------------------|
| 1                  |            | CptBaseInfo | Credential template information |
| <b>CptBaseInfo</b> |            |             |                                 |
| 1                  | cptId      | Long        | Credential template ID          |
| 2                  | cptVersion | Integer     | Version                         |

#### 11.2.2.4 Query Credential Template List

|                              |               |  |                                       |                   |
|------------------------------|---------------|--|---------------------------------------|-------------------|
| Interface Address            |               | /did/queryCptList  |                                       |                   |
| Description                  |               | Anyone can check all their credential templates by DID. It is possible for the same individual/organization to register multiple credential templates. For example, a university may have a degree template, an incomplete template, etc. in addition to a diploma template. |                                       |                   |
| Interface request parameter  |               |  |                                       |                   |
| No.                          | Parameter     | Type   | Required                              | Description       |
| 1                            |               | QueryCptListWrapper  | Y                                     | Wrapper class     |
| QueryCptListWrapper          |               |  |                                       |                   |
| 1                            | page          | Integer  | Y                                     | Page              |
| 2                            | size          | Integer  | Y                                     | Number of entries |
| 3                            | did           | String   | Y                                     | DID               |
| Interface response parameter |               |  |                                       |                   |
| No.                          | Parameter     | Type   | Description                           |                   |
| 1                            |               | Pages<CptInfo>   | Credential template list info         |                   |
| Pages<CptInfo>               |               |  |                                       |                   |
| 1                            | page          | Integer  | Page                                  |                   |
| 2                            | size          | Integer  | Number of entries                     |                   |
| 3                            | totalNum      | Integer  | Number of entries in total            |                   |
| 4                            | totalPage     | Integer  | Number of pages in total              |                   |
| 5                            | result        | List<CptInfo>  | Result list                           |                   |
| CptInfo                      |               |  |                                       |                   |
| 1                            | cptVersion    | Integer  | Credential template version           |                   |
| 2                            | cptJsonSchema | Map<String, JsonSchema>  | JsonSchema information for MapType    |                   |
| 3                            | title         | String   | Credential template title             |                   |
| 4                            | description   | String   | Credential template description       |                   |
| 5                            | publisherDid  | String   | DID of the credential template issuer |                   |
| 6                            | proof         | Proof  | Signature                             |                   |
| 7                            | cptId         | Long   | Credential template ID                |                   |
| 8                            | create        | String   | Created date                          |                   |
| 9                            | update        | String   | Updated date                          |                   |
| JsonSchema                   |               |  |                                       |                   |
| 1                            | type          | String   | Field type                            |                   |
| 2                            | description   | String   | Field description                     |                   |
| 3                            | required      | Boolean  | whether is required to fill           |                   |

| Proof |                |        |  |
|-------|----------------|--------|--|
| 1     | creator        | String | DID involved in the calculation of the Signature value |
| 2     | type           | String | Algorithm type   |
| 3     | signatureValue | String | Signature value  |

#### 11.2.2.5 Query Credential Template

| Interface Address            |                | /did/queryCptById   |  |                        |
|------------------------------|----------------|---|--|------------------------|
| Description                  |                | Query the contents of a specific credential template by its ID. |  |                        |
| Interface request parameter  |                |   |  |                        |
| No.                          | Parameter      | Type  | Required   | Description            |
| 1                            |                | QueryCptByIdWrapper   | Y  | Wrapper class          |
| QueryCptByIdWrapper          |                |   |  |                        |
| 1                            | cptId          | Long  | Y  | Credential template ID |
| Interface response parameter |                |   |  |                        |
| No.                          | Parameter      | Type  | Description  |                        |
| 1                            |                | CptInfo   | Credential template information                        |                        |
| CptInfo                      |                |   |  |                        |
| 1                            | cptVersion     | Integer   | Version  |                        |
| 2                            | cptJsonSchema  | Map<String, JsonSchema>   | JsonSchema information for MapType                     |                        |
| 3                            | title          | String  | Credential template title                              |                        |
| 4                            | description    | String  | Credential template description                        |                        |
| 5                            | publisherDid   | String  | DID of the credential template issuer                  |                        |
| 6                            | proof          | Proof   | Signature  |                        |
| 7                            | cptId          | Long  | Credential template ID                                 |                        |
| 8                            | create         | String  | Created date   |                        |
| 9                            | update         | String  | Updated date   |                        |
| JsonSchema                   |                |   |  |                        |
| 1                            | type           | String  | Field type   |                        |
| 2                            | description    | String  | Field description                                      |                        |
| 3                            | required       | boolean   | WHETHER IS REQUIRED TO FILL                            |                        |
| Proof                        |                |   |  |                        |
| 1                            | creator        | String  | DID involved in the calculation of the Signature value |                        |
| 2                            | type           | String  | Algorithm type   |                        |
| 3                            | signatureValue | String  | Signature value  |                        |

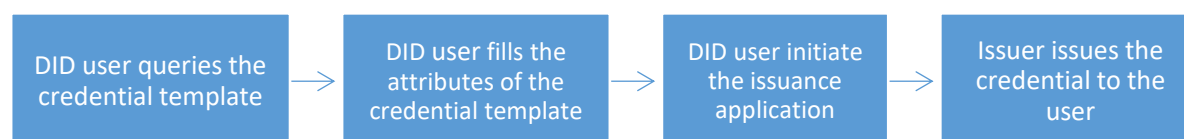
#### 11.2.2.6 Update Credential Template

| Interface Address |  | /did/updateCpt   |  |  |
|-------------------|--|--|--|--|
| Description       |  | The issuer updates the content of its own registered credential templates. The update of credential template does not affect credentials already issued. |  |  |

| Interface request parameter  |                |                         |   |  |
|------------------------------|----------------|-------------------------|---|--|
| No.                          | Parameter      | Type                    | Required  | Description  |
| 1                            |                | RegisterCptWrapper      | Y   | Wrapper class  |
| RegisterCptWrapper           |                |                         |   |  |
| 1                            | did            | String                  | Y   | DID  |
| 2                            | cptJsonSchema  | Map<String, JsonSchema> | Y   | JsonSchema information for MapType                     |
| 3                            | title          | String                  | Y   | Credential template title                              |
| 4                            | description    | String                  | Y   | Credential template description                        |
| 5                            | type           | String                  | Y   | Credential type, fill in proof                         |
| 6                            | proof          | Proof                   | Y   | Signature  |
| 7                            | cptId          | Long                    | Y   | Credential template ID                                 |
| 8                            | create         | String                  | Y   | Created date   |
| 9                            | update         | String                  | Y   | Updated date   |
| Proof                        |                |                         |   |  |
| 1                            | creator        | String                  | Y   | DID involved in the calculation of the Signature value |
| 2                            | type           | String                  | Y   | Algorithm type   |
| 3                            | signatureValue | String                  | Y   | Signature value  |
| JsonSchema                   |                |                         |   |  |
| 1                            | type           | String                  | Y   | Field type   |
| 2                            | description    | String                  | Y   | Field description                                      |
| 3                            | required       | boolean                 | Y   | WHETHER IS REQUIRED TO FILL                            |
| Interface response parameter |                |                         |   |  |
| No.                          | Parameter      | Type                    | Description                                       |  |
| 1                            |                | CptBaseInfo             | Credential template information                   |  |
| CptBaseInfo                  |                |                         |   |  |
| 1                            | cptId          | Long                    | Credential template ID                            |  |
| 2                            | cptVersion     | Integer                 | Version, plus one after each update is successful |  |

### 11.2.3 Credential

The credential is generated based on the credential template. The application of the credential is made by the user, and then the issuer issues the credential. The credential issuance process is generally as follows:



Once the user has the credentials issued by the issuer, he/she can present them to the verifier for further use.

## 11.2.3.1 Issue Credential

| Interface Address            |                | /did/createCredential  |  |  |
|------------------------------|----------------|--|--|--|
| Description                  |                | The attribute values defined in the credential template are provided by the issuer for the DID user to obtain on the front page. The issuer issues the credentials for the DID user through this interface. If there are more Claim parameters than defined in the credential template, the server side will discard them. |  |  |
| Interface request parameter  |                |  |  |  |
| No.                          | Parameter      | Type   | Required                                   | Description  |
| 1                            |                | CreateCredentialReq  | Y  | Wrapper class  |
| CreateCredentialReq          |                |  |  |  |
| 1                            | cptId          | Long   | Y  | Credential template ID   |
| 2                            | issuerDid      | String   | Y  | DID of the credential template issuer  |
| 3                            | userDid        | String   | Y  | DID of the user requesting the credentials   |
| 4                            | expirationDate | String   | Y  | Credential expiration date   |
| 5                            | claim          | Map<String,Object>   | Y  | Claim data   |
| 6                            | type           | String   | Y  | Credential type, fill in Proof   |
| 7                            | shortDesc      | String   | N  | Brief description of the credential template. If this field is null, the value of the title field in the credential template is displayed. If not, the input value is displayed. |
| 8                            | longDesc       | String   | N  | Detailed description of the credential template  |
| Interface response parameter |                |  |  |  |
| No.                          | Parameter      | Type   | Description                                |  |
| 1                            |                | CredentialWrapper  | Credential issuance information            |  |
| CredentialWrapper            |                |  |  |  |
| 1                            | context        | String   | Specification                              |  |
| 2                            | id             | String   | Credential ID                              |  |
| 3                            | type           | String   | Credential type, fill in proof             |  |
| 4                            | cptId          | Long   | Credential template ID                     |  |
| 5                            | issuerDid      | String   | DID of credential issuer                   |  |
| 6                            | userDid        | String   | DID of the user requesting the credentials |  |
| 7                            | expirationDate | String   | Expiration date                            |  |
| 8                            | created        | String   | Created date                               |  |
| 9                            | shortDesc      | String   | Brief description of the credential        |  |
| 10                           | longDesc       | String   | Detailed description of the credential     |  |
| 11                           | claim          | Map<String, Object>  | Claim data                                 |  |
| 12                           | proof          | Map<String, Object>  | Signature                                  |  |

## 11.2.3.2 Verify Credential

| Interface Address            | /did/verifyCredential  |                     |  |  |
|------------------------------|--|---------------------|--|--|
| Description                  | Generally called by the verifier. It can verify whether a particular credential is valid or not. Verify the signature of the credential, whether the credential is expired, and whether the credential is revoked, respectively. |                     |  |  |
| Interface request parameter  |  |                     |  |  |
| No.                          | Parameter  | Type                | Required                                       | Description                                |
| 1                            |  | VerifyCredentialReq | Y  | Wrapper class                              |
| VerifyCredentialReq          |  |                     |  |  |
| 1                            | credentialWrapper  | CredentialWrapper   | Y  | Credential information                     |
| 2                            | publicKey  | PublicKey           | Y  | Issuer’s public key                        |
| CredentialWrapper            |  |                     |  |  |
| 1                            | context  | String              | Y  | Specification                              |
| 2                            | id   | String              | Y  | Credential ID                              |
| 3                            | type   | String              | Y  | Credential type, fill in Proof             |
| 4                            | cptId  | Long                | Y  | Credential template ID                     |
| 5                            | issuerDid  | String              | Y  | DID of credential issuer                   |
| 6                            | userId   | String              | Y  | DID of the user requesting the credentials |
| 7                            | expirationDate   | String              | Y  | Expiration date                            |
| 8                            | created  | String              | Y  | Created date                               |
| 9                            | shortDesc  | String              | N  | Brief description of the credential        |
| 10                           | longDesc   | String              | N  | Detailed description of the credential     |
| 11                           | claim  | Map<String, Object> | Y  | Claim data                                 |
| 12                           | proof  | Map<String, Object> | Y  | Signature                                  |
| PublicKey                    |  |                     |  |  |
| 1                            | publicKey  | String              | Y  | Public key                                 |
| 2                            | type   | String              | Y  | Algorithm type                             |
| Interface response parameter |  |                     |  |  |
| No.                          | Parameter  | Type                | Description                                    |  |
| 1                            |  | Boolean             | Return true if success, return false if failed |  |

### 11.2.3.3 Revoke Credential

| Interface Address           |           | /did/revokeCredential   |          |               |
|-----------------------------|-----------|---|----------|---------------|
| Description                 |           | Called by the issuer to revoke or void a credential that has been issued. Since the issued credential is already in the custody of the user, the revocation of the credential is followed by the upload of its credential ID. |          |               |
| Interface request parameter |           |   |          |               |
| No.                         | Parameter | Type  | Required | Description   |
| 1                           |           | RevokCredentialReq  | Y        | Wrapper class |
| RevokCredentialReq          |           |   |          |               |

| 1                                   | credId        | String  | Y  | Credential ID to be revoked  |
|-------------------------------------|---------------|---------|--|--|
| 2                                   | cptId         | Long    | Y  | Credential template ID   |
| 3                                   | did           | String  | Y  | Issuer's DID   |
| 4                                   | revokeDate    | String  | Y  | Revoked date   |
| 5                                   | publicKeySign | String  | Y  | The primary private key performs a k1 signature on the primary public key                                      |
| 6                                   | revokeSign    | String  | Y  | After splicing the certificate ID and revocation time, use the primary private key for k1 revocation signature |
| <b>Interface response parameter</b> |               |         |  |  |
| No.                                 | Parameter     | Type    | Description                                    |  |
| 1                                   |               | Boolean | Return true if success, return false if failed |  |

#### 11.2.3.4 Query Revoked Credential

|                              |           |   |                             |                   |
|------------------------------|-----------|---|-----------------------------|-------------------|
| Interface Address            |           | /did/getRevokedCredList   |                             |                   |
| Description                  |           | Called when verifying credentials. Find out all its revoked credential IDs by Issuer's DID. |                             |                   |
| Interface request parameter  |           |   |                             |                   |
| No.                          | Parameter | Type  | Required                    | Description       |
| 1                            |           | QueryCredentialWrapper  | Y                           | Wrapper class     |
| QueryCredentialWrapper       |           |   |                             |                   |
| 1                            | Page      | Integer   | Y                           | Page              |
| 2                            | Size      | Integer   | Y                           | Number of entries |
| 3                            | did       | String  | Y                           | Issuer's DID      |
| Interface response parameter |           |   |                             |                   |
| No.                          | Parameter | Type  | Description                 |                   |
| 1                            |           | Pages<BaseCredential>   | Revocation list information |                   |
| Pages<BaseCredential>        |           |   |                             |                   |
| 1                            | page      | Integer   | Page                        |                   |
| 2                            | size      | Integer   | Number of entries           |                   |
| 3                            | totalNum  | Integer   | Number of entries in total  |                   |
| 4                            | totalPage | Integer   | Number of pages in total    |                   |
| 5                            | result    | List<BaseCredential >   | Result list                 |                   |
| BaseCredential               |           |   |                             |                   |
| 1                            | id        | String  | Credential ID               |                   |
| 2                            | created   | String  | Revoked date                |                   |

#### 11.2.4 Identity Hub

BSN provides users with a privacy data storage area (Identity Hub, hereinafter referred to as Hub) where users can choose to store their credentials or other data (both are called “resource”)

into the Hub, which verifies the identity of visitors while the data is encrypted during transmission and storage, leaving users in full control of access to their own data.

#### 11.2.4.1 Register Hub User by DID

| Interface Address            |           | /hub/register   |                                     |               |
|------------------------------|-----------|---|-------------------------------------|---------------|
| Description                  |           | Register by DID, if successful, returns the user's ID in the Hub. |                                     |               |
|                              |           |   |                                     |               |
| Interface request parameter  |           |   |                                     |               |
| No.                          | Parameter | Type  | Required                            | Description   |
| 1                            |           | RegisterHubReq  | Y                                   | Wrapper class |
| RegisterHubReq               |           |   |                                     |               |
| 1                            | did       | String  | Y                                   | DID           |
| 2                            | publicKey | String  | Y                                   | Public key    |
| Interface response parameter |           |   |                                     |               |
| No.                          | Parameter | Type  | Description                         |               |
| 1                            |           | RegisterHubResult   | Wrapper class of the returned value |               |
| RegisterHubResult            |           |   |                                     |               |
| 1                            | success   | Boolean   | Whether it is successful            |               |
| 2                            | uid       | String  | ID in the Hub                       |               |
| 3                            | message   | String  | Result description                  |               |

#### 11.2.4.2 Register Hub User by Public Key

| Interface Address            |            | /hub/registerByIdPublicKey   |                                     |                                |
|------------------------------|------------|--|-------------------------------------|--------------------------------|
| Description                  |            | Register by Public key. This function allows users define the ID by themselves. If successful, returns the user's ID in the Hub. |                                     |                                |
| Interface request parameter  |            |  |                                     |                                |
| No.                          | Parameter  | Type   | Required                            | Description                    |
| 1                            |            | RegisterHubReq   | Y                                   | Wrapper class                  |
| RegisterHubReq               |            |  |                                     |                                |
| 1                            | id         | String   | N                                   | Self-defined Hub ID            |
| 2                            | publicKey  | String   | Y                                   | Public key                     |
| 3                            | cryptoType | String   | Y                                   | Encryption Algorithm:<br>ECDSA |
| Interface response parameter |            |  |                                     |                                |
| No.                          | Parameter  | Type   | Description                         |                                |
| 1                            |            | RegisterHubResult  | Wrapper class of the returned value |                                |
| RegisterHubResult            |            |  |                                     |                                |
| 1                            | success    | Boolean  | Whether it is successful            |                                |
| 2                            | uid        | String   | ID in the Hub                       |                                |
| 3                            | message    | String   | Result description                  |                                |

#### 11.2.4.3 Save Resource

| Interface Address | /hub/saveResource |
|-------------------|-------------------|
|-------------------|-------------------|



|                              |            |  |   |  |
|------------------------------|------------|--|---|--|
| Description                  |            | Store the resource to the Hub. If the user stores it himself, the uid and the ownerUid should be the same. At this time, there is no need to create permissions to call directly; if the issuer stores it for the user after issuing the credentials, the uid should be the issuer, and the ownerUid should be the user. At this time, the user must have created “WRITE” permissions for it, otherwise the storage will fail. |   |  |
| Interface request parameter  |            |  |   |  |
| No.                          | Parameter  | Type   | Required  | Description  |
| 1                            |            | SaveResourceReq  | Y   | Wrapper class  |
| SaveResourceReq              |            |  |   |  |
| 1                            | uid        | String   | Y   | ID in the Hub  |
| 2                            | content    | String   | Y   | Resource information   |
| 3                            | url        | String   | N   | The path of the resource storage. Null when it is user self-storage; required when the issuer stores it for the user |
| 4                            | ownerUid   | String   | Y   | Resource owner   |
| 5                            | grant      | String   | Y   | WRITE means add; UPDATE means update   |
| 6                            | key        | String   | Y   | Key  |
| 7                            | sign       | String   | Y   | Signature  |
| Interface response parameter |            |  |   |  |
| No.                          | Parameter  | Type   | Description   |  |
| 1                            |            | SaveResourceResp   | Wrapper class   |  |
| SaveResourceResp             |            |  |   |  |
| 1                            | url        | String   | The path of the stored resource   |  |
| 2                            | encryptKey | String   | Null when the issuer stores it for users; if it is the user self-storage, returns the KeyA encrypted by the user's Public key, and the user's private key decrypts KeyA to derive the plaintext Key. The resource stored in the Hub is encrypted by Key with AES-ECB algorithm. |  |

#### 11.2.4.4 Get Resource

|                              |           |   |          |                                 |
|------------------------------|-----------|---|----------|---------------------------------|
| Interface Address            |           | /hub/getResource  |          |                                 |
| Description                  |           | Access the Hub and read the specified resource. Users have direct access, third parties need to obtain authorization from the user to access. |          |                                 |
| Interface request parameter  |           |   |          |                                 |
| No.                          | Parameter | Type  | Required | Description                     |
| 1                            |           | QueryResourceReq  | Y        | Wrapper class                   |
| QueryResourceReq             |           |   |          |                                 |
| 1                            | uid       | String  | Y        | ID in the Hub                   |
| 2                            | url       | String  | Y        | The path of the stored resource |
| 3                            | sign      | String  | Y        | Signature                       |
| Interface response parameter |           |   |          |                                 |

| No.                 | Parameter    | Type         | Description   |
|---------------------|--------------|--------------|---|
| 1                   | ResourceInfo | ResourceInfo | Wrapper class   |
| <b>ResourceInfo</b> |              |              |   |
| 1                   | content      | String       | Content of cryptographic resource   |
| 2                   | key          | String       | For the ciphertext key, you need to use your own private key to decrypt it first, and then decrypt the content. |

#### 11.2.4.5 Delete Resource

|                              |           |   |  |                                 |
|------------------------------|-----------|---|--|---------------------------------|
| Interface Address            |           | /hub/deleteResource   |  |                                 |
| Description                  |           | The owner of the resource can call this function to delete certain resource within the Hub. |  |                                 |
| Interface request parameter  |           |   |  |                                 |
| No.                          | Parameter | Type  | Required                                       | Description                     |
| 1                            |           | DeleteResourceReq   | Y  | Wrapper class                   |
| DeleteResourceReq            |           |   |  |                                 |
| 1                            | uid       | String  | Y  | ID in the Hub                   |
| 2                            | url       | String  | Y  | The path of the stored resource |
| 3                            | sign      | String  | Y  | Signature                       |
| Interface response parameter |           |   |  |                                 |
| No.                          | Parameter | Type  | Description                                    |                                 |
| 1                            |           | Boolean   | Return true if success, return false if failed |                                 |

#### 11.2.4.6 Create Permissions

| Interface Address           |                 | /hub/createPerm  |          |   |
|-----------------------------|-----------------|--|----------|---|
| Description                 |                 | Resource owner creates the permissions of accessing to the resource in the Hub for third parties. Permissions WRITE means store resource, UPDATE means update resource, READ means read resource. An authorization can only be accessed once, and an authorization with the same uid and the same permission cannot be created again without access. However, the permissions of UPDATE and READ from the server side will return the same result for each call. |          |   |
| Interface request parameter |                 |  |          |   |
| No.                         | Parameter       | Type   | Required | Description   |
| 1                           |                 | CreatePermissionReq  | Y        | Wrapper class   |
| CreatePermissionReq         |                 |  |          |   |
| 1                           | uid             | String   | Y        | ID in the Hub   |
| 2                           | grant           | String   | Y        | Permissions   |
| 3                           | grantUid        | String   | Y        | Authorized ID in the Hub                              |
| 4                           | grantPublicKey  | String   | Y        | Authorized public key                                 |
| 5                           | grantEncryptKey | String   | N        | Encryption key.<br>READ/UPDATE permission is required |
| 6                           | url             | String   | N        | The path of the stored resource.                      |

|                              |           |                     |  | READ/UPDATE permission is required |
|------------------------------|-----------|---------------------|--|------------------------------------|
| 7                            | sign      | String              | Y  | Signature                          |
| Interface response parameter |           |                     |  |                                    |
| No.                          | Parameter | Type                | Description  |                                    |
| 1                            |           | AddPermissionResult | Response data  |                                    |
| AddPermissionResult          |           |                     |  |                                    |
| 1                            | url       | String              | The path of the stored resource.   |                                    |
| 2                            | Key       | String              | Ciphertext key to encrypt the resource (encrypted using the authorized public key) |                                    |

#### 11.2.4.7 Delete Permissions

| Interface Address            |           | /hub/deletePermission  |                              |                                  |
|------------------------------|-----------|--|------------------------------|----------------------------------|
| Description                  |           | Resource owner calls this function for permissions that have been created but not yet accessed by third parties. |                              |                                  |
|                              |           |  |                              |                                  |
| Interface request parameter  |           |  |                              |                                  |
| No.                          | Parameter | Type   | Required                     | Description                      |
| 1                            |           | DeletePermissionReq  | Y                            | Wrapper class                    |
| DeletePermissionReq          |           |  |                              |                                  |
| 1                            | uid       | String   | Y                            | ID in the Hub                    |
| 2                            | grant     | String   | Y                            | Permissions                      |
| 3                            | grantUid  | String   | Y                            | Authorized ID in the Hub         |
| 4                            | url       | String   | Y                            | The path of the stored resource. |
| 5                            | sign      | String   | Y                            | Signature                        |
| Interface response parameter |           |  |                              |                                  |
| No.                          | Parameter | Type   | Description                  |                                  |
| 1                            |           | DeletePermissionResult   | Response data                |                                  |
| DeletePermissionResult       |           |  |                              |                                  |
| 1                            | succes    | boolean  | Whether deleted successfully |                                  |
| 2                            | message   | String   | Result description           |                                  |

#### 11.2.4.8 Query Permissions

| Interface Address           |           | /hub/queryPermission  |          |               |
|-----------------------------|-----------|---|----------|---------------|
| Description                 |           | Resource owner calls this function to query Permissions that have already been created. |          |               |
|                             |           |   |          |               |
| Interface request parameter |           |   |          |               |
| No.                         | Parameter | Type  | Required | Description   |
| 1                           |           | QueryPermissionReq  | Y        | Wrapper class |
| QueryPermissionReq          |           |   |          |               |
| 1                           | uid       | String  | Y        | ID in the Hub |

|                              |            |                      |   |  |
|------------------------------|------------|----------------------|---|--|
| 2                            | flag       | String               | N | YES means accessed; NO means not accessed. |
| 3                            | grantUid   | String               | N | Authorized ID in the Hub                   |
| 4                            | sign       | String               | Y | Signature                                  |
| Interface response parameter |            |                      |   |  |
| No.                          | Parameter  | Type                 |   | Description                                |
| 1                            |            | List<PermissionInfo> |   | Response data                              |
| PermissionInfo               |            |                      |   |  |
| 1                            | url        | String               |   | The path of the stored resource            |
| 2                            | grant      | String               |   | Permissions                                |
| 3                            | grantUid   | String               |   | Authorized ID in the Hub                   |
| 4                            | status     | Integer              |   | 0 means deleted; 1 means not deleted       |
| 5                            | createTime | LocalDateTime        |   | Created date                               |
| 6                            | readTime   | LocalDateTime        |   | Read time                                  |
| 7                            | flag       | UsedFlag             |   | YES means accessed; NO means not accessed. |
| 8                            | uid        | String               |   | ID in the Hub                              |
| 9                            | key        | String               |   | Ciphertext key                             |
| 10                           | ownerKey   | String               |   | Owner’s key                                |

#### 11.2.4.9 Query Granted Permissions

| Interface Address            |           | /hub/queryGrantedPermission  |                            |  |
|------------------------------|-----------|--|----------------------------|--|
| Description                  |           | Users can look up all or part of the permission records authorized to them in three dimensions: the uid of the resource owner, whether it has been accessed and the permission type. |                            |  |
| Interface request parameter  |           |  |                            |  |
| No.                          | Parameter | Type   | Required                   | Description  |
| 1                            |           | QueryGrantedPermissionReq  | Y                          | Wrapper class  |
| QueryGrantedPermissionReq    |           |  |                            |  |
| 1                            | uid       | String   | Y                          | ID in the Hub  |
| 2                            | ownerUid  | String   | N                          | Resource owner’s ID in the Hub   |
| 3                            | flag      | String   | N                          | Access status. 0: query accessed permissions; 1: query unaccessed records; if null, query both |
| 4                            | grant     | Sting  | N                          | Authorization type. READ: Read; WRITE: Add; UPDATE: Update                                     |
| 5                            | sign      | String   | Y                          | Signature  |
| Interface response parameter |           |  |                            |  |
| No.                          | Parameter | Type   | Description                |  |
| 1                            |           | List<GrantPermissionInfo>  | List of authorized records |  |

| GrantPermissionInfo |            |         |   |
|---------------------|------------|---------|---|
| 1                   | url        | String  | The path of the resource storage.                             |
| 2                   | grant      | String  | Authorization type.<br>READ: Read; WRITE: Add; UPDATE: Update |
| 3                   | status     | Integer | 0: deleted; 1: not deleted                                    |
| 4                   | createTime | Date    | Time to create authorization                                  |
| 5                   | readTime   | Date    | Accessed time. Null if not accessed.                          |
| 6                   | flag       | Integer | 0: accessed; 1: not accessed.                                 |
| 7                   | ownerUid   | String  | ID in the Hub   |
| 8                   | OwnerKey   | String  | Ciphertext key  |
| 9                   | key        | String  | Owner's key   |

#### 11.2.4.10 Query Operation Record of the Resource

| Interface Address            |               | /hub/getResourceHistory   |          |  |
|------------------------------|---------------|---|----------|--|
| Description                  |               | Resource owner calls this function to query the operation record of the resource. |          |  |
| Interface request parameter  |               |   |          |  |
| No.                          | Parameter     | Type  | Required | Description  |
| 1                            |               | QueryResourceHistoryReq   | Y        | Wrapper class  |
| QueryPermissionReq           |               |   |          |  |
| 1                            | uid           | String  | Y        | ID in the Hub  |
| 2                            | url           | String  | N        | The path of the stored resource.                               |
| 3                            | operation     | String  | N        | Authorization type. WRITE: Add; UPDATE: Update; DELETE: delete |
| 4                            | sign          | String  | Y        | Signature  |
| Interface response parameter |               |   |          |  |
| No.                          | Parameter     | Type  |          | Description  |
| 1                            |               | List<ResourceHistoryInfo>   |          | Resource operation record list                                 |
| PermissionInfo               |               |   |          |  |
| 1                            | OperationUid  | String  |          | Operator's ID in the Hub                                       |
| 2                            | ownerUid      | String  |          | Owner's ID in the Hub  |
| 3                            | operation     | String  |          | Authorization type. WRITE: Add; UPDATE: Update; DELETE: delete |
| 4                            | OperationTime | LocalDateTime   |          | Operation time   |
| 5                            | url           | String  |          | The path of the stored resource.                               |
| 6                            | key           | String  |          | Ciphertext key   |

|   |         |        |                             |
|---|---------|--------|-----------------------------|
| 7 | content | String | Ciphertext resource content |
|---|---------|--------|-----------------------------|

#### 11.2.4.11 Change Resource Owner

| Interface Address            |                   | /hub/transferowner                    |          |   |
|------------------------------|-------------------|---------------------------------------|----------|---|
| Description                  |                   | Change the resource owner in the Hub. |          |   |
| Interface request parameter  |                   |                                       |          |   |
| No.                          | Parameter         | Type                                  | Required | Description                                     |
| 1                            |                   | TransferOwnerReq                      | Y        | Wrapper class                                   |
| QueryPermissionReq           |                   |                                       |          |   |
| 1                            | uid               | String                                | Y        | ID in the Hub                                   |
| 2                            | url               | String                                | Y        | The path of the stored resource.                |
| 3                            | newOwnerUid       | String                                | Y        | New owner’s ID in the Hub                       |
| 4                            | newOwnerPublicKey | String                                | Y        | New owner’s public key                          |
| 5                            | newKey            | String                                | N        | New ciphertext key                              |
| 6                            | sign              | String                                | Y        | Signature                                       |
| Interface response parameter |                   |                                       |          |   |
| No.                          | Parameter         | Type                                  |          | Description                                     |
| 1                            |                   | Boolean                               | Y        | Return true if success, return false if failure |

### 11.3 Response Code

| Response code | Description                                 |
|---------------|---|
| 0             | Success                                     |
| 9999          | Unknown exception                           |
| 1001          | {attribute} is null                         |
| 1002          | The format of {attribute} is invalid        |
| 1003          | {attribute} contains a null attribute value |
| 1004          | {attribute} is too long                     |
| 1005          | Transaction timeout                         |
| 1006          | Transaction error                           |
| 1008          | Config file does not exist                  |
| 1009          | Node private key is empty                   |
| 1010          | DID contract address is empty               |
| 1011          | CPT contract address is empty               |
| 1012          | Auth issuer contract address is empty       |
| 1013          | DID blockchain type is empty                |

|      |   |
|------|---|
| 1014 | Failed to initialize the DID SDK  |
| 1020 | Failed to create the key pair   |
| 1021 | Public and private keys do not match  |
| 1022 | Public key is empty   |
| 1023 | Invalid public key format   |
| 1024 | Private key is empty  |
| 1025 | Invalid private key format  |
| 1027 | Encryption Type is empty  |
| 1028 | Invalid Encryption Type   |
| 1029 | Failed to sign the data   |
| 1030 | Signer and DID do not match   |
| 1031 | Signature verification failed   |
| 1032 | Public key and document's primary public key do not match                   |
| 1033 | Public key and document's recovery public key do not match                  |
| 1040 | DID already exists  |
| 1041 | DID does not exist  |
| 1042 | Failed to create DID  |
| 1043 | Invalid DID   |
| 1044 | Failed to generate the DID  |
| 1045 | Failed to generate the DID Document   |
| 1046 | DID Document verification success   |
| 1050 | DID is registered as the issuer   |
| 1051 | DID is not registered as the issuer   |
| 1052 | Failed to register as the issuer  |
| 1054 | Issuer does not exist   |
| 1060 | CPT does not exist  |
| 1062 | Issuer and publisherDid in the CPT do not match                             |
| 1070 | The credential has been revoked   |
| 1071 | The credential has expired  |
| 1072 | Failed to revoke the credential   |
| 1073 | CPT and credential do not match   |
| 1074 | Failed to create credential   |
| 1075 | Credential verification success   |
| 1076 | The credential is not in the revoke list                                    |
| 1077 | Computed DID from the document is not the same with the DID in the document |
| 1078 | Created time is different with updated time in DID Document                 |
| 1079 | Public key signature verification failed                                    |
| 1080 | DID is not the same with the proof creator in CPT                           |
| 1081 | The DID Document version does not match the one on-chain                    |

|                     |   |
|---------------------|---|
| 1082                | The DID Document created time does not match the one on-chain |
| 1083                | The DID Document recovery key does not match the one on-chain |
| 1084                | Failed to add DID Document to the chain                       |
| 1085                | Failed to create the key pair                                 |
| 1086                | Failed to calculate the DID                                   |
| 1087                | Failed to calculate the DID Document signature                |
| 1088                | Failed to create the DID Document                             |
| 1090                | The mnemonic is empty   |
| 1337                | Failed to encrypt the key                                     |
| 1338                | Failed to sign the data                                       |
| <b>Identity Hub</b> |   |
| 1303                | Private key is empty  |
| 1304                | Private key format is invalid                                 |
| 1305                | Public key is empty   |
| 1306                | Public key format is invalid                                  |
| 1307                | Public key and private key do not match                       |
| 1309                | The content is empty  |
| 1310                | The URL is empty  |
| 1321                | The URL of the Identity Hub cannot be empty                   |
| 1322                | The public key of the Identity Hub cannot be empty            |
| 1327                | Failed to send the request                                    |
| 1328                | The format of the grant is invalid                            |
| 1329                | Grant cannot be empty   |
| 1335                | Resource does not exist                                       |
| 1336                | The key is empty  |
| 1337                | Failed to encrypt the key                                     |
| 1338                | Failed to sign the data                                       |
| 1341                | Failed to delete permission                                   |
| 1342                | Failed to query permission                                    |
| 1343                | Grant is empty  |
| 1344                | Failed to check permission                                    |
| 1347                | Failed to query publicKey                                     |
| 1350                | Config file does not exist                                    |
| 1351                | The public key is empty                                       |
| 1352                | You cannot add permissions to yourself                        |
| 1354                | Illegal flag  |
| 1361                | The user ID is empty  |
| 1362                | The granted user ID is empty                                  |
| 1363                | The public key of the granted user is empty                   |
| 1364                | Failed to generate the user ID                                |



|      |   |
|------|---|
| 1365 | Failed to decrypt the data  |
| 1366 | Failed to encrypt the data  |
| 1367 | Failed to decrypt Identity Hub's private key                      |
| 1368 | The private key of Identity Hub is empty                          |
| 1369 | The public key of Identity Hub is empty                           |
| 1370 | No permission to save resource                                    |
| 1371 | No permission to update resource                                  |
| 1372 | The resource has been saved, cannot be saved again                |
| 1373 | The granted user already has an unused permission                 |
| 1400 | Permission does not exist   |
| 1402 | Failed to delete permission                                       |
| 1403 | Failed to query permission  |
| 1406 | Failed to get the public key                                      |
| 1407 | Failed to update the public key                                   |
| 1408 | Failed to save the resource                                       |
| 1409 | Failed to query the resource                                      |
| 1410 | Failed to delete the resource                                     |
| 1411 | Failed to update the resource                                     |
| 1412 | Failed to get the HTTP request                                    |
| 1413 | Failed to get the private key of the Identity Hub                 |
| 1414 | Missing request data  |
| 1415 | Failed to convert the request parameter                           |
| 1416 | Unknown client data or not connected                              |
| 1417 | Parse return code error   |
| 1418 | Resource does not exist   |
| 1419 | The URL format is invalid   |
| 1422 | Signature verification failed                                     |
| 1423 | The user is not registered  |
| 1424 | User registration failed, the format of the public key is invalid |
| 1425 | The user is already registered                                    |
| 1426 | The granted user does not exist                                   |
| 1427 | The resource owner does not exist                                 |
| 1428 | Failed to close the permission                                    |
| 1429 | Only the resource owner can delete it                             |
| 1430 | Failed to add operation record                                    |
| 1431 | Failed to register the user                                       |
| 1432 | The user ID is empty  |
| 1433 | Failed to parse the request parameter                             |
| 1434 | Failed to add the operation record                                |
| 1435 | Failed to update the resource                                     |

|      |   |
|------|---|
| 1436 | The resource does not exist   |
| 1437 | Failed to save the resource   |
| 1438 | Failed to save the user information to the database                       |
| 1441 | The current permission has been used or the user does not have permission |
| 1442 | Failed to check the permission  |
| 1455 | decrypt encptyKey failed  |
| 1456 | decrypt content failed  |
| 1460 | New owner's user ID is empty  |
| 1461 | Failed to change the owner  |
| 1462 | The uid and the ownerId cannot be the same                                |
| 1463 | The ownerId is not exist  |
| 1465 | The new owner and new owner's public key do not match                     |
| 1501 | Failed to query the encryption key of the granted resource                |
| 1502 | Failed to decrypt the key of the granted resource                         |
| 1503 | The resource does not exist   |
| 1504 | The recovery key pair is incorrect, cannot reset DID authentication       |
| 1505 | The primary private key and public key do not match                       |

*Note: {attribute} is a dynamic parameter.*

## 11.4 SDK

BSN provides a Java version SDK, which implements signature, verification, communication and other methods, so Java developers can quickly make API calls through the SDK.

### 1. Steps to Use

- 1) Download the SDK source code and compile and package it as a jar named did-sdk-1.0.jar;
- 2) Add the did-sdk-1.0.jar to the classpath directory of the project project;
- 3) Create an instance of DidClient:  
`DidClient didClient = new DidClient(URL,PROJECTID,TOKEN);`
- 4) Call the method in the SDK, as follows:  
`DidDataWrapper didData = didClient.createDid(true);`

### 1. Specifications

#### ☐ Timestamp

The format of the time is a string in the form of yyyy-MM-dd HH:mm:ss, for example: 2021-05-25 12:30:59 means May 25, 2021 at 12:30:59.

#### ☐ Exception

A runtime exception "RuntimeException" is thrown when there is a runtime error.

### 2. Function Description

The methods in the SDK can be divided into four categories according to their functional properties: DID, issuer, credential and privacy area, and each method is described below:

### 11.4.1 DID

#### 11.4.1.1 Generate Private and Public Keys by Mnemonics

The user can customize mnemonics and call this function to generate a pair of public-private keys for the k1 algorithm offline. As long as mnemonics are the same, the generated public and private keys must be the same for each call.

| Function name       |            | createKeyPair(List<String> mnemList)                           |          |                |
|---------------------|------------|--|----------|----------------|
| Description         |            | The user can generate the private and public keys by mnimonics |          |                |
| Request Parameters  |            |  |          |                |
| No.                 | Parameter  | Type   | Required | Description    |
| 1                   | mnemList   | List<String>   | Y        | Mnemonics      |
| Response Parameters |            |  |          |                |
| No.                 | Parameter  | Type   | Required | Description    |
| 1                   |            | DidDataWrapper   | Y        | Private key    |
| KeyPair             |            |  |          |                |
| 1                   | privateKey | String   | Y        | Private key    |
| 2                   | publicKey  | String   | Y        | Public key     |
| 3                   | type       | String   | Y        | Algorithm Type |

#### 11.4.1.2 Create DID

| Function name       |                  | createDid(Boolean isStorageOnChain)  |          |   |
|---------------------|------------------|--|----------|---|
| Description         |                  | Call this function to create a DID. isStorageOnChain indicates whether the DID Document is stored on-chain or not. |          |   |
| Request Parameters  |                  |  |          |   |
| No.                 | Parameter        | Type   | Required | Description   |
| 1                   | isStorageOnChain | Boolean  | Y        | On-chain marker. true means DID Document is stored on-chain; false means DID Document is not stored on-chain. |
| Response Parameters |                  |  |          |   |
| No.                 | Parameter        | Type   | Required | Description   |
| 1                   |                  | DidDataWrapper   | Y        |   |

#### DidDataWrapper

| No. | Parameter     | Type         | Required | Description                             |
|-----|---------------|--------------|----------|---|
| 1   | did           | String       | Y        | DID                                     |
| 2   | authPublicKey | KeyPair      | Y        | Primary public/private key information  |
| 3   | recyPublicKey | KeyPair      | Y        | Recovery public/private key information |
| 4   | document      | DocumentInfo | N        | DID Document                            |

|   |         |        |   |                 |
|---|---------|--------|---|-----------------|
| 5 | didSign | String | Y | DID signature   |
| 6 | address | String | Y | Account address |

**DocumentInfo**

| No. | Parameter      | Type      | Required | Description         |
|-----|----------------|-----------|----------|---------------------|
| 1   | did            | String    | Y        | DID                 |
| 2   | version        | String    | Y        | Version             |
| 3   | created        | String    | Y        | Created date        |
| 4   | updated        | String    | Y        | Updated date        |
| 5   | authentication | PublicKey | Y        | Primary public key  |
| 6   | recovery       | PublicKey | Y        | Recovery public key |
| 7   | proof          | Proof     | Y        | Signature           |

**KeyPair**

| No. | Parameter  | Type   | Required | Description    |
|-----|------------|--------|----------|----------------|
| 1   | privateKey | String | Y        | Private key    |
| 2   | publicKey  | String | Y        | Public key     |
| 3   | type       | String | Y        | Algorithm type |

**PublicKey**

| No. | Parameter | Type   | Required | Description    |
|-----|-----------|--------|----------|----------------|
| 1   | type      | String | Y        | Algorithm type |
| 2   | publicKey | String | Y        | Public key     |

**Proof**

| No. | Parameter      | Type   | Required | Description     |
|-----|----------------|--------|----------|-----------------|
| 1   | type           | String | Y        | Algorithm type  |
| 2   | creator        | String | Y        | DID             |
| 3   | signatureValue | String | Y        | Signature value |

**11.4.1.3 Verify DID Document**

| Function name      | verifyDidDocument(DidDocument didDocument)   |             |          |                     |
|--------------------|--|-------------|----------|---------------------|
| Description        | Verify the content format and signature value of the offline generated DID Document. |             |          |                     |
| Request Parameters |  |             |          |                     |
| No.                | Parameter  | Type        | Required | Description         |
| 1                  |  | DidDocument | Y        |                     |
| DidDocument        |  |             |          |                     |
| 1                  | did  | String      | Y        | DID                 |
| 2                  | version  | String      | Y        | Version             |
| 3                  | created  | String      | Y        | Created date        |
| 4                  | updated  | String      | Y        | Updated date        |
| 5                  | authentication   | PublicKey   | Y        | Primary public key  |
| 6                  | recovery   | PublicKey   | Y        | Recovery public key |

| 7                          | proof          | Proof   | Y        | Signature                                       |
|----------------------------|----------------|---------|----------|---|
| <b>PublicKey</b>           |                |         |          |   |
| 1                          | type           | String  | Y        | Algorithm type                                  |
| 2                          | publicKey      | String  | Y        | Public key                                      |
| <b>Proof</b>               |                |         |          |   |
| 1                          | type           | String  | Y        | Algorithm type                                  |
| 2                          | creator        | String  | Y        | DID   |
| 3                          | signatureValue | String  | Y        | Signature value                                 |
| <b>Response Parameters</b> |                |         |          |   |
| No.                        | Parameter      | Type    | Required | Description                                     |
| 1                          |                | Boolean | Y        | Return true if success, return false if failure |

#### 11.4.1.4 Upload DID Document

| Function name       | storeDidDocumentOnChain(DidDocument didDocument)   |             |          |                     |
|---------------------|--|-------------|----------|---------------------|
| Description         | Store the DID document on-chain. Firstly to execute the verification, so that you can call this function if you want to store the DID Document on chain. |             |          |                     |
| Request Parameters  |  |             |          |                     |
| No.                 | Parameter  | Type        | Required | Description         |
| 1                   |  | DidDocument | Y        |                     |
| DidDocument         |  |             |          |                     |
| 1                   | did  | String      | Y        | DID                 |
| 2                   | version  | String      | Y        | Version             |
| 3                   | created  | String      | Y        | Created date        |
| 4                   | updated  | String      | Y        | Updated date        |
| 5                   | authentication   | PublicKey   | Y        | Primary public key  |
| 6                   | recovery   | PublicKey   | Y        | Recovery public key |
| 7                   | proof  | Proof       | Y        | Signature           |
| PublicKey           |  |             |          |                     |
| 1                   | type   | String      | Y        | Algorithm type      |
| 2                   | publicKey  | String      | Y        | Public key          |
| Proof               |  |             |          |                     |
| 1                   | type   | String      | Y        | Algorithm type      |
| 2                   | creator  | String      | Y        | DID                 |
| 3                   | signatureValue   | String      | Y        | Signature value     |
| Response Parameters |  |             |          |                     |
| No.                 | Parameter  | Type        | Required | Description         |
| 1                   |  | Boolean     | Y        | Storage result      |

#### 11.4.1.5 Get DID Document

| Function name | <b>getDidDocument(String did)</b>   |
|---------------|---|
| Description   | The information in the DID Document is a record and description of the DID, and |

|                     |  |             |          |              |
|---------------------|--|-------------|----------|--------------|
|                     | anyone can query the corresponding DID Document from the chain by the DID. It can be used to verify the DID and obtain the DID public key. |             |          |              |
| Request Parameters  |  |             |          |              |
| No.                 | Parameter  | Type        | Required | Description  |
| 1                   | did  | String      | Y        | DID          |
| Response Parameters |  |             |          |              |
| No.                 | Parameter  | Type        | Required | Description  |
| 1                   | didDocument  | DidDocument | Y        | DID Document |

**DidDocument**

| No. | Parameter      | Type      | Required | Description         |
|-----|----------------|-----------|----------|---------------------|
| 1   | did            | String    | Y        | DID                 |
| 2   | version        | String    | Y        | Version             |
| 3   | created        | String    | Y        | Created date        |
| 4   | updated        | String    | Y        | Updated date        |
| 5   | authentication | PublicKey | Y        | Primary public key  |
| 6   | recovery       | PublicKey | Y        | Recovery public key |
| 7   | proof          | Proof     | Y        | Signature           |

**PublicKey**

|   |           |        |   |                |
|---|-----------|--------|---|----------------|
| 1 | type      | String | Y | Algorithm type |
| 2 | publicKey | String | Y | Public key     |

**Proof**

|   |                |        |   |                 |
|---|----------------|--------|---|-----------------|
| 1 | type           | String | Y | Algorithm type  |
| 2 | creator        | String | Y | DID             |
| 3 | signatureValue | String | Y | Signature value |

**11.4.1.6 Verify DID**

| Function name       | verifyDIdSign(String did, String didSign)  |         |          |   |
|---------------------|--|---------|----------|---|
| Description         | Verify the digital signature value of the DID, so that it can ensure the authenticity and validity of the current DID. |         |          |   |
| Request Parameters  |  |         |          |   |
| No.                 | Parameter  | Type    | Required | Description                                     |
| 1                   | did  | String  | Y        | DID   |
| 2                   | didSign  | String  | Y        | DID signature                                   |
| Response Parameters |  |         |          |   |
| No.                 | Parameter  | Type    | Required | Description                                     |
| 1                   |  | Boolean | Y        | Return true if success, return false if failure |

**11.4.1.7 Key Update**

| Function name | resetDidAuth(ResetDidAuth restDidAuth)  |  |  |  |
|---------------|---|--|--|--|
| Description   | If the primary private key is lost or leaked, a pair of primary public and private keys can be regenerated by the recovery private key. The user completes the primary public-private keys update with the recovery public-private keys. After the key is |  |  |  |

|  |  |
|--|--|
|  | updated, the user's DID Document will also be updated, but the DID remains the same. If the user fills in the primary public-private keys, the primary public keys in the DID Document is updated and the signature is recalculated using the filled-in primary public key; otherwise, a new pair of primary public private keys is automatically generated and the primary public key and signature calculation of the DID Document are updated. <i>Note: If the issuer updates the key, all the previously issued credentials will not pass the signature verification (if the issuer records the master public key of the credential in the business system, it can transmit the old master public key information to the user, then it can also pass the credential verification).</i> |
|--|--|

**Request Parameters**

| No. | Parameter | Type         | Required | Description |
|-----|-----------|--------------|----------|-------------|
| 1   |           | ResetDidAuth | Y        |             |

**ResetDidAuth**

|   |                |         |   |                                 |
|---|----------------|---------|---|---------------------------------|
| 1 | did            | String  | Y | DID                             |
| 2 | primaryKeyPair | KeyPair | N | Primary public and private key  |
| 3 | recoveryKey    | KeyPair | Y | Recovery public and private key |

**KeyPair**

|   |            |        |   |                |
|---|------------|--------|---|----------------|
| 1 | privateKey | String | Y | Private Key    |
| 2 | publicKey  | String | Y | Public Key     |
| 3 | type       | String | Y | Algorithm type |

**Response Parameters**

| No. | Parameter | Type    | Required | Description                     |
|-----|-----------|---------|----------|---------------------------------|
| 1   |           | KeyPair | Y        | New public and private key pair |

**KeyPair**

| No. | Parameter  | Type   | Required | Description    |
|-----|------------|--------|----------|----------------|
| 1   | privateKey | String | Y        | Private key    |
| 2   | publicKey  | String | Y        | Public key     |
| 3   | type       | String | Y        | Algorithm type |

**11.4.2 Issuer****11.4.2.1 Register Issuer**

|                         |            |   |          |               |
|-------------------------|------------|---|----------|---------------|
| Function name           |            | registerAuthIssuer(RegisterAuthorityIssuer register)  |          |               |
| Description             |            | The DID user becomes the issuer, and the issuer information is uploaded to the chain if the registration is successful. |          |               |
| Request Parameters      |            |   |          |               |
| No.                     | Parameter  | Type  | Required | Description   |
| 1                       |            | RegisterAuthorityIssuer   | Y        |               |
| RegisterAuthorityIssuer |            |   |          |               |
| 1                       | privateKey | String  | Y        | Private key   |
| 2                       | did        | String  | Y        | DID           |
| 3                       | name       | String  | Y        | Issuer’s name |
| Response Parameters     |            |   |          |               |

| No. | Parameter | Type    | Required | Description                                     |
|-----|-----------|---------|----------|---|
| 1   |           | Boolean | Y        | Return true if success, return false if failure |

#### 11.4.2.2 Query Issuer

|                     |  |                        |          |                                   |
|---------------------|--|------------------------|----------|-----------------------------------|
| Function name       | queryAuthIssuerList(AuthIssuerList query)  |                        |          |                                   |
| Description         | You can query whether it is the issuer through DID and identify the type of credential that can be issued by name. |                        |          |                                   |
| Request Parameters  |  |                        |          |                                   |
| No.                 | Parameter  | Type                   | Required | Description                       |
| 1                   |  | AuthIssuerList         | Y        |                                   |
| AuthIssuerList      |  |                        |          |                                   |
| 1                   | page   | Integer                | Y        | Number of pages                   |
| 2                   | size   | Integer                | Y        | Number of entries per page        |
| 3                   | did  | String                 | Y        | DID                               |
| Response Parameters |  |                        |          |                                   |
| No.                 | Parameter  | Type                   | Required | Description                       |
| 1                   |  | Pages<AuthorityIssuer> | Y        | Query result, the list of issuers |

#### Pages

| No. | Parameter | Type                   | Required | Description     |
|-----|-----------|------------------------|----------|-----------------|
| 1   | page      | Integer                | Y        | Page number     |
| 2   | size      | Integer                | Y        | Paging Size     |
| 3   | totalNum  | Integer                | Y        | Total number    |
| 4   | totalPage | Integer                | Y        | Total pages     |
| 5   | result    | List< AuthorityIssuer> | Y        | List of issuers |

#### AuthorityIssuer

| No. | Parameter | Type   | Required | Description   |
|-----|-----------|--------|----------|---------------|
| 1   | did       | String | Y        | DID           |
| 2   | name      | String | Y        | Issuer's name |

#### 11.4.2.3 Register credential template

|                    |            |   |          |             |
|--------------------|------------|---|----------|-------------|
| Function name      |            | registerCpt(RegisterCpt registerCpt)  |          |             |
| Description        |            | The issuer customizes the credential template and can agree on which attribute values must be provided by the applicant. For example, in the template of college diploma, you can agree that "name" and "student number" are mandatory information. |          |             |
| Request Parameters |            |   |          |             |
| No.                | Parameter  | Type  | Required | Description |
| 1                  |            | RegisterCpt   | Y        |             |
| RegisterCpt        |            |   |          |             |
| 1                  | did        | String  | Y        | DID         |
| 2                  | privateKey | String  | Y        | Private key |



| 3                          | cptJsonSchema | Map<String, JsonSchema> | Y        | JsonSchema of credential template                             |
|----------------------------|---------------|-------------------------|----------|---|
| 4                          | title         | String                  | Y        | Title   |
| 5                          | description   | String                  | Y        | Description   |
| 6                          | type          | String                  | Y        | Credential Type, fill in Proof                                |
| 7                          | cptId         | Long                    | Y        | Credential template ID  |
| <b>JsonSchema</b>          |               |                         |          |   |
| 1                          | type          | String                  | Y        | Field type  |
| 2                          | description   | String                  | Y        | Field description   |
| 3                          | required      | Boolean                 | Y        | true: required; false: optional                               |
| <b>Response Parameters</b> |               |                         |          |   |
| No.                        | Parameter     | Type                    | Required | Description   |
| 1                          |               | CptBaseInfo             | Y        | Registration result, basic information of credential template |

**CptBaseInfo**

| No. | Parameter  | Type    | Required | Description                 |
|-----|------------|---------|----------|-----------------------------|
| 1   | cptId      | Long    | Y        | Credential template ID      |
| 2   | cptVersion | Integer | Y        | Credential template Version |

**11.4.2.4 Query Credential Template List**

| Function name       |           | queryCptListByDid(QueryCptList query)  |          |  |
|---------------------|-----------|--|----------|--|
| Description         |           | Anyone can check all their credential templates by DID. It is possible for the same individual/organization to register multiple credential templates. For example, a university may have a degree template, an incomplete template, etc. in addition to a diploma template. |          |  |
| Request Parameters  |           |  |          |  |
| No.                 | Parameter | Type   | Required | Description  |
| 1                   |           | QueryCpt   | Y        |  |
| QueryCpt            |           |  |          |  |
| 1                   | page      | Integer  | Y        | Number of pages                                    |
| 2                   | size      | Integer  | Y        | Number of entries per page                         |
| 3                   | did       | String   | Y        | DID  |
| Response Parameters |           |  |          |  |
| No.                 | Parameter | Type   | Required | Description  |
| 1                   |           | Pages<CptInfo>   | Y        | Query result, credential template information list |

**Pages**

| No. | Parameter | Type    | Required | Description  |
|-----|-----------|---------|----------|--------------|
| 1   | page      | Integer | Y        | Page number  |
| 2   | size      | Integer | Y        | Paging Size  |
| 3   | totalNum  | Integer | Y        | Total number |
| 4   | totalPage | Integer | Y        | Total pages  |

|   |        |               |   |                              |
|---|--------|---------------|---|------------------------------|
| 5 | result | List<CptInfo> | Y | List of credential templates |
|---|--------|---------------|---|------------------------------|

**CptInfo**

| No. | Parameter     | Type                    | Required | Description                        |
|-----|---------------|-------------------------|----------|------------------------------------|
| 1   | cptJsonSchema | Map<String, JsonSchema> | Y        | JsonSchema for Credential template |
| 2   | title         | String                  | Y        | Title                              |
| 3   | description   | String                  | Y        | Description                        |
| 4   | publisherDid  | String                  | Y        | DID to create credential template  |
| 5   | proof         | Proof                   | Y        | Signature                          |
| 6   | create        | String                  | Y        | Created date                       |
| 7   | update        | String                  | Y        | Updated date                       |
| 8   | cptId         | Long                    | Y        | Credential template ID             |
| 9   | cptVersion    | Integer                 | Y        | Credential template version        |

**Proof**

| No. | Parameter      | Type   | Required | Description     |
|-----|----------------|--------|----------|-----------------|
| 1   | type           | String | Y        | Algorithm type  |
| 2   | creator        | String | Y        | DID             |
| 3   | signatureValue | String | Y        | Signature value |

**JsonSchema**

| No. | Parameter   | Type    | Required | Description                     |
|-----|-------------|---------|----------|---------------------------------|
| 1   | type        | String  | Y        | Type                            |
| 2   | description | String  | Y        | Description                     |
| 3   | required    | Boolean | Y        | true: required; false: optional |

**11.4.2.5 Query Credential Template**

| Function name       |           | queryCptById(Long cptId)  |          |   |
|---------------------|-----------|---|----------|---|
| Description         |           | Query the contents of a specific credential template by its ID. |          |   |
| Request Parameters  |           |   |          |   |
| No.                 | Parameter | Type  | Required | Description                                   |
| 1                   | cptId     | Long  | Y        | Credential template ID                        |
| Response Parameters |           |   |          |   |
| No.                 | Parameter | Type  | Required | Description                                   |
| 1                   |           | CptInfo   | Y        | Query result, credential template information |

**CptInfo**

| No. | Parameter     | Type                    | Required | Description                           |
|-----|---------------|-------------------------|----------|---------------------------------------|
| 1   | cptJsonSchema | Map<String, JsonSchema> | Y        | JsonSchema for Credential template    |
| 2   | title         | String                  | Y        | Title                                 |
| 3   | description   | String                  | Y        | Description                           |
| 4   | publisherDid  | String                  | Y        | DID to create the credential template |

|                   |             |         |   |                                 |
|-------------------|-------------|---------|---|---------------------------------|
| 5                 | proof       | Proof   | Y | Signature                       |
| 6                 | create      | String  | Y | Created date                    |
| 7                 | update      | String  | Y | Updated date                    |
| 8                 | cptId       | Long    | Y | Credential template ID          |
| 9                 | cptVersion  | Integer | Y | Credential template version     |
| <b>JsonSchema</b> |             |         |   |                                 |
| 1                 | type        | String  | Y | Type                            |
| 2                 | description | String  | Y | Description                     |
| 3                 | required    | Boolean | Y | true: required; false: optional |

**Proof**

| No. | Parameter      | Type   | Required | Description     |
|-----|----------------|--------|----------|-----------------|
| 1   | type           | String | Y        | Algorithm type  |
| 2   | creator        | String | Y        | DID             |
| 3   | signatureValue | String | Y        | Signature value |

**11.4.2.6 Update Credential Template**

|                     |   |                         |          |   |
|---------------------|---|-------------------------|----------|---|
| Function name       | updateCpt(RegisterCpt registerCpt)  |                         |          |   |
| Description         | The issuer updates the content of its own registered credential templates. The update of the credential template ID does not affect issued credentials. |                         |          |   |
| Request Parameters  |   |                         |          |   |
| No.                 | Parameter   | Type                    | Required | Description   |
| 1                   |   | RegisterCpt             | Y        |   |
| RegisterCpt         |   |                         |          |   |
| 1                   | did   | String                  | Y        | DID   |
| 2                   | privateKey  | String                  | Y        | Private key   |
| 3                   | cptJsonSchema   | Map<String, JsonSchema> | Y        | JsonSchema for Credential template                      |
| 4                   | title   | String                  | Y        | Credential template title                               |
| 5                   | description   | String                  | Y        | Credential template description                         |
| 6                   | type  | String                  | Y        | Credential Type, fill in proof                          |
| 7                   | cptId   | Long                    | Y        | Credential template ID                                  |
| JsonSchema          |   |                         |          |   |
| 1                   | type  | String                  | Y        | Type  |
| 2                   | description   | String                  | Y        | Description   |
| 3                   | required  | Boolean                 | Y        | true: required; false: optional                         |
| Response Parameters |   |                         |          |   |
| No.                 | Parameter   | Type                    | Required | Description   |
| 1                   |   | CptBaseInfo             | Y        | Update result, basic information of credential template |

**CptBaseInfo**

| No. | Parameter | Type | Required | Description |
|-----|-----------|------|----------|-------------|
|-----|-----------|------|----------|-------------|

|   |            |         |   |   |
|---|------------|---------|---|---|
| 1 | cptId      | Long    | Y | Credential template ID  |
| 2 | cptVersion | Integer | Y | Credential template version, add 1 after each successful update |

### 11.4.3 Credential

#### 11.4.3.1 Create Credential

| Function name       | createCredential(CreateCredential createCredential)  |                     |          |  |
|---------------------|--|---------------------|----------|--|
| Description         | The attribute values defined in the credential template are provided by the issuer for the DID user to obtain on the front page. The issuer issues the credentials for the DID user through this interface. If there are more Claim parameters than defined in the credential template, the server side will discard them. |                     |          |  |
| Request Parameters  |  |                     |          |  |
| No.                 | Parameter  | Type                | Required | Description  |
| 1                   |  | CreateCredential    | Y        |  |
| CreateCredential    |  |                     |          |  |
| 1                   | cptId  | Long                | Y        | Credential template ID   |
| 2                   | issuerDid  | String              | Y        | DID of the credential template issuer  |
| 3                   | userDid  | String              | Y        | DID of the user who created the credentials  |
| 4                   | expirationDate   | String              | Y        | Credential expiration date. Should be greater than today. In the form of yyyy-mm-dd                    |
| 5                   | claim  | Map<String, Object> | Y        | Content of the credential. The claim data needs to correspond to the format of the credential template |
| 6                   | type   | String              | Y        | Credential type, input Proof   |
| 7                   | privateKey   | String              | Y        | Private key  |
| 8                   | shortDesc  | String              | N        | Brief description of the credential. The default value is the credential template title.               |
| 9                   | longDesc   | String              | N        | Detailed description of the credential   |
| Response Parameters |  |                     |          |  |
| No.                 | Parameter  | Type                | Required | Description  |
| 1                   |  | CredentialWrapper   | Y        | Creation result, Credential information  |

#### CredentialWrapper

| No. | Parameter | Type   | Required | Description            |
|-----|-----------|--------|----------|------------------------|
| 1   | context   | String | Y        | Version                |
| 2   | id        | String | Y        | Credential ID          |
| 3   | type      | String | Y        | Credential type, Proof |

|    |                |                     |   |   |
|----|----------------|---------------------|---|---|
| 4  | cptId          | Long                | Y | Credential template Id                      |
| 5  | issuerDid      | String              | Y | DID of the credential template issuer       |
| 6  | userDid        | String              | Y | DID of the user who created the credentials |
| 7  | expirationDate | String              | Y | Credential expiration date                  |
| 8  | created        | String              | Y | Created date                                |
| 9  | shortDesc      | String              | Y | Brief description of the credential         |
| 10 | longDesc       | String              | N | Detailed description of the credential      |
| 11 | claim          | Map<String, Object> | Y | Claim data                                  |
| 12 | proof          | Map<String, Object> | Y | Signature                                   |

#### 11.4.3.2 Verify Credential

| Function name      | verifyCredential(CredentialWrapper createCredential,PublicKey publicKey)   |                     |          |   |
|--------------------|--|---------------------|----------|---|
| Description        | Generally called by the verifier. It can verify whether a particular credential is valid or not. Verify the signature of the credential, whether the credential is expired, and whether the credential is revoked, respectively. |                     |          |   |
| Request Parameters |  |                     |          |   |
| No.                | Parameter  | Type                | Required | Description                                 |
| 1                  | createCredential   | CredentialWrapper   | Y        |   |
| 2                  | publicKey  | PublicKey           | Y        | Public key                                  |
| CredentialWrapper  |  |                     |          |   |
| 1                  | context  | String              | Y        | Version                                     |
| 2                  | id   | String              | Y        | Credential ID                               |
| 3                  | type   | String              | Y        | Credential type, Proof                      |
| 4                  | cptId  | Long                | Y        | Credential template ID                      |
| 5                  | issuerDid  | String              | Y        | DID of the credential template issuer       |
| 6                  | userId   | String              | Y        | DID of the user who created the credentials |
| 7                  | expirationDate   | String              | Y        | Credential expiration date                  |
| 8                  | created  | String              | Y        | Created date                                |
| 9                  | shortDesc  | String              | N        | Brief description of the credential         |
| 10                 | longDesc   | String              | N        | Detailed description of the credential      |
| 11                 | claim  | Map<String, Object> | Y        | Claim data                                  |
| 12                 | proof  | Map<String, Object> | Y        | Signature                                   |
| PublicKey          |  |                     |          |   |

| 1                          | type      | String  | Y        | Algorithm type                                  |
|----------------------------|-----------|---------|----------|---|
| 2                          | publicKey | String  | Y        | Public key                                      |
| <b>Response Parameters</b> |           |         |          |   |
| No.                        | Parameter | Type    | Required | Description                                     |
| 1                          |           | Boolean | Y        | Return true if success, return false if failure |

#### 11.4.3.3 Revoke Credential

|                     |   |                  |          |   |
|---------------------|---|------------------|----------|---|
| Function name       | revokeCredential(RevokeCredential cred)   |                  |          |   |
| Description         | Called by the issuer to revoke or void a credential that has been issued. Since the issued credential is already in the custody of the user, the revocation of the credential is followed by the upload of its credential ID. |                  |          |   |
| Request Parameters  |   |                  |          |   |
| No.                 | Parameter   | Type             | Required | Description                                     |
| 1                   |   | RevokeCredential | Y        |   |
| RevokeCredential    |   |                  |          |   |
| 1                   | credId  | String           | Y        | Credential ID                                   |
| 2                   | cptId   | Long             | Y        | Credential template Id                          |
| 3                   | did   | String           | Y        | DID   |
| 4                   | privateKey  | String           | Y        | Private key                                     |
| Response Parameters |   |                  |          |   |
| No.                 | Parameter   | Type             | Required | Description                                     |
| 1                   |   | Boolean          | Y        | Return true if success, return false if failure |

#### 11.4.3.4 Query Revoked Credential

|                     |   |                       |          |   |
|---------------------|---|-----------------------|----------|---|
| Function name       | getRevokedCredList(QueryCredentialList queryCredentialList)                                 |                       |          |   |
| Description         | Called when verifying credentials. Find out all its revoked credential IDs by Issuer's DID. |                       |          |   |
| Request Parameters  |   |                       |          |   |
| No.                 | Parameter   | Type                  | Required | Description                                     |
| 1                   |   | QueryCredential       | Y        |   |
| QueryCredential     |   |                       |          |   |
| 1                   | page  | Integer               | Y        | Number of pages                                 |
| 2                   | size  | Integer               | Y        | Number of entries per page                      |
| 3                   | did   | String                | Y        | DID   |
| Response Parameters |   |                       |          |   |
| No.                 | Parameter   | Type                  | Required | Description                                     |
| 1                   |   | Pages<BaseCredential> | Y        | Query result, basic info list of the credential |

**Pages**

| No. | Parameter | Type                 | Required | Description               |
|-----|-----------|----------------------|----------|---------------------------|
| 1   | page      | Integer              | Y        | Page number               |
| 2   | size      | Integer              | Y        | Paging Size               |
| 3   | totalNum  | Integer              | Y        | Total number              |
| 4   | totalPage | Integer              | Y        | Total pages               |
| 5   | result    | List<BaseCredential> | Y        | List of revoked documents |

**BaseCredential**

| No. | Parameter | Type   | Required | Description   |
|-----|-----------|--------|----------|---------------|
| 1   | id        | String | Y        | Credential ID |
| 2   | created   | String | Y        | Revoked time  |

**11.4.4 Identity Hub****11.4.4.1 Register Hub User by DID**

| Function name       |           | registerHub(String did)   |          |                     |
|---------------------|-----------|---|----------|---------------------|
| Description         |           | Register by DID, if successful, returns the user's ID in the Hub. |          |                     |
| Request Parameters  |           |   |          |                     |
| No.                 | Parameter | Type  | Required | Description         |
| 1                   | did       | String  | Y        | DID                 |
| Response Parameters |           |   |          |                     |
| No.                 | Parameter | Type  | Required | Description         |
| 1                   |           | RegisterHubResult   | Y        | Registration result |

**RegisterHubResult**

| No. | Parameter | Type    | Required | Description                                     |
|-----|-----------|---------|----------|---|
| 1   | success   | Boolean | Y        | Return true if success, return false if failure |
| 2   | uid       | String  | Y        | ID in the Hub                                   |
| 3   | message   | String  | Y        | Result description                              |

**11.4.4.2 Register Hub User by Public Key**

|                    |  |                   |          |                     |
|--------------------|--|-------------------|----------|---------------------|
| Function name      | registerHub(String id, String publicKey, CryptoType cryptoType)  |                   |          |                     |
| Description        | Register by Public key. This function allows users define the ID by themselves. If successful, returns the user's ID in the Hub. |                   |          |                     |
| Request Parameters |  |                   |          |                     |
| No.                | Parameter  | Type              | Required | Description         |
| 1                  | id   | String            | N        | Self-defined hub ID |
| 2                  | publicKey  | String            | Y        | Public key          |
| 3                  | CryptoType   | String            | Y        | ECDSA               |
| Response parameter |  |                   |          |                     |
| No.                | Parameter  | Type              | Required | Description         |
| 1                  |  | RegisterHubResult | Y        | Registration result |

| RegisterHubResult |         |         |   |
|-------------------|---------|---------|---|
| 1                 | success | Boolean | Return true if success, return false if failure |
| 2                 | uid     | String  | ID in the Hub                                   |
| 3                 | message | String  | Result description                              |

#### 11.4.4.3 Save Resource

| Function name       |            | saveResource(SaveResource saveResource)  |          |  |
|---------------------|------------|--|----------|--|
| Description         |            | Store the resource to the Hub. If the user stores it himself, the uid and the ownerId should be the same. At this time, there is no create permission need and can call directly; if the issuer stores it for the user after issuing the credentials, the uid should be the issuer, and the ownerId should be the user. At this time, the user must have created WRITE permission for it, otherwise the storage will fail. |          |  |
| Request Parameters  |            |  |          |  |
| No.                 | Parameter  | Type   | Required | Description  |
| 1                   |            | SaveResource   | Y        |  |
| SaveResource        |            |  |          |  |
| 1                   | did        | String   | Y        | ID in the Hub  |
| 2                   | content    | String   | Y        | Resource content   |
| 3                   | url        | String   | N        | The path of the stored resource. Null when stored by the user; required when stored by the issuer. |
| 4                   | ownerUid   | String   | Y        | Resource owner’s ID in the Hub   |
| 5                   | grant      | Operation  | Y        | Operation permissions: WRITE/UPDATE  |
| 6                   | privateKey | String   | Y        | Private key  |
| Response Parameters |            |  |          |  |
| No.                 | Parameter  | Type   | Required | Description  |
| 1                   |            | SaveResourceResult   | Y        | Save result  |

#### SaveResourceResult

| No. | Parameter  | Type   | Required | Description                      |
|-----|------------|--------|----------|----------------------------------|
| 1   | url        | String | Y        | The path of the stored resource. |
| 2   | encryptKey | String | Y        | Ciphertext key                   |

#### 11.4.4.4 Get Resource

| Function name      | getResource(String did,String privateKey, String url)  |        |          |                                  |
|--------------------|--|--------|----------|----------------------------------|
| Description        | Access the Hub and read the specified resource. Users can directly access, third parties need to obtain authorization from the user to access. |        |          |                                  |
| Request Parameters |  |        |          |                                  |
| No.                | Parameter  | Type   | Required | Description                      |
| 1                  | uid  | String | Y        | ID in the Hub                    |
| 2                  | privateKey   | String | Y        | Private key                      |
| 3                  | url  | String | Y        | The path of the stored resource. |



| Response Parameters |           |                   |          |              |
|---------------------|-----------|-------------------|----------|--------------|
| No.                 | Parameter | Type              | Required | Description  |
| 1                   |           | QueryResourceResp | Y        | Query result |

**QueryResourceResp**

| No. | Parameter | Type   | Required | Description                 |
|-----|-----------|--------|----------|-----------------------------|
| 1   | content   | String | Y        | Ciphertext resource content |
| 2   | key       | String | Y        | Ciphertext key              |

**11.4.4.5 Delete Resource**

|                     |  |         |          |   |
|---------------------|--|---------|----------|---|
| Function name       | deleteResource(String did,String privateKey, String url)                           |         |          |   |
| Description         | The resource owner can call this function to delete a certain resource in the Hub. |         |          |   |
| Request Parameters  |  |         |          |   |
| No.                 | Parameter  | Type    | Required | Description                                     |
| 1                   | uid  | String  | Y        | ID in the Hub                                   |
| 2                   | privateKey   | String  | Y        | Private key                                     |
| 3                   | url  | String  | Y        | The path of the stored resource.                |
| Response Parameters |  |         |          |   |
| No.                 | Parameter  | Type    | Required | Description                                     |
| 1                   |  | Boolean | Y        | Return true if success, return false if failure |

**11.4.4.6 Create Permissions**

| Function name      | createPermission(CreatePermission createPermission)  |                  |          |  |
|--------------------|--|------------------|----------|--|
| Description        | Resource owner creates the permissions of accessing to the resource in the Hub for third parties. Permissions WRITE means store resource, UPDATE means update resource, READ means read resource. An authorization can only be accessed once, and an authorization with the same uid and the same permission cannot be created again without access. However, the permissions of UPDATE and READ from the server side will return the same result for each call. |                  |          |  |
| Request Parameters |  |                  |          |  |
| No.                | Parameter  | Type             | Required | Description  |
| 1                  |  | CreatePermission | Y        |  |
| CreatePermission   |  |                  |          |  |
| 1                  | uid  | String           | Y        | ID in the Hub  |
| 2                  | url  | String           | N        | The path of the stored resource. Can be null if the permission is WRITE, required when the permission is READ/UPDATE |
| 3                  | grant  | Operation        | Y        | Operation permissions: WRITE/UPDATE/READ   |
| 4                  | grantUid   | String           | Y        | Authorized ID in the Hub   |
| 5                  | grantPublicKey   | String           | Y        | Authorized public key  |
| 6                  | privateKey   | String           | Y        | Private key  |

| Response Parameters |           |                      |          |                 |
|---------------------|-----------|----------------------|----------|-----------------|
| No.                 | Parameter | Type                 | Required | Description     |
| 1                   |           | CreatePermissionResp | Y        | Creation result |

**CreatePermissionResp**

| No. | Parameter | Type   | Required | Description                      |
|-----|-----------|--------|----------|----------------------------------|
| 1   | url       | String | Y        | The path of the stored resource. |
| 2   | key       | String | Y        | Ciphertext key                   |

**11.4.4.7 Delete Permissions**

| Function name | <b>deletePermission(DeletePermission deletePermission)</b>   |
|---------------|--|
| Description   | Resource owner calls this function to delete permissions that are not yet accessed by a third party. |

**Request Parameters**

| No. | Parameter | Type             | Required | Description |
|-----|-----------|------------------|----------|-------------|
| 1   |           | DeletePermission | Y        |             |

**DeletePermission**

|   |            |           |   |  |
|---|------------|-----------|---|--|
| 1 | uid        | String    | Y | ID in the Hub                            |
| 2 | url        | String    | Y | The path of the stored resource.         |
| 3 | grantUid   | String    | Y | Authorized ID in the Hub                 |
| 4 | grant      | Operation | Y | Operation permissions: WRITE/UPDATE/READ |
| 5 | privateKey | String    | Y | Private key                              |

**Response Parameters**

| No. | Parameter | Type                 | Required | Description     |
|-----|-----------|----------------------|----------|-----------------|
| 1   |           | DeletePermissionResp | Y        | Deletion result |

**DeletePermissionResp**

| No. | Parameter | Type    | Required | Description                                     |
|-----|-----------|---------|----------|---|
| 1   | success   | Boolean | Y        | Return true if success, return false if failure |
| 2   | message   | String  | Y        | Result description                              |

**11.4.4.8 Query Permissions**

| Function name | <b>queryPermission(QueryPermission queryPermission)</b>                            |
|---------------|--|
| Description   | Resource owner can call this function to query Permissions that have been created. |

**Request Parameters**

| No. | Parameter | Type            | Required | Description |
|-----|-----------|-----------------|----------|-------------|
| 1   |           | QueryPermission | Y        |             |

**QueryPermission**

|   |          |          |   |  |
|---|----------|----------|---|--|
| 1 | uid      | String   | Y | ID in the Hub                                |
| 2 | grantUid | String   | N | Authorized ID in the Hub                     |
| 3 | flag     | UsedFlag | N | Access flag. YES: accessed; NO: Not accessed |

| 4                          | privateKey | String               | Y        | Private key                                  |
|----------------------------|------------|----------------------|----------|--|
| <b>Response Parameters</b> |            |                      |          |  |
| No.                        | Parameter  | Type                 | Required | Description                                  |
| 1                          |            | List<PermissionInfo> | Y        | Query result, list of Permissions            |
| <b>PermissionInfo</b>      |            |                      |          |  |
| 1                          | uid        | String               | Y        | ID in the Hub                                |
| 2                          | grantUid   | String               | Y        | Authorized ID in the Hub                     |
| 3                          | url        | String               | Y        | The path of the stored resource.             |
| 4                          | grant      | String               | Y        | Operation permissions: WRITE/UPDATE/READ     |
| 5                          | createTime | LocalDateTime        | Y        | Authorization created time                   |
| 6                          | readTime   | LocalDateTime        | N        | Authorization used time                      |
| 7                          | flag       | UsedFlag             | Y        | Access flag. YES: accessed; NO: Not accessed |
| 8                          | status     | Integer              | Y        | Delete flag. 0: Deleted; 1: Not deleted      |
| 9                          | key        | String               | N        | Ciphertext key                               |
| 10                         | ownerKey   | String               | N        | Owner's key                                  |

#### 11.4.4.9 Query Granted Permissions

| Function name          | queryGrantedPermission(QueryGrantedPermission queryPermission)   |                           |          |  |
|------------------------|--|---------------------------|----------|--|
| Description            | Users can look up all or part of the permission records authorized to them in three dimensions: the uid of the resource owner, whether it has been accessed and the permission type. |                           |          |  |
| Request Parameters     |  |                           |          |  |
| No.                    | Parameter  | Type                      | Required | Description                                  |
| 1                      |  | QueryGrantedPermission    | Y        |  |
| QueryGrantedPermission |  |                           |          |  |
| 1                      | uid  | String                    | Y        | ID in the Hub                                |
| 2                      | grantUid   | String                    | N        | Resource owner’s ID in the Hub               |
| 3                      | grant  | Operation                 | N        | Operation permissions: WRITE/UPDATE/READ     |
| 4                      | flag   | UsedFlag                  | N        | Access flag. YES: accessed; NO: Not accessed |
| 5                      | privateKey   | String                    | Y        | Private key                                  |
| Response Parameters    |  |                           |          |  |
| No.                    | Parameter  | Type                      | Required | Description                                  |
| 1                      |  | List<GrantPermissionInfo> | Y        | Query result, list of Permissions            |
| GrantPermissionInfo    |  |                           |          |  |

|   |            |               |   |  |
|---|------------|---------------|---|--|
| 1 | url        | String        | Y | The path of the stored resource.             |
| 2 | grant      | String        | Y | Operation permissions: WRITE/UPDATE/READ     |
| 3 | status     | Integer       | Y | Delete flag. 0: Deleted; 1: Not deleted      |
| 4 | createTime | LocalDateTime | Y | Authorization created time                   |
| 5 | readTime   | LocalDateTime | N | Authorization used time                      |
| 6 | flag       | UsedFlag      | Y | Access flag. YES: accessed; NO: Not accessed |
| 7 | ownerUid   | String        | Y | Resource owner's ID in the Hub               |
| 8 | key        | String        | N | Ciphertext key                               |
| 9 | ownerKey   | String        | N | Owner's key                                  |

#### 11.4.4.10 Query Resource Operation Record

|                      |   |                           |          |   |
|----------------------|---|---------------------------|----------|---|
| Function name        | queryResourceHistory(QueryResourceHistory queryResourceHistory)                   |                           |          |   |
| Description          | Resource owner calls this function to query the operation record of the resource. |                           |          |   |
| Request Parameters   |   |                           |          |   |
| No.                  | Parameter   | Type                      | Required | Description                             |
| 1                    |   | QueryResourceHistory      | Y        |   |
| QueryResourceHistory |   |                           |          |   |
| 1                    | uid   | String                    | Y        | ID in the Hub                           |
| 2                    | url   | String                    | N        | The path of the stored resource.        |
| 3                    | operation   | Operation                 | N        | Authorization type: WRITE/UPDATE/DELETE |
| 4                    | privateKey  | String                    | Y        | Private key                             |
| Response Parameters  |   |                           |          |   |
| No.                  | Parameter   | Type                      | Required | Description                             |
| 1                    |   | List<ResourceHistoryInfo> | Y        | Query result, list of Permissions       |
| ResourceHistoryInfo  |   |                           |          |   |
| 1                    | operationUid  | String                    | Y        | Operator’s ID in the Hub                |
| 2                    | ownerUid  | String                    | Y        | Owner’s ID in the Hub                   |
| 3                    | operation   | String                    | Y        | Operation type: WRITE/UPDATE/READ       |
| 4                    | content   | String                    | Y        | Ciphertext resource content             |
| 5                    | url   | String                    | Y        | The path of the stored resource.        |
| 6                    | key   | String                    | N        | Ciphertext key                          |
| 7                    | operationTime   | LocalDateTime             | Y        | Operated time                           |

**11.4.4.11 Change Resource Owner**

|                     |                   |  |          |   |
|---------------------|-------------------|--|----------|---|
| Function name       |                   | transferOwner(TransferOwner transferOwner) |          |   |
| Description         |                   | Change the resource owner in the Hub.      |          |   |
| Request Parameters  |                   |  |          |   |
| No.                 | Parameter         | Type                                       | Required | Description                                     |
| 1                   |                   | TransferOwner                              | Y        |   |
| TransferOwner       |                   |  |          |   |
| 1                   | uid               | String                                     | Y        | ID in the Hub                                   |
| 2                   | url               | String                                     | Y        | The path of the stored resource.                |
| 3                   | newOwnerUid       | String                                     | Y        | New owner’s ID in the Hub                       |
| 4                   | newOwnerPublicKey | String                                     | Y        | New owner’s public key                          |
| 5                   | privateKey        | String                                     | Y        | Private key                                     |
| Response Parameters |                   |  |          |   |
| No.                 | Parameter         | Type                                       | Required | Description                                     |
| 1                   |                   | Boolean                                    | Y        | Return true if success, return false if failure |

**11.4.4.12 Decrypt Resource**

|                     |            |  |          |                             |
|---------------------|------------|--|----------|-----------------------------|
| Function name       |            | decrypt(String content, String encptyKey, String privateKey)   |          |                             |
| Description         |            | Decrypt the ciphertext resource content returned from the Get Resource interface using the ciphertext key returned from the Get Resource interface to get the plaintext resource content |          |                             |
| Request Parameters  |            |  |          |                             |
| No.                 | Parameter  | Type   | Required | Description                 |
| 1                   | content    | String   | Y        | Ciphertext resource content |
| 2                   | encptyKey  | String   | Y        | Ciphertext key              |
| 3                   | privateKey | String   | Y        | Private key                 |
| Response Parameters |            |  |          |                             |
| No.                 | Parameter  | Type   | Required | Description                 |
| 1                   |            | String   | Y        | Plaintext resource content  |

## 12 Account Management

In the **My Account** page, the user can view details of their card and transactions they performed on the network. To work with **My Account**, follow these steps:

1. In the **User Center** menu, click the dropdown to reveal the list, in the menu list, click **My Account** to display the page.
2. To update the user **Card Information**, click the **Update card information** to display the **My Credit Card** page. The user will be redirected to the Stripe website. The BSN portal can never see and does not store credit card information.
3. Update the card details as needed and click **Update**.

Card number MM / YY CVC

\*Only the last 4 digits of the credit card number are retained on this website, and the rest of the information is not retained.

Update Go Back

4. To search a bill in the **My Bills** section, enter or select the following:
  - **Bill Number** - Enter the bill number if known
  - **Created Date** - Select a start and end date
  - **Service Name** - Enter a service name if known
  - **Status** - Select from the options available in the dropdown
  - **Bill Type** - Select from the options available in the dropdown
  - Click **Search** to display the bill information.

My Bills

Bill Number Service Name Bill Type All

Created Date Start ~ End Status All Search Reset

5. In the **Bill list**, under the **Status** and **Action** columns, the user can perform certain actions including **Pay** and **Details** on each bill. To **pay** a bill, click **Pay** and to **View** a bill, click **Details**.

| Bill Number                   | Service Name       | Bill Type       | Total Amount (USD) | Payment Amount (USD) | Created Date                   | Status         | Action                  |
|-------------------------------|--------------------|-----------------|--------------------|----------------------|--------------------------------|----------------|-------------------------|
| 5F39EA142CC44C9E8CF9D49E01... | WineTrace          | Data Usage      | 0.00               | 0.00                 | (UTC+8:00) 07/21/2021 02:00:05 | Paid           | <a href="#">Details</a> |
| 33B3753E168E469FA77F1EC73...  | WineTrace          | Service Publish | 860.03             | 0.00                 | (UTC+8:00) 07/14/2021 10:30:40 | Partial Refund | <a href="#">Details</a> |
| 4E34B5D582E34AB8ADF5F7052D... | WineTrace          | Service Publish | 863.35             | 0.00                 | (UTC+8:00) 06/28/2021 14:34:42 | Expired        | <a href="#">Details</a> |
| 9D6C81B4D9FA4DE1A7F6881B1...  | Team demonstration | Service Publish | 863.35             | 0.00                 | (UTC+8:00) 06/28/2021 13:47:22 | Expired        | <a href="#">Details</a> |

## 13 Online Documentation

| White Papers                 |         |                                |                     |
|------------------------------|---------|--------------------------------|---------------------|
| Name                         | Version | Update                         | Details             |
| BSN Introduction White paper | V1.05   | February 5 <sup>th</sup> ,2020 | <a href="#">PDF</a> |
| BSN Technical White Paper    | V1.0.0  | April 25 <sup>th</sup> ,2020   | <a href="#">PDF</a> |

| Site Documents      |         |                              |  |
|---------------------|---------|------------------------------|--|
| Name                | Version | Update                       | Details                                    |
| User Manual         | 1.8.4   | June19 <sup>th</sup> ,2023   | <a href="#">Online</a> <a href="#">PDF</a> |
| Fabric Examples     | 1.0.1   | April 24 <sup>th</sup> ,2020 | <a href="#">Github</a>                     |
| FISCO BCOS Examples | 1.0.1   | April 24 <sup>th</sup> ,2020 | <a href="#">Github</a>                     |
| SDK Examples        | 1.0.1   | April 24 <sup>th</sup> ,2020 | <a href="#">Github</a>                     |

| Permissioned Frameworks |   |  |
|-------------------------|---|--|
| Name                    | Official Website  | Details  |
| Hyperledger Fabric      | <a href="https://www.hyperledger.org/">https://www.hyperledger.org/</a>                 | <a href="#">Github</a> <a href="#">Documentation</a> |
| FISCO BCOS              | <a href="http://fisco-bcos.org/">http://fisco-bcos.org/</a>                             | <a href="#">Github</a> <a href="#">Documentation</a> |
| ConsenSys Quorum        | <a href="https://consensys.net/quorum/">https://consensys.net/quorum/</a>               | <a href="#">Github</a> <a href="#">Documentation</a> |
| Hyperledger Besu        | <a href="https://www.hyperledger.org/use/besu">https://www.hyperledger.org/use/besu</a> | <a href="#">Github</a> <a href="#">Documentation</a> |

| Public Chains |   |  |
|---------------|---|--|
| Name          | Official Website  | Details  |
| ETH           | <a href="https://ethereum.org/">https://ethereum.org/</a>   | <a href="#">Github</a> <a href="#">Documentation</a> |
| Tezos         | <a href="https://tezos.com/">https://tezos.com/</a>   | <a href="#">Github</a> <a href="#">Documentation</a> |
| EOS           | <a href="https://eos.io/">https://eos.io/</a>   | <a href="#">Github</a> <a href="#">Documentation</a> |
| dfuse-eos     | <a href="https://www.dfuse.io/en/home/?utm_source=BSN">https://www.dfuse.io/en/home/?utm_source=BSN</a> | <a href="#">Github</a> <a href="#">Documentation</a> |
| Findora       | <a href="https://findora.org/">https://findora.org/</a>   | <a href="#">Github</a> <a href="#">Documentation</a> |
| Near          | <a href="https://near.org/">https://near.org/</a>   | <a href="#">Github</a> <a href="#">Documentation</a> |
| Klaytn        | <a href="https://www.klaytn.com/">https://www.klaytn.com/</a>   | <a href="#">Github</a> <a href="#">Documentation</a> |





## 14 Contact Us

If you have any questions or find any errors in this manual, please contact us:

Customer service hotline: +86-400-071-8215 (workday: 08:00 - 17:30)

Email: [support@bsnbase.com](mailto:support@bsnbase.com)

Telegram BSN Support Group: <https://t.me/bsnsupport>

International Social Media:

