bubi Documentation

发布 *1.0.0*

wilson

2018 年 **09** 月 **06** 日

Contents

Contents:

CHAPTER 1

Bumo keypair

1.1 Overview

This document describes in detail the process of generating Keypairs (public and private key pairs) and how to generate an address and sign a transaction based on keypairs. It introduces two interface methods and related processes for executing the transaction call. It provides reference information for ProtoBuf data structures. Finally, it illustrates two methods to submit transactions by showing how to generate transaction_blob with interface call and how to generate transaction_blob by yourself.

1.2 Terminology

This section gives details about the terms used in this document.

Keypair

In the BUMO project, the keypair is the interface that generates the public key, private key, address, and signature. Only the `ED25519` signature algorithm is supported during the signing process.

Private Key

The private key is a string generated by the algorithm. The private key is a prerequisite for generating the public key and the address, and is also the basic element for completing the signature. The private key cannot be changed after it is generated. Once it is lost, it cannot be retrieved, so it needs to be kept safely.

Public Key

The public key is a string generated based on the private key. It can verify the string encrypted by the private key. It does not expose the private key when transmitted between networks. It is also a necessary condition for generating an address.

Address

The address is a string generated upon the public key. Similar to real-life addresses, contacts cannot be found without an address, so transactions cannot be completed.

Signature

The Signature refers to the process of encrypting and confirming transaction data by algorithm and private key and obtaining signature data. The user can verify the integrity and correctness of the transaction data through the signature data.

Transaction

All operations that modify blockchain data in BUMO are called transactions, such as issuing assets, transferring assets, sending BUs, creating accounts, setting metadata and setting permissions, etc.

Transaction Blob

The Transaction Blob is a hexadecimal string obtained by serializing a transaction object. Transaction serialization refers to the process of converting the state information of a transaction object into a string that can be stored and transmitted through the ProtoBuf data structure.

Raw Private Key

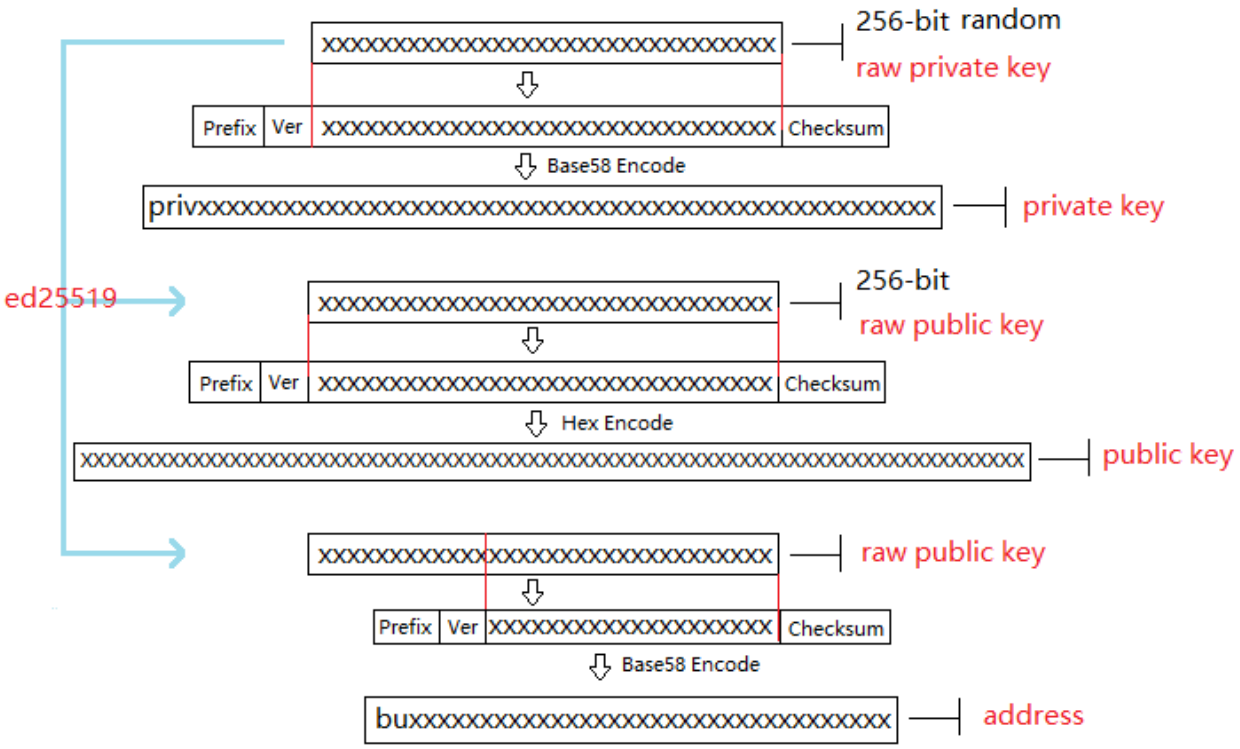
The Raw Private Key is a byte array obtained by a random algorithm. The Raw Private Key is a prerequisite for generating a private key.

Raw Public Key

The Raw Public Key is a byte array generated by processing the raw private key with the `ED25519` algorithm. The Raw Public Key is a prerequisite for generating a public key.

1.3 Schematic Diagram

The following diagram illustrates how the private, public keys and address are generated.



1.4 Generating Private Keys

Generating a private key requires multiple algorithms such as a random algorithm and `SHA256`. Generating a private key includes the following steps:

1. Generate a 256-bit random number (a private key in the mathematical sense) with a random algorithm and get a byte array, the raw private key, as shown below:

```
[17,236,24,183,207,250,207,180,108,87,224,39,189,99,246,85,138,120,236,78,228,233,41,
↪192,124,109,156,104,235,66,194,24]
```

2. Add a 3-byte prefix in the raw private key, and then add a 1-byte version number to get a new byte array, as shown below:

```
[218,55,159,1,17,236,24,183,207,250,207,180,108,87,224,39,189,99,246,85,138,120,236,
↪78,228,233,41,192,124,109,156,104,235,66,194,24]
```

3. Perform `SHA256` calculations twice on the byte array obtained in Step 2. Take the first 4 bytes of the operation result as the byte array of the Checksum, as shown below:

```
[30,19,80,117]
```

4. Combine the byte array in Step 2 and the checksum byte array in Step 3 in order, resulting in a new byte array, as shown below:

```
[218,55,159,1,17,236,24,183,207,250,207,180,108,87,224,39,189,99,246,85,138,120,236,
↪78,228,233,41,192,124,109,156,104,235,66,194,24,30,19,80,117]
```

5. Encode the byte array generated in Step 4 with Base58, and get the string starting with priv, namely the private key, as shown below:

```
privbsGZFUoRv8aXZbSGd3bwzZWFn3L5QKq74RXAQYcmfXhhZ54CLr9z
```

Table 1

Name	Data	Length
Prefix	0xDA 0x37 0x9F	3 bytes
Version	0x01	1 byte
Checksum	After performing `SHA256` calculation twice on the byte array obtained in Step 2,take the first 4 bytes of the operation result	4 bytes

This table illustrates the Prefix, Version and Checksum used in generating the private key.

1.5 Generating Public Keys

The public key can be generated with the `ED25519` algorithm after the private key is generated. Generating a public key includes the following steps:

1. Generate a 32-bit byte array (raw public key) by processing the raw private key with the `ED25519` algorithm. For example, the raw public key of the private key `privbsGZFUoRv8aXZbSGd3bwzZWFn3L5QKq74RXAQYcmfXhhZ54CLr9z` is shown below:

```
[21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,240,212,
↪54,18,225,158,198,50,87,10]
```

2. Add a 1-byte prefix in the raw public key, and then add a 1-byte version number to get a new byte array, as shown below:

```
[176,1,21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,
↪240,212,54,18,225,158,198,50,87,10]
```

注解： For the Prefix, Version and Checksum, please refer to Table 2.

3. Perform `SHA256` calculation twice on the byte array in Step 2. Take the first 4 bytes of the operation result as the byte array of the Checksum, as shown below:

```
[116,171,22,107]
```

4. Combine the byte array in Step 2 and the checksum byte array in Step 3 in order, resulting in a new byte array, as shown below:

```
[176,1,21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,
↪240,212,54,18,225,158,198,50,87,10,116,171,22,107]
```

5. Encode the byte array in Step 4 into hexadecimal and get a hexadecimal string, namely the public key, as shown below:

```
b00115764cd017e0da753271fa26cd529451a21b8253d001f0d43612e19ec632570a74ab166b
```

注解： Now the public key is generated.

Table 2

Name	Data	Length
Prefix	0xB0	1 Byte
Version	0x01	1 Byte
Checksum	After performing `SHA256` calculation twice on the byte array obtained in Step 2, take the first 4 bytes of the operation result	4 Bytes

This table illustrates the Prefix, Version and Checksum used in generating the public key.

1.6 Generating Addresses

The address can be further generated by an algorithm after generating the private key and the public key. Generating an address includes the following steps:

1. Generate a 32-bit byte array (raw public key) by processing the raw private key with the `ED25519` algorithm. For example, the raw public key of the private key `privbsGZFUoRv8aXZbSGd3bwzZWFn3L5QKq74RXAQYcmfXhhZ54CLr9z` is shown below:

```
[21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,240,212,
↪54,18,225,158,198,50,87,10]
```

2. Perform SHA256 calculation twice on the raw public key and take the last 20 bytes of the operation result as the byte array, as shown below:

```
[173,148,59,51,183,193,55,160,1,133,247,80,65,13,67,190,164,114,18,220]
```

3. Add a 2-byte prefix in the byte array generated in Step 2, and then add a 1-byte version number to get a new byte array, as shown below:

```
[1,86,1,173,148,59,51,183,193,55,160,1,133,247,80,65,13,67,190,164,114,18,220]
```

注解： For the Prefix, Version and Checksum, please refer to Table 3.

4. Perform SHA256 calculation twice on the byte array in Step 3. Take the first 4 bytes of the operation result as the byte array of the Checksum, as shown below:

```
[167,127,34,35]
```

5. Combine the byte array in Step 3 and the Checksum byte array in Step 4 in order, resulting in a new byte array, as shown below:

```
[1,86,1,173,148,59,51,183,193,55,160,1,133,247,80,65,13,67,190,164,114,18,220,167,127,
↪34,35]
```

6. Encode the byte array generated in Step 5 with Base58, and get the string starting with bu, namely the address, as shown below:

```
buQmWJrdYJP5CPKTbkQUqscwvTGaU44dord8
```

注解： Now the address is generated.

Table 3

Name	Data	Length
Prefix	0x01 0x56	2 Bytes
Version	0x01	1 Byte
PublicKey	Take the last 20 bytes in raw public key	20 Bytes
Checksum	After performing SHA256 calculation twice on the byte array obtained in step 3, take the first 4 bytes of the operation result	4 Bytes

This table illustrates the Prefix, Version and Checksum used in generating the address.

1.7 Signing Transactions

Sign the pending transaction (the byte array obtained by the inverse hexadecimal encoding of the transaction_blob) with the ED25519 algorithm and the private key, and perform hexadecimal conversion to get sign_data, the signature

string.

The following example shows how to sign the transaction_blob with ED25519 and the private key.

The private key:

```
b00115764cd017e0da753271fa26cd529451a21b8253d001f0d43612e19ec632570a74ab166b
```

The transaction_blob:

```
0A24627551566B5555424B70444B526D48595777314D553855376E676F5165686E6F31363569109F0818C0843D20E8073214
```

After signing the transaction_blob with the signature interface of ED25519 and performing hexadecimal conversion, the resulting sign_data is:

```
a46ee590a84abdeb8cc38ade1ae8e8a2c71bb69bdc4cd7dc0de1b74b37e2cbd1696229687f80dff4276b1a3dd3f95a9bc1d5
```

1.8 Methods of Submitting Transactions

There are two methods of calling the interface to execute transactions: Generating Transaction_blobs by Calling the Interface and Generating Transaction_blobs by Yourself.

1.8.1 Generating Transaction_blobs by Calling the Interface

> **注意:** As the transaction_blob is likely to be intercepted and tampered with, it is not recommended to generate transaction_blobs in this way.

If you need to call the interface to generate transaction_blobs, sign and submit transactions, please refer to the BUMO development documentation at the following address:

https://github.com/bumoproject/bumo/blob/master/docs/develop.md

Calling the interface to generate a transaction_blob includes the following steps:

1. Call the getAccount interface to get the nonce value of the account that is to initiate a transaction. The code is shown below:

```
HTTP GET host:port/getAccount?address=account address
```

2. Populate the json data as needed and complete filling the transaction data. The format is shown below:

```
{
"source_address":"xxxxxxxxxx", //The source transaction account, the originator of␣
↪the transaction
"nonce":2, //Nonce value
"ceil_ledger_seq": 0, //Optional
"fee_limit":1000, //Fee paid in transaction
"gas_price": 1000, //Gas price (Not less than the configured value)
"metadata":"0123456789abcdef", //Optional, metadata for the transaction given by␣
↪users, in hexadecimal format
"operations":[
{
//Populate according to specific operations
```

(continues on next page)

(续上页)

```
},
{
//Populate according to specific operations
}
......
]
}
```

注解： The nonce value needs to be incremented by 1 based on the value obtained in Step 1.

3. By calling the getTransactionBlob interface, the json data generated in Step 2 is passed as a parameter, and a transaction hash and a transaction_blob are obtained to implement transaction serialization. The format is shown below:

```
{
"error_code": 0,
"error_desc": "",
"result": {
"hash": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //Transaction hash
"transaction_blob": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //The hexadecimal␣
↪representation after the transaction is serialized
}
}
```

4. Sign the transaction and populate the transaction data. Sign the transaction_blob according to the previously generated private key, and then populate the json data of the submitted transaction. The format is shown below:

```
{
"items" : [{
"transaction_blob" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //The␣
↪hexadecimal representation after the transaction is serialized
"signatures" : [{//The first signature
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //Signature data
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Public key
}, {//The second signature
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //Signature data
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Public key
}
]
}
]
}
```

5. By calling the submitTransaction interface, the json data generated in Step 4 is passed as a parameter, the response result is obtained and transaction submission is completed. The format of the response result is shown below:

```
{
"results": [
{
"error_code": 0,
"error_desc": "",
"hash": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Transaction hash
}
],
```

(continues on next page)

(续上页)

```
"success_count": 1
}
```

1.8.2 Generating Transaction_blobs by Yourself

Generating the transaction_blob by yourself, signing, and submitting the transaction include the following steps:

1. Call the getAccount interface to get the nonce value of the account that is to initiate a transaction. The code is shown below:

```
HTTP GET host:port/getAccount?address=account address
```

2. Populate the transaction object (Transaction) of the protocol buffer and serialize it to get the transaction_blob. For details of the specific transaction data structure, please refer to *ProtoBuf Data Structure*.

3. Sign the transaction and populate the transaction data. Generate a public key based on the private key, sign the transaction _blob with the private key, and then populate the json data of the submitted transaction. The format is shown below:

```
{
"items" : [{
"transaction_blob" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //The␣
↪hexadecimal representation after the transaction is serialized
"signatures" : [{//The first signature
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //Signature data
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Public key
}, {//The second signature
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //Signature data
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Public key
}
]
}
]
}
```

4. By calling the submitTransaction interface, the json data generated in Step 3 is passed as a parameter to complete the transaction submission. The response result format is shown below:

```
{
"results": [
{
"error_code": 0,
"error_desc": "",
"hash": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //Transaction hash
}
],
"success_count": 1
}
```

1.9 ProtoBuf Data Structure

Protocol Buffer (ProtoBuf) is a lightweight and efficient structured data storage format that can be used for serializing structured data. It is ideal for data storage or RPC data exchange formats. It can be used in communication proto-

cols, data storage and other fields of language-independent, platform-independent, scalable serialized structured data formats. Currently the APIs in `C++`, `Java`, and `Python` are available.

For more information aobut ProtoBuf, please refer to the following link:

https://developers.google.com/protocol-buffers/docs/overview

Now, we will introduce the data structure details of Protocol Buffer, and provide the file and simple test program for the protocol buffer of various languages generated by the script.

1.9.1 Data Structure

The following section describes the various ProtoBuf data structures that might be used in transactions and their uses for your reference.

Transaction

This data structure is for complete transactions.

```
message Transaction {
enum Limit{
UNKNOWN = 0;
OPERATIONS = 1000;
};
string source_address = 1; // Account address of the transaction initiator
int64 nonce = 2; // Transaction sequence number
int64 fee_limit = 3; // The transaction fee, by default is 1000Gas; the unit is MO, 1␣
↪BU = 10^8 MO
int64 gas_price = 4; // The packaging fee of transactions, by default is 1000; the␣
↪unit is MO,1 BU = 10^8 MO
int64 ceil_ledger_seq = 5; // Block bound
bytes metadata = 6; // Transaction metadata
repeated Operation operations = 7; // Operation list
}
```

Operation

This data structure is for operations in transactions.

```
message Operation {
enum Type {
UNKNOWN = 0;
CREATE_ACCOUNT = 1;
ISSUE_ASSET = 2;
PAY_ASSE = 3;
SET_METADATA = 4;
SET_SIGNER_WEIGHT = 5;
SET_THRESHOLD = 6;
PAY_COIN = 7;
LOG = 8;
SET_PRIVILEGE = 9;
};
Type type = 1; // Operation type
string source_address = 2; // Source account address for the operation
bytes metadata = 3; // Operation metadata

OperationCreateAccount create_account = 4; // Create an account operation
OperationIssueAsset issue_asset = 5; // Issue assets operation
OperationPayAsset pay_asset = 6; // Transfer assets operation
```

(continues on next page)

(续上页)

```
OperationSetMetadata set_metadata = 7; // Set metadata
OperationSetSignerWeight set_signer_weight = 8; // Set privilege for signer
OperationSetThreshold set_threshold = 9; // Set transaction threshold
OperationPayCoin pay_coin = 10; // Transfer coin
OperationLog log = 11; // Record log
OperationSetPrivilege set_privilege = 12; // Set privilege
}
```

OperationCreateAccount

This data structure is for creating accounts.

```
message OperationCreateAccount{
string dest_address = 1; // Target account address to be created
Contract contract = 2; // Contract
AccountPrivilege priv = 3; // Privilege
repeated KeyPair metadatas = 4; // Additional info
int64 init_balance = 5; // Initiation balance
string init_input = 6; // Input parameter for contracts
}
```

Contract

This data structure is for setting contracts.

```
message Contract{
enum ContractType{
JAVASCRIPT = 0;
}
ContractType type = 1; // Contract type
string payload = 2; // Contract code
}
```

AccountPrivilege

This data structure is for setting account privilege.

```
message AccountPrivilege {
int64 master_weight = 1; // Account weight
repeated Signer signers = 2; // Signer weight list
AccountThreshold thresholds = 3; // Threshold
}
```

Signer

This data structure is for setting signer weight.

```
message Signer {
enum Limit{
SIGNER_NONE = 0;
SIGNER = 100;
};
string address = 1; // Signer account address
int64 weight = 2; // Signer weight
}
```

AccountThreshold

This data structure is for setting account threshold.

```
message AccountThreshold{
int64 tx_threshold = 1; // Transaction threshold
repeated OperationTypeThreshold type_thresholds = 2; // Specify the transaction␣
↪threshold list for the operations. The threshold for the transactions with␣
↪unspecified operation is set by tx_threshold
}
```

OperationTypeThreshold

This data structure is for operation threshold of specified types.

```
message OperationTypeThreshold{
Operation.Type type = 1; // Operation type
int64 threshold = 2; // Corresponding threshold of this operation
}
```

OperationIssueAsset

This data structure is for issuing assets.

```
message OperationIssueAsset{
string code = 1; // Asset encoding to be issued
int64 amount = 2; // Asset amount to be issued
}
```

OperationPayAsset

This data structure is for transferring assets.

```
message OperationPayAsset {
string dest_address = 1; // Target account address
Asset asset = 2; // Asset
string input = 3; // Input parameter for contracts
}
```

Asset

This data structure is for asset.

```
message Asset{
AssetKey key = 1; // Asset identification
int64 amount = 2; // Asset amount
}
```

AssetKey

This data structure is for identifying the uniqueness of asset.

```
message AssetKey{
string issuer = 1; // Account address of asset issuer
string code = 2; // Asset encoding
int32 type = 3; // Asset type(by default is 0, which indicates the amount is not␣
↪limited)
}
```

OperationSetMetadata

This data structure is for setting Metadata.

```
message OperationSetMetadata{
string key = 1; // keyword, unique
string value = 2; // Content
int64 version = 3; // Version control, optional
bool delete_flag = 4; // Whether it is deletable
}
```

OperationSetSignerWeight

This data structure is for setting signer weight.

```
message OperationSetSignerWeight{
int64 master_weight = 1; // Self weight
repeated Signer signers = 2; // Signer weight list
}
```

OperationSetThreshold

This data structure is for setting threshold.

```
message OperationSetThreshold{
int64 tx_threshold = 1; // Transaction threshold
repeated OperationTypeThreshold type_thresholds = 2; // The transaction threshold␣
↪list for specified operations. The threshold for the transactions with unspecified␣
↪operation is set by tx_threshold

}
```

OperationPayCoin

This data structure is for sending coin.

```
message OperationPayCoin{
string dest_address = 1; // Target account address
int64 amount = 2; // Coin amount
string input = 3; // Input parameter for contracts
}
```

OperationLog

This data structure is for recording log information.

```
message OperationLog{
string topic = 1; // Log theme
repeated string datas = 2; // Log content
}
```

OperationSetPrivilege

This data structure is for setting account privilege.

```
message OperationSetPrivilege{
string master_weight = 1; // Account weight
repeated Signer signers = 2; // Signer weight list
string tx_threshold = 3; // Transaction threshold
repeated OperationTypeThreshold type_thresholds = 4; // The transaction threshold␣
↪list for specified operations. The threshold for the transactions with unspecified␣
↪operation is set by tx_threshold

```

(continues on next page)

(续上页)

```
}
```

1.9.2 Examples

This section provides examples of proto scripts, as well as proto source code generated by `cpp`, `java`, `javascript`, `pyton`, `object-c`, and `php`. For more information, please refer to the following link:

https://github.com/bumoproject/bumo/tree/develop/src/proto

Description of the directory structure in the above link is shown below:

1. cpp: C++ source code
2. io: Java source code
3. go: Go source and test program
4. js: Javascript source code and test program
5. Python: Python source code and test program
6. ios: Object-c source code and test program
7. php: PHP source code and test program

1.10 Examples for Transaction Submission

Scenario: Account A（`buQVkUUBKpDKRmHYWw1MU8U7ngoQehno165i`）creates account B (Generate an address by *Generating Addresses* in Keypair).

1.10.1 Generating Transaction_blobs by Interface

Generating transaction_blobs by the interface includes the following steps:

1. Obtain the nonce value of the account to initiate a transaction by GET.

```
GET http://seed1.bumotest.io:26002/getAccount?
↪address=buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw
```

Response message:

```
{
"error_code" : 0,
"result" : {
"address" : "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw",
"assets" : [
{
"amount" : 1000000000,
"key" : {
"code" : "HNC",
"issuer" : "buQBjJD1BSJ7nzAbzdTenAhpFjmxRVEEtmxH"
}
}
],
```

(continues on next page)

(续上页)

```
"assets_hash" : "3bf279af496877a51303e91c36d42d64ba9d414de8c038719b842e6421a9dae0",
"balance" : 27034700,
"metadatas" : null,
"metadatas_hash" : "ad67d57ae19de8068dbcd47282146bd553fe9f684c57c8c114453863ee41abc3",
"nonce" : 5,
"priv" : {
"master_weight" : 1,
"thresholds" : [{
"tx_threshold" : 1
}
]
}
}
}
address: Current query account address
assets: Account asset list
assets_hash: Asset list hash
balance: Account balance
metadata: Account metadata in hexadecimal format
metadatas_hash: Transaction metadata hash
nonce: The sending transaction serial number, the nonce+1 returned by querying the␣
↪account information interface
priv: Privilege
master_weight: Current account weight
thresholds: Threshold
tx_threshold: Transaction default threshold
```

2. Complete populating the transaction data.

The account address of account B generated by *Generating Address* in Keypair is `buQoP2eRymAcUm3uvWgQ8RnjtrSnXBXfAzsV`, the populated json data is shown below:

```
{
"source_address":"buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw",
"nonce":7,
"ceil_ledger_seq": 0,
"fee_limit":1000000,
"gas_price": 1000,
"metadata":"",
"operations":[
{
"type": 1,
"create_account": {
"dest_address": "buQoP2eRymAcUm3uvWgQ8RnjtrSnXBXfAzsV",
"init_balance": 10000000,
"priv": {
"master_weight": 1,
"thresholds": {
"tx_threshold": 1
}
}
}
}
]
}
```

注解：The nonce value is not 6, so this transaction would fail.

3. Serialize the transaction data.

```
POST http://seed1.bumotest.io:26002/getTransactionBlob
```

Request message:

```
4.1.2 populated jason data
```

Response message:

```
{
"error_code": 0,
"error_desc": "",
"result": {
"hash": "be4953bce94ecd5c5a19c7c4445d940c6a55fb56370f7f606e127776053b3b51",
"transaction_blob":
↪"0a24627551737572483l4d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3
↪"
}
}
```

4. Sign the transaction_blob with the private key.

Import package: import io.bumo.encryption.key.PrivateKey;

```
Private key:
privbvTuL1k8z27i9eyBrFDUvAVVCSxKeLtzjMMZEqimFwbNchnejS81

The sign_data after being signed:
9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A306
```

5. Complete populating the transaction data.

```
{
"items" : [{
"transaction_blob" :
↪"0a24627551737572483l4d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3
↪",
"signatures" : [{
"sign_data" :
↪"9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A
↪",
"public_key" :
↪"b00179b4adb1d3188aa1b98d6977a837bd4afdbb4813ac65472074fe3a491979bf256ba63895"
}
]
}
]
}
```

6. Submit the transaction by POST.

```
POST http://seed1.bumotest.io/submitTransaction
```

Response message:

```
{
"results": [{
"error_code": 0,
"error_desc": "",
"hash": "be4953bce94ecd5c5a19c7c4445d940c6a55fb56370f7f606e127776053b3b51"
}
],
"success_count": 1
}
```

1.10.2 Generating Transaction_blobs by Yourself

Generating transaction_blobs by yourself (take `Java` as an example) includes the following steps:

1. Obtain the nonce value of the account that is to initiate a transaction by GET.

```
GET http://seed1.bumotest.io:26002/getAccount?
↪address=buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw
```

Response message:

```
{
"error_code" : 0,
"result" : {
"address" : "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw",
"assets" : [
{
"amount" : 1000000000,
"key" : {
"code" : "HNC",
"issuer" : "buQBjJD1BSJ7nzAbzdTenAhpFjmxRVEEtmxH"
}
}
],
"assets_hash" : "3bf279af496877a51303e91c36d42d64ba9d414de8c038719b842e6421a9dae0",
"balance" : 27034700,
"metadatas" : null,
"metadatas_hash" : "ad67d57ae19de8068dbcd47282146bd553fe9f684c57c8c114453863ee41abc3",
"nonce" : 5,
"priv" : {
"master_weight" : 1,
"thresholds" : [{
"tx_threshold" : 1
}
]
}
}
}
address: Current query account address
assets: Account asset list
assets_hash: Asset list hash
balance: Account balance
metadata: Account metadata in hexadecimal format
metadatas_hash: Transaction metadata hash
nonce: The sending transaction serial number, the nonce+1 returned by querying the␣
↪account information interface
```

(continues on next page)

(续上页)

```
priv: Privilege
master_weight: Current account weight
thresholds: Threshold
tx_threshold: Transaction default threshold
```

2. Populate the transaction data structure and generate a transaction_blob.

Import package: import io.bumo.sdk.core.extend.protobuf.Chain;

```
Chain.Transaction.Builder builder = Chain.Transaction.newBuilder();
builder.setSourceAddress("buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw");
builder.setNonce(7);

builder.setFeeLimit(1000 * 1000);
builder.setGasPrice(1000);
builder.setCeilLedgerSeq(0);
builder.setMetadata(ByteString.copyFromUtf8(""));

Chain.Operation.Builder operation = builder.addOperationsBuilder();
operation.setType(Chain.Operation.Type.CREATE_ACCOUNT);

Chain.OperationCreateAccount.Builder operationCreateAccount = Chain.
↪OperationCreateAccount.newBuilder();
operationCreateAccount.setDestAddress("buQoP2eRymAcUm3uvWgQ8RnjtrSnXBXfAzsV");
operationCreateAccount.setInitBalance(10000000);

Chain.AccountPrivilege.Builder accountPrivilegeBuilder = Chain.AccountPrivilege.
↪newBuilder();
accountPrivilegeBuilder.setMasterWeight(1);

Chain.AccountThreshold.Builder accountThresholdBuilder = Chain.AccountThreshold.
↪newBuilder();
accountThresholdBuilder.setTxThreshold(1);

accountPrivilegeBuilder.setThresholds(accountThresholdBuilder);
operationCreateAccount.setPriv(accountPrivilegeBuilder);
operation.setCreateAccount(operationCreateAccount);
String transaction_blob = HexFormat.byteToHex(builder.build().toByteArray());

The transaction_blob obtained:
0a2462755173757248314d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3708
```

注解： The nonce value is not 6, so this transaction would fail.

3. Sign the transaction_blob with the private key.

Import package: import io.bumo.encryption.key.PrivateKey;

```
The private key:
privbvTuL1k8z27i9eyBrFDUvAVVCSxKeLtzjMMZEqimFwbNchnejS81

The sign_data after being signed:
9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A30
```

4. Complete populating the transaction data.

```
{
"items" : [{
"transaction_blob" :
↪"0a2462755173757248314d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3
↪",
"signatures" : [{
"sign_data" :
↪"9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A
↪",
"public_key" :
↪"b00179b4adb1d3188aa1b98d6977a837bd4afdbb4813ac65472074fe3a491979bf256ba63895"
}
]
}
]
}
```


5. Submit the transaction by POST.

```
POST http://seed1.bumotest.io/submitTransaction
```

Response message:


```
{
"results": [{
"error_code": 0,
"error_desc": "",
"hash": "be4953bce94ecd5c5a19c7c4445d940c6a55fb56370f7f606e127776053b3b51"
}
],
"success_count": 1
}
```


注解: “success_count”:1 represents that the submission succeeded.

CHAPTER 2

BUMO JAVA SDK

2.1 Overview

This document details the common interfaces of the Bumo Java SDK, making it easier for developers to operate and query the BU blockchain.

2.2 Terminology

This section gives details about the terms used in this document.

Operate the BU Blockchain

Operate the BU Blockchain refers to writing data to or modifying data in the BU blockchain.

Submit Transactions

Submit Transactions refers to sending a request to write data to or modify data in the BU blockchain.

Query the BU Blockchain

Query the BU Blockchain refers to querying data in the BU blockchain.

Account Services

Account Services provide account validity checking and query interfaces.

Asset Services

Asset Services provide an asset-related query interface that follows the ATP 1.0 protocol.

Ctp10Token Services

Ctp10Token Services provide a validity check and query interfaces related to contract assets, which follows the CTP 1.0 protocol.

Contract Services

Contract Services provide a contract-related validity checking and query interfaces.

Transaction Services

Transaction Services provide a build transaction Blob interface, a signature interface, a query and a submit transaction interface.

Block Services

Block Services provide an interface to query the block.

Account Nonce Value

Account Nonce Value is used to identify the order in which the transaction is executed when the user submits the transaction.

2.3 Format of Request Parameters and Response Data

This section details the format of the request parameters and response data.

2.3.1 Request Parameters

The class name of the request parameter of the interface is composed of **Service Name+Method Name+Request**. For example, the request parameter format of the `getInfo` interface in Account Services is `AccountGetInfoRequest`.

The member of the request parameter is the member of the input parameter of each interface. For example, if the input parameter of the `getInfo` interface in Account Services is `address`, the complete structure of the request parameters of the interface is as follows:

```
Class AccountGetInfoRequest {
String address;
}
```

2.3.2 Response Data

The class name of the response data of the interface is composed of **Service Name+Method Name+Response**. For example, the response data format of the `getNonce` interface in Account Services is `AccountGetNonceResponse`.

The members of the response data include error codes, error descriptions, and return results. For example, the members of the response data of the `getInfo` interface in Assets Services are as follows:

```
Class AccountGetNonceResponse {
Integer errorCode;
String errorDesc;
AccountGetNonceResult result;
}
```

注解:

- errorCode: error code. 0 means no error, greater than 0 means there is an error
- errorDesc: error description

- result: returns the result. A structure whose class name is **Service Name+Method Name+Result**, whose members are members of the return value of each interface. For example, the result class name of the `getNonce` interface in Account Services is AccountGetNonceResult, and the member has a nonce. The complete structure is as follows:

```
Class AccountGetNonceResult {
Long nonce;
}
```

2.4 Usage

This section describes the process of using the SDK. First you need to generate the SDK implementation and then call the interface of the corresponding service. Services include account services, asset services, Ctp1.0Token services, contract services, transaction services, and block services. Interfaces are classified into public-private key address interfaces, validity check interfaces, query interfaces, and broadcast transaction-related interfaces.

2.4.1 Generating SDK Instances

The SDK instance is generated by calling the getInstance interface of the SDK. The specific call is as follows:

```
String url = "http://seed1.bumotest.io";
SDK sdk = SDK.getInstance(url);
```

2.4.2 Generating Public-Private Keys and Addresses

The public-private key address interface is used to generate the public key, private key, and address for the account on the BU blockchain. This can be achieved by directly calling the `Keypair.generator` interface. The specific call is as follows:

```
Keypair keypair = Keypair.generator();
System.out.println(keypair.getPrivateKey());
System.out.println(keypair.getPublicKey());
System.out.println(keypair.getAddress());
```

2.4.3 Checking Validity

The validity check interface is used to verify the validity of the information, and the information validity check can be achieved by directly invoking the corresponding interface. For example, to verify the validity of the account address, the specific call is as follows:

```
//
Initialize request parameters
String address = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
AccountCheckValidRequest request = new AccountCheckValidRequest();
request.setAddress(address);

// Call the ``checkValid`` interface
AccountCheckValidResponse response =
sdk.getAccountService().checkValid(request);
```

(continues on next page)

(续上页)

```
if(0 == response.getErrorCode()) {
System.out.println(response.getResult().isValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.4.4 Querying

The query interface is used to query data on the BU blockchain, and data query can be implemented by directly invoking the corresponding interface. For example, to query the account information, the specific call is as follows:

```
// Initialize request parameters
String accountAddress = "buQemmMwmRQY1JkcU7w3nhruo%X5N3j6C29uo";
AccountGetInfoRequest request = new AccountGetInfoRequest();
request.setAddress(accountAddress);

// Call the getInfo interface
AccountGetInfoResponse response = sdk.getAccountService().getInfo(request);
if (response.getErrorCode() == 0) {
AccountGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result,true));
}
else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.4.5 Broadcasting Transactions

Broadcasting transactions refers to the initiation of a transaction by means of broadcasting. The broadcast transaction consists of the following steps:

1. Obtaining the Nonce Value of the Account Initiating the Transaction
2. Building Operations
3. Serializing Transactions
4. Signing Transactions
5. Commiting Transactions

2.4.5.1 Obtaining the Nonce Value of the Account Initiating the Transaction

The developer can maintain the nonce value of each account, and automatically increments by 1 for the nounce value after submitting a transaction, so that multiple transactions can be sent in a short time; otherwise, the nonce value of the account must be added 1 after the execution of the previous transaction is completed. The specific interface call is as follows:

```
// Initialize request parameters
String senderAddress = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
AccountGetNonceRequest getNonceRequest = new AccountGetNonceRequest();
getNonceRequest.setAddress(senderAddress);
```

(continues on next page)

(续上页)

```
// Call the getNonce interface
AccountGetNonceResponse getNonceResponse = sdk.getAccountService().
↪getNonce(getNonceRequest);

// Assign nonce value
if (getNonceResponse.getErrorCode() == 0) {
AccountGetNonceResult result = getNonceResponse.getResult();
System.out.println("nonce: " + result.getNonce());
}
else {
System.out.println("error" + getNonceResponse.getErrorDesc());
}
```

2.4.5.2 Building Operations

The operations refer to some of the actions that are done in the transaction to facilitate serialization of transactions and evaluation of fees. For example, to build an operation to send BU (BUSendOperation), the specific interface call is as follows:

```
String senderAddress = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
String destAddress = "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw";
Long buAmount = ToBaseUnit.BU2MO("10.9");

BUSendOperation operation = new BUSendOperation();
operation.setSourceAddress(senderAddress);
operation.setDestAddress(destAddress);
operation.setAmount(buAmount);
```

2.4.5.3 Serializing Transactions

The transaction serialization interface is used to serialize transactions and generate transaction blob strings for network transmission. The nonce value and operation are obtained from the interface called, and the specific interface call is as follows:

```
// Initialize variables
String senderAddress = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
Long gasPrice = 1000L;
Long feeLimit = ToBaseUnit.BU2MO("0.01");

// Initialize request parameters
TransactionBuildBlobRequest buildBlobRequest = new TransactionBuildBlobRequest();
buildBlobRequest.setSourceAddress(senderAddress);
buildBlobRequest.setNonce(nonce + 1);
buildBlobRequest.setFeeLimit(feeLimit);
buildBlobRequest.setGasPrice(gasPrice);
buildBlobRequest.addOperation(operation);

// Call the buildBlob interface
TransactionBuildBlobResponse buildBlobResponse = sdk.getTransactionService().
↪buildBlob(buildBlobRequest);
if (buildBlobResponse.getErrorCode() == 0) {
TransactionBuildBlobResult result = buildBlobResponse.getResult();
System.out.println("txHash: " + result.getHash() + ", blob: " + result.
↪getTransactionBlob());
```

(continues on next page)

(续上页)

```
} else {
System.out.println("error: " + buildBlobResponse.getErrorDesc());
}
```

2.4.5.4 Signing Transactions

The `signature transaction` interface is used by the transaction initiator to sign the transaction using the private key of the account. The transactionBlob is obtained from the interface called. The specific interface call is as follows:

```
// Initialize request parameters
String senderPrivateKey = "privbyQCRp7DLqKtRFCqKQJr81TurTqG6UKXMMtGAmPG3abcM9XHjWvq";
String []signerPrivateKeyArr = {senderPrivateKey};
TransactionSignRequest signRequest = new TransactionSignRequest();
signRequest.setBlob(transactionBlob);
for (int i = 0; i < signerPrivateKeyArr.length; i++) {
signRequest.addPrivateKey(signerPrivateKeyArr[i]);
}

// Call the sign interface
TransactionSignResponse signResponse = sdk.getTransactionService().sign(signRequest);
if (signResponse.getErrorCode() == 0) {
TransactionSignResult result = signResponse.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + signResponse.getErrorDesc());
}
```

2.4.5.5 Submitting Transactions

The submit interface is used to send a transaction request to the BU blockchain, triggering the execution of the transaction. transactionBlob and signResult are obtained from the interfaces called. The specific interface call is as follows:

```
// Initialize request parameters
TransactionSubmitRequest submitRequest = new TransactionSubmitRequest();
submitRequest.setTransactionBlob(transactionBlob);
submitRequest.setSignatures(signResult.getSignatures());

// Call the submit interface
TransactionSubmitResponse response = sdk.getTransactionService().
↪submit(submitRequest);
if (0 == response.getErrorCode()) {
System.out.println("Broadcast transactions successfully, hash=" + response.getResult().
↪getHash());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.5 Account Services

Account Services provide account-related interfaces, which include six interfaces: `checkValid`, `getInfo`, `getNonce`, `getBalance`, `getAssets` and `getMetadata`.

2.5.1 checkValid

The `checkValid` interface is used to check the validity of the account address on the blockchain.

The method call is as follows:

```
AccounCheckValidResponse checkValid(AccountCheckValidRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
address	String	Required, the account address to be checked on the blockchain

The response data is shown in the following table:

Parameter	Type	Description
isValid	String	Whether the response data is valid

The error code is shown in the following table:

Exception	Error Code	Description
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
String address = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
AccountCheckValidRequest request = new AccountCheckValidRequest();
request.setAddress(address);

// Call the checkValid interface
AccountCheckValidResponse response = sdk.getAccountService().checkValid(request);
if(0 == response.getErrorCode()) {
System.out.println(response.getResult().isValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.5.2 getInfo

The `getInfo` interface is used to obtain the specified account information.

The method call is as follows:

```
AccountGetInfoResponse GetInfo(AccountGetInfoRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
address	String	Required, the account address to be queried on the blockchain

The response data is shown in the following table:

Parameter	Type	Description
address	String	Account address
balance	Long	Account balance, unit is MO, 1 BU = 10^8 MO, the account balance must be > 0
nonce	Long	Account transaction serial number must be greater than 0
priv	*Priv*	Account privilege

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters

String accountAddress = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
AccountGetInfoRequest request = new AccountGetInfoRequest();
request.setAddress(accountAddress);

// Call the getInfo interface
AccountGetInfoResponse response = sdk.getAccountService().getInfo(request);
if (response.getErrorCode() == 0) {
AccountGetInfoResult result = response.getResult();
System.out.println("Account info: \n" + JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.5.2.1 Priv

The specific information of Priv is shown in the following table:

Member	Type	Description
masterWeight	Long	Account weight, size limit[0, (Integer.MAX_VALUE * 2L + 1)]
signers	*Signer* []	Signer weight list
threshold	*Threshold*	Threshold

2.5.2.2 Signer

The specific information of Signer is shown in the following table:

Member	Type	Description
address	String	The account address of the signer on the blockchain
weight	Long	Signer weight, size limit[0, (Integer.MAX_VALUE * 2L + 1)]

2.5.2.3 Threshold

The specific information of Signer is shown in the following table:

Member	Type	Description
txThreshold	Long	Transaction default threshold, size limit[0, Long.MAX_VALUE]
typeThresholds	*TypeThreshold*[]	Thresholds for different types of transactions

2.5.2.4 TypeThreshold

The specific information of Signer is shown in the following table:

Member	Type	Description
type	Long	The operation type must be greater than 0
threshold	Long	Threshold, size limit[0, Long.MAX_VALUE]

2.5.3 getNonce

The `getNonce` interface is used to obtain the nonce value of the specified account.

The method call is as follows:

```
AccountGetNonceResponse getNonce(AccountGetNonceRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
address	String	Required, the account address to be queried on the blockchain

The response data is shown in the following table:

Parameter	Type	Description
nonce	Long	Account transaction serial number

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters

String accountAddress = "buQswSaKDACkrFsnP1wcVsLAUzXQsemauEjf";
AccountGetNonceRequest request = new AccountGetNonceRequest();
request.setAddress(accountAddress);

```

(continues on next page)

(续上页)

```
// Call the getNonce interface
AccountGetNonceResponse response = sdk.getAccountService().getNonce(request);
if(0 == response.getErrorCode()){
System.out.println("Account nonce:" + response.getResult().getNonce());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.5.4 getBalance

The `getBalance` interface is used to obtain the BU balance of the specified account.

The method call is as follows:

```
AccountGetBalanceResponse getBalance(AccountGetBalanceRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
address	String	Required, the account address to be queried on the blockchain

The response data is shown in the following table:

Parameter	Type	Description
balance	Long	BU balance, unit MO, 1 BU = 10^8 MO

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters

String accountAddress = "buQswSaKDACkrFsnP1wcVsLAUzXQsemauEjf";
AccountGetBalanceRequest request = new AccountGetBalanceRequest();
request.setAddress(accountAddress);

// Call the getBalance interface
AccountGetBalanceResponse response = sdk.getAccountService().getBalance(request);
if(0 == response.getErrorCode()){
AccountGetBalanceResult result = response.getResult();
System.out.println("BU balance: " + ToBaseUnit.MO2BU(result.getBalance().toString()) +
↪" BU");
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.5.5 getAssets

The `getAssets` interface is used to get all the asset information of the specified account.

The method call is as follows:

```
AccountGetAssets getAssets(AccountGetAssetsRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
address	String	Required, the account address to be queried

The response data is shown in the following table:

Parameter	Type	Description
asset	*AssetInfo*[]	Account asset

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
NO_ASSET_ERROR	11009	The account does not have the asset
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
AccountGetAssetsRequest request = new AccountGetAssetsRequest();
request.setAddress("buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw");

// Call the getAssets interface
AccountGetAssetsResponse response = sdk.getAccountService().getAssets(request);
if (response.getErrorCode() == 0) {
AccountGetAssetsResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.5.5.1 AssetInfo

The specific information of AssetInfo is shown in the following table:

Member	Type	Description
key	*Key*	Unique identifier for asset
assetAmount	Long	Amount of assets

2.5.5.2 Key

The specific information of Key is shown in the following table:

Member	Type	Description
code	String	Asset code
issuer	String	The account address for issuing assets

2.5.6 getMetadata

The `getMetadata` interface is used to obtain the metadata information of the specified account.

The method call is as follows:

```
AccountGetMetadataResponse getMetadata(AccountGetMetadataRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
address	String	Required, the account address to be queried
key	String	Optional, metadata keyword, length limit [1, 1024]

The response data is shown in the following table:

Parameter	Type	Description
metadata	*MetadataInfo*	Account

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
NO_METADATA_ERROR	11010	The account does not have the metadata
INVALID_DATAKEY_ERROR	11011	The length of key must be between 1 and 1024
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
String accountAddress = "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw";
AccountGetMetadataRequest request = new AccountGetMetadataRequest();
request.setAddress(accountAddress);
request.setKey("20180704");

// Call the getMetadata interface
AccountGetMetadataResponse response = sdk.getAccountService().getMetadata(request);
if (response.getErrorCode() == 0) {
AccountGetMetadataResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.5.6.1 MetadataInfo

The specific information of MetadataInfo is shown in the following table:

Member	Type	Description
key	String	Metadata keyword
value	String	Metadata content
version	Long	Metadata version

2.6 Asset Services

Asset Services follow the ATP 1.0 protocol, and Account Services provide an asset-related interface. Currently there is one interface: `getInfo`.

2.6.1 getInfo

The `getInfo` interface is used to obtain the specified asset information of the specified account.

The method call is as follows:

```
AssetGetInfoResponse getInfo(AssetGetInfoRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
address	String	Required, the account address to be queried
code	String	Required, asset code, length limit [1, 64]
issuer	String	Required, the account address for issuing assets

The response data is shown in the following table:

Parameter	Type	Description
asset	*AssetInfo*[]	Account asset

The error code is shown in the following table:

The specific example is as follows:

```
// Initialize request parameters

AssetGetInfoRequest request = new AssetGetInfoRequest();
request.setAddress("buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw");
request.setIssuer("buQBjJD1BSJ7nzAbzdTenAhpFjmxRVEEtmxH");
request.setCode("HNC");

// Call the getInfo interface
AssetGetInfoResponse response = sdk.getAssetService().getInfo(request);
if (response.getErrorCode() == 0) {
AssetGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
```

(continues on next page)

(续上页)

```
System.out.println("error: " + response.getErrorDesc());
}
```

2.7 Ctp10Token Services

Ctp10Token Services follow the CTP 1.0 protocol and mainly provide contract Token-related interfaces. Currently there are 8 interfaces: `checkValid`, `allowance`, `getInfo`, `getName`, `getSymbol`, `getDecimals`, `getTotalSupply`, and `getBalance`.

2.7.1 checkValid

The `checkValid` interface is used to verify the validity of the contract token.

The method call is as follows:

```
Ctp10TokenCheckValidResponse checkValid(Ctp10TokenCheckValidRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Required, contract address of token to be verified

The response data is shown in the following table:

Parameter	Type	Description
isValid	String	Whether the response data is valid

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
Ctp10TokenCheckValidRequest request = new Ctp10TokenCheckValidRequest();
request.setContractAddress("buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea");

// Call the checkValid interface
Ctp10TokenCheckValidResponse response = sdk.getTokenService().checkValid(request);
if (response.getErrorCode() == 0) {
Ctp10TokenCheckValidResult result = response.getResult();
System.out.println(result.getValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.7.2 allowance

The `allowance` interface is used to obtain the amount that the spender allows to extract from the owner.

The method call is as follows:

```
Ctp10TokenAllowanceResponse allowance(Ctp10TokenAllowanceRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
contractAddress	String	Required, contract account address
tokenOwner	String	Required, the account address holding the contract Token
spender	String	Required, authorized account address

The response data is shown in the following table:

Parameter	Type	Description
allowance	String	Allowed amount to be withdrawn

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
INVALID_TOKENOWNER_ERRPR	11035	Invalid token owner
INVALID_SPENDER_ERROR	11043	Invalid spender
GET_ALLOWNANCE_ERROR	11036	Failed to get allowance
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
Ctp10TokenAllowanceRequest request = new Ctp10TokenAllowanceRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");
request.setTokenOwner("buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp");
request.setSpender("buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp");

// Call the allowance interface
Ctp10TokenAllowanceResponse response = sdk.getTokenService().allowance(request);
if (response.getErrorCode() == 0) {
Ctp10TokenAllowanceResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.7.3 getInfo-Ctp10Token

The `getInfo-Ctp10Token` interface is used to obtain information about the contract token.

The method call is as follows:

```
Ctp10TokenGetInfoResponse getInfo(Ctp10TokenGetInfoRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract token address to be queried

The response data is shown in the following table:

Parameter	Type	Description
ctp	String	Contract Token version number
symbol	String	Contract Token symbol
decimals	Integer	Accuracy of the number of contracts
totalSupply	String	Total supply of contracts
name	String	The name of the contract Token
contractOwner	String	Owner of the contract Token

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
Ctp10TokenGetInfoRequest request = new Ctp10TokenGetInfoRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// Call the allowance interface
Ctp10TokenGetInfoResponse response = sdk.getTokenService().getInfo(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.7.4 getName

The `getName` interface is used to get the name of the contract Token.

The method call is as follows:

```
Ctp10TokenGetNameResponse getName(Ctp10TokenGetNameRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract account address to be queried

The response data is shown in the following table:

Parameter	Type	Description
name	String	The name of the contract Token

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
Ctp10TokenGetNameRequest request = new Ctp10TokenGetNameRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// Call the getName interface
Ctp10TokenGetNameResponse response = sdk.getTokenService().getName(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetNameResult result = response.getResult();
System.out.println(result.getName());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.7.5 getSymbol

The `getSymbol` interface is used to get the symbol of the contract Token.

The method call is as follows:

```
Ctp10TokenGetSymbolResponse getSymbol (Ctp10TokenGetSymbolRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract account address to be queried

The response data is shown in the following table:

Parameter	Type	Description
symbol	String	Contract Token symbol

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters

Ctp10TokenGetSymbolRequest request = new Ctp10TokenGetSymbolRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// Call the getSymbol interface
Ctp10TokenGetSymbolResponse response = sdk.getTokenService().getSymbol(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetSymbolResult result = response.getResult();
System.out.println(result.getSymbol());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.7.6 getDecimals

The `getDecimals` interface is used to get the precision of the contract Token.

The method call is as follows:

```
Ctp10TokenGetDecimalsResponse getDecimals (Ctp10TokenGetDecimalsRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract account address to be queried

The response data is shown in the following table:

Parameter	Type	Description
decimals	Integer	Contract token precision

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters

Ctp10TokenGetDecimalsRequest request = new Ctp10TokenGetDecimalsRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// Call the getDecimals interface
Ctp10TokenGetDecimalsResponse response = sdk.getTokenService().getDecimals(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetDecimalsResult result = response.getResult();
System.out.println(result.getDecimals());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.7.7 getTotalSupply

The `getTotalSupply` interface is used to get the total supply of contract tokens.

The method call is as follows:

```
Ctp10TokenGetTotalSupplyResponse getTotalSupply(Ctp10TokenGetTotalSupplyRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract account address to be queried

The response data is shown in the following table:

Parameter	Type	Description
totalSupply	String	Total supply of contract Token

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
Ctp10TokenGetTotalSupplyRequest request = new Ctp10TokenGetTotalSupplyRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// Call the getDecimals interface
Ctp10TokenGetTotalSupplyResponse response = sdk.getTokenService().
↪getTotalSupply(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetTotalSupplyResult result = response.getResult();
```

(continues on next page)

(续上页)

```
System.out.println(result.getTotalSupply());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.7.8 getBalance-Ctp10Token

The `getBalance-Ctp10Token` interface is used to get the account balance of the contract Token owner.

The method call is as follows:

```
Ctp10TokenGetBalanceResponse getBalance(Ctp10TokenGetBalanceRequest)
```

The request parameters are shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract account address to be queried
tokenOwner	String	Required, the account address holding the contract Token

The response data is shown in the following table:

Parameter	Type	Description
balance	Long	Token balance

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_TOKENOWNER_ERRPR	11035	Invalid token owner
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
Ctp10TokenGetBalanceRequest request = new Ctp10TokenGetBalanceRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");
request.setTokenOwner("buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp");

// Call the getBalance interface
Ctp10TokenGetBalanceResponse response = sdk.getTokenService().getBalance(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetBalanceResult result = response.getResult();
System.out.println(result.getBalance());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.8 Contract Services

Contract Services provide contract-related interfaces and currently have four interfaces: `checkValid`, `getInfo`, `getAddress`, and `call`.

2.8.1 checkValid

The `checkValid` interface is used to check the validity of the contract account.

The method call is as follows:

```
ContractCheckValidResponse checkValid(ContractCheckValidRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract account address to be tested

The response data is shown in the following table:

Parameter	Type	Description
isValid	Boolean	Whether the response data is valid

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
ContractCheckValidRequest request = new ContractCheckValidRequest();
request.setContractAddress("buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea");

// Call the getDecimals interface
ContractCheckValidResponse response = sdk.getContractService().checkValid(request);
if (response.getErrorCode() == 0) {
ContractCheckValidResult result = response.getResult();
System.out.println(result.getValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.8.2 getInfo

The `getInfo` interface is used to query the contract code.

The method call is as follows:

```
ContractGetInfoResponse getInfo (ContractGetInfoRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
contractAddress	String	Contract account address to be queried

The response data is shown in the following table:

Parameter	Type	Description
contract	ContractInfo	Contract info

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_CONTRACTADDRESS _ERROR	11037	Invalid contract address
CONTRACTADDRESS_NOT_CON TRACTAC-COUNT_ERROR	11038	contractAddress is not a contract account
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
ContractGetInfoRequest request = new ContractGetInfoRequest();
request.setContractAddress("buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea");

// Call the getInfo interface
ContractGetInfoResponse response = sdk.getContractService().getInfo(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.8.2.1 ContractInfo

The specific information of ContractInfo is shown in the following table:

Member	Type	Description
type	Integer	Contract type, default is 0
payload	String	Contract code

2.8.3 getAddress

The `getAddress` interface is used to query the contract address.

The method call is as follows:

```
ContractGetAddressResponse getInfo (ContractGetAddressRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
hash	String	The hash used to create a contract transaction

The response data is shown in the following table:

Parameter	Type	Description
contractAddressList	List (ContractAddressInfo)	Contract address list

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_HASH_ERROR	11055	Invalid transaction hash
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
ContractGetAddressRequest request = new ContractGetAddressRequest();
request.setHash("44246c5ba1b8b835a5cbc29bdc9454cdb9a9d049870e41227f2dcfbcf7a07689");

// Call the getAddress interface
ContractGetAddressResponse response = sdk.getContractService().getAddress(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.8.3.1 ContractAddressInfo

The specific information of ContractAddressInfo is shown in the following table:

Member	Type	Description
contractAddress	String	Contract address
operationIndex	Integer	The subscript of the operation

2.8.4 call

The call `interface` is used to debug the contract code.

The method call is as follows:

```
ContractCallesponse call(ContractCallRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
sourceAddress	String	Optional, the account address to trigger the contract
contractAddress	String	Optional, the contract account address and code cannot be empty at the same time
code	String	Optional, the contract code and contractAddress cannot be empty at the same time, length limit [1, 64]
input	String	Optional, input parameter for the contract
contractBalance	String	Optional, the initial BU balance given to the contract, unit MO, 1 BU = 10^8 MO, size limit [1, Long.MAX_VALUE]
optType	Integer	Required, 0: Call the read/write interface of the contract init, 1: Call the read/write interface of the contract main, 2: Call the read-only interface query
feeLimit	Long	Minimum fee required for the transaction, size limit [1, Long.MAX_VALUE]
gasPrice	Long	Transaction fuel price, size limit [1000, Long.MAX_VALUE]

The response data is shown in the following table:

Parameter	Type	Description
logs	JSONObject	Log information
queryRets	JSONArray	Query the result set
stat	*ContractStat*	Contract resource occupancy
txs	*TransactionEnvs*[]	Transaction set

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_SOURCEADDRESS_E RROR	11002	Invalid sourceAddress
INVALID_CONTRACTADDRESS _ERROR	11037	Invalid contract address
CONTRACTADDRESS_CODE_BO TH_NULL_ERROR	11063	ContractAddress and code cannot be empty at the same time
INVALID_OPTTYPE_ERROR	11064	OptType must be between 0 and 2
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
ContractCallRequest request = new ContractCallRequest();
request.setCode("\"use strict\";log(undefined);function query() {␣
↪getBalance(thisAddress); }");
request.setFeeLimit(1000000000L);
request.setOptType(2);

// Call the ``call`` interface
ContractCallResponse response = sdk.getContractService().call(request);
if (response.getErrorCode() == 0) {
```

(continues on next page)

(续上页)

```
ContractCallResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.8.4.1 ContractStat

The specific information of ContractStat is shown in the following table:

Member	Type	Description
applyTime	Long	Receipt time
memoryUsage	Long	Memory footprint
stackUsage	Long	Stack occupancy
step	Long	Steps needed

2.8.4.2 TransactionEnvs

The specific information of TransactionEnvs is shown in the following table:

Member	Type	Description
transactionEnv	*TransactionEnv*	Transaction

2.8.4.3 TransactionEnv

The specific information of TransactionEnv is shown in the following table:

Member	Type	Description
transaction	*TransactionInfo*	Transaction content
trigger	*ContractTrigger*	Contract trigger

2.8.4.4 TransactionInfo

The specific information of TransactionInfo is shown in the following table:

Member	Type	Description
sourceAddress	String	The source account address initiating the transaction
feeLimit	Long	Minimum fees required for the transaction
gasPrice	Long	Transaction fuel price
nonce	Long	Transaction serial number
operations	Operation[]	Operation list

2.8.4.5 ContractTrigger

The specific information of ContractTrigger is shown in the following table:

Member	Type	Description
transaction	*TriggerTransaction*	Trigger transactions

2.8.4.6 Operation

The specific information of Operation is shown in the following table:

Member	Type	Description
type	Integer	Operation type
sourceAddress	String	The source account address initiating operations
metadata	String	Note
createAccount	*OperationCreateAccount*	Operation of creating accounts
issueAsset	*OperationIssueAsset*	Operation of issuing assets
payAsset	*OperationPayAsset*	Operation of transferring assets
payCoin	*OperationPayCoin*	Operation of sending BU
setMetadata	*OperationSetMetadata*	Operation of setting metadata
setPrivilege	*OperationSetPrivilege*	Operation of setting account privilege
log	*OperationLog*	Record logs

2.8.4.7 TriggerTransaction

The specific information of TriggerTransaction is shown in the following table:

Member	Type	Description
hash	String	Transaction hash

2.8.4.8 OperationCreateAccount

The specific information of OperationCreateAccount is shown in the following table:

Member	Type	Description
destAddress	String	Target account address
contract	Contract	Contract info
priv	*Priv*	Account privilege
metadata	*MetadataInfo*[]	Account
initBalance	Long	Account assets, unit MO, 1 BU = 10^8 MO,
initInput	String	The input parameter for the init function of the contract

2.8.4.9 Contract

The specific information of Contract is shown in the following table:

Member	Type	Description
type	Integer	The contract language is not assigned value by default
payload	String	The contract code for the corresponding language

2.8.4.10 MetadataInfo

The specific information of MetadataInfo is shown in the following table:

Member	Type	Description
key	String	metadata keyword
value	String	metadata content
version	Long	metadata version

2.8.4.11 OperationIssueAsset

The specific information of OperationIssueAsset is shown in the following table:

Member	Type	Description
code	String	Assets encoding
assetAmount	Long	Assets amount

2.8.4.12 OperationPayAsset

The specific information of OperationPayAsset is shown in the following table:

Member	Type	Description
destAddress	String	The target account address to which the asset is transferred
asset	*AssetInfo*	Account asset
input	String	Input parameters for the main function of the contract

2.8.4.13 OperationPayCoin

The specific information of OperationPayCoin is shown in the following table:

Member	Type	Description
destAddress	String	The target account address to which the asset is transferred
buAmount	Long	BU amounts to be transferred
input	String	Input parameters for the main function of the contract

2.8.4.14 OperationSetMetadata

The specific information of OperationSetMetadata is shown in the following table:

Member	Type	Description
key	String	metadata keyword
value	String	metadata content
version	Long	metadata version
deleteFlag	boolean	Whether to delete metadata

2.8.4.15 OperationSetPrivilege

The specific information of OperationSetPrivilege is shown in the following table:

2.8.4.16 OperationLog

The specific information of OperationLog is shown in the following table:

Member	Type	Description
topic	String	Log theme
data	String[]	Log content

2.9 Transaction Services

Transaction Services provide transaction-related interfaces and currently have five interfaces: `buildBlob`, `evaluateFee`, `sign`, `submit`, and `getInfo`.

2.9.1 buildBlob

The `buildBlob` interface is used to serialize transactions and generate transaction blob strings for network transmission.

Before you can call buildBlob, you need to build some operations. There are 16 operations: `AccountActivateOperation`, `AccountSetMetadataOperation`, `AccountSetPrivilegeOperation`, `AssetIssueOperation`, `AssetSendOperation`, `BUSendOperation`, `TokenIssueOperation`, `TokenTransferOperation`, `TokenTransferFromOperation`, `TokenApproveOperation`, `TokenAssignOperation`, `TokenChangeOwnerOperation`, `ContractCreateOperation`, `ContractInvokeByAssetOperation`, `ContractInvokeByBUOperation`, and `LogCreateOperation`.

The method call is as follows:

```
TransactionBuildBlobResponse buildBlob(TransactionBuildBlobRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
sourceAddress	String	Required, the source account address initiating the operation
nonce	Long	Required, the transaction serial number to be initiated, add 1 in the function, size limit [1, Long.MAX_VALUE]
gasPrice	Long	Required, transaction gas price, unit MO, 1 BU = 10^8 MO, size limit [1000, Long.MAX_VALUE]
feeLimit	Long	Required, the minimum fees required for the transaction, unit MO, 1 BU = 10^8 MO, size limit [1, Long.MAX_VALUE]
operation	BaseOperation[]	Required, list of operations to be committed which cannot be empty
ceilLedgerSeq	Long	Optional, set a value which will be combined with the current block height to restrict transactions. If transactions do not complete within the set value plus the current block height, the transactions fail. The value you set must be greater than 0. If the value is set to 0, no limit is set.
metadata	String	Optional, note

The response data is shown in the following table:

Parameter	Type	Description
transactionBlob	String	Serialized transaction hex string
hash	String	Transaction hash

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_SOURCEADDRESS_E RROR	11002	Invalid sourceAddress
INVALID_NONCE_ERROR	11048	Nonce must be between 1 and Long.MAX_VAL
INVALID_DESTADDRESS_ERR OR	11003	Invalid destAddress
INVALID_INITBALANCE_ERR OR	11004	InitBalance must be between 1 and Long.MAX_
SOURCEADDRESS_EQUAL_DES TADDRESS_ERROR	11005	SourceAddress cannot be equal to destAddress
INVALID_ISSUE_AMMOUNT_E RROR	11008	AssetAmount that will be issued must be betwee
INVALID_DATAKEY_ERROR	11011	The length of key must be between 1 and 1024
INVALID_DATAVALUE_ERROR	11012	The length of value must be between 0 and 2560
INVALID_DATAVERSION_ERR OR	11013	The version must be equal to or greater than 0
INVALID_MASTERWEIGHT _ERROR	11015	MasterWeight must be between 0 and (Integer.M
INVALID_SIGNER_ADDRESS _ERROR	11016	Invalid signer address
INVALID_SIGNER_WEIGHT _ERROR	11017	Signer weight must be between 0 and (Integer.M
INVALID_TX_THRESHOLD_ER ROR	11018	TxThreshold must be between 0 and Long.MAX
INVALID_OPERATION_TYPE _ERROR	11019	Operation type must be between 1 and 100
INVALID_TYPE_THRESHOLD _ERROR	11020	TypeThreshold must be between 0 and Long.MA
INVALID_ASSET_CODE _ERROR	11023	The length of key must be between 1 and 64
INVALID_ASSET_AMOUNT_ER ROR	11024	AssetAmount must be between 0 and Long.MA
INVALID_BU_AMOUNT_ERROR	11026	BuAmount must be between 0 and Long.MAX_
INVALID_ISSUER_ADDRESS _ERROR	11027	Invalid issuer address
NO_SUCH_TOKEN_ERROR	11030	No such token
INVALID_TOKEN_NAME_ERRO R	11031	The length of token name must be between 1 an
INVALID_TOKEN_SYMBOL_ER ROR	11032	The length of symbol must be between 1 and 10
INVALID_TOKEN_DECIMALS _ERROR	11033	Decimals must be between 0 and 8
INVALID_TOKEN_TOTALSUPP LY_ERROR	11034	TotalSupply must be between 1 and Long.MAX_
INVALID_TOKENOWNER_ERRP R	11035	Invalid token owner
INVALID_CONTRACTADDRESS _ERROR	11037	Invalid contract address
CONTRACTADDRESS_NOT_CON TRACTACCOUNT_ERROR	11038	ContractAddress is not a contract account
INVALID_TOKEN_AMOUNT_ER ROR	11039	Token amount must be between 1 and Long.MA
SOURCEADDRESS_EQUAL_CON TRACTADDRESS_ERROR	11040	SourceAddress cannot be equal to contractAddr
INVALID_FROMADDRESS_ERR OR	11041	Invalid fromAddress
FROMADDRESS_EQUAL_DESTA DDRESS_ERROR	11042	FromAddress cannot be equal to destAddress
INVALID_SPENDER_ERROR	11043	Invalid spender
PAYLOAD_EMPTY_ERROR	11044	Payload cannot be empty
INVALID_LOG_TOPIC _ERROR	11045	The length of key must be between 1 and 128
INVALID_LOG_DATA _ERROR	11046	The length of value must be between 1 and 1024
INVALID_CONTRACT_TYPE_E RROR	11047	Type must be equal to or greater than 0
INVALID_NONCE_ERROR	11048	Nonce must be between 1 and Long.MAX_VAL
INVALID_ GASPRICE_ERROR	11049	GasPrice must be between 1000 and Long.MAX
INVALID_FEELIMIT_ERROR	11050	FeeLimit must be between 1 and Long.MAX_V
OPERATIONS_EMPTY_ERROR	11051	Operations cannot be empty
INVALID_CEILLEDGERSEQ_E RROR	11052	CeilLedgerSeq must be equal or greater than 0
OPERATIONS_ONE_ERROR	11053	One of the operations cannot be resolved

表 1 – 续上页

Exception	Error Code	Description
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize variables
String senderAddresss = "buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea";
String destAddress = "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw";
Long buAmount = ToBaseUnit.BU2MO("10.9");
Long gasPrice = 1000L;
Long feeLimit = ToBaseUnit.BU2MO("0.01");
Long nonce = 1L;

// Build the sendBU operation
BUSendOperation operation = new BUSendOperation();
operation.setSourceAddress(senderAddresss);
operation.setDestAddress(destAddress);
operation.setAmount(buAmount);

// Initialize request parameters
TransactionBuildBlobRequest request = new TransactionBuildBlobRequest();
request.setSourceAddress(senderAddresss);
request.setNonce(nonce);
request.setFeeLimit(feeLimit);
request.setGasPrice(gasPrice);
request.addOperation(operation);

// Call the buildBlob interface
String transactionBlob = null;
TransactionBuildBlobResponse response = sdk.getTransactionService().
↪buildBlob(request);
if (response.getErrorCode() == 0) {
TransactionBuildBlobResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.9.1.1 BaseOperation

BaseOperation is the base class for all operations in the `buildBlob` interface. The following table describes BaseOperation:

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
metadata	String	Optional, note

2.9.1.2 AccountActivateOperation

AccountActivateOperation inherits from BaseOperation, and feeLimit is currently fixed at 0.01 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
destAddress	String	Required, target account address
initBalance	Long	Required, initialize the asset, unit MO, 1 BU = 10^8 MO, size (0, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.3 AccountSetMetadataOperation

AccountSetMetadataOperation is inherited from BaseOperation, and feeLimit is currently fixed at 0.01 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
key	String	Required, metadata keyword, length limit [1, 1024]
value	String	Required, metadata content, length limit [0, 256000]
version	Long	Optional, metadata version
deleteFlag	Boolean	Optional, whether to delete metadata
metadata	String	Optional, note

2.9.1.4 AccountSetPrivilegeOperation

AccountSetPrivilegeOperation inherits from BaseOperation, and feeLimit is currently fixed at 0.01 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
masterWeight	String	Optional, account weight, size limit [0, (Integer.MAX_VALUE * 2L + 1)]
signers	Signer[]	Optional, signer weight list
txThreshold	String	Optional, transaction threshold, size limit [0, Long.MAX_VALUE]
typeThreshold	TypeThreshold[]	Optional, specify transaction threshold
metadata	String	Optional, note

2.9.1.5 AssetIssueOperation

AssetIssueOperation inherits from BaseOperation, and feeLimit is currently fixed at 50.01 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
code	String	Required, asset code, length limit [1, 64]
assetAmount	Long	Required, number of asset issues, size limit [0, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.6 AssetSendOperation

AssetSendOperation inherits from BaseOperation, and feeLimit is currently fixed at 0.01 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
destAddress	String	Required, target account address
code	String	Required, asset code, length limit [1, 64]
issuer	String	Required, account address issuing assets
assetAmount	Long	Required, asset quantity, size limit [0, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.7 BUSendOperation

BUSendOperation inherits from BaseOperation, feeLimit is currently (2018.07.26) fixed at 0.01 BU.

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
destAddress	String	Required, target account address
buAmount	Long	Required, amount of asset issued, size limit [0, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.8 Ctp10TokenIssueOperation

Ctp10TokenIssueOperation inherits from BaseOperation, and feeLimit is currently fixed at 10.08 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
initBalance	Long	Required, initial assets for the contract account, unit MO, 1 BU = 10^8 MO, size limit [1, max(64)]
name	String	Required, ctp10Token name, length limit [1, 1024]
symbol	String	Required, ctp10Token symbol, length limit [1, 1024]
decimals	Integer	Required, the precision of the number of ctp10Token, size limit [0, 8]
supply	String	Required, total supply issued by ctp10Token (without precision), size limit [1, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.9 Ctp10TokenTransferOperation

Ctp10TokenTransferOperation inherits from BaseOperation, and feeLimit currently (2018.07.26) is fixed at 0.02 BU.

Member	Type	Description
sourceAddress	String	Optional, account address holding the contract token
contractAddress	String	Required, contract account address
destAddress	String	Required, target account address to which token is transferred
tokenAmount	String	Required, amount of tokens to be transferred, size limit [1, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.10 TokenTransferFromOperation

TokenTransferFromOperation inherits from BaseOperation, and feeLimit is currently fixed at 0.02 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
contractAddress	String	Required, contract account address
fromAddress	String	Required, source account address from which token is transferred
destAddress	String	Required, target account address to which token is transferred
tokenAmount	String	Required, amount of ctp10Tokens to be transferred, size limit [1, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.11 Ctp10TokenApproveOperation

Ctp10TokenApproveOperation inherits from BaseOperation, and feeLimit is currently fixed at 0.02 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, account address holding the contract token
contractAddress	String	Required, contract account address
spender	String	Required, authorized account address
tokenAmount	String	Required, the number of authorized ctp10Tokens to be transferred, size limit [1, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.12 Ctp10TokenAssignOperation

Ctp10TokenAssignOperation inherits from BaseOperation, feeLimit is currently (2018.07.26) fixed at 0.02 BU.

Member	Type	Description
sourceAddress	String	Optional, account address holding the contract token
contractAddress	String	Required, contract account address
destAddress	String	Required, target account address to be assigned
tokenAmount	String	Required, amount of ctp10Tokens to be allocated, size limit [1, Long.MAX_VALUE]
metadata	String	Optional, note

2.9.1.13 Ctp10TokenChangeOwnerOperation

Ctp10TokenChangeOwnerOperation inherits from BaseOperation, and feeLimit is currently fixed at 0.02 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, account address holding the contract token
contractAddress	String	Required, contract account address
tokenOwner	String	Required, target account address to which token is transferred
metadata	String	Optional, note

2.9.1.14 ContractCreateOperation

ContractCreateOperation inherits from BaseOperation, and feeLimit is currently fixed at 10.01 BU (2018.07.26).

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
initBalance	Long	Required, initial asset for contract account, unit MO, 1 BU = 10^8 MO, size limit [1, Long.MAX_VALUE]
type	Integer	Optional, the language of the contract, the default is 0
payload	String	Required, contract code for the corresponding language
initInput	String	Optional, the input parameters of the init method in the contract code
metadata	String	Optional, note

2.9.1.15 ContractInvokeByAssetOperation

ContractInvokeByAssetOperation inherits from BaseOperation. FeeLimit requires to add fees according to the execution of the transaction in the contract. First, the transaction fee is initiated. At present the fee (2018.07.26) is 0.01BU, and then the transaction in the contract also requires the transaction initiator to add the transaction fees.

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
contractAddress	String	Required, contract account address
code	String	Optional, asset code, length limit [0, 1024]; when it is empty, only the contract is triggered
issuer	String	Optional, the account address issuing assets; when it is null, only trigger the contract
assetAmount	Long	Optional, asset quantity, size limit [0, Long.MAX_VALUE], when it is 0, only trigger the contract
input	String	Optional, the input parameter of the main() method for the contract to be triggered
metadata	String	Optional, note

2.9.1.16 ContractInvokeByBUOperation

ContractInvokeByBUOperation inherits from BaseOperation. FeeLimit requires to add fees according to the execution of the transaction in the contract. First, the transaction fee is initiated. At present the fee (2018.07.26) is 0.01BU, and then the transaction in the contract also requires the transaction initiator to add the transaction fees.

Member	Type	Description
sourceAddress	String	Optional, source account address of the operation
contractAddress	String	Required, contract account address
buAmount	Long	Optional, number of asset issues, size limit [0, Long.MAX_VALUE], when it is 0 only triggers the contract
input	String	Optional, the input parameter of the main() method for the contract to be triggered
metadata	String	Optional, note

2.9.2 evaluateFee

The evaluateFee `interface` implements the cost estimate for the transaction.

The method call is as follows:

```
TransactionEvaluateFeeResponse evaluateFee (TransactionEvaluateFeeRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
sourceAddress	String	Required, the source account address issuing the operation
nonce	Long	Required, transaction serial number to be initiated, size limit [1, Long.MAX_VALUE]
operation	BaseOperation[]	Required, list of operations to be committed which cannot be empty
signtureNumber	Integer	Optional, the number of people to sign, the default is 1, size limit [1, Integer.MAX_VALUE]
ceilLedgerSeq	Long	Optional, set a value which will be combined with the current block height to restrict transactions. If transactions do not complete within the set value plus the current block height, the transactions fail. The value you set must be greater than 0. If the value is set to 0, no limit is set.
metadata	String	Optional, note

The response data is shown in the following table:

Parameter	Type	Description
txs	*TestTx*[]	Evaluation transaction set

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_SOURCEADDRESS_ERROR	11002	Invalid sourceAddress
INVALID_NONCE_ERROR	11045	Nonce must be between 1 and Long.MAX_VALUE
OPERATIONS_EMPTY_ERROR	11051	Operations cannot be empty
OPERATIONS_ONE_ERROR	11053	One of operations cannot be resolved
INVALID_SIGNATURENUMBER_ERROR	11054	SignagureNumber must be between 1 and Integer.MAX_VALU E
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize variables
String senderAddresss = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
```

(continues on next page)

(续上页)

```
String destAddress = "buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea";
Long buAmount = ToBaseUnit.BU2MO("10.9");
Long gasPrice = 1000L;
Long feeLimit = ToBaseUnit.BU2MO("0.01");
Long nonce = 51L;

// Build the sendBU operation
BUSendOperation buSendOperation = new BUSendOperation();
buSendOperation.setSourceAddress(senderAddresss);
buSendOperation.setDestAddress(destAddress);
buSendOperation.setAmount(buAmount);

// Initialize request parameters for transaction evaluation

TransactionEvaluateFeeRequest request = new TransactionEvaluateFeeRequest();
request.addOperation(buSendOperation);
request.setSourceAddress(senderAddresss);
request.setNonce(nonce);
request.setSignatureNumber(1);
request.setMetadata(HexFormat.byteToHex("evaluate fees".getBytes()));

// Call the evaluateFee interface
TransactionEvaluateFeeResponse response = sdk.getTransactionService().
↪evaluateFee(request);
if (response.getErrorCode() == 0) {
TransactionEvaluateFeeResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.9.2.1 TestTx

The specific information of TestTx is shown in the following table:

Member	Type	Description
transactionEnv	*TestTransactionFees*	Assess transaction costs

2.9.2.2 TestTransactionFees

The specific information of TestTransactionFees is shown in the following table:

Member	Type	Description
transactionFees	*TransactionFees*	Transaction fees

2.9.2.3 TransactionFees

The specific information of TransactionFees is shown in the following table:

Member	Type	Description
feeLimit	Long	Minimum fees required for the transaction
gasPrice	Long	Transaction gas price

2.9.3 sign

The `sign` interface is used to implement the signature of the transaction.

The method call is as follows:

```
TransactionSignResponse sign(TransactionSignRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
blob	String	Required, pending transaction blob to be signed
privateKeys	String[]	Required, private key list

The response data is shown in the following table:

Parameter	Type	Description
signatures	Signature	Signed data list

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_BLOB_ERROR	11056	Invalid blob
PRIVATEKEY_NULL_ERROR	11057	PrivateKeys cannot be empty
PRIVATEKEY_ONE_ERROR	11058	One of the privateKeys is invalid
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
String issuePrivateKey = "privbyQCRp7DLqKtRFCqKQJr81TurTqG6UKXMMtGAmPG3abcM9XHjWvq";
String []signerPrivateKeyArr = {issuePrivateKey};
String transactionBlob =
↪"0A246275516E6E5545425245773268423670574847507A77616E5837643238786B364B566370102118C0843D20E8073A5
↪";
TransactionSignRequest request = new TransactionSignRequest();
request.setBlob(transactionBlob);
for (int i = 0; i < signerPrivateKeyArr.length; i++) {
request.addPrivateKey(signerPrivateKeyArr[i]);
}
TransactionSignResponse response = sdk.getTransactionService().sign(request);
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

2.9.3.1 Signature

The specific information of signature is shown in the following table:

Member	Type	Description
signData	Long	Data signed
publicKey	Long	Public key

2.9.4 submit

The `submit` interface is used to implement the submission of the transaction.

The method call is as follows:

```
TransactionSubmitResponse submit(TransactionSubmitRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
blob	String	Required, transaction blob
signature	Signature[]	Required, signature list

The response data is shown in the following table:

Parameter	Type	Description
hash	String	Transaction hash

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_BLOB_ERROR	11056	Invalid blob
SIGNATURE_EMPTY_ERROR	11067	The signatures cannot be empty
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
String transactionBlob =
↪"0A246275516E6E554542524577326842367057484750 7A77616E5837643238786B364B566370102118C0843D20E8073A5
↪";
Signature signature = new Signature();
signature.setSignData(
↪"D2B5E3045F2C1B7D363D4F58C1858C30ABBBB0F41E4B2E18AF680553CA9C3689078E215C097086E47A4393BCA715C7A5D2
↪");
signature.setPublicKey(
↪"b0011765082a9352e04678ef38d38046dc01306edef676547456c0c23e270aaed7ffe9e31477");
TransactionSubmitRequest request = new TransactionSubmitRequest();
request.setTransactionBlob(transactionBlob);
request.addSignature(signature);

// Call the submit interface
```

(continues on next page)

(续上页)

```
TransactionSubmitResponse response = sdk.getTransactionService().submit(request);
if (0 == response.getErrorCode()) { // Committed transactions successfully
System.out.println(JSON.toJSONString(response.getResult(), true));
} else{
System.out.println("error: " + response.getErrorDesc());
}
```

2.9.5 getInfo

The `getInfo` interface is used to implement query transactions based on transaction hashes.

The method call is as follows:

```
TransactionGetInfoResponse getInfo (TransactionGetInfoRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
hash	String	Transaction hash

The response data is shown in the following table:

Parameter	Type	Description
totalCount	Long	Total number of transactions returned
transactions	*TransactionHistory* []	Transaction content

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_HASH_ERROR	11055	Invalid transaction hash
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
String txHash = "1653f54fbba1134f7e35acee49592a7c29384da10f2f629c9a214f6e54747705";
TransactionGetInfoRequest request = new TransactionGetInfoRequest();
request.setHash(txHash);

// Call the getInfo interface
TransactionGetInfoResponse response = sdk.getTransactionService().getInfo(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.9.5.1 TransactionHistory

The specific information of TransactionHistory is shown in the following table:

Member	Type	Description
actualFee	String	Actual transaction cost
closeTime	Long	Transaction closure time
errorCode	Long	Transaction error code
errorDesc	String	Transaction description
hash	String	Transaction hash
ledgerSeq	Long	Block serial number
transaction	*TransactionInfo*	List of transaction contents
signatures	*Signature*[]	Signature list
txSize	Long	Transaction size

2.10 Block Services

Block services provide block-related interfaces. There are currently 11 interfaces: `getNumber`, `checkStatus`, `getTransactions`, `getInfo`, `getLatestInfo`, `getValidators`, `getLatestValidators`, `getReward`, `getLatestReward`, `getFees` and `getLatestFees`.

2.10.1 getNumber

The `getNumber` interface is used to query the latest block height.

The method call is as follows:

```
BlockGetNumberResponse getNumber();
```

The response data is shown in the following table:

Parameter	Type	Description
header	BlockHeader	Block head
blockNumber	Long	The latest block height,corresponding to the underlying field sequence

The error code is shown in the following table:

Exception	Error Code	Description
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Call the getNumber interface
BlockGetNumberResponse response = sdk.getBlockService().getNumber();
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.2 checkStatus

The `checkStatus` interface is used to check if the local node block is synchronized.

The method call is as follows:

```
BlockCheckStatusResponse checkStatus();
```

The response data is shown in the following table:

Parameter	Type	Description
isSynchronous	Boolean	Whether the block is synchronized

The error code is shown in the following table:

Exception	Error Code	Description
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Call the checkStatus interface
BlockCheckStatusResponse response = sdk.getBlockService().checkStatus();
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.3 getTransactions

The `getTransactions` interface is used to query all transactions at the specified block height.

The method call is as follows:

```
BlockGetTransactionsResponse getTransactions(BlockGetTransactionsRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
blockNumber	Long	Required, the height of the block to be queried must be greater than 0

The response data is shown in the following table:

Parameter	Type	Description
totalCount	Long	Total number of transactions returned
transactions	TransactionHistory[]	Transaction content

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
Long blockNumber = 617247L;// Block 617247
BlockGetTransactionsRequest request = new BlockGetTransactionsRequest();
request.setBlockNumber(blockNumber);

// Call the getTransactions interface
BlockGetTransactionsResponse response = sdk.getBlockService().
↪getTransactions(request);
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.4 getInfo

The `getInfo` interface is used to obtain block information.

The method call is as follows:

```
BlockGetInfoResponse getInfo(BlockGetInfoRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
blockNumber	Long	Required, the height of the block to be queried

The response data is shown in the following table:

Parameter	Type	Description
closeTime	Long	Block closure time
number	Long	Block height
txCount	Long	Total transactions amount
version	String	Block version

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
BlockGetInfoRequest request = new BlockGetInfoRequest();
request.setBlockNumber(629743L);

// Call the getInfo interface
BlockGetInfoResponse response = sdk.getBlockService().getInfo(request);
if (response.getErrorCode() == 0) {
BlockGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.5 getLatestInfo

The `getLatestInfo` interface is used to get the latest block information.

The method call is as follows:

```
BlockGetLatestInfoResponse getLatestInfo();
```

The response data is shown in the following table:

Parameter	Type	Description
closeTime	Long	Block closure time
number	Long	Block height,corresponding to the underlying field seq
txCount	Long	Total transactions amount
version	String	Block version

The error code is shown in the following table:

Exception	Error Code	Description
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Call the getLatestInfo interface
BlockGetLatestInfoResponse response = sdk.getBlockService().getLatestInfo();
if (response.getErrorCode() == 0) {
BlockGetLatestInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.6 getValidators

The `getValidators` interface is used to get the number of all the authentication nodes in the specified block.

The method call is as follows:

```
BlockGetValidatorsResponse getValidators(BlockGetValidatorsRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
blockNumber	Long	Required, the height of the block to be queried must be greater than 0

The response data is shown in the following table:

Parameter	Type	Description
validators	ValidatorInfo[]	Validators list

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
BlockGetValidatorsRequest request = new BlockGetValidatorsRequest();
request.setBlockNumber(629743L);

// Call the getValidators interface
BlockGetValidatorsResponse response = sdk.getBlockService().getValidators(request);
if (response.getErrorCode() == 0) {
BlockGetValidatorsResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.6.1 ValidatorInfo

The specific information of ValidatorInfo is shown in the following table:

Member	Type	Description
address	String	Consensus node address
plegeCoinAmount	Long	Validators' deposit

2.10.7 getLatestValidators

The `getLatestValidators` interface is used to get the number of all validators in the latest block.

The method call is as follows:

```
BlockGetLatestValidatorsResponse getLatestValidators();
```

The response data is shown in the following table:

Parameter	Type	Description
validators	ValidatorInfo[]	Validators list

The error code is shown in the following table:

Exception	Error Code	Description
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Call the getLatestValidators interface
BlockGetLatestValidatorsResponse response = sdk.getBlockService().
↪getLatestValidators();
if (response.getErrorCode() == 0) {
BlockGetLatestValidatorsResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.8 getReward

The `getReward` interface is used to retrieve the block reward and valicator node rewards in the specified block. The method call is as follows:

```
BlockGetRewardResponse getReward(BlockGetRewardRequest);
```

The request parameters are shown in the following table:

Parameter	Type	Description
blockNumber	Long	Required, the height of the block to be queried must be greater than 0

The response data is shown in the following table:

Parameter	Type	Description
blockReward	Long	Block rewards
validatorsReward	ValidatorReward[]	Validators rewards

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
BlockGetRewardRequest request = new BlockGetRewardRequest();
request.setBlockNumber(629743L);

// Call the getReward interface
BlockGetRewardResponse response = sdk.getBlockService().getReward(request);
if (response.getErrorCode() == 0) {
BlockGetRewardResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.8.1 ValidatorReward

The specific information of ValidatorReward is shown in the following table:

Member	Type	Description
validator	String	Validator address
reward	Long	Validator reward

2.10.9 getLatestReward

The `getLatestReward` interface gets the block rewards and validator rewards in the latest block. The method call is as follows:

```
BlockGetLatestRewardResponse getLatestReward();
```

The response data is shown in the following table:

Parameter	Type	Description
blockReward	Long	Block rewards
validatorsReward	*ValidatorReward*[]	Validator rewards

The error code is shown in the following table:

Exception	Error Code	Description
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Call the getLatestReward interface
BlockGetLatestRewardResponse response = sdk.getBlockService().getLatestReward();
if (response.getErrorCode() == 0) {
BlockGetLatestRewardResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.10 getFees

The `getFees` interface gets the minimum asset limit and fuel price of the account in the specified block.

The method call is as follows:

```
BlockGetFeesResponse getFees(BlockGetFeesRequest);
```

The request parameter is shown in the following table:

Parameter	Type	Description
blockNumber	Long	Required, the height of the block to be queried must be greater than 0

The response data is shown in the following table:

Parameter	Type	Description
fees	Fees	Fees

The error code is shown in the following table:

Exception	Error Code	Description
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Initialize request parameters
BlockGetFeesRequest request = new BlockGetFeesRequest();
request.setBlockNumber(629743L);

// Call the getFees interface
BlockGetFeesResponse response = sdk.getBlockService().getFees(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.10.10.1 Fees

The specific information of Fees is shown in the following table:

Member	Type	Description
baseReserve	Long	Minimum asset limit for the account
gasPrice	Long	Transaction fuel price, unit MO, 1 BU = 10^8 MO

2.10.11 getLatestFees

The `getLatestFees` interface is used to obtain the minimum asset limit and fuel price of the account in the latest block.

The method call is as follows:

```
BlockGetLatestFeesResponse getLatestFees();
```

The response data is shown in the following table:

Parameter	Type	Description
fees	Fees	Fees

The error code is shown in the following table:

Exception	Error Code	Description
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

The specific example is as follows:

```
// Call the getLatestFees interface
BlockGetLatestFeesResponse response = sdk.getBlockService().getLatestFees();
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

2.11 Error Code

The following table describes the error messages that may appear.

Exception	Error code	Description
ACCOUNT_CREATE_ERROR	11001	Failed to create the account
INVALID_SOURCEADDRESS _ERROR	11002	Invalid sourceAddress
INVALID_DESTADDRESS_E RROR	11003	Invalid destAddress
INVALID_INITBALANCE_E RROR	11004	InitBalance must be between 1 and Long.MAX_
SOURCEADDRESS_EQUAL_D ESTADDRESS_ERROR	11005	SourceAddress cannot be equal to destAddress
INVALID_ADDRESS_ERROR	11006	Invalid address
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
INVALID_ISSUE_AMOUNT _ERROR	11008	Amount of the token to be issued must be betwe
NO_ASSET_ERROR	11009	The account does not have the asset
NO_METADATA_ERROR	11010	The account does not have the metadata
INVALID_DATAKEY_ERROR	11011	The length of key must be between 1 and 1024
INVALID_DATAVALUE_ERR OR	11012	The length of value must be between 0 and 2560
INVALID_DATAVERSION_E RROR	11013	The version must be equal to or greater than 0
INVALID_MASTERWEIGHT _ERROR	11015	MasterWeight must be between 0 and (Integer.M
INVALID_SIGNER_ADDRES S_ERROR	11016	Invalid signer address
INVALID_SIGNER_WEIGHT _ERROR	11017	Signer weight must be between 0 and (Integer.M
INVALID_TX_THRESHOLD _ERROR	11018	TxThreshold must be between 0 and Long.MAX
INVALID_OPERATION_TYP E_ERROR	11019	Operation type must be between 1 and 100
INVALID_TYPE_THRESHOL D_ERROR	11020	TypeThreshold must be between 0 and Long.MA
INVALID_ASSET_CODE_ER ROR	11023	The length of key must be between 1 and 64

表 2 – 续上页

Exception	Error code	Description
INVALID_ASSET_AMOUNT _ERROR	11024	AssetAmount must be between 0 and Long.MAX
INVALID_BU_AMOUNT_ERR OR	11026	BuAmount must be between 0 and Long.MAX_V
INVALID_ISSUER_ADDRES S_ERROR	11027	Invalid issuer address
NO_SUCH_TOKEN_ERROR	11030	No such token
INVALID_TOKEN_NAME_ER ROR	11031	The length of token name must be between 1 and
INVALID_TOKEN_SIMBOL _ERROR	11032	The length of symbol must be between 1 and 102
INVALID_TOKEN_DECIMAL S_ERROR	11033	Decimals must be between 0 and 8
INVALID_TOKEN_TOTALSU PPLY_ERROR	11034	TotalSupply must be between 1 and Long.MAX_
INVALID_TOKENOWNER_ER RPR	11035	Invalid token owner
INVALID_CONTRACTADDRE SS_ERROR	11037	Invalid contract address
CONTRACTADDRESS_NOT_C ONTRACTACCOUNT_ERROR	11038	contractAddress is not a contract account
INVALID_TOKEN_AMOUNT _ERROR	11039	TokenAmount must be between 1 and Long.MAX
SOURCEADDRESS_EQUAL_C ONTRACTADDRESS_ERROR	11040	SourceAddress cannot be equal to contractAddre
INVALID_FROMADDRESS_E RROR	11041	Invalid fromAddress
FROMADDRESS_EQUAL_DES TADDRESS_ERROR	11042	FromAddress cannot be equal to destAddress
INVALID_SPENDER_ERROR	11043	Invalid spender
PAYLOAD_EMPTY_ERROR	11044	Payload cannot be empty
INVALID_LOG_TOPIC_ERR OR	11045	The length of one log topic must be between 1 ar
INVALID_LOG_DATA_ERRO R	11046	The length of one piece of log data must be betw
INVALID_CONTRACT_TYPE _ERROR	11047	Invalid contract type
INVALID_NONCE_ERROR	11048	Nonce must be between 1 and Long.MAX_VALU
INVALID_GASPRICE_ERRO R	11049	GasPrice must be between 1000 and Long.MAX_
INVALID_FEELIMIT_ERRO R	11050	FeeLimit must be between 1 and Long.MAX_VA
OPERATIONS_EMPTY_ERRO R	11051	Operations cannot be empty
INVALID_CEILLEDGERSEQ _ERROR	11052	CeilLedgerSeq must be equal to or greater than 0
OPERATIONS_ONE_ERROR	11053	One of the operations cannot be resolved
INVALID_SIGNATURENUMB ER_ERROR	11054	SignagureNumber must be between 1 and Intege
INVALID_HASH_ERROR	11055	Invalid transaction hash
INVALID_BLOB_ERROR	11056	Invalid blob
PRIVATEKEY_NULL_ERROR	11057	PrivateKeys cannot be empty
PRIVATEKEY_ONE_ERROR	11058	One of privateKeys is invalid
PUBLICKEY_NULL_ERROR	11061	PublicKey cannot be empty
URL_EMPTY_ERROR	11062	Url cannot be empty
CONTRACTADDRESS_CODE _BOTH_NULL_ERROR	11063	ContractAddress and code cannot be empty at th
INVALID_OPTTYPE_ERROR	11064	OptType must be between 0 and 2
GET_ALLOWANCE_ERROR	11065	Failed to get allowance
GET_TOKEN_INFO_ERROR	11066	Failed to get the token info
SIGNATURE_EMPTY_ERROR	11067	The signatures cannot be empty
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTN_BLOCKCHAIN_E RROR	19999	Failed to connect to the blockchain
SYSTEM_ERROR	20000	System error

CHAPTER 3

API Guide for Exchanges

3.1 Overview

This document is for exchanges to install the BUMO node, and use the BUMO SDK.

3.2 BUMO Node Installation

Support for most operating systems such as Ubuntu, Centos, etc. It is recommended to use Ubuntu 14.04 or Centos 7. This example uses Ubuntu 14.04 as an example of a node installation.

3.2.1 Installing Dependencies

You need to install the dependencies required by the system before compiling the source code for BUMO. To install dependencies, you need to complete the following steps:

1. Input the following command to install `automake`.

```
sudo apt-get install automake
```

2. Input the following command to install `autoconf`.

```
sudo apt-get install autoconf
```

3. Input the following command to install `libtool`.

```
sudo apt-get install libtool
```

4. Input the following command to install `g++`.

```
sudo apt-get install g++
```

5. Input the following command to install `libssl-dev`.

```
sudo apt-get install libssl-dev
```

6. Input the following command to install `cmake`.

```
sudo apt-get install cmake
```

7. Input the following command to install `libbz2-dev`.

```
sudo apt-get install libbz2-dev
```

8. Input the following command to install `python`.

```
sudo apt-get install python
```

9. Input the following command to install `unzip`.

```
sudo apt-get install unzip
```

3.2.2 Compiling Source Code

The source code of BUMO can be compiled after the dependencies are successfully installed. If `git` is not installed, you can install `git` with the `sudo apt-get install git` command. To compile the source code of BUMO, you need to complete the following steps:

1. Download the BUMO source code file by inputting the following command in the root directory.

```
git clone https://github.com/bumoproject/bumo.git
```

注解： The bumo/ directory will be created automatically during the BUMO source code being downloaded, and the source code files will be stored in this directory.

---

2. Input the following command to enter the directory where the source code files are located.

```
cd /bumo/build/
```

3. Input the following command to download the dependencies and initialize the development environment.

```
./install-build-deps-linux.sh
```

4. Input the following command to return to the bumo/ directory.

```
cd ../
```

5. Input the following command to complete the compilation of the BUMO source code. The message below shows that the compilation is successful.

```
make
```

```
make[3]: Leaving directory `/bumo/build/linux'
/usr/bin/cmake -E cmake_progress_report /bumo/build/linux/CMakeFiles  1 2 3 4 5 6 7 8 9
[100%] Built target bumo
make[2]: Leaving directory `/bumo/build/linux'
/usr/bin/cmake -E cmake_progress_start /bumo/build/linux/CMakeFiles 0
make[1]: Leaving directory `/bumo/build/linux'
```

注解: The executable files **bumo** and **bumod** generated after compilation are stored in the /bumo/bin directory.

3.2.3 Installing the BUMO Node

The BUMO node can be installed after the compilation is completed. To install the BUMO node, you need to complete the following steps:

1. Input the following command to enter the installation directory.

```
cd /bumo/
```

2. Input the following command to complete the installation. The message below shows that the installation is successful.

```
make install
```

```
sudo mkdir -p /usr/local/buchain/coredump;
make[1]: Leaving directory `/bumo/build/linux'
```

注解:

- By default the service is installed in the /usr/local/buchain/ directory.
- After the installation is complete, you can start the bumo service with the `service bumo start` command without additional configuration.
- After installing the BUMO node, the directory structure in the buchain/ directory is as follows:

Directory	Description
bin	The directory stores the executable file (compiled bumo executable)
jslib	The directory stores the third-party `js` library
config	The configuration file directory contains: bumo.json
data	The database directory stores account ledger data
scripts	The directory stores scripts to start and stop the node
log	The directory stores logs

3.2.4 Changing the Operating Environment

You need to stop the BUMO service before changing the operating environment of BUMO. You can modify it by following the steps below:

1. Input the following command to enter the configuration file directory.

```
cd /usr/local/buchain/config/
```

注解:

The following runtime environment configuration files are provided in this directory.

- Bumo-mainnet.json (This file is the configuration file of the main network environment, which is applied in the production environment)
- bumo-testnet.json (This file is the configuration file applied in the test network environment)
- bumo-single.json (This file is the configuration file applied in the single-node debugging environment)

2. Change the configuration file (bumo.json) of the current running environment to another name, for example:

```
mv bumo.json bumoprevious.json
```

3. Change the environment configuration file to be run to bumo.json, for example:

```
mv bumo-mainnet.json bumo.json
```

注解:

- In this example, the main network environment is set to the running environment.
- After changing the operating environment, you need to clear the database to restart the bumo service.

3.3 DevOps Services

How to start, stop, query the service and the system, as well as how to clear the database are described in this section.

Starting BUMO Service

Input the following command to start the bumo service.

```
service bumo start
```

Stopping BUMO Service

Input the following command stop the bumo service.

```
service bumo stop
```

Querying BUMO Service Status

Input the following command to query the bumo service.

```
service bumo status
```

Querying System Status

Input the following command to query the detailed status of the system:

```
curl 127.0.0.1:19333/getModulesStatus
```

The following response message is received:

```
{
"glue_manager":{
"cache_topic_size":0,
"ledger_upgrade":{
"current_states":null,
"local_state":null
},
"system":{
"current_time":"2017-07-20 10:32:22", //Current system time
"process_uptime":"2017-07-20 09:35:06", //When bumo was started
"uptime":"2017-05-14 23:51:04"
},
"time":"0 ms",
"transaction_size":0
},
"keyvalue_db":Object{...},
"ledger_db":Object{...},
"ledger_manager":{
"account_count":2316, //Total accounts
"hash_type":"sha256",
"ledger_sequence":12187,
"time":"0 ms",
"tx_count":1185 //Total transactions
},
"peer_manager":Object{...},
"web server":Object{...},
```

3.3.1 Clearing Database

You need to stop the BUMO service before clearing the data. To clear the database, you need to complete the following steps:

1. Input the following command to enter the bumo service directory.

```
/usr/local/buchain/bin
```

2. Input the following command to clear the database.

```
./bumo --dropdb
```

注解： After the database is successfully cleared, you can see the information shown below.

```
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(153):Initialize db succes
sful
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(156):Drop db successfully
```

3.4 JAVA SDK Usage

The use of the JAVA SDK includes *Generating Addresses for Users to Recharge*, *Checking the Legality of Account Addresses*, and *Asset Transactions*.

3.4.1 Generating Addresses for Users to Recharge

The exchange needs to generate an address for each user to recharge. The exchange can create the user's address to recharge through `Keypair.generator()` provided in Bumo-sdk-java. The specific example is as follows:

```
/**

     * Generate an account private key, public key and address
     */
    @Test
    public void createAccount() {
        Keypair keypair = Keypair.generator();
        System.out.println(JSON.toJSONString(keypair, true));
    }
```

The return value is shown below:

```
{
    "address":"buQsG8XBSNSR5xzpRzRACfQNgxL5SMr2fenF",
    "privateKey":"privbyzj3bKdsB8SkVGmfp77JM2CZ4bsHei38GkSUHx3nHhApuGxtVvo",
    "publicKey":"b0012adc1eab24afdb266eb6f0341c893cff92aaef4dbc9e21d645f6accb48e6834b6336eb56"
}
```

3.4.2 Checking the Legality of Account Addresses

Check the validity of the account address by the code shown below.

```
/**

     * Check whether the account address is valid
     */
    @Test
    public void checkAccountAddress() {
        String address = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
        AccountCheckValidRequest accountCheckValidRequest = new␣
↪AccountCheckValidRequest();
        accountCheckValidRequest.setAddress(address);
        AccountCheckValidResponse accountCheckValidResponse = sdk.getAccountService().
↪checkValid(accountCheckValidRequest);
        if (0 == accountCheckValidResponse.getErrorCode()) {
            System.out.println(accountCheckValidResponse.getResult().isValid());
        } else {
            System.out.println(JSON.toJSONString(accountCheckValidResponse, true));
        }
    }
```

注解:

- If the return value is true, the account address is legal.

- If the return value is false, the account address is illegal.

3.4.3 Asset Transactions

In the BUMO network, a block is generated every 10 seconds, and each transaction only needs one confirmation to get its final state. In this section, we will introduce *Detecting User Recharging*, *Withdrawing or Transferring BU by Users* and *Querying Transactions*.

3.4.3.1 Detecting User Recharging

The exchange needs to monitor block generation, and then parse the transaction records in the block to confirm the user's recharge behavior. The specific steps are as follows:

1. Make sure that the node block status is normal.
2. Analyze the transactions contained in the block (for parsing methods, see parsing transactions in the block).
3. Record the results after parsing.

Viewing the Block Status

View the block status by the code shown below.

```
/**

     * Check whether the connected node is synchronous in the blockchain
     */
    @Test
    public void checkBlockStatus() {
        BlockCheckStatusResponse response = sdk.getBlockService().checkStatus();
        System.out.println(response.getResult().getSynchronous());
    }
```

注解:

- If the return value is true, the block is normal.
- If the return value is false, the block is abnormal.

Parsing Transactions in the Block

The exchange can query the transactions in the block according to the block height, and then analyze each transaction.

Example of request:

```
/**

     * Detect user recharge operations
     * <p>
     *Detect user recharge actions by parsing transactions in the block
     */
    @Test
    public void getTransactionOfBolck() {
        Long blockNumber = 617247L;// Block 617247
```

(continues on next page)

(续上页)

```
        BlockGetTransactionsRequest request = new BlockGetTransactionsRequest();
        request.setBlockNumber(blockNumber);
        BlockGetTransactionsResponse response = sdk.getBlockService().
↪getTransactions(request);
        if (0 == response.getErrorCode()) {
            System.out.println(JSON.toJSONString(response, true));
        } else {
            System.out.println("Failure\n" + JSON.toJSONString(response, true));
        }
        //Detect whether an account has recharged BU
        // Analyze transactions[n].transaction.operations[n].pay_coin.dest_address

        // Note:
        // Operations are arrays, there may be multiple transfer operations
    }
```

The response message is shown below:

```
{
    "total_count": 1,
    "transactions": [{
        "close_time": 1524467568753121,
        "error_code": 0,
        "error_desc": "",
        "hash": "89402813097402d1983c178c5ec271c6890db40c3beb9f06db71c8d52dab6c86",
        "ledger_seq": 33063,
        "signatures": [{
            "public_key":
↪"b001dbf0942450f5601e39ac1f7223e332fe0324f1f91ec16c286258caba46dd29f6ef9bf93b",
            "sign_data":
↪"668984fc7ded2dd30d87a1577f78eeb34d2198de3485be14ea66d9ca18f21aa21b2e0461ad8fedefc1abcb4221d346b404
↪"
        }],
        "transaction": {
            "fee_limit": 1000000,
            "gas_price": 1000,
            "metadata": "333133323333",
            "nonce": 25,
            "operations": [{
                "pay_coin": {
                    "amount": 3000,
                    "dest_address": "buQctxUa367fjw9jegzMVvdux5eCdEhX18ME"
                },
                "type": 7
            }],
            "source_address": "buQhP7pzmjoRsNG7AkhfNxiWd7HuYsYnLa4x"
        }
    }]
}

Details on the response message:

total_count   The total number of transactions (generally 1)
transactions  Query the transaction object in the block; the array size is the total␣
↪number of transactions in the block
|__ actual_fee    Transaction fees in MO
```

(continues on next page)

(续上页)

```
|__close_time     Transaction time
|__error_code     Transaction status, 0 indicates success, otherwise, failure
|__error_desc     Transaction status information
|__hash           Transaction hash
|__ledger_seq     Block height
|__signatures     Signature information
|__public_key     Public key for the signer
|__sign_data      Signature data for the signer
|__transaction    Signature object
|__fee_limit      Minimum fee, in MO
|__gas_price      Gas price in MO
|__metadata       Metadata for the transaction
|__nonce          Transactions in the original account
|__operations     Operation objects (multiple objects supported)
|__pay_coin     Operation type: built-in token
|__amount       Amount of BU transferred, in MO
|__dest_address       Recipient address
|__type        Operation type: 7 stands for built-in token transfer
|__source_address  Source account address
```

注解:

- For how to use Bumo-sdk-java, visit the following link:

  https://github.com/bumoproject/bumo-sdk-java/tree/release2.0.0

- For the example of API guide for the exchange, visit the following link:

  https://github.com/bumoproject/bumo-sdk-java/blob/release2.0.0/examples/src/main/java/io/bumo/sdk/example/ExchangeDemo.java

3.4.3.2 Withdrawing or Transferring BU by Users

For BU withdrawal operations, refer to the transfer example provided by bumo-sdk-java as follows:

```
/**
     * Send a transaction of sending bu
     *
     * @throws Exception
     */
    @Test
    public void sendBu() throws Exception {
        // Init variable
        // The account private key to send bu
        String senderPrivateKey =
↪"privbyQCRp7DLqKtRFCqKQJr81TurTqG6UKXMMtGAmPG3abcM9XHjWvq";
        // The account address to receive bu
        String destAddress = "buQswSaKDACkrFsnP1wcVsLAUzXQsemauE";
        // The amount to be sent
        Long amount = ToBaseUnit.BU2MO("0.01");
        // The fixed write 1000L, the unit is MO
        Long gasPrice = 1000L;
        // Set up the maximum cost 0.01BU
```

(continues on next page)

(续上页)

```
        Long feeLimit = ToBaseUnit.BU2MO("0.01");
        // Transaction initiation account's nonce + 1
        Long nonce = 1L;

        // Record txhash for subsequent confirmation of the real result of the␣
↪transaction.
        // After recommending five blocks, call again through txhash `Get the␣
↪transaction information
        // from the transaction Hash'(see example: getTxByHash ()) to confirm the␣
↪final result of the transaction
        String txhash = sendBu(senderPrivateKey, destAddress, amount, nonce, gasPrice,
↪ feeLimit);

    }
```

注解:

- Record the hash value of the BU withdrawal operation to view the final result of the BU withdrawal operation
- The current (2018-04-23) lowest value of gasPrice is 1000MO
- It is recommended to fill in 1000000 MO for feeLimit, which equals to 0.01BU

3.4.3.3 Querying Transactions

The final result of the BU withdrawal operation can be queried by the hash value returned when the BU withdrawal operation is initiated.

The call example is as follows:

```
/**
     * Get transaction information based on the transaction Hash
     */
    @Test
    public void getTxByHash() {
        String txHash =
↪"fba9c3f73705ca3eb865c7ec2959c30bd27534509796fd5b208b0576ab155d95";
        TransactionGetInfoRequest request = new TransactionGetInfoRequest();
        request.setHash(txHash);
        TransactionGetInfoResponse response = sdk.getTransactionService().
↪getInfo(request);
        if (0 == response.getErrorCode()) {
            System.out.println(JSON.toJSONString(response, true));
        } else {
            System.out.println("Failure\n" + JSON.toJSONString(response, true));
        }
    }
```

```
public static void queryTransactionByHash(BcQueryService queryService) {
   String txHash = "";
   TransactionHistory tx = queryService.getTransactionHistoryByHash(txHash);
   System.out.println(tx);
```

(continues on next page)

(续上页)

```
}

Note:
- When the number of tx.totalCount is greater than or equal to 1, the transaction␣
↪history exists
- When tx.transactions.errorCode equals 0, it indicates that the transaction is␣
↪successful, otherwise the transaction is not successful.
- For the withdrawal operation, the exchange should pay attention to the pay_coin␣
↪operation
- Example of a complete BU withdrawal response:
{
    "total_count": 1,
    "transactions": [{
        "close_time": 1524467568753121,
        "error_code": 0,
        "error_desc": "",
        "hash": "89402813097402d1983c178c5ec271c6890db40c3beb9f06db71c8d52dab6c86",
        "ledger_seq": 33063,
        "signatures": [{
            "public_key":
↪"b001dbf0942450f5601e39ac1f7223e332fe0324f1f91ec16c286258caba46dd29f6ef9bf93b",
            "sign_data":
↪"668984fc7ded2dd30d87a1577f78eeb34d2198de3485be14ea66d9ca18f21aa21b2e0461ad8fedefc1abcb4221d346b404
↪"
        }],
        "transaction": {
            "fee_limit": 1000000,
            "gas_price": 1000,
            "metadata": "333133323333",
            "nonce": 25,
            "operations": [{
                "pay_coin": {
                    "amount": 3000,
                    "dest_address": "buQctxUa367fjw9jegzMVvdux5eCdEhX18ME"
                },
                "type": 7
            }],
            "source_address": "buQhP7pzmjoRsNG7AkhfNxiWd7HuYsYnLa4x"
        }
    }]
}

total_count   The total number of transactions (generally 1)
transactions  Query the transaction object in the block, the array size is the total␣
↪number of transactions in the block
|__ actual_fee    Transaction fees in MO
|__close_time     Transaction time
|__error_code     Transaction status, 0 indicates success, otherwise, failure
|__error_desc     Transaction status information
|__hash           Transaction hash
|__ledger_seq     Block height
|__signatures     Signature information
|__public_key     Public key for the signer
|__sign_data      Signature data for the signer
|__transaction    Signature object
|__fee_limit     Minimum fee, in MO
```

(continues on next page)

(续上页)

```
|__gas_price     Gas price, in MO
|__metadata      Metadata for the transaction
|__nonce         Transactions in the original account
|__operations    Operation objects (multiple objects supported)
|__pay_coin      Operation type: built-in token
|__amount        Amount of BU transferred, in MO
|__dest_address       Recipient address
|__type         Operation type: 7 stands for built-in token transfer
|__source_address  Source account address
```

3.5 BU-Explorer

BUMO provides a blockchain data browsing tool for users to query block data.

You can visit the following links to query blockchain data:

- Testnet: http://explorer.bumotest.io
- Mainnet: http://explorer.bumo.io

3.6 BUMO Wallet

BUMO provides a full-node wallet for Windows and Mac, allowing users to manage their private keys, view BU transfers, and sign transactions offline.

You can download the BUMO wallet by the following link:

https://github.com/bumoproject/bumo-wallet/releases

3.7 FAQ

Start node in BUChain command line

Q: Do I need to start the node when using the BUChain command line?

A: No.

Are the values of gas_price and fee_limit fixed

Q: Are Gas_price fixed at 1000MO and fee_limit fixed at 1000000MO?

A: They are not fixed. But at present (2018-04-23) gas_price is 1000MO, the larger the gas_price is, the higher the priority for transactions to be packaged. The fee_limit is the maximum transaction fees for the blockchain when the transaction is initiated. If the transaction is legal, the actual fees charged are less than the fee_limit filled by the caller. (gas_price can be obtained from the result.fees.gas_price field in the query result via the following link:

http://seed1.bumo.io:16002/getLedger?with_fee=true

Transfer account balance

Q: Can I transfer all the balance from my account?

A: No. In order to prevent DDOS attacks, and prevent creating a large number of spam accounts, the activated accounts of BUMO must reserve a certain amount of BU, currently at 0.1 BU (it can be obtained from the result.fees.base_reserve field in the query result via the following link:

http://seed1.bumo.io:16002/getLedger?with_fee=true

CHAPTER 4

Installation Guide for BUMO

4.1 Overview

This document will walk you through the process of installing and configuring the BUMO node in both Linux and MacOS systems.

4.2 System Requirements

Before installing a BUMO node, you must make sure that your system meets the following conditions.

Hardware Requirements

The hardware requirements must meet the following configurations:

- **Recommended**：CPU 8 cores, memory 32G, bandwidth 20M, SSD disk 500G
- **Minimum**：CPU 4 cores, memory 16G, bandwidth 10M, SSD disk 500G

Software Requirements

You can choose Ubuntu, Centos or MacOS systems. The following systems are supported.

- Ubuntu 14.04
- Centos 7
- Mac OS X 10.11.4

4.3 Installing the BUMO Node in Linux

The following installation example is based on Ubuntu 14.04.

Two installation methods are supported on Linux systems: *Installing by Compilation* and *Installing with a Package*.

注解:

- The root directory in the root account is used as the installation directory in this installation document. You can choose your own installation directory
- Before installing the BUMO node, you must make sure that the device's network connection is normal

4.3.1 Installing by Compilation

Installing by Compilation means that the source code of the BUMO node is first compiled into machine code that can be recognized by the computer and then installed. Installing by Compilation consists of three parts: *Installing Dependencies*, *Compiling the BUMO Source Code*, and *Installing the BUMO Node*.

4.3.1.1 Installing Dependencies

You must install the dependencies required by the system before compiling the source code of the BUMO node. You must complete the following steps to install dependencies.

1. Input the following command to install `automake`.

```
sudo apt-get install automake
```

2. Input the following command to install `autoconf`.

```
sudo apt-get install autoconf
```

3. Input the following command to install `libtool`.

```
sudo apt-get install libtool
```

4. Input the following command to install `g++`.

```
sudo apt-get install g++
```

5. Input the following command to install `libssl-dev`.

```
sudo apt-get install libssl-dev
```

6. Input the following command to install `cmake`.

```
sudo apt-get install cmake
```

7. Input the following command to install `libbz2-dev`.

```
sudo apt-get install libbz2-dev
```

8. Input the following command to install `python`.

```
sudo apt-get install python
```

9. Input the following command to install `unzip`.

```
sudo apt-get install unzip
```

4.3.1.2 Compiling the BUMO Source Code

The source code of BUMO can be compiled after the dependencies have been successfully installed. You must complete the following steps to compile the source code:

1. In the root directory, input the following command to download the source code file of BUMO. If `git` is not installed, you can install `git` with the `sudo apt-get install git` command.

```
git clone https://github.com/bumoproject/bumo.git
```

```
root@ubuntu:/# git clone https://github.com/bumoproject/bumo.git
Cloning into 'bumo'...
remote: Counting objects: 22085, done.
remote: Compressing objects: 100% (265/265), done.
remote: Total 22085 (delta 200), reused 232 (delta 87), pack-reused 21695
Receiving objects: 100% (22085/22085), 185.05 MiB | 432.00 KiB/s, done.
Resolving deltas: 100% (10447/10447), done.
Checking connectivity... done.
Checking out files: 100% (13744/13744), done.
```

注解: The bumo/ directory will be created automatically during BUMO source code being downloaded, and the source code files will be stored in this directory.

2. Input the following command to enter the file directory of the source code.

```
cd /bumo/build/
```

3. Input the following command to download the dependencies and initialize the development environment.

```
./install-build-deps-linux.sh
```

4. Input the following command to return to the bumo/ directory.

```
cd ../
```

5. Input the following command to complete the compilation of the BUMO source code. The message below shows that the compilation is successful.

```
make
```

```
make[3]: Leaving directory `/bumo/build/linux'
/usr/bin/cmake -E cmake_progress_report /bumo/build/linux/CMakeFiles  1 2 3 4 5 6 7 8 9
[100%] Built target bumo
make[2]: Leaving directory `/bumo/build/linux'
/usr/bin/cmake -E cmake_progress_start /bumo/build/linux/CMakeFiles 0
make[1]: Leaving directory `/bumo/build/linux'
```

注解: The executable files generated after compilation are **bumo** and **bumod** which are stored in the /bumo/bin directory.

4.3.1.3 Installing the BUMO Node

The BUMO node can be installed after the compilation is finished. You must complete the following steps to install a BUMO node:

1. Input the following command to enter the installation directory.

```
cd /bumo/
```

2. Input the following command to complete the installation. The message below shows that the installation is successful.

```
make install
```

```
sudo mkdir -p /usr/local/buchain/coredump;
make[1]: Leaving directory `/bumo/build/linux'
```

注解:

- By default, the service is installed in the /usr/local/buchain/ directory.
- After the installation is finished, you can start the bumo service with the `service bumo start` command without additional configuration.
- After installing the BUMO node, the directory structure in the buchain/ directory is as follows:

Directory	Description
bin	The directory stores the executable file (compiled bumo executable)
jslib	The directory stores the third-party `js` library
config	The configuration file directory contains: bumo.json
data	The database directory stores account ledger data
scripts	The directory stores scripts to start and stop the node
log	The directory stores logs. Available after bumo is started

4.3.2 Installing with a Package

Installing with a package refers to installing the BUMO node with an installation package. Installing the BUMO node with the installation package consists of five parts: *Obtaining the Installation Package and Extracting It*, *Registering the Services*, *Modifying the Service Startup Directory*, *Setting the Boot Start*, and *Selecting the Configuration File for the Running Environment*.

4.3.2.1 Obtaining the Installation Package and Extracting It

You must complete the following steps to obtain the installation package of BUMO and extract it.

1. Input the following command to download the installation package of BUMO.

```
wget https://github.com/bumoproject/bumo/releases/download/1.0.0.7/buchain-1.0.0.7-
↪linux-x64.tar.gz
```

注解:

- If you don't have `wget` installed, you can use the `apt-get install wget` command to install `wget`.
- You can find the version you need from the https://github.com/bumoproject/bumo/releases link and then right-click the version to copy the download link.

- In this example the file is downloaded to the root directory.

2. Copy the installation package to the /usr/local/ directory by inputting the following command.

```
cp buchain-1.0.0.7-linux-x64.tar.gz /usr/local/
```

注解： The above copy operation is done in the directory where the file is downloaded. You must copy the file according to the specific download directory.

3. Input the following command to go to the /usr/local/ directory.

```
cd /usr/local/
```

4. Input the following command to extract the file.

```
tar -zxvf buchain-1.0.0.7-linux-x64.tar.gz
```

注解： After extracting the file, the buchain/ directory is generated.

4.3.2.2 Registering the Services

After extracting the file, you must register the services of bumo and bumod. You must complete the following steps to register services:

1. Input the following command to register the service of bumo.

```
ln -s /usr/local/buchain/scripts/bumo /etc/init.d/bumo
```

2. Input the following command to register the service of bumod.

```
ln -s /usr/local/buchain/scripts/bumod /etc/init.d/bumod
```

4.3.2.3 Modifying the Service Startup Directory

You must complete the following steps to modify the boot directory of bumo and bumod:

1. Open the bumo file by inputting the following command in the local/ directory.

```
vim buchain/scripts/bumo
```

2. Locate `install_dir` and change the installation directory of bumo.

```
install_dir=/usr/local/buchain
```

```
#!/bin/bash
install_dir=/usr/local/buchain
script_dir=/usr/local/buchain/scripts
```

注解： By default, the directory of `install_dir` is in the /usr/local/buchain directory; you can modify it according to the specific installation directory of bumo.

3. Press `Esc` to exit editing.
4. Input `:wq` to save the file.
5. Open the bumod file by inputting the following command in the local/ directory.

```
vim /buchain/scripts/bumod
```

6. Locate `install_dir` and change the installation directory for bumod.

```
install_dir=/usr/local/buchain
```

注解： By default, the directory of `install_dir` is in the /usr/local/buchain directory; you can modify it according to the specific installation directory of bumod.

7. Press `Esc` to exit editing.
8. Input `:wq` to save the file.

4.3.2.4 Setting the Boot Start

Setting up booting includes setting the startup level, adding startup commands, and modifying file permissions. You must complete the following steps to set up the boot:

1. Input the following command to set level 1.

```
ln -s -f /etc/init.d/bumod /etc/rc1.d/S99bumod
```

2. Input the following command to set level 2.

```
ln -s -f /etc/init.d/bumod /etc/rc2.d/S99bumod
```

3. Input the following command to set level 3.

```
ln -s -f /etc/init.d/bumod /etc/rc3.d/S99bumod
```

4. Input the following command to set level 4.

```
ln -s -f /etc/init.d/bumod /etc/rc4.d/S99bumod
```

5. Input the following command to set level 5.

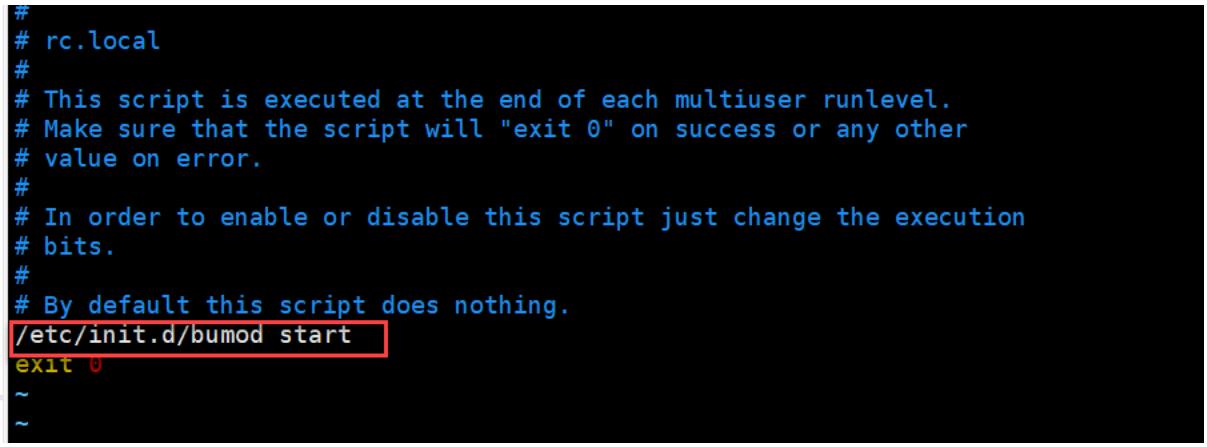
```
ln -s -f /etc/init.d/bumod /etc/rc5.d/S99bumod
```

6. Input the following command to open the rc.local file.

```
vim /etc/rc.local
```

7. Append the following command to the end of the rc.local file.

```
/etc/init.d/bumod start
```



8. Press `Esc` to exit editing.
9. Input `:wq` to save the file.
10. Execute the following command to set the permission of the rc.local file.

```
chmod +x /etc/rc.local
```

注解: Now the BUMO node is installed. Before starting the bumo service, you must select the configuration file for the running environment.

4.3.2.5 Selecting the Configuration File for the Running Environment

After installing the BUMO node, you must select the configuration file of the running environment to start the bumo service. You must complete the following steps to select the configuration file for the runtime environment:

1. Input the following command to go to the configuration file directory.

```
cd /usr/local/buchain/config/
```

注解:

The configuration files for the following runtime environments are available in this directory.

- bumo-mainnet.json：This file is the configuration file of the main network environment and is applied in the production environment
- bumo-testnet.json：This file is the configuration file of the test network environment
- bumo-single.json：This file is the configuration file for the single-node debugging environment

---

2. Input the following command to rename the configuration file for the runtime environment.

```
mv bumo-testnet.json bumo.json
```

注解:

- In this example, the test network environment is selected as the running environment. You can also select other files as your running environment according to your needs.
- After renaming the file, the bumo service can be started by the `service start bumo` command.
- After installing the BUMO node, you can view the directory structure of the installation file in the buchain/ directory.

4.4 Installing the BUMO Node in MacOS

Two installation methods are supported on MacOS systems: *Installing by Compilation in MacOS* and *Installing with a Package in MacOS*.

4.4.1 Installing by Compilation in MacOS

Installing by Compilation means that the source code of the BUMO node is first compiled into machine code that can be recognized by the computer and then installed. Installing by Compilation consists of six parts: *Installing Xcode*, *Installing Command Line Tools*, *Installing Homebrew*, *Installing Dependencies in MacOS*, *Compiling the BUMO Source Code in MacOS*, and *Installing the BUMO Node in MacOS*.

4.4.1.1 Installing Xcode

You must complete the following steps to install Xcode:

1. Click Software Download.
2. Input `Apple ID` and `Password`.
3. Click `Sign in` to go to the download page.
4. Click `Xcode 9.4.1` to start downloading `Xcode`.
5. Unzip the `Xcode_9.4.1.xip` file.
6. Double-click the extracted file `Xcode` to complete the installation.

注解: When choosing the version of `Xcode`, you must select one which is suitable to your MacOS system.

4.4.1.2 Installing Command Line Tools

安装 `Command Line Tools` 需要完成以下步骤:

1. Click Software Download .
2. Input `Apple ID` and `Password`.
3. Click `Sign in` to go to the download page.
4. Click `Command Line Tools(macOS 10.14)for Xcode 10 Beta 6` to start downloading `Command Line Tools`.
5. Double-click `Command_Line_Tools_macOS_10.14_for_Xcode_10Beta_6.dmg`.
6. Click the `Command Line Tools` icon.

7. Click **Next**
8. Select a language and then click **Next**.
9. Click **Agree**.
10. Click **Install**.
11. Input password for you mac and then click **Install software**.

注解: When choosing the version of `Command Line Tools`, you must select one which is suitable to your MacOS system.

4.4.1.3 Installing Homebrew

You must complete following steps to install Homebrew:

1. Open the terminal in the MacOS system.
2. Input the following code in the terminal:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
↪master/install)"
```

3. Press `Enter` to install.

4.4.1.4 Installing Dependencies in MacOS

1. Input the following command to set `Homebrew` without automatic update.

```
export HOMEBREW_NO_AUTO_UPDATE=true
```

2. Input the following command to install `autoconf`.

```
brew install autoconf
```

3. Input the following command to install `automake`.

```
brew install automake
```

4. Input the following command to install `libtool`.

```
brew install libtool
```

5. Input the following command to install `cmake`.

```
brew install cmake
```

6. Input the following command to install `python`.

```
brew install python
```

7. Input the following command to install `m4`.

```
brew install m4
```

8. Input the following command to install `wget`.

```
brew install wget
```

4.4.1.5 Compiling the BUMO Source Code in MacOS

1. In the root directory, input the following command to download the source code file of BUMO. If `git` is not installed, you can install `git` with the `sudo apt-get install git` command.

```
sudo git clone https://github.com/bumoproject/bumo.git
```

```
root@ubuntu:/# git clone https://github.com/bumoproject/bumo.git
Cloning into 'bumo'...
remote: Counting objects: 22085, done.
remote: Compressing objects: 100% (265/265), done.
remote: Total 22085 (delta 200), reused 232 (delta 87), pack-reused 21695
Receiving objects: 100% (22085/22085), 185.05 MiB | 432.00 KiB/s, done.
Resolving deltas: 100% (10447/10447), done.
Checking connectivity... done.
Checking out files: 100% (13744/13744), done.
```

注解： The bumo/ directory will be created automatically during the BUMO source code being downloaded, and the source code file will be stored in this directory.

2. Input the following command to go to the file directory of the source code.

```
cd /bumo/build/
```

3. Input the following command to download the dependencies and initialize the development environment.

```
sudo ./install-build-deps-mac.sh
```

4. Input the following command to return to the bumo/ directory.

```
cd ../
```

5. Input the following command to complete the compilation of the BUMO source code.

```
sudo make
```

注解： The executable files generated after compilation are **bumo** and **bumod** which are stored in the /bumo/bin directory.

4.4.1.6 Installing the BUMO Node in MacOS

The BUMO node can be installed after the compilation is finished. You must complete the following steps to install a BUMO node:

1. Input the following command to go to the installation directory.

```
cd /bumo/
```

2.Input the following command to complete the installation.

```
sudo make install
```

注解:

- By default, the service is installed in the /usr/local/buchain/ directory.
- After installing the BUMO node, the directory structure in the buchain/ directory is as follows:

Directory	Description
bin	The directory stores the executable file (compiled bumo executable)
jslib	The directory stores the third-party `js` library
config	The configuration file directory contains: bumo.json
data	The database directory stores account ledger data
log	The directory stores logs. Available after bumo is started

4.4.2 Installing with a Package in MacOS

Installing with a Package refers to installing the BUMO node with an installation package. Installing the BUMO node as an installation package consists of two parts: *Obtaining the Installation Package and Extracting It in MacOS*, and *Selecting the Configuration File for the Running Environment in MacOS*.

4.4.2.1 Obtaining the Installation Package and Extracting It in MacOS

1. Download the required installation package from the address below.

```
sudo wget https://github.com/bumoproject/bumo/releases/download/1.0.0.7/buchain-1.0.0.
↪7-macOS-x64.tar.gz
```

注解:

- If you don't have `wget` installed, you can use the `apt-get install wget` command to install `wget`.
- You can find the version you need from the https://github.com/bumoproject/bumo/releases link and then right-click the version to copy the download link.
- In this example the file is downloaded to the root directory.

2. Copy the installation package to the /usr/local/ directory by inputting the following command.

```
sudo cp buchain-1.0.0.7-macOS-x64.tar.gz /usr/local/
```

注解: The above copy operation is done in the directory where the file is downloaded. You must copy the file according to the specific download directory.

3. Input the following command to go to the /usr/local/ directory.

```
cd /usr/local/
```

4. Input the following command to extract the file.

```
sudo tar -zxvf buchain-1.0.0.7-macOS-x64.tar.gz
```

注解: After extracting the file, the buchain/ directory is generated.

Directory	Description
bin	The directory stores the executable file (compiled bumo executable)
jslib	The directory stores the third-party `js` library
config	The configuration file directory contains: bumo.json
data	The database directory stores account ledger data
log	The directory stores logs. Available after bumo is started

4.4.2.2 Selecting the Configuration File for the Running Environment in MacOS

After installing the BUMO node, you must select the configuration file of the running environment to start the bumo service. You must complete the following steps to select the configuration file for the runtime environment:

1. Input the following command to go to the configuration file directory.

```
cd /usr/local/buchain/config/
```

注解:

The configuration files for the following runtime environments are available in this directory.

- bumo-mainnet.json：This file is the configuration file of the main network environment and is applied in the production environment
- bumo-testnet.json：This file is the configuration file of the test network environment
- bumo-single.json：This file is the configuration file for the single-node debugging environment

2. Input the following command to rename the configuration file for the runtime environment.

```
mv bumo-testnet.json bumo.json
```

注解:

- In this example, the test network environment is selected as the running environment. You can also select other files as your running environment according to your needs.
- After renaming the file, the bumo service can be started by the `service start bumo` command.
- After installing the BUMO node, you can view the directory structure of the installation file in the buchain/ directory.

4.5 Configuration

The configuration is divided into *General Configuration* and *Multi-Node Configuration Example*.

4.5.1 General Configuration

General configuration includes data storage, communication between nodes, WEB API, WebSocket API, blocks, genesis, and log. The general configuration is configured in the bumo.json file in the /usr/local/buchain/config directory.

Data Storage

```
"db":{
"account_path": "data/account.db", //Store account data
"ledger_path": "data/ledger.db", //Store block data
"keyvalue_path": "data/keyvalue.db" //Store consensus data
}
```

Communication between Nodes

```
"p2p":
{
"network_id":30000,//Network ID
//Consensus network
"consensus_network":
{
"heartbeat_interval":60, //Heartbeat cycle, in second
"listen_port":36001,//Port monitored
"target_peer_connection":50, //Maximum number of active connections
"known_peers":
[
"127.0.0.1:36001"//Connect to other nodes
]
}
}
```

WEB API Configuration

```
"webserver":{
"listen_addresses":"0.0.0.0:16002"
}
```

WebSocket API Configuration

```
"wsserver":
{
"listen_address":"0.0.0.0:36003"
}
```

Block Configuration

```
"ledger":
{
"validation_address":"buQmtDED9nFcCfRkwAF4TVhg6SL1FupDNhZY",//The address of␣
↪validation node; the sync node or wallet does not need to be configured
"validation_private_key":
↪"e174929ecec818c0861aeb168ebb800f6317dae1d439ec85ac0ce4ccdb88487487c3b74a316ee777a3a7a77e5b12efd724
↪", //The private key of validation node; the sync node or wallet does not need to␣
↪be configured
```

(continues on next page)

(续上页)

```
"max_trans_per_ledger":1000, //Maximum number of transactions per block
"tx_pool": //Transaction pool configuration
{
"queue_limit":10240, // Limited transactions in the transaction pool
"queue_per_account_txs_limit":64 //Maximum transaction buffer for a single account
}
}
```

注解: `Validation_address` and `validation_private_key` can be obtained through the bumo program command line tool. Please save the account information properly and you will not be able to retrieve it if it is lost.

```
[root@bumo ~]# cd /usr/local/buchain/bin
[root@bumo bin]#./bumo --create-account

{
"address" : "buQmtDED9nFcCfRkwAF4TVhg6SL1FupDNhZY", //Address
"private_key" : "privbsZozNs3q9aixZWEUzL9ft8AYph5DixN1sQccYvLs2zPsPhPK1Pt", //Private
↪key
"private_key_aes" :
↪"e174929ecec818c0861aeb168ebb800f6317dae1d439ec85ac0ce4ccdb88487487c3b74a316ee777a3a7a77e5b12efd72
↪", //AES encrypted private key
"public_key" :
↪"b00108d329d5ff69a70177a60bf1b68972576b35a22d99d0b9a61541ab568521db5ee817fea6", //
↪Public key
"public_key_raw" : "08d329d5ff69a70177a60bf1b68972576b35a22d99d0b9a61541ab568521db5e",
↪ //Original public key
"sign_type" : "ed25519" //ed25519 encrypted
}
```

Genesis

```
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3", //Genesis address
"slogan" : "a new era of value", //Slogan stored in genesis
"fees":
{
"base_reserve": 10000000, //Base reserve for the account
"gas_price": 1000 //Byte fee
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY"] //The block list of validation
↪node
}
```

注解: The `genesis` configuration on the same blockchain must be consistent. `account` can be obtained by the bumo program command line tool `./bumo --create-account`. Please save the account information properly and you will not be able to retrieve it if it is lost.

Log Configuration

```
"logger":
{
```

(continues on next page)

(续上页)

```
"path":"log/buchain.log", // Log directory
"dest":"FILE|STDOUT|STDERR", //Output file classification
"level":"TRACE|INFO|WARNING|ERROR|FATAL",//Log level
"time_capacity":1, //Time span, day
"size_capacity":10, //Capacity, Megabyte
"expire_days":10 //Cycle of cleaning up the log, day
}
```

4.5.2 Multi-Node Configuration Example

In this section, two verification nodes and one synchronization node are taken as examples to describe the configuration of multiple nodes in the same blockchain. The three modules p2p, ledger and genesis need to be modified.

Configuration of p2p Module

The `known_peers` of p2p must be the IP and port of other known nodes for the interconnection between nodes.

```
verification node one:
"p2p":
{
"network_id":30000,
"consensus_network":
{
"heartbeat_interval":60,
"listen_port":36001,
"target_peer_connection":50,
"known_peers":
[
"192.168.1.102:36001", //IP and port of node two
"192.168.1.103:36001" //IP and port of node three
]
}
}

verification node two:
"p2p":
{
"network_id":30000,
"consensus_network":
{
"heartbeat_interval":60,
"listen_port":36001,
"target_peer_connection":50,
"known_peers":
[
"192.168.1.101:36001", //IP and port of node one
"192.168.1.103:36001" //IP and port of node three
]
}
}

synchronization node three:
"p2p":
{
"network_id":30000,
```

(continues on next page)

(续上页)

```
"consensus_network":
{
"heartbeat_interval":60,
"listen_port":36001,
"target_peer_connection":50,
"known_peers":
[
"192.168.1.101:36001", //IP and port of node one
"192.168.1.102:36001" //IP and port of node two
]
}
}
```

Configuration of Leger Module

The `validation_address` and `validation_private_key` from the ledger of verification node must match. And you must input `validation_address` of all the verification nodes into `genesis.validators`.

```
Verification node one:
"ledger":
{
"validation_address":"buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",//The address of␣
↪verification node one; the sync node or wallet does not need to be configured
"validation_private_key":
↪"66932f19d5be465ea9e7cfcb3ea7326d81953b9f99bc39ddb437b5367937f234b866695e1aae9be4bae27317c9987f80be
↪", //The private key of verification node two; the synchronization node or wallet␣
↪does not need to be configured
"max_trans_per_ledger":1000,
"tx_pool":
{
"queue_limit":10240,
"queue_per_account_txs_limit":64
}
}

Verification node two:
"ledger":
{
"validation_address":"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm",//The address of␣
↪verification node two; the sync node or wallet does not need to be configured
"validation_private_key":
↪"1cb0151ec2b23cb97bf94d86ee1100582f9f5fbfdfe40a69edae2d2b8711395c40c1da859ac0bc93240a8a70c4a06779ec
↪", //The private key of verification node two; the synchronization node or wallet␣
↪does not need to be configured
"max_trans_per_ledger":1000,
"tx_pool":
{
"queue_limit":10240,
"queue_per_account_txs_limit":64
}
}

Verification node three:
"ledger":
{
"max_trans_per_ledger":1000,
"tx_pool":
```

(continues on next page)

(续上页)

```
{
"queue_limit":10240,
"queue_per_account_txs_limit":64
}
}
```

Configuration of Genesis Module

The genesis configuration on the same blockchain must be consistent.

```
Verification note one:
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
"slogan" : "a new era of value",
"fees":
{
"base_reserve": 10000000,
"gas_price": 1000
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",
 ↪"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm"] //All verification node addresses need to␣
 ↪be configured. If there are two verification nodes, configure two addresses.
}

Verification note two:
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
"slogan" : "a new era of value",
"fees":
{
"base_reserve": 10000000,
"gas_price": 1000
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",
 ↪"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm"] //All verification node addresses need to␣
 ↪be configured. If there are two verification nodes, configure two addresses.
}

Verification note three:
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
"slogan" : "a new era of value",
"fees":
{
"base_reserve": 10000000,
"gas_price": 1000
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",
 ↪"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm"] //All verification node addresses need to␣
 ↪be configured. If there are two verification nodes, configure two addresses.
}
```

注解:

- Before running, please make sure that the initial data of each node is consistent, otherwise you will not be able to reach consensus to generate the block.
- `account`, `validation_address` can be obtained by the bumo program command line tool `./bumo --create-account`. Please save the account information properly and you will not be able to retrieve it if it is lost.

4.6 Maintenance Service

In the maintenance service, the BUMO service operations such as startup, shutdown, status query, system details query, clear database, create a hard fork, and change the running environment, are described in detail.

Starting the BUMO Service

Input the following command to start the bumo service.

```
service bumo start
```

注解： To start the bumo service in MacOS, you must enter the /usr/local/buchain/bin directory and start the bumo service with the `./bumo` command.

Stopping the BUMO Service

Input the following command to stop the bumo service.

```
service bumo stop
```

注解： In the MacOS system, you can stop the bumo service by pressing `control+c`.

Querying the BUMO Service Status

Input the following command to query the bumo service.

```
service bumo status
```

注解： No service is available in MacOS.

Querying the Detailed System Status

Input the following command to query the detailed system status.

```
curl 127.0.0.1:19333/getModulesStatus
```

The result is shown below:

```
{
 "glue_manager":{
     "cache_topic_size":0,
     "ledger_upgrade":{
```

(continues on next page)

(续上页)

```
        "current_states":null,
        "local_state":null
    },
    "system":{
        "current_time":"2017-07-20 10:32:22", //Current system time
        "process_uptime":"2017-07-20 09:35:06", //When bumo is started
        "uptime":"2017-05-14 23:51:04"
    },
    "time":"0 ms",
    "transaction_size":0
},
"keyvalue_db":Object{...},
"ledger_db":Object{...},
"ledger_manager":{
    "account_count":2316,  //Total accounts
    "hash_type":"sha256",
    "ledger_sequence":12187,
    "time":"0 ms",
    "tx_count":1185   //Total transactions
},
"peer_manager":Object{...},
"web server":Object{...},
```

注解： No service is available in MacOS.

Clearing Database

You must stop the BUMO service before clearing the database. You must complete the following steps to clear the database:

1. Input the following command to enter the bumo service directory.

```
cd /usr/local/buchain/bin
```

2. Input the following command to clear the database.

```
./bumo --dropdb
```

注解： After the database is successfully cleared, you can see the information shown below.

```
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(153):Initialize db succes
sful
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(156):Drop db successfully
```

Creating a Hard Fork

You must complete the following steps to create a hard fork.

1. Create the hard fork by inputting the following command in the /usr/local directory.

```
buchain/bin/bumo --create-hardfork
```

2. Enter `y` when prompted and then press `Enter`. The message shown below indicates the hard fork is created successfully.

```
[2018-07-19 11:45:40.464 - INF] <7F2A786AA8C0> ledger_manager.cpp(324):Are you su
re to create hardfork ledger? Press y to continue.
y
[2018-07-19 11:47:05.599 - INF] <7F2A786AA8C0> ledger_manager.cpp(341):Max closed ledger seq=290524
[2018-07-19 11:47:05.599 - INF] <7F2A786AA8C0> ledger_frm.cpp(551):total reward(800000000) = total fee(0) + block reward(800000000) in ledger(290525)
[2018-07-19 11:47:05.605 - INF] <7F2A786AA8C0> ledger_manager.cpp(438):Create hard fork ledger successful, seq(290525), consensus value hash(4b9ad78065c
65aaf1280edf6129ab2da93c99c42f2bcd380b5966750ccd5d80d)
```

注解:

- After executing the above command, the new blockchain network has only one verification node.
- After executing the hard fork command, the following hash value is displayed:

```
4b9ad78065c65aaf1280edf6129ab2da93c99c42f2bcd380b5966750ccd5d80d
```

3. Input the following command to clear the consensus status data. When clearing the consensus status data, you must ensure that the bumo service is not running, otherwise it cannot be cleared.

```
buchain/bin/bumo --clear-consensus-status
```

4. Add the hash value to the bumo.json file in the /usr/local/buchain/config directory of the node or synchronization node.

```
"ledger": {
"genesis_account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
"max_trans_per_ledger": 1000,
"hardfork_points" :
[
"4b9ad78065c65aaf1280edf6129ab2da93c99c42f2bcd380b5966750ccd5d80d
"
]
},
```

5. Start the node service for the configuration to take effect.

Changing the Running Environment

Before changing the running environment, you must make sure that the BUMO service is down. If you want to change the running environment of the BUMO node, you can modify it by following the steps below.

1. Input the following command to enter the directory where the configuration file is located.

```
cd /usr/local/buchain/config/
```

注解:

The configuration files for the following runtime environments are available in this directory.

- bumo-mainnet.json：This file is the configuration file of the main network environment and is applied in the production environment
- bumo-testnet.json：This file is the configuration file of the test network environment
- bumo-single.json：This file is the configuration file for the single-node debugging environment

2. Change the configuration file name (bumo.json) for the current running environment, for example:

```
mv bumo.json bumoprevious.json
```

3. Change the environment configuration file to run to bumo.json, for example:

```
mv bumo-mainnet.json bumo.json
```

注解:

- In this example, the main network environment is set to the running environment.
- After changing the running environment, you must clear the database to restart the bumo service.

4.7 Uninstalling the BUMO Node

Uninstalling BUMO nodes is divided into two categories, one for uninstalling the BUMO node installed by compilation and the other is for uninstalling the BUMO node installed with a package.

4.7.1 Uninstalling the BUMO Node Installed by Compilation

If you installed the BUMO node by compilation, you can uninstall the BUMO node by the following steps:

1. Input the following command to enter the BUMO installation directory.

```
cd /bumo
```

2. Input the following command to delete the BUMO node.

```
make uninstall
```

注解: Now the BUMO node is uninstalled.

4.7.2 Uninstalling the BUMO Node Installed with a Package

If you installed the BUMO node with the installation package, you can uninstall the BUMO node by the following steps:

1. Input the following command to delete the directory of the buchain.

```
sudo rm -rf /usr/local/buchain/
```

2. Input the following command to delete the soft link of bumo.

```
sudo rm -rf /etc/init.d/bumo
```

3. Input the following command to delete the soft link of bumod.

```
sudo rm -rf /etc/init.d/bumod
```

注解：Now the BUMO node is uninstalled.

CHAPTER 5

Keypair手册

5.1 概述

本文档详细介绍了Keypair（公、私钥对）的生成过程以及在此基础上如何生成地址（address）并对交易签名，介绍了执行交易调用的两种接口方式以及相关流程，提供了多种ProtoBuf数据结构参考信息，最后以示例的方式详细介绍了两种交易提交方式，即调用接口生成transaction*blob*和自己生成*transaction*blob。

5.2 术语

本章节对该文档中使用到的术语进行了详细说明。

Keypair

keypair是BUMO工程中生成公钥、私钥、地址及签名的接口。在签名过程中仅支持 `ED25519` 签名算法。

私钥

私钥是通过算法生成的一串字符串，是生成公钥和地址的前提条件，同时也是完成签名的基本要素。私钥生成后不能更改，一旦丢失将无法找回，因此需要妥善保管。

公钥

公钥是基于私钥产生的一串字符串，可以对私钥加密的字符串进行验证，在网络间传输时不会导致私钥泄露，同时也是生成地址的必要条件。

地址

地址是基于公钥产生的一串字符串。与现实生活中的地址类似，没有地址就无法找到联系人，因此也就无法完成交易。

签名

签名是指通过算法和私钥对交易数据进行加密确认并得到签名数据的过程。用户可以通过签名数据判断交易数据的完整性和正确性。

交易

在BUMO中所有修改区块链数据的操作都称为交易，比如发行资产、转移资产、发送BU、创建账号、设置metadata、设置权限等都是交易。

Transaction Blob

Transaction Blob是指对一个交易对象进行序列化处理之后得到的16进制字符串。交易序列化是指通过ProtoBuf数据结构将交易对象的状态信息转换成可以存储和传输的字符串的过程。

Raw Private Key

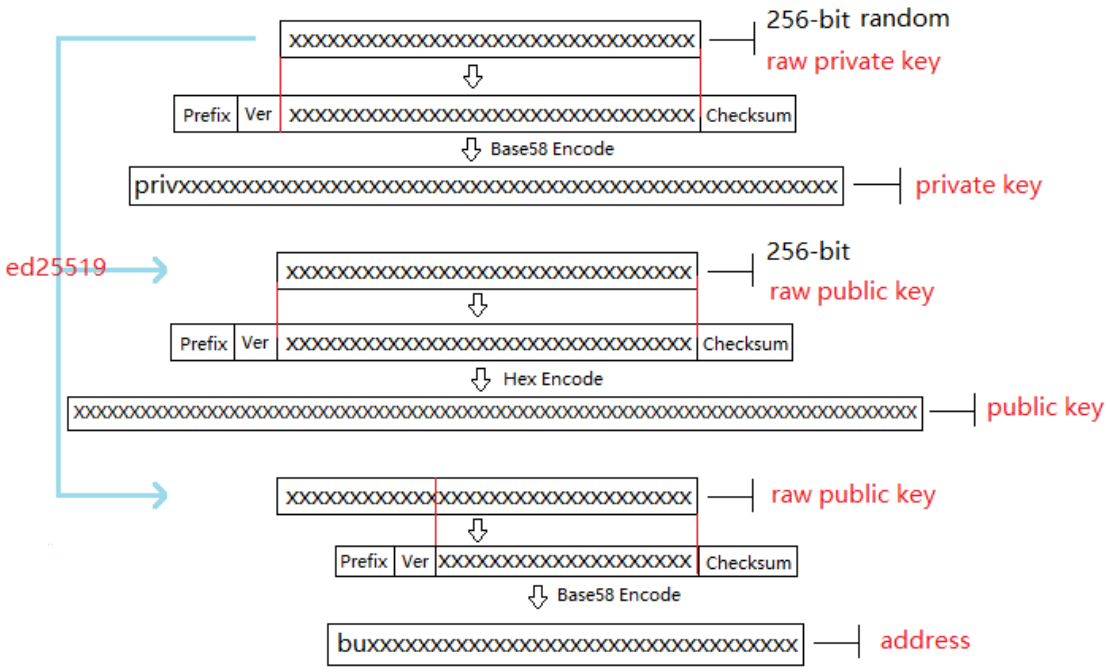
Raw Private Key 是指通过随机算法得到的字节数组，该字节数组是生成私钥的前提条件。

Raw Public Key

Raw Public Key 是指通过 `ED25519` 算法对 raw private key 进行处理生成的字节数组，该字节数组是生成公钥的前提条件。

5.3 原理图

下图说明了私钥、公钥和地址的生成原理。



5.4 生成私钥

生成私钥需要使用随机算法、`SHA256` 等多个算法才能实现。生成私钥包括以下步骤:

1. 利用随机算法生成一个256位的随机数（数学意义上的私钥），得到字节数组即 raw private key，如下所示:

```
[17,236,24,183,207,250,207,180,108,87,224,39,189,99,246,85,138,120,236,78,228,233,41,
↪192,124,109,156,104,235,66,194,24]
```

2. 在 raw private key 前面加上3个字节的前缀（Prefix），然后再加上1个字节的版本号（Version），得到新的字节数组，如下所示：

```
[218,55,159,1,17,236,24,183,207,250,207,180,108,87,224,39,189,99,246,85,138,120,236,
↪78,228,233,41,192,124,109,156,104,235,66,194,24]
```

注解： 关于Prefix、Version以及Checksum请查看表1。

3. 对第2步中得到的字节数组进行两次 `SHA256` 计算，取运算结果的前4个字节，得到校验码（Checksum）的字节数组，如下所示：

```
[30,19,80,117]
```

4. 将第2步中的字节数组和第3步中的校验码字节数组按照先后顺序连接在一起，得到新的字节数组，如下所示：

```
[218,55,159,1,17,236,24,183,207,250,207,180,108,87,224,39,189,99,246,85,138,120,236,
↪78,228,233,41,192,124,109,156,104,235,66,194,24,30,19,80,117]
```

5. 对第4步中产生的字节数组进行Base58编码，得到以priv开始的字符串，即私钥（private key），如下所示：

```
privbsGZFUoRv8aXZbSGd3bwzZWFn3L5QKq74RXAQYcmfXhhZ54CLr9z
```

注解： 至此就完成了私钥的生成。

表1

名称	数据内容	长度
Prefix	0xDA 0x37 0x9F	3个字节
Version	0x01	1个字节
Checksum	对第2步中得到的字节数组进行两次SHA256运算之后，取运算结果的前4个字节	4个字节

该表对生成私钥中使用到的Prefix、Version以及Checksum进行了说明。

5.5 生成公钥

生成公钥需要在生成私钥之后才能实现，需要用到 `ED25519` 算法。生成公钥包含以下步骤：

1. 通过 `ED25519` 算法对raw private key进行处理生成32位的字节数组，即raw public key。例如私钥是 `privbsGZFUoRv8aXZbSGd3bwzZWFn3L5QKq74RXAQYcmfXhhZ54CLr9z`，其raw public key如下所示：

```
[21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,240,212,
↪54,18,225,158,198,50,87,10]
```

2. 在raw public key 前面加上1个字节的前缀（Prefix），然后再加上1个字节的版本号（Version），得到新的字节数组，如下所示：

```
[176,1,21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,
↪240,212,54,18,225,158,198,50,87,10]
```

注解： 关于Prefix、Version以及Checksum请查看表2。

3. 对第2步中的字节数组进行两次 SHA256 计算，取运算结果的前4个字节，得到校验码（Checksum）的字节数组，如下所示：

```
[116,171,22,107]
```

4. 将第2步中的字节数组和3步的校验码字节数组按照先后顺序连接在一起，得到新的字节数组，如下所示：

```
[176,1,21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,
↪240,212,54,18,225,158,198,50,87,10,116,171,22,107]
```

5. 对第4步中的字节数组进行16进制编码，得到16进制字符串，即公钥（public key），如下所示：

```
b00115764cd017e0da753271fa26cd529451a21b8253d001f0d43612e19ec632570a74ab166b
```

注解： 至此就完成了公钥的生成。

表2

名称	数据内容	长度
Prefix	0xB0	1个字节
Version	0x01	1个字节
Checksum	对第2步中得到的字节数组进行两次SHA256运算之后，取运算结果的前4个字节	4个字节

该表对生成公钥中使用到的Prefix、Version以及Checksum进行了说明。

5.6 生成地址

在生成私钥和公钥后可以进一步通过算法生成地址。生成地址包含以下步骤：

1. 通过 ED25519 算法对raw private key进行处理生成32位的字节数组，即 raw public key。例如私钥为 privbsGZFUoRv8aXZbSGd3bwzZWFn3L5QKq74RXAQYcmfXhhZ54CLr9z，其raw public key 如下所示：

```
[21,118,76,208,23,224,218,117,50,113,250,38,205,82,148,81,162,27,130,83,208,1,240,212,
↪54,18,225,158,198,50,87,10]
```

2. 对 raw public key 进行两次 SHA256 运算，并取运算结果的后20位字节，得到字节数组，如下所示：

```
[173,148,59,51,183,193,55,160,1,133,247,80,65,13,67,190,164,114,18,220]
```

3. 在第2步产生的字节数组前面加上2个字节的前缀（Prefix），然后再加上1个字节的版本号（Version），得到新的字节数组，如下所示：

```
[1,86,1,173,148,59,51,183,193,55,160,1,133,247,80,65,13,67,190,164,114,18,220]
```

注解： 关于Prefix、Version以及Checksum请查看表3。

4. 对第3步中的字节数组进行两次 SHA256 计算，取运算结果的前4个字节，得到校验码（Checksum）的字节数组，如下所示：

```
[167,127,34,35]
```

5. 将第3步中的字节数组和第4步的校验码字节数组按照先后顺序连接在一起，得到新的字节数组，如下所示：

```
[1,86,1,173,148,59,51,183,193,55,160,1,133,247,80,65,13,67,190,164,114,18,220,167,127,
↪34,35]
```

6. 对第5步中产生的字节数组进行Base58编码，得到以bu开头的字符串，即地址（address），如下所示：

```
buQmWJrdYJP5CPKTbkQUqscwvTGaU44dord8
```

注解： 至此就完成了地址的生成。

表3

名称	数据内容	长度
Prefix	0x01 0x56	2个字节
Version	0x01	1个字节
PublicKey	取raw public key的后20个字节	20个字节
Checksum	对第3步中得到的字节数组进行两次SHA256运算之后，取运算结果的前4个字节	4个字节

该表对生成地址中使用到的Prefix、Version以及Checksum进行了说明。

5.7 交易签名

借助 ED25519 算法和私钥对待签名的交易（transaction*blob*的反*16*进制编码得到的字节数组）进行签名，并进行*16*进制转换，得到签名字符串*sign*data。

下面的示例展示了如何用 ED25519 和私钥对transaction_blob签名。

私钥是：

```
b00115764cd017e0da753271fa26cd529451a21b8253d001f0d43612e19ec632570a74ab166b
```

Transaction_blob是:

```
0A24627551566B5555424B70444B526D48595777314D553855376E676F5165686E6F31363569109F0818C0843D20E8073214
```

用 ED25519 的签名接口对transaction*blob*进行签名，并进行*16*进制转换后，得到的*sign*data是：

```
a46ee590a84abdeb8cc38ade1ae8e8a2c71bb69bdc4cd7dc0de1b74b37e2cbd1696229687f80dff4276b1a3dd3f95a9bc1d5
```

5.8 交易提交方式

交易的执行有两种接口调用方式：调用接口生成transaction_blob 和自己生成transaction_blob。

5.8.1 调用接口生成:

注意: 由于transaction*blob*很可能被截取和篡改，因此不建议用这种方式生成*transaction*blob。

如果需要调用接口生成transaction_blob、签名并提交交易，请查看bumo的开发文档，地址如下:

https://github.com/bumoproject/bumo/blob/master/docs/develop.md

调用接口生成transation_blob包含以下步骤:

1. 调用`getAccount`接口获取待发起交易账户的nonce值，代码如下所示:

```
HTTP GET host:port/getAccount?address=账户地址
```

2. 根据需要填充json数据并完成交易数据填充，格式如下所示:

```
{
"source_address":"xxxxxxxxxx", //交易源账号，即交易的发起方
"nonce":2, //nonce的值
"ceil_ledger_seq": 0, //可选
"fee_limit":1000, //交易支付的费用
"gas_price": 1000, //gas价格(不小于配置的最低值)
"metadata":"0123456789abcdef", //可选，用户自定义给交易的备注，16进制格式
"operations":[
{
//根据不同的操作填写
},
{
//根据不同的操作填写
}
......
]
}
```

注解: nonce值需要在第1步中获取值的基础上加1。

3. 通过调用`getTransactionBlob`接口将第2步中生成的json数据作为参数传入，得到一个交易hash和transaction_blob，实现交易序列化，格式如下所示:

```
{
"error_code": 0,
"error_desc": "",
"result": {
"hash": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //交易的hash
"transaction_blob": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //交易序列化之后的16进制表示
}
}
```

4. 对交易进行签名并填充交易数据。根据之前生成的私钥对transaction_blob签名，然后填充提交交易的json数据，格式如下所示:

```
{
"items" : [{
"transaction_blob" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //一个交易序列化之
后的16进制表示
```

(continues on next page)

(续上页)

```
"signatures" : [{//第一个签名
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //签名数据
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //公钥
}, {//第二个签名
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //签名数据
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //公钥
}
]
}
]
}
```

5. 通过调用submitTransaction接口，将第4步中生成的json数据作为参数传入，得到响应结果，完成交易提交。响应结果的格式如下所示：

```
{
"results": [
{
"error_code": 0,
"error_desc": "",
"hash": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //交易的hash
}
],
"success_count": 1
}
```

5.8.2 自己生成

自己生成transaction_blob、签名，并提交交易，具体操作包括以下步骤：

1. 通过调用getAccount接口获取待发起交易的账户的nonce值，如下所示：

```
HTTP GET host:port/getAccount?address=账户地址
```

2. 填充protocol buffer的交易对象Transaction，并进行序列化操作，从而得到transaction_blob。具体的交易数据结构详情请看 *ProtoBuf数据结构*。

3. 签名交易，并填充交易数据。根据私钥生成公钥，并用私钥对transaction_blob签名，然后填充提交交易的json数据，格式如下：

```
{
"items" : [{
"transaction_blob" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //一个交易序列化之后的16进制表示
"signatures" : [{//第一个签名
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //签名数据
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //公钥
}, {//第二个签名
"sign_data" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", //签名数据
"public_key" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //公钥
}
]
}
]
}
```

4. 通过调用submitTransaction接口，将第3步生成的json数据作为参数传入，完成交易提交。响应结果格式如下：

```
{
"results": [
{
"error_code": 0,
"error_desc": "",
"hash": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" //交易的hash
}
],
"success_count": 1
}
```

5.9 ProtoBuf数据结构

Protocol Buffers（ProtoBuf） 是一种轻便高效的结构化数据存储格式，可以用于结构化数据串行化，或者说序列化。它很适合做数据存储或 RPC 数据交换格式。可用于通讯协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据格式。目前提供了 C++、Java、Python 三种语言的 API。

要了解更多关于ProtoBuf的信息，请查看以下链接：

https://developers.google.com/protocol-buffers/docs/overview

接下来将介绍Protocol Buffer的数据结构详情，并提供针对脚本生成的各种语言的protocol buffer的文件和简单测试程序。

5.9.1 数据结构

下面介绍了交易中可能用到的各种ProtoBuf数据结构及其用途，供用户参考使用。

Transaction

该数据结构适用于完整的交易。

```
message Transaction {
enum Limit{
UNKNOWN = 0;
OPERATIONS = 1000;
};
string source_address = 1; // 交易发起账户地址
int64 nonce = 2; // 交易序列号
int64 fee_limit = 3; // 交易费用，默认1000Gas，单位是MO，1 BU = 10^8 MO
int64 gas_price = 4; // 交易打包费用，默认是1000，单位是MO，1 BU = 10^8 MO
int64 ceil_ledger_seq = 5; // 区块高度限制
bytes metadata = 6; // 交易备注
repeated Operation operations = 7; // 操作列表
}
```

Operation

该数据结构适用于交易中的操作。

```
message Operation {
enum Type {
```

(continues on next page)

(续上页)

```
UNKNOWN = 0;
CREATE_ACCOUNT = 1;
ISSUE_ASSET = 2;
PAY_ASSE = 3;
SET_METADATA = 4;
SET_SIGNER_WEIGHT = 5;
SET_THRESHOLD = 6;
PAY_COIN = 7;
LOG = 8;
SET_PRIVILEGE = 9;
};
Type type = 1; // 操作类型
string source_address = 2; // 操作源账户地址
bytes metadata = 3; // 操作备注
OperationCreateAccount create_account = 4; // 创建账户操作
OperationIssueAsset issue_asset = 5; // 发行资产操作
OperationPayAsset pay_asset = 6; // 转移资产操作
OperationSetMetadata set_metadata = 7; // 设置metadata
OperationSetSignerWeight set_signer_weight = 8; // 设置签名者权限
OperationSetThreshold       set_threshold = 9; // 设置交易门限
OperationPayCoin pay_coin = 10; // 转移coin
OperationLog log = 11; // 记录log
OperationSetPrivilege set_privilege = 12; // 设置权限
}
```

OperationCreateAccount

该数据结构用于创建账户。

```
message OperationCreateAccount{
string dest_address = 1; // 待创建的目标账户地址
Contract contract = 2; // 合约
AccountPrivilege priv = 3; // 权限
repeated KeyPair metadatas = 4; // 附加信息
int64       init_balance = 5; // 初始化余额
string init_input = 6; // 合约入参
}
```

Contract

该数据结构用于设置合约。

```
message Contract{
enum ContractType{
JAVASCRIPT = 0;
}
ContractType type = 1; // 合约类型
string payload = 2; // 合约代码
}
```

AccountPrivilege

该数据结构用于设置账户权限。

```
message AccountPrivilege {
int64 master_weight = 1; // 账户自身权重
repeated Signer signers = 2; // 签名者权重列表
```

(continues on next page)

(续上页)

```
AccountThreshold thresholds = 3; // 门限
}
```

Signer

该数据结构用于设置签名者权重。

```
message Signer {
enum Limit{
SIGNER_NONE = 0;
SIGNER = 100;
};
string address = 1; // 签名者账户地址
int64 weight = 2; // 签名者权重
}
```

AccountThreshold

该数据结构用于设置账户门限。

```
message AccountThreshold{
int64 tx_threshold = 1; // 交易门限
repeated OperationTypeThreshold type_thresholds = 2; // 指定操作的交易门限列表，未指定的操作的交易以tx_threshold为门限
}
```

OperationTypeThreshold

该数据结构用于指定类型的操作门限。

```
message OperationTypeThreshold{
Operation.Type type = 1; // 操作类型
int64 threshold = 2; // 该操作对应的门限
}
```

OperationIssueAsset

该数据结构用于发行资产。

```
message OperationIssueAsset{
string code = 1; // 待发行的资产编码
int64 amount = 2; // 待发行的资产数量
}
```

OperationPayAsset

该数据结构用于转移资产。

```
message OperationPayAsset {
string dest_address = 1; // 目标账户地址
Asset asset = 2; // 资产
string input = 3; // 合约入参
}
```

Asset

该数据结构适用于资产。

```
message Asset{
AssetKey    key = 1; // 资产标识
int64       amount = 2; // 资产数量
}
```

AssetKey

该数据结构用于标识资产唯一性。

```
message AssetKey{
string issuer = 1; // 资产发行账户地址
string code = 2; // 资产编码
int32 type = 3; // 资产类型（默认为0，表示不限制数量）
}
```

OperationSetMetadata

该数据结构用于设置Metadata。

```
message OperationSetMetadata{
string      key = 1; // 关键字，惟一
string value = 2; // 内容
int64 version = 3; // 版本控制，可不设置
bool delete_flag = 4; // 是否删除
}
```

OperationSetSignerWeight

该数据结构用于设置签名者权重。

```
message OperationSetSignerWeight{
int64 master_weight = 1; // 自身权重
repeated Signer signers = 2; // 签名者权重列表
}
```

OperationSetThreshold

该数据结构用于设置门限。

```
message OperationSetThreshold{
int64 tx_threshold = 1; // 交易门限
repeated OperationTypeThreshold type_thresholds = 2; // 指定操作的交易门限列表，未指定的操作
的交易以tx_threshold为门限
}
```

OperationPayCoin

该数据结构用于发送coin。

```
message OperationPayCoin{
string dest_address = 1; // 目标账户地址
int64 amount = 2; // coin的数量
string input = 3; // 合约入参
}
```

OperationLog数据结构

该数据结构用于记录log信息。

```
message OperationLog{
string topic = 1; // 日志主题
repeated string datas = 2; // 日志内容
}
```

OperationSetPrivilege数据结构

该数据结构用于设置账户权限。

```
message OperationSetPrivilege{
string master_weight = 1; // 账户自身权重
repeated Signer signers = 2; // 签名者权重列表
string tx_threshold = 3; // 交易门限
repeated OperationTypeThreshold type_thresholds = 4; // 指定操作的交易门限列表，未指定的操作
的交易以tx_threshold为门限
}
```

5.9.2 使用示例

本节中提供了proto脚本，以及 `cpp`、`java`、`javascript`、`pyton`、`object-c` 和 `php` 生成的proto源码的示例，详细信息请查看以下链接:

https://github.com/bumoproject/bumo/tree/develop/src/proto

链接中的目录结构说明:

1. cpp: C++的源码
2. io: Java的源码
3. go: Go的源码及测试程序
4. js: Javascript的源码及测试程序
5. python: Python的源码及测试程序
6. ios: Object-c的源码及测试程序
7. php: PHP的源码及测试程序

5.10 交易提交示例

场景：账户A （`buQVkUUBKpDKRmHYWw1MU8U7ngoQehno165i`）创建账户B（通过Keypair中的 生成地址 来生成新账户地址）。

5.10.1 接口生成transaction_blob示例

通过接口生成transaction_blob包含以下步骤:

1. 通过GET获取待发起交易账户的nonce值。

```
GET http://seed1.bumotest.io:26002/getAccount?
↪address=buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw
```

得到的响应报文:

```
{
"error_code" : 0,
"result" : {
"address" : "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw",
"assets" : [
{
"amount" : 1000000000,
"key" : {
"code" : "HNC",
"issuer" : "buQBjJD1BSJ7nzAbzdTenAhpFjmxRVEEtmxH"
}
}
],
"assets_hash" : "3bf279af496877a51303e91c36d42d64ba9d414de8c038719b842e6421a9dae0",
"balance" : 27034700,
"metadatas" : null,
"metadatas_hash" : "ad67d57ae19de8068dbcd47282146bd553fe9f684c57c8c114453863ee41abc3",
"nonce" : 5,
"priv" : {
"master_weight" : 1,
"thresholds" : [{
"tx_threshold" : 1
}
]
}
}
}
address: 当前查询的账户地址
assets: 账户资产列表
assets_hash: 资产列表hash
balance: 账户资产余额
metadata: 交易备注，必须是16进制
metadatas_hash: 交易备注hash
nonce: 转出方交易序列号，通过查询账户信息接口返回的nonce + 1
priv: 权限
master_weight: 当前账户权重
thresholds: 门限
tx_threshold: 交易默认门限
```

2. 完成交易数据填充。

通过 Keypair 中的 生成地址 生成的新账户B的地址是buQoP2eRymAcUm3uvWgQ8RnjtrSnXBXfAzsV，填充的json数据如下：

```
{
"source_address":"buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw",
"nonce":7,
"ceil_ledger_seq": 0,
"fee_limit":1000000,
"gas_price": 1000,
"metadata":"",
"operations":[
{
"type": 1,
"create_account": {
"dest_address": "buQoP2eRymAcUm3uvWgQ8RnjtrSnXBXfAzsV",
"init_balance": 10000000,
"priv": {
```

(continues on next page)

(续上页)

```
"master_weight": 1,
"thresholds": {
"tx_threshold": 1
}
}
}
}
]
}
```

注解: 这里的nonce值不是6，没有连续，因此该交易会超时，不会成功。

3. 对交易数据进行序列化处理。

```
POST http://seed1.bumotest.io:26002/getTransactionBlob
```

请求报文: 4.1.2中填充的json数据 响应报文:

```
{
"error_code": 0,
"error_desc": "",
"result": {
"hash": "be4953bce94ecd5c5a19c7c4445d940c6a55fb56370f7f606e127776053b3b51",
"transaction_blob":
↪"0a2462755173757248314d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3
↪"
}
}
```

4. 通过私钥对交易（transaction_blob）签名。

导入包:import io.bumo.encryption.key.PrivateKey;

私钥是:

```
privbvTuL1k8z27i9eyBrFDUvAVVCSxKeLtzjMMZEqimFwbNchnejS81
```

签名后的sign_data是:

```
9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A306
```

5. 完成交易数据填充。

```
{
"items" : [{
"transaction_blob" :
↪"0a2462755173757248314d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3
↪",
"signatures" : [{
"sign_data" :
↪"9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A
↪",
"public_key" :
↪"b00179b4adb1d3188aa1b98d6977a837bd4afdbb4813ac65472074fe3a491979bf256ba63895"
}
```

(continues on next page)

(续上页)

```
]
}
]
}
```

6. 通过POST提交交易。

```
POST http://seed1.bumotest.io/submitTransaction
```

得到如下的响应报文:

```
{
"results": [{
"error_code": 0,
"error_desc": "",
"hash": "be4953bce94ecd5c5a19c7c4445d940c6a55fb56370f7f606e127776053b3b51"
}
],
"success_count": 1
}
```

注解: “success_count”:1表示提交成功。

5.10.2 自己生成transaction_blob示例

自己生成transaction_blob（以Java为例）包含以下步骤:

1. 通过GET获取待发起交易账户的nonce值。

```
GET http://seed1.bumotest.io:26002/getAccount?
↪address=buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw
```

得到的响应报文:

```
{
"error_code" : 0,
"result" : {
"address" : "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw",
"assets" : [
{
"amount" : 1000000000,
"key" : {
"code" : "HNC",
"issuer" : "buQBjJD1BSJ7nzAbzdTenAhpFjmxRVEEtmxH"
}
}
],
"assets_hash" : "3bf279af496877a51303e91c36d42d64ba9d414de8c038719b842e6421a9dae0",
"balance" : 27034700,
"metadatas" : null,
"metadatas_hash" : "ad67d57ae19de8068dbcd47282146bd553fe9f684c57c8c114453863ee41abc3",
"nonce" : 5,
"priv" : {
```

(continues on next page)

(续上页)

```
"master_weight" : 1,
"thresholds" : [{
"tx_threshold" : 1
}
]
}
}
}
address: 当前查询的账户地址
assets: 账户资产列表
assets_hash: 资产列表hash
balance: 账户资产余额
metadata: 交易备注，必须是16进制
metadatas_hash: 交易备注hash
nonce: 转出方交易序列号，通过查询账户信息接口返回的nonce + 1
priv: 权限
master_weight: 当前账户权重
thresholds: 门限
tx_threshold: 交易默认门限
```

2. 填充交易（Transaction）数据结构，并生成transaction_blob。

导入包:import io.bumo.sdk.core.extend.protobuf.Chain;

```
Chain.Transaction.Builder builder = Chain.Transaction.newBuilder();
builder.setSourceAddress("buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw");
builder.setNonce(7);
builder.setFeeLimit(1000 * 1000);
builder.setGasPrice(1000);
builder.setCeilLedgerSeq(0);
builder.setMetadata(ByteString.copyFromUtf8(""));
Chain.Operation.Builder operation = builder.addOperationsBuilder();
operation.setType(Chain.Operation.Type.CREATE_ACCOUNT);
Chain.OperationCreateAccount.Builder operationCreateAccount = Chain.
↪OperationCreateAccount.newBuilder();
operationCreateAccount.setDestAddress("buQoP2eRymAcUm3uvWgQ8RnjtrSnXBXfAzsV");
operationCreateAccount.setInitBalance(10000000);
Chain.AccountPrivilege.Builder accountPrivilegeBuilder = Chain.AccountPrivilege.
↪newBuilder();
accountPrivilegeBuilder.setMasterWeight(1);
Chain.AccountThreshold.Builder accountThresholdBuilder = Chain.AccountThreshold.
↪newBuilder();
accountThresholdBuilder.setTxThreshold(1);
accountPrivilegeBuilder.setThresholds(accountThresholdBuilder);
operationCreateAccount.setPriv(accountPrivilegeBuilder);
operation.setCreateAccount(operationCreateAccount);
String transaction_blob = HexFormat.byteToHex(builder.build().toByteArray());
得到的transaction_blob是:
0a2462755173757248314d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3708
```

注解: 这里的nonce值不是6，没有连续，因此该交易会超时，不会成功。

3. 通过私钥对交易（transaction_blob）签名。

导入包:import io.bumo.encryption.key.PrivateKey;

私钥是：

```
privbvTuL1k8z27i9eyBrFDUvAVVCSxKeLtzjMMZEqimFwbNchnejS81
```

签名后的sign_data是：

```
9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A306
```

4. 完成交易数据填充。

```
{
"items" : [{
"transaction_blob" :
↪"0a2462755173757248314d34726a4c6b666a7a6b7852394b584a366a537532723978424e4577100718c0843d20e8073a3
↪",
"signatures" : [{
"sign_data" :
↪"9C86CE621A1C9368E93F332C55FDF423C087631B51E95381B80F81044714E3CE3DCF5E4634E5BE77B12ABD3C54554E834A
↪",
"public_key" :
↪"b00179b4adb1d3188aa1b98d6977a837bd4afdbb4813ac65472074fe3a491979bf256ba63895"
}
]
}
]
}
```

5. 通过POST提交交易。

```
POST http://seed1.bumotest.io/submitTransaction
```

得到的响应报文：

```
{
"results": [{
"error_code": 0,
"error_desc": "",
"hash": "be4953bce94ecd5c5a19c7c4445d940c6a55fb56370f7f606e127776053b3b51"
}
],
"success_count": 1
}
```

注解： “success_count”:1表明交易提交成功。

CHAPTER 6

交易所对接指南

6.1 概述

本文档用于交易所对接BUMO节点、安装部署以及使用BUMO SDK。

6.2 安装BUMO节点

支持 Ubuntu、Centos 等大多数操作系统编译，推荐使用版本Ubuntu 14.04或Centos 7。本示例将以 Ubuntu 14.04 作为节点的安装示例。

6.2.1 安装依赖

在编译BUMO的源代码之前需要安装系统所需的依赖。安装依赖需要完成以下步骤:

1. 输入以下命令安装 `automake`。

```
sudo apt-get install automake
```

2. 输入以下命令安装 `autoconf`。

```
sudo apt-get install autoconf
```

3. 输入以下命令安装 `libtool`。

```
sudo apt-get install libtool
```

4. 输入以下命令安装 `g++`。

```
sudo apt-get install g++
```

5. 输入以下命令安装 `libssl-dev`。

```
sudo apt-get install libssl-dev
```

6. 输入以下命令安装 cmake。

```
sudo apt-get install cmake
```

7. 输入以下命令安装 libbz2-dev。

```
sudo apt-get install libbz2-dev
```

8. 输入以下命令安装 python。

```
sudo apt-get install python
```

9. 输入以下命令安装 unzip。

```
sudo apt-get install unzip
```

6.2.2 编译源代码

在成功安装依赖后才能编译BUMO的源代码。如果没有安装 git，可以通过 sudo apt-get install git 命令来安装 git。编译BUMO的源代码需要完成以下步骤:

1. 在根目录下输入以下命令下载BUMO的源代码文件。

```
git clone https://github.com/bumoproject/bumo.git
```

注解: 在BUMO的源代码下载过程中将自动创建bumo/目录，源代码文件将存放到该目录下。

2. 输入以下命令进入到源代码的文件目录。

```
cd /bumo/build/
```

3. 输入以下命令下载依赖并初始化开发环境。

```
./install-build-deps-linux.sh
```

4. 输入以下命令回到bumo/目录下。

```
cd ../
```

5. 输入以下命令完成BUMO源代码的编译。出现下图所示信息则表示编译成功。

```
make
```

```
make[3]: Leaving directory `/bumo/build/linux'
/usr/bin/cmake -E cmake_progress_report /bumo/build/linux/CMakeFiles  1 2 3 4 5 6 7 8 9
[100%] Built target bumo
make[2]: Leaving directory `/bumo/build/linux'
/usr/bin/cmake -E cmake_progress_start /bumo/build/linux/CMakeFiles 0
make[1]: Leaving directory `/bumo/build/linux'
```

注解: 编译完成后生成的可执行文件 **bumo** 和 **bumod** 存放在/bumo/bin目录下。

6.2.3 安装BUMO节点

在编译完成后才能安装BUMO节点。安装BUMO节点需要完成以下步骤:

1. 输入以下命令进入到安装目录。

```
cd /bumo/
```

2. 输入以下命令完成安装。出现下图所示信息则表示安装成功。

```
make install
```

```
sudo mkdir -p /usr/local/buchain/coredump;
make[1]: Leaving directory `/bumo/build/linux'
```

注解:

- 默认情况下服务安装在/usr/local/buchain/目录下。
- 安装完成后无需其他配置即可通过 `service bumo start` 命令来启动bumo服务。
- 安装完BUMO节点后在buchain/目录下有如下目录结构:

目录	说明
bin	存放可执行文件（编译后的bumo可执行程序）
jslib	存放第三方js库
config	配置文件目录包含：bumo.json
data	数据库目录，存放账本数据
scripts	启停脚本目录
log	运行日志存储目录

6.2.4 更改运行环境

在更改BUMO的运行环境之前需要关闭BUMO服务 。您可按照以下步骤进行修改。

1. 输入以下命令进入到配置文件目录。

```
cd /usr/local/buchain/config/
```

注解:

在该目录下提供了以下运行环境的配置文件。

- bumo-mainnet.json（该文件是主网环境的配置文件，应用在生成环境中）
- bumo-testnet.json（该文件是测试网环境的配置文件）
- bumo-single.json（该文件是单节点调试环境的配置文件）

2. 把当前运行环境的配置文件（bumo.json）更改为其他名称，例如：

```
mv bumo.json bumoprevious.json
```

3. 把要运行的环境配置文件更改为bumo.json，例如：

```
mv bumo-mainnet.json bumo.json
```

注解： 本示例中把主网环境设置成了运行环境。更改运行环境后需要清空数据库才能重启bumo服务。

6.3 运维服务

在运维服务中对BUMO服务的启动、关闭、状态查询、系统详情查询、清空数据库进行了详细说明。

启动BUMO服务

输入以下命令启动bumo服务。

```
service bumo start
```

关闭BUMO服务

输入以下命令关闭bumo服务。

```
service bumo stop
```

查询BUMO服务状态

输入以下命令查询bumo服务。

```
service bumo status
```

查询系统详细状态

输入以下命令查询系统详细状态：

```
curl 127.0.0.1:19333/getModulesStatus
```

得到如下结果：

```
{
 "glue_manager":{
     "cache_topic_size":0,
     "ledger_upgrade":{
         "current_states":null,
         "local_state":null
     },
     "system":{
         "current_time":"2017-07-20 10:32:22", //当前系统时间
```

(continues on next page)

(续上页)

```
        "process_uptime":"2017-07-20 09:35:06", //bumo启动时间
        "uptime":"2017-05-14 23:51:04"
    },
    "time":"0 ms",
    "transaction_size":0
},
"keyvalue_db":Object{...},
"ledger_db":Object{...},
"ledger_manager":{
    "account_count":2316,  //账户数
    "hash_type":"sha256",
    "ledger_sequence":12187,
    "time":"0 ms",
    "tx_count":1185   //交易数
},
"peer_manager":Object{...},
"web server":Object{...},
```

清空数据库

在清空数据之前需要停止BUMO服务。清空数据库需要完成以下步骤:

1. 输入以下命令进入bumo的服务目录。

```
/usr/local/buchain/bin
```

2. 输入以下命令清空数据库。

```
./bumo --dropdb
```

注解: 数据库成功清空后能看到如下所示的信息。

```
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(153):Initialize db succes
sful
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(156):Drop db successfully
```

6.4 JAVA SDK 用法说明

JAVA SDK的使用包括了 生成用户充值地址 、检测账户地址的合法性 以及 资产交易。

6.4.1 生成用户充值地址

交易所需要给每一个用户生成一个充值地址，交易所可通过 Bumo-sdk-java 中提供的Keypair.generator()创建用户的充值地址，具体示例如下所示:

```
/**

     * 生成账户私钥、公钥以及地址
     */
    @Test
    public void createAccount() {
```

(continues on next page)

(续上页)

```
        Keypair keypair = Keypair.generator();
        System.out.println(JSON.toJSONString(keypair, true));
    }
```

返回值如下所示：

```
{
    "address":"buQsG8XBSNSR5xzpRzRACfQNgxL5SMr2fenF",
    "privateKey":"privbyzj3bKdsB8SkVGmfp77JM2CZ4bsHei38GkSUHx3nHhApuGxtVvo",
    "publicKey":"b0012adc1eab24afdb266eb6f0341c893cff92aaef4dbc9e21d645f6accb48e6834b6336eb56"
}
```

6.4.2 检测账户地址的合法性

通过如下所示代码检测账户地址的合法性。

```
/**
     * 检验账户地址是否合法
     */
    @Test
    public void checkAccountAddress() {
        String address = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
        AccountCheckValidRequest accountCheckValidRequest = new
↪AccountCheckValidRequest();
        accountCheckValidRequest.setAddress(address);
        AccountCheckValidResponse accountCheckValidResponse = sdk.getAccountService().
↪checkValid(accountCheckValidRequest);
        if (0 == accountCheckValidResponse.getErrorCode()) {
            System.out.println(accountCheckValidResponse.getResult().isValid());
        } else {
            System.out.println(JSON.toJSONString(accountCheckValidResponse, true));
        }
    }
```

注解:

- 如果返回值为 `true` 则表示账户地址合法
- 如果返回值为 `false` 则表示账户地址非法

6.4.3 资产交易

在BUMO 网络里，每10秒产生一个区块，每个交易只需要一次确认即可得到交易终态。在本章节将介绍 探测用户充值 、用户提现或转账 以及 查询交易 。

6.4.3.1 探测用户充值

交易所需要开发监听区块生成，然后解析区块里的交易记录，从而确认用户充值行为。具体步骤如下:

1. 确保节点区块状态正常。

2. 解析区块里包含的交易（解析方法见解析区块交易）。
3. 记录解析后的结果。

查看区块状态

通过如下所示代码查看区块状态。

```
/**

     * 检测连接的节点是否区块同步正常
     */
    @Test
    public void checkBlockStatus() {
        BlockCheckStatusResponse response = sdk.getBlockService().checkStatus();
        System.out.println(response.getResult().getSynchronous());
    }
```

注解:

- 如果返回值为 `true` 则表示区块正常
- 如果返回值为 `false` 则表示区块异常

解析区块交易

交易所可根据区块高度查询该区块里的交易信息，然后分析每条交易信息。

请求示例:

```
/**

     * 探测用户充值操作
     *
     * 通过解析区块中的交易来探测用户的充值动作
     */
    @Test
    public void getTransactionOfBolck() {
        Long blockNumber = 617247L;// 第617247个区块
        BlockGetTransactionsRequest request = new BlockGetTransactionsRequest();
        request.setBlockNumber(blockNumber);
        BlockGetTransactionsResponse response = sdk.getBlockService().
↪getTransactions(request);
        if (0 == response.getErrorCode()) {
            System.out.println(JSON.toJSONString(response, true));
        } else {
            System.out.println("Failure\n" + JSON.toJSONString(response, true));
        }
        // 探测某个账户是否充值BU
        // 解析transactions[n].transaction.operations[n].pay_coin.dest_address

        // 注意:
        // Operations是数组，有可能有多笔转账操作
    }
```

响应报文如下:

```
{
      "total_count": 1,
      "transactions": [{
              "close_time": 1524467568753121,
              "error_code": 0,
              "error_desc": "",
              "hash":
↪"89402813097402d1983c178c5ec271c6890db40c3beb9f06db71c8d52dab6c86",
              "ledger_seq": 33063,
              "signatures": [{
                      "public_key":
↪"b001dbf0942450f5601e39ac1f7223e332fe0324f1f91ec16c286258caba46dd29f6ef9bf93b",
                      "sign_data":
↪"668984fc7ded2dd30d87a1577f78eeb34d2198de3485be14ea66d9ca18f21aa21b2e0461ad8fedefc1abcb4221d346b404
↪"
              }],
              "transaction": {
                      "fee_limit": 1000000,
                      "gas_price": 1000,
                      "metadata": "333133323333",
                      "nonce": 25,
                      "operations": [{
                              "pay_coin": {
                                      "amount": 3000,
                                      "dest_address":
↪"buQctxUa367fjw9jegzMVvdux5eCdEhX18ME"
                              },
                              "type": 7
                      }],
                      "source_address": "buQhP7pzmjoRsNG7AkhfNxiWd7HuYsYnLa4x"
              }
      }]
}
```


响应报文解释:

total_count	交易总数（一般情况下都是1）
transactions	查询区块中交易对象，数组大小是该区块的交易总数
actual_fee	交易费用，单位是MO
close_time	交易时间
error_code	交易状态 0 是成功 非0 为失败
error_desc	交易状态信息
hash	交易哈希
ledger_seq	区块高度
signatures	签名信息
public_key	签名者公钥
sign_data	签名者签名数据
transaction	签名对象
fee_limit	费用最小值，单位 MO
gas_price	Gas，单位 MO
metadata	交易附加信息
nonce	交易原账号交易数
operations	操作对象(支持多个)
pay_coin	操作类型：内置token
amount	转移BU数量，单位 MO
dest_address	接收方地址
type	操作类型：7 为内置token转移
source_address	转出方地址

注解:

- 关于Bumo-sdk-java 如何使用，请访问以下链接：

  https://github.com/bumoproject/bumo-sdk-java/tree/release2.0.0

- 关于交易所对接示例，请访问以下链接：

  https://github.com/bumoproject/bumo-sdk-java/blob/release2.0.0/examples/src/main/java/io/bumo/sdk/example/ExchangeDemo.java

6.4.3.2 用户提现或转账

用户提现操作可参考bumo-sdk-java 提供的转账示例，如下所示：

```
/**
     * 发送一笔BU交易
     *
     * @throws Exception
     */
    @Test
    public void sendBu() throws Exception {
        // 初始化变量
        // 发送方私钥
        String senderPrivateKey =
↪"privbyQCRp7DLqKtRFCqKQJr81TurTqG6UKXMMtGAmPG3abcM9XHjWvq";
        // 接收方账户地址
        String destAddress = "buQswSaKDACkrFsnP1wcVsLAUzXQsemauE";
        // 发送BU数量
        Long amount = ToBaseUnit.BU2MO("0.01");
        // 固定写 1000L，单位是MO
        Long gasPrice = 1000L;
        // 设置最大费用 0.01BU
        Long feeLimit = ToBaseUnit.BU2MO("0.01");
        // 参考getAccountNonce()获取账户Nonce+ 1
        Long nonce = 1L;

        // 记录 txhash，以便后续再次确认交易真实结果
        // 推荐5个区块后通过txhash再次调用`根据交易Hash获取交易信息`（参考提示：getTxByHash()）
来确认交易终态结果
        String txhash = sendBu(senderPrivateKey, destAddress, amount, nonce, gasPrice,
↪ feeLimit);

    }
```

注解:

- 记录提现操作的hash值，以便后续查看该笔提现操作的终态结果
- `gasPrice` 目前（2018-04-23）最低值是1000MO
- `feeLimit` 建议填写1000000MO，即0.01BU

6.4.3.3 查询交易

用户提现操作的终态结果可通过当时发起提现操作时返回的hash值进行查询。

调用示例如下所示：

```
public static void queryTransactionByHash(BcQueryService queryService) {
  String txHash = "";
  TransactionHistory tx = queryService.getTransactionHistoryByHash(txHash);
  System.out.println(tx);
}
```

注解：

- `tx.totalCount` 数量大于等于1时说明交易历史存在
- `tx.transactions.errorCode` 等于0表示交易成功，非0表示交易失败，具体原因查看 `errorDesc`
- 用户提现操作，交易所请关注 `pay_coin` 操作
- 完整用户提现响应示例：

```
{
      "total_count": 1,
      "transactions": [{
              "close_time": 1524467568753121,
              "error_code": 0,
              "error_desc": "",
              "hash":
↪"89402813097402d1983c178c5ec271c6890db40c3beb9f06db71c8d52dab6c86",
              "ledger_seq": 33063,
              "signatures": [{
                      "public_key":
↪"b001dbf0942450f5601e39ac1f7223e332fe0324f1f91ec16c286258caba46dd29f6ef9bf93b",
                      "sign_data":
↪"668984fc7ded2dd30d87a1577f78eeb34d2198de3485be14ea66d9ca18f21aa21b2e0461ad8fedefc1abcb4221d346b404
↪"
              }],
              "transaction": {
                      "fee_limit": 1000000,
                      "gas_price": 1000,
                      "metadata": "333133323333",
                      "nonce": 25,
                      "operations": [{
                              "pay_coin": {
                                      "amount": 3000,
                                      "dest_address":
↪"buQctxUa367fjw9jegzMVvdux5eCdEhX18ME"
                              },
                              "type": 7
                      }],
                      "source_address": "buQhP7pzmjoRsNG7AkhfNxiWd7HuYsYnLa4x"
              }
      }]
}
```

(continues on next page)

(续上页)

total_count	交易总数（一般情况下都是1）
transactions	查询区块中交易对象，数组大小是该区块的交易总数
actual_fee	交易费用，单位是MO
close_time	交易时间
error_code	交易状态 0 是成功 非0 为失败
error_desc	交易状态信息
hash	交易哈希
ledger_seq	区块高度
signatures	签名信息
public_key	签名者公钥
sign_data	签名者签名数据
transaction	签名对象
fee_limit	费用最小值，单位 MO
gas_price	Gas，单位 MO
metadata	交易附加信息
nonce	交易原账号交易数
operations	操作对象(支持多个)
pay_coin	操作类型：内置token
amount	转移BU数量，单位 MO
dest_address	接收方地址
type	操作类型：7 为内置token转移
source_address	转出方地址

6.5 BU-Explorer

BUMO提供了区块链数据浏览工具，可供用户查询区块数据。

您访问以下链接查询区块链数据：

- 测试网：http://explorer.bumotest.io
- 主网：http://explorer.bumo.io

6.6 BUMO钱包

BUMO提供了Windows和Mac版全节点钱包，可供用户管理用户私钥、查看BU余额转账以及离线签名交易等功能。

您可以通过以链接下载BUMO钱包：

https://github.com/bumoproject/bumo-wallet/releases

6.7 常见问题

BUChain命令行的节点启动

问：使用BUChain命令行时是否需要启动该节点?

答：不用。

gas_price和fee_limit的值是否固定

问：gas_price 是固定1000MO，fee_limit 是1000000MO 吗?

答：不是固定。但目前(2018-04-23) gas_price 是1000MO，gas_price 越大越优先打包。fee_limit 是交易时交易发起方最多给区块链的交易费用，在正常合法的交易情况下区块链收取的真实费用小于调用方填写的 fee_limit 。(gas_price 可通过http://seed1.bumo.io:16002/getLedger?with_fee=true查询的结果 result.fees.gas_price 字段得到）。

账户余额转出

问：账户的余额能否全部转出?

答：不能。为了防止DDOS 攻击，防止创建大量垃圾账户，BUMO激活的账户必须保留一定数量的BU，目前是0.1BU（可通过http://seed1.bumo.io:16002/getLedger?with_fee=true 查询的结果 result.fees.base_reserve 字段得到）。

CHAPTER 7

BUMO JAVA SDK指南

7.1 概述

本文档对Bumo Java SDK常用的接口进行了详细说明, 使开发者能够更方便地操作和查询BU区块链。

7.2 术语

本章节对该文档中使用到的术语进行了详细说明。

操作BU区块链

操作BU区块链是指向BU区块链写入或修改数据。

提交交易

提交交易是指向BU区块链发送写入或修改数据的请求。

查询BU区块链

查询BU区块链是指查询BU区块链中的数据。

账户服务

账户服务提供了账户相关的有效性校验与查询接口。

资产服务

资产服务提供了资产相关的查询接口，该资产遵循ATP1.0协议。

Ctp10Token服务

Ctp10Token服务提供了合约资产相关的有效性校验与查询接口，该资产遵循CTP1.0协议。

合约服务

合约服务提供了合约相关的有效性校验与查询接口。

交易服务

交易服务提供了构建交易Blob接口、签名接口、查询与提交交易接口。

区块服务

区块服务提供了区块的查询接口。

账户nonce值

账户nonce值用于标识用户提交交易时交易执行的顺序。

7.3 请求参数与响应数据格式

本章节将详细介绍请求参数与响应数据的格式。

7.3.1 请求参数

接口的请求参数的类名，由 **服务名+方法名+Request** 构成，例如: 账户服务下的 `getInfo` 接口的请求参数格式是 `AccountGetInfoRequest`。

请求参数的成员，是各个接口的入参成员。例如：账户服务下的 `getInfo` 接口的入参成员是 `address`，那么该接口的请求参数的完整结构如下：

```
Class AccountGetInfoRequest {
String address;
}
```

7.3.2 响应数据

接口的响应数据的类名，由 **服务名+方法名+Response** 构成，例如：账户服务下的 `getNonce` 接口的响应数据格式是 `AccountGetNonceResponse`。

响应数据的成员包括错误码、错误描述和返回结果。例如，资产服务下的 `getInfo` 接口的响应数据的成员如下：

```
Class AccountGetNonceResponse {
Integer errorCode;
String errorDesc;
AccountGetNonceResult result;
}
```

注解:

- errorCode: 错误码。0表示无错误，大于0表示有错误
- errorDesc: 错误描述
- result: 返回结果。一个结构体，其类名由 **服务名+方法名+Result** 构成，其成员是各个接口返回值的成员，例如：账户服务下的 `getNonce` 接口的结果类名是 `AccountGetNonceResult` ，成员有nonce，完整结构如下：

```
Class AccountGetNonceResult {
Long nonce;
}
```

7.4 使用方法

本章节介绍SDK的使用流程。首先需要生成SDK实现，然后调用相应服务的接口。服务包括账户服务、资产服务、Ctp1.0Token服务、合约服务、交易服务、区块服务。接口按用途分为生成公私钥地址接口、有效性校验接口、查询接口、广播交易相关接口。

7.4.1 生成SDK实例

生成SDK实例需调用SDK的接口 `getInstance` 来实现。具体调用如下所示：

```
String url = "http://seed1.bumotest.io";
SDK sdk = SDK.getInstance(url);
```

7.4.2 生成公私钥地址

生成公私钥地址接口用于生成BU区块链账户的公钥、私钥和地址。直接调用 `Keypair.generator` 接口即可实现。具体调用如下所示：

```
Keypair keypair = Keypair.generator();
System.out.println(keypair.getPrivateKey());
System.out.println(keypair.getPublicKey());
System.out.println(keypair.getAddress());
```

7.4.3 有效性校验

有效性校验接口用于校验信息的有效性，直接调用相应的接口即可实现。比如校验账户地址的有效性，具体调用如下所示：

```
// 初始化请求参数
String address = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
AccountCheckValidRequest request = new AccountCheckValidRequest();
request.setAddress(address);

// 调用checkValid接口
AccountCheckValidResponse response =
sdk.getAccountService().checkValid(request);
if(0 == response.getErrorCode()) {
System.out.println(response.getResult().isValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.4.4 查询

查询接口用于查询BU区块链上的数据，直接调用相应的接口即可实现。比如查询账户信息，具体调用如下所示：

```
// 初始化请求参数
String accountAddress = "buQemmMwmRQY1JkcU7w3nhruo%X5N3j6C29uo";
AccountGetInfoRequest request = new AccountGetInfoRequest();
request.setAddress(accountAddress);

// 调用getInfo接口
AccountGetInfoResponse response = sdk.getAccountService().getInfo(request);
if (response.getErrorCode() == 0) {
AccountGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result,true));
}
else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.4.5 广播交易

广播交易是指通过广播的方式发起交易。广播交易包括以下步骤：

1. 获取交易发起的账户*nonce*值
2. 构建操作
3. 序列化交易
4. 签名交易
5. 提交交易

7.4.5.1 获取交易发起的账户nonce值

开发者可自己维护各个账户的nonce值，在提交完一个交易后，自动为nonce值递增1，这样可以在短时间内发送多笔交易；否则，必须等上一个交易执行完成后，账户的nonce值才会加1。具体接口调用如下所示：

```
// 初始化请求参数
String senderAddress = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
AccountGetNonceRequest getNonceRequest = new AccountGetNonceRequest();
getNonceRequest.setAddress(senderAddress);

// 调用getNonce接口
AccountGetNonceResponse getNonceResponse = sdk.getAccountService().
↪getNonce(getNonceRequest);

// 赋值nonce
if (getNonceResponse.getErrorCode() == 0) {
AccountGetNonceResult result = getNonceResponse.getResult();
System.out.println("nonce: " + result.getNonce());
}
else {
System.out.println("error" + getNonceResponse.getErrorDesc());
}
```

7.4.5.2 构建操作

这里的操作是指在交易中做的一些动作，便于序列化交易和评估费用。例如，构建发送BU操作（BUSendOperation），具体接口调用如下所示：

```
String senderAddress = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
String destAddress = "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw";
Long buAmount = ToBaseUnit.BU2MO("10.9");

BUSendOperation operation = new BUSendOperation();
operation.setSourceAddress(senderAddress);
operation.setDestAddress(destAddress);
operation.setAmount(buAmount);
```

7.4.5.3 序列化交易

序列化交易接口用于序列化交易，并生成交易Blob串，便于网络传输。其中nonce和operation是上面接口得到的，具体接口调用如下所示：

```
// 初始化变量
String senderAddress = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
Long gasPrice = 1000L;
Long feeLimit = ToBaseUnit.BU2MO("0.01");

// 初始化请求参数
TransactionBuildBlobRequest buildBlobRequest = new TransactionBuildBlobRequest();
buildBlobRequest.setSourceAddress(senderAddress);
buildBlobRequest.setNonce(nonce + 1);
buildBlobRequest.setFeeLimit(feeLimit);
buildBlobRequest.setGasPrice(gasPrice);
buildBlobRequest.addOperation(operation);

// 调用buildBlob接口
TransactionBuildBlobResponse buildBlobResponse = sdk.getTransactionService().
↪buildBlob(buildBlobRequest);
if (buildBlobResponse.getErrorCode() == 0) {
TransactionBuildBlobResult result = buildBlobResponse.getResult();
System.out.println("txHash: " + result.getHash() + ", blob: " + result.
↪getTransactionBlob());
} else {
System.out.println("error: " + buildBlobResponse.getErrorDesc());
}
```

签名交易

签名交易接口用于交易发起者使用其账户私钥对交易进行签名。其中 transactionBlob 是上面接口得到的，具体接口调用如下所示：

```
// 初始化请求参数
String senderPrivateKey = "privbyQCRp7DLqKtRFCqKQJr81TurTqG6UKXMMtGAmPG3abcM9XHjWvq";
String []signerPrivateKeyArr = {senderPrivateKey};
TransactionSignRequest signRequest = new TransactionSignRequest();
signRequest.setBlob(transactionBlob);
for (int i = 0; i < signerPrivateKeyArr.length; i++) {
```

(continues on next page)

(续上页)

```
signRequest.addPrivateKey(signerPrivateKeyArr[i]);
}

// 调用sign接口
TransactionSignResponse signResponse = sdk.getTransactionService().sign(signRequest);
if (signResponse.getErrorCode() == 0) {
TransactionSignResult result = signResponse.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + signResponse.getErrorDesc());
}
```

7.4.5.4 提交交易

提交交易接口用于向BU区块链发送交易请求，触发交易的执行。其中 `transactionBlob` 和 `signResult` 是上面接口得到的，具体接口调用如下所示：

```
// 初始化请求参数
TransactionSubmitRequest submitRequest = new TransactionSubmitRequest();
submitRequest.setTransactionBlob(transactionBlob);
submitRequest.setSignatures(signResult.getSignatures());

// 调用submit接口
TransactionSubmitResponse response = sdk.getTransactionService().
↪submit(submitRequest);
if (0 == response.getErrorCode()) {
System.out.println("交易广播成功，hash=" + response.getResult().getHash());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.5 账户服务

账户服务提供账户相关的接口，包括6个接口：`checkValid`、`getInfo`、`getNonce`、`getBalance`、`getAssets`、`getM`

7.5.1 checkValid

`checkValid` 接口用于检查区块链账户地址的有效性。

调用方法如下所示：

```
AccounCheckValidResponse checkValid(AccountCheckValidRequest);
```

请求参数如下表所示：

参数	类型	描述
address	String	必填，待检查的区块链账户地址

响应数据如下表所示：

参数	类型	描述
isValid	String	是否有效

错误码如下表所示：

异常	错误码	描述
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String address = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
AccountCheckValidRequest request = new AccountCheckValidRequest();
request.setAddress(address);

// 调用checkValid
AccountCheckValidResponse response = sdk.getAccountService().checkValid(request);
if(0 == response.getErrorCode()) {
System.out.println(response.getResult().isValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.5.2 getInfo

`getInfo` 接口用于获取指定账户的信息。

调用方法如下所示:

```
AccountGetInfoResponse GetInfo(AccountGetInfoRequest);
```

请求参数如下表所示:

参数	类型	描述
address	String	必填，待查询的区块链账户地址

响应数据如下表所示:

参数	类型	描述
address	String	账户地址
balance	Long	账户余额，单位MO， 1BU = 10^8 MO，必须大于0
nonce	Long	账户交易序列号，必须大于0
priv	*Priv*	账户权限

错误码如下表所示：

异常	错误码	描述
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String accountAddress = "buQemmMwmRQY1JkcU7w3nhruoX5N3j6C29uo";
AccountGetInfoRequest request = new AccountGetInfoRequest();
request.setAddress(accountAddress);

// 调用getInfo接口
AccountGetInfoResponse response = sdk.getAccountService().getInfo(request);
if (response.getErrorCode() == 0) {
AccountGetInfoResult result = response.getResult();
System.out.println("账户信息: \n" + JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.5.2.1 Priv

Priv的具体信息如下表所示：

成员	类型	描述
masterWeight	Long	账户自身权重，大小限制[0,(Integer.MAX_VALUE *2L + 1)]
signers	*Signer* []	签名者权重列表
threshold	*Threshold*	门限

7.5.2.2 Signer

Signer的具体信息如下表所示：

成员	类型	描述
address	String	签名者区块链账户地址
weight	Long	签名者权重，大小限制[0, (Integer.MAX_VALUE * 2L + 1)]

7.5.2.3 Threshold

Threshold的具体信息如下表所示：

成员	类型	描述
txThreshold	Long	交易默认门限，大小限制[0,Long.MAX_VALUE]
typeThresholds	*TypeThreshold* []	不同类型交易的门限

7.5.2.4 TypeThreshold

TypeThreshold的具体信息如下表所示：

成员	类型	描述
type	Long	操作类型，必须大于0
threshold	Long	门限值，大小限制[0, Long.MAX_VALUE]

7.5.3 getNonce

getNonce接口用于获取指定账户的nonce值。

调用方法如下所示:

```
AccountGetNonceResponse getNonce(AccountGetNonceRequest);
```

请求参数如下表所示:

参数	类型	描述
address	String	必填，待查询的区块链账户地址

响应数据如下表所示:

参数	类型	描述
nonce	Long	账户交易序列号

错误码如下表所示：

异常	错误码	描述
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String accountAddress = "buQswSaKDACkrFsnP1wcVsLAUzXQsemauEjf";
AccountGetNonceRequest request = new AccountGetNonceRequest();
request.setAddress(accountAddress);

// 调用getNonce接口
AccountGetNonceResponse response = sdk.getAccountService().getNonce(request);
if(0 == response.getErrorCode()){
System.out.println("账户nonce:" + response.getResult().getNonce());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.5.4 getBalance

`getBalance` 接口用于获取指定账户的BU余额。

调用方法如下所示：

```
AccountGetBalanceResponse getBalance(AccountGetBalanceRequest);
```

请求参数如下表所示：

参数	类型	描述
address	String	必填，待查询的区块链账户地址

响应数据如下表所示：

参数	类型	描述
balance	Long	BU的余额, 单位MO，1 BU = 10^8 MO

错误码如下表所示：

异常	错误码	描述
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String accountAddress = "buQswSaKDACkrFsnP1wcVsLAUzXQsemauEjf";
AccountGetBalanceRequest request = new AccountGetBalanceRequest();
request.setAddress(accountAddress);

// 调用getBalance接口
AccountGetBalanceResponse response = sdk.getAccountService().getBalance(request);
if(0 == response.getErrorCode()){
AccountGetBalanceResult result = response.getResult();
System.out.println("BU余额: " + ToBaseUnit.MO2BU(result.getBalance().toString()) + " BU
↪");
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.5.5 getAssets

`getAssets` 接口用于获取指定账户的所有资产信息。

调用方法如下所示：

```
AccountGetAssets getAssets(AccountGetAssetsRequest);
```

请求参数如下表所示：

参数	类型	描述
address	String	必填，待查询的账户地址

响应数据如下表所示：

参数	类型	描述
asset	*AssetInfo* []	账户资产

错误码如下表所示：

异常	错误码	描述
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
NO_ASSET_ERROR	11009	The account does not have the asset
SYSTEM_ERROR	20000	System error

具体示例如下所示:

```
// 初始化请求参数
AccountGetAssetsRequest request = new AccountGetAssetsRequest();
request.setAddress("buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw");

// 调用getAssets接口
AccountGetAssetsResponse response = sdk.getAccountService().getAssets(request);
if (response.getErrorCode() == 0) {
AccountGetAssetsResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.5.5.1 AssetInfo

AssetInfo的具体信息如下表所示:

成员	类型	描述
key	*Key*	资产惟一标识
assetAmount	Long	资产数量

7.5.5.2 Key

Key的具体信息如下表所示:

成员	类型	描述
code	String	资产编码
issuer	String	资产发行账户地址

7.5.6 getMetadata

`getMetadata` 接口用于获取指定账户的metadata信息。

调用方法如下所示:

```
AccountGetMetadataResponse getMetadata(AccountGetMetadataRequest);
```

请求参数如下表所示:

参数	类型	描述
address	String	必填，待查询的账户地址
key	String	选填，metadata关键字，长度限制[1, 1024]

响应数据如下表所示：

参数	类型	描述
metadata	*MetadataInfo*	账户metadata

错误码如下表所示：

异常	错误码	描述
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
NO_METADATA_ERROR	11010	The account does not have the metadata
INVALID_DATAKEY_ERROR	11011	The length of key must be between 1 and 1024
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String accountAddress = "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw";
AccountGetMetadataRequest request = new AccountGetMetadataRequest();
request.setAddress(accountAddress);
request.setKey("20180704");

// 调用getMetadata接口
AccountGetMetadataResponse response = sdk.getAccountService().getMetadata(request);
if (response.getErrorCode() == 0) {
AccountGetMetadataResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.5.6.1 MetadataInfo

MetadataInfo的具体信息如下表所示：

成员	类型	描述
key	String	metadata的关键词
value	String	metadata的内容
version	Long	metadata的版本

7.6 资产服务

资产服务遵循ATP1.0协议，账户服务提供资产相关的接口，目前有1个接口：`getInfo`。

7.6.1 getInfo

`getInfo` 接口用于获取指定账户的指定资产信息。

调用方法如下所示：

```
AssetGetInfoResponse getInfo(AssetGetInfoRequest);
```

请求参数如下表所示：

参数	类型	描述
address	String	必填，待查询的账户地址
code	String	必填，资产编码，长度限制[1, 64]
issuer	String	必填，资产发行账户地址

响应数据如下表所示：

参数	类型	描述
asset	*AssetInfo* []	账户资产

错误码如下表所示：

异常	错误码	描述
INVALID_ADDRESS_ERROR	11006	Invalid address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
INVALID_ASSET_CODE_ERRO R	11023	The length of asset code must be between 1 and 64
INVALID_ISSUER_ADDRESS _ERROR	11027	Invalid issuer address
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
AssetGetInfoRequest request = new AssetGetInfoRequest();
request.setAddress("buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw");
request.setIssuer("buQBjJD1BSJ7nzAbzdTenAhpFjmxRVEEtmxH");
request.setCode("HNC");

// 调用getInfo消息
AssetGetInfoResponse response = sdk.getAssetService().getInfo(request);
if (response.getErrorCode() == 0) {
AssetGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.2 Ctp10Token服务

Ctp10Token服务遵循CTP1.0协议，主要提供合约Token相关的接口，目前有8个接口：checkValid、allowance、getInfo、getName、getSymbol、getDecimals、getTotalSupply、getBalan

7.6.3 checkValid

checkValid 接口用于验证合约Token的有效性。

调用方法如下所示：

```
Ctp10TokenCheckValidResponse checkValid(Ctp10TokenCheckValidRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	必填，待验证的Token合约地址

响应数据如下表所示：

参数	类型	描述
isValid	String	是否有效

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenCheckValidRequest request = new Ctp10TokenCheckValidRequest();
request.setContractAddress("buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea");

// 调用checkValid接口
Ctp10TokenCheckValidResponse response = sdk.getTokenService().checkValid(request);
if (response.getErrorCode() == 0) {
Ctp10TokenCheckValidResult result = response.getResult();
System.out.println(result.getValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.4 allowance

`allowance` 接口用于获取spender允许从owner提取的金额。

调用方法如下所示：

```
Ctp10TokenAllowanceResponse allowance(Ctp10TokenAllowanceRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	必填，合约账户地址
tokenOwner	String	必填，合约Token的持有者账户地址
spender	String	必填，被授权账户地址

响应数据如下表所示：

参数	类型	描述
allowance	String	允许提取的金额

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
INVALID_TOKENOWNER_ERRPR	11035	Invalid token owner
INVALID_SPENDER_ERROR	11043	Invalid spender
GET_ALLOWNANCE_ERROR	11036	Failed to get allowance
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenAllowanceRequest request = new Ctp10TokenAllowanceRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");
request.setTokenOwner("buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp");
request.setSpender("buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp");

// 调用allowance接口
Ctp10TokenAllowanceResponse response = sdk.getTokenService().allowance(request);
if (response.getErrorCode() == 0) {
Ctp10TokenAllowanceResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.5 getInfo-Ctp10Token

`getInfo-Ctp10Token` 接口用于获取合约Token的信息。

调用方法如下所示：

```
Ctp10TokenGetInfoResponse getInfo(Ctp10TokenGetInfoRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待查询的合约Token地址

响应数据如下表所示：

参数	类型	描述
ctp	String	合约Token版本号
symbol	String	合约Token符号
decimals	Integer	合约数量的精度
totalSupply	String	合约的总供应量
name	String	合约Token的名称
contractOwner	String	合约Token的拥有者

具体错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenGetInfoRequest request = new Ctp10TokenGetInfoRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// 调用getInfo接口
Ctp10TokenGetInfoResponse response = sdk.getTokenService().getInfo(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.6 getName

`getName` 接口用于获取合约Token的名称。

调用方法如下所示：

```
Ctp10TokenGetNameResponse getName(Ctp10TokenGetNameRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待查询的合约账户地址

响应数据如下表所示：

参数	类型	描述
name	String	合约Token的名称

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenGetNameRequest request = new Ctp10TokenGetNameRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// 调用getName接口
Ctp10TokenGetNameResponse response = sdk.getTokenService().getName(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetNameResult result = response.getResult();
System.out.println(result.getName());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.7 getSymbol

`getSymbol` 接口用于获取合约Token的符号。

调用方法如下所示：

```
Ctp10TokenGetSymbolResponse getSymbol (Ctp10TokenGetSymbolRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待查询的合约账户地址

响应数据如下表所示：

参数	类型	描述
symbol	String	合约Token的符号

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenGetSymbolRequest request = new Ctp10TokenGetSymbolRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// 调用getSymbol接口
Ctp10TokenGetSymbolResponse response = sdk.getTokenService().getSymbol(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetSymbolResult result = response.getResult();
System.out.println(result.getSymbol());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.8 getDecimals

`getDecimals` 接口用于获取合约Token的精度。

调用方法如下所示：

```
Ctp10TokenGetDecimalsResponse getDecimals (Ctp10TokenGetDecimalsRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待查询的合约账户地址

响应数据如下表所示：

参数	类型	描述
decimals	Integer	合约token精度

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenGetDecimalsRequest request = new Ctp10TokenGetDecimalsRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// 调用getDecimals接口
Ctp10TokenGetDecimalsResponse response = sdk.getTokenService().getDecimals(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetDecimalsResult result = response.getResult();
System.out.println(result.getDecimals());
} else {
```

(continues on next page)

(续上页)

```
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.9 getTotalSupply

getTotalSupply 接口用于获取合约Token的总供应量。

调用方法如下所示：

```
Ctp10TokenGetTotalSupplyResponse getTotalSupply(Ctp10TokenGetTotalSupplyRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待查询的合约账户地址

响应数据如下表所示：

参数	类型	描述
totalSupply	String	合约Token的总供应量

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenGetTotalSupplyRequest request = new Ctp10TokenGetTotalSupplyRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");

// 调用getTotalSupply接口
Ctp10TokenGetTotalSupplyResponse response = sdk.getTokenService().
↪getTotalSupply(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetTotalSupplyResult result = response.getResult();
System.out.println(result.getTotalSupply());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.6.10 getBalance-Ctp10Token

getBalance-Ctp10Token 接口获取合约Token拥有者的账户余额。

调用方法如下所示：

```
Ctp10TokenGetBalanceResponse getBalance(Ctp10TokenGetBalanceRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待查询的合约账户地址
tokenOwner	String	必填，合约Token持有者的账户地址

响应数据如下表所示：

参数	类型	描述
balance	Long	token的余额

错误码如下表所示：

异常	错误码	描述
INVALID_TOKENOWNER_ERRPR	11035	Invalid token owner
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Ctp10TokenGetBalanceRequest request = new Ctp10TokenGetBalanceRequest();
request.setContractAddress("buQhdBSkJqERBSsYiUShUZFMZQhXvkdNgnYq");
request.setTokenOwner("buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp");

// 调用getBalance接口
Ctp10TokenGetBalanceResponse response = sdk.getTokenService().getBalance(request);
if (response.getErrorCode() == 0) {
Ctp10TokenGetBalanceResult result = response.getResult();
System.out.println(result.getBalance());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.7 合约服务

合约服务提供合约相关的接口，目前有4个接口：`checkValid`、`getInfo`、`getAddress`、`call`。

7.7.1 checkValid

`checkValid` 接口用于检测合约账户的有效性。

调用方法如下所示：

```
ContractCheckValidResponse checkValid(ContractCheckValidRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待检测的合约账户地址

响应数据如下表所示：

参数	类型	描述
isValid	Boolean	是否有效

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS_ERROR	11037	Invalid contract address
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
ContractCheckValidRequest request = new ContractCheckValidRequest();
request.setContractAddress("buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea");

// 调用checkValid接口
ContractCheckValidResponse response = sdk.getContractService().checkValid(request);
if (response.getErrorCode() == 0) {
ContractCheckValidResult result = response.getResult();
System.out.println(result.getValid());
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.7.2 getInfo

getInfo 接口用于查询合约代码。

调用方法如下所示:

```
ContractGetInfoResponse getInfo (ContractGetInfoRequest);
```

请求参数如下表所示：

参数	类型	描述
contractAddress	String	待查询的合约账户地址

响应数据如下表所示：

参数	类型	描述
contract	*ContractInfo*	合约信息

错误码如下表所示：

异常	错误码	描述
INVALID_CONTRACTADDRESS _ERROR	11037	Invalid contract address
CONTRACTADDRESS_NOT_CON TRACTAC-COUNT_ERROR	11038	contractAddress is not a contract account
NO_SUCH_TOKEN_ERROR	11030	No such token
GET_TOKEN_INFO_ERROR	11066	Failed to get token info
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
ContractGetInfoRequest request = new ContractGetInfoRequest();
request.setContractAddress("buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea");

// 调用getInfo接口
ContractGetInfoResponse response = sdk.getContractService().getInfo(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.7.2.1 ContractInfo

ContractInfo的具体信息如下表所示：

成员	类型	描述
type	Integer	合约类型，默认为0
payload	String	合约代码

7.7.3 getAddress

`getAddress` 接口用于查询合约地址。

调用方法如下所示：

```
ContractGetAddressResponse getInfo (ContractGetAddressRequest);
```

请求参数如下表所示：

参数	类型	描述
hash	String	创建合约交易的hash

响应数据如下表所示：

参数	类型	描述
contractAddressList	List (*ContractAddressInfo*)	合约地址列表

错误码如下表所示：

异常	错误码	描述
INVALID_HASH_ERROR	11055	Invalid transaction hash
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
ContractGetAddressRequest request = new ContractGetAddressRequest();
request.setHash("44246c5ba1b8b835a5cbc29bdc9454cdb9a9d049870e41227f2dcfbcf7a07689");

// 调用getAddress接口
ContractGetAddressResponse response = sdk.getContractService().getAddress(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.7.4 ContractAddressInfo

ContractAddressInfo的具体信息如下表所示：

成员	类型	描述
contractAddress	String	合约地址
operationIndex	Integer	所在操作的下标

7.7.5 call

`call` 接口用于调试合约代码。

调用方法如下所示：

```
ContractCallesponse call(ContractCallRequest);
```

请求参数如下表所示：

参数	类型	描述
sourceAddress	String	选填，合约触发账户地址
contractAddress	contractAddress	选填，合约账户地址，与code不能同时为空
code	String	选填，合约代码，与contractAddress不能同时为空，长度限制[1,64]
input	String	选填，合约入参
contractBalance	String	选填，赋予合约的初始 BU余额, 单位MO，1 BU = 10^8MO, 大小限制[1, Long.MAX_VALUE]
optType	Integer	必填，0:调用合约的读写接口 init,1: 调用合约的读写接口main, 2 :调用只读接口 query
feeLimit	Long	交易要求的最低手续费，大小限制[1, Long.MAX_VALUE]
gasPrice	Long	交易燃料单价，大小限制[1000,Long.MAX_VALUE]

响应数据如下表所示：

参数	类型	描述
logs	JSONObject	日志信息
queryRets	JSONArray	查询结果集
stat	*ContractStat*	合约资源占用信息
txs	*TransactionEnvs*	交易集

错误码如下表所示：

异常	错误码	描述
INVALID_SOURCEADDRESS_E RROR	11002	Invalid sourceAddress
INVALID_CONTRACTADDRESS _ERROR	11037	Invalid contract address
CONTRACTADDRESS_CODE_BO TH_NULL_ERROR	11063	ContractAddress and code cannot be empty at the same time
INVALID_OPTTYPE_ERROR	11064	OptType must be between 0 and 2
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
ContractCallRequest request = new ContractCallRequest();
request.setCode("\"use strict\";log(undefined);function query() {␣
↪getBalance(thisAddress); }");
request.setFeeLimit(1000000000L);
request.setOptType(2);

// 调用call接口
ContractCallResponse response = sdk.getContractService().call(request);
if (response.getErrorCode() == 0) {
ContractCallResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.7.5.1 ContractStat

ContractStat的具体信息如下表所示：

成员	类型	描述
applyTime	Long	接收时间
memoryUsage	Long	内存占用量
stackUsage	Long	堆栈占用量
step	Long	完成需要的步数

7.7.5.2 TransactionEnvs

TransactionEnvs的具体信息如下表所示：

成员	类型	描述
transactionEnv	*TransactionEnv*	交易

7.7.5.3 TransactionEnv

TransactionEnv的具体信息如下表所示：

成员	类型	描述
transaction	*TransactionInfo*	交易内容
trigger	*ContractTrigger*	合约触发者

7.7.5.4 TransactionInfo

TransactionInfo的具体信息如下表所示：

成员	类型	描述
sourceAddress	String	交易发起的源账户地址
feeLimit	Long	交易要求的最低费用
gasPrice	Long	交易燃料单价
nonce	Long	交易序列号
operations	*Operation*	操作列表

7.7.5.5 ContractTrigger

ContractTrigger的具体信息如下表所示：

成员	类型	描述
transaction	*TriggerTransaction*	触发交易

7.7.5.6 Operation

Operation的具体信息如下表所示：

成员	类型	描述
type	Integer	操作类型
sourceAddress	String	操作发起源账户地址
metadata	String	备注
createAccount	*OperationCreateAccount*	创建账户操作
issueAsset	*OperationIssueAsset*	发行资产操作
payAsset	*OperationPayAsset*	转移资产操作
payCoin	*OperationPayCoin*	发送BU操作
setMetadata	*OperationSetMetadata*	设置metadata操作
setPrivilege	*OperationSetPrivilege*	设置账户权限操作
log	*OperationLog*	记录日志

7.7.5.7 TriggerTransaction

TriggerTransaction的具体信息如下表所示：

成员	类型	描述
hash	String	交易hash

7.7.5.8 OperationCreateAccount

OperationCreateAccount的具体信息如下表所示：

成员	类型	描述
destAddress	String	目标账户地址
contract	*Contract*	合约信息
priv	*Priv*	账户权限
metadata	MetadataInfo	账户
initBalance	Long	账户资产, 单位MO，1 BU = 10^8 MO,
initInput	String	合约init函数的入参

7.7.5.9 Contract

Contract的具体信息如下表所示：

成员	类型	描述
type	Integer	合约的语种，默认不赋值
payload	String	对应语种的合约代码

7.7.5.10 MetadataInfo

MetadataInfo的具体信息如下表所示:

成员	类型	描述
key	String	metadata的关键词
value	String	metadata的内容
version	Long	metadata的版本

7.7.5.11 OperationIssueAsset

OperationIssueAsset的具体信息如下表所示:

成员	类型	描述
code	String	资产编码
assetAmount	Long	资产数量

7.7.5.12 OperationPayAsset

OperationPayAsset的具体信息如下表所示:

成员	类型	描述
destAddress	String	待转移资产的目标账户地址
asset	*AssetInfo*	账户资产
input	String	合约main函数入参

7.7.5.13 OperationPayCoin

OperationPayCoin的具体信息如下表所示:

成员	类型	描述
destAddress	String	待转移的目标账户地址
buAmount	Long	待转移的BU数量
input	String	合约main函数入参

7.7.5.14 OperationSetMetadata

OperationSetMetadata的具体信息如下表所示:

成员	类型	描述
key	String	metadata的关键词
value	String	metadata的内容
version	Long	metadata的版本
deleteFlag	boolean	是否删除metadata

7.7.5.15 OperationSetPrivilege

OperationSetPrivilege的具体信息如下表所示:

成员	类型	描述
masterWeight	String	账户自身权重，大小限制[0, (Integer.MAX_VALUE *2L + 1)]
signers	*Signer*	签名者权重列表
txThreshold	String	交易门限，大小限制[0, Long.MAX_VALUE]
typeThreshold	*TypeThreshold*	指定类型交易门限

7.7.5.16 OperationLog

OperationLog的具体信息如下表所示:

成员	类型	描述
topic	String	日志主题
data	String[]	日志内容

7.8 交易服务

交易服务提供交易相关的接口，目前有5个接口：`buildBlob`, `evaluateFee`, `sign`, `submit`, `getInfo`。

7.8.1 buildBlob

`buildBlob` 接口用于序列化交易，生成交易Blob串，便于网络传输。

注 解: 在调用buildBlob之前需要构建一些操作，目前操作有16种，分别是：`AccountActivateOperation`、`AccountSetMetadataOperation`、`AccountSetPrivilegeOperation`、`Ass`

调用方法如下所示：

```
TransactionBuildBlobResponse buildBlob(TransactionBuildBlobRequest);
```

请求参数如下表所示：

参数	类型	描述
sourceAddress	String	必填，发起该操作的源账户地址
nonce	Long	必填，待发起的交易序列号，函数里+1，大小限制[1,Long.MAX_VALUE]
gasPrice	Long	必填，交易燃料单价，单位MO，1 BU = 10^8MO，大小限制[1000, Long.MAX_VALUE]
feeLimit	Long	必填，交易要求的最低的手续费，单位MO，1MO，大小限制[1000, Long.MAX_VALUE]
operation	BaseOperation[]	必填，待提交的操作列表，不能为空
ceilLedgerSeq	Long	选填，距离当前区块高度指定差值的区块内执行的限制，当区块超出当时区块高度与所设差值的和后，交易执行失败。必须大于等于0，是0时不限制
metadata	String	选填，备注

响应数据如下表所示：

参数	类型	描述
transactionBlob	String	Transaction序列化后的16进制字符串
hash	String	交易hash

错误码如下表所示：

异常	错误码	描述
INVALID_SOURCEADDRESS_E RROR	11002	Invalid sourceAddress
INVALID_NONCE_ERROR	11048	Nonce must be between 1 and Long.MAX_VALUE
INVALID_DESTADDRESS_ERR OR	11003	Invalid destAddress
INVALID_INITBALANCE_ERR OR	11004	InitBalance must be between 1 and Long.MAX_VAL
SOURCEADDRESS_EQUAL_DES TADDRESS_ERROR	11005	SourceAddress cannot be equal to destAddress
INVALID_ISSUE_AMMOUNT_E RROR	11008	AssetAmount this will be issued must be between 1 a
INVALID_DATAKEY_ERROR	11011	The length of key must be between 1 and 1024
INVALID_DATAVALUE_ERROR	11012	The length of value must be between 0 and 256000
INVALID_DATAVERSION_ERR OR	11013	The version must be equal to or greater than 0
INVALID_MASTERWEIGHT _ERROR	11015	MasterWeight must be between 0 and (Integer.MAX_
INVALID_SIGNER_ADDRESS _ERROR	11016	Invalid signer address
INVALID_SIGNER_WEIGHT _ERROR	11017	Signer weight must be between 0 and (Integer.MAX_
INVALID_TX_THRESHOLD_ER ROR	11018	TxThreshold must be between 0 and Long.MAX_VA
INVALID_OPERATION_TYPE _ERROR	11019	Operation type must be between 1 and 100
INVALID_TYPE_THRESHOLD _ERROR	11020	TypeThreshold must be between 0 and Long.MAX_V
INVALID_ASSET_CODE _ERROR	11023	The length of key must be between 1 and 64
INVALID_ASSET_AMOUNT_ER ROR	11024	AssetAmount must be between 0 and Long.MAX_VA
INVALID_BU_AMOUNT_ERROR	11026	BuAmount must be between 0 and Long.MAX_VAL
INVALID_ISSUER_ADDRESS _ERROR	11027	Invalid issuer address
NO_SUCH_TOKEN_ERROR	11030	No such token
INVALID_TOKEN_NAME_ERRO R	11031	The length of token name must be between 1 and 102
INVALID_TOKEN_SYMBOL_ER ROR	11032	The length of symbol must be between 1 and 1024
INVALID_TOKEN_DECIMALS _ERROR	11033	Decimals must be between 0 and 8
INVALID_TOKEN_TOTALSUPP LY_ERROR	11034	TotalSupply must be between 1 and Long.MAX_VAL
INVALID_TOKENOWNER_ERRP R	11035	Invalid token owner
INVALID_CONTRACTADDRESS _ERROR	11037	Invalid contract address
CONTRACTADDRESS_NOT_CON TRACTACCOUNT_ERROR	11038	ContractAddress is not a contract account
INVALID_TOKEN_AMOUNT_ER ROR	11039	Token amount must be between 1 and Long.MAX_V
SOURCEADDRESS_EQUAL_CON TRACTADDRESS_ERROR	11040	SourceAddress cannot be equal to contractAddress
INVALID_FROMADDRESS_ERR OR	11041	Invalid fromAddress
FROMADDRESS_EQUAL_DESTA DDRESS_ERROR	11042	FromAddress cannot be equal to destAddress
INVALID_SPENDER_ERROR	11043	Invalid spender
PAYLOAD_EMPTY_ERROR	11044	Payload cannot be empty
INVALID_LOG_TOPIC _ERROR	11045	The length of key must be between 1 and 128
INVALID_LOG_DATA _ERROR	11046	The length of value must be between 1 and 1024
INVALID_CONTRACT_TYPE_E RROR	11047	Type must be equal to or greater than 0
INVALID_NONCE_ERROR	11048	Nonce must be between 1 and Long.MAX_VALUE
INVALID_ GASPRICE_ERROR	11049	GasPrice must be between 1000 and Long.MAX_VA
INVALID_FEELIMIT_ERROR	11050	FeeLimit must be between 1 and Long.MAX_VALU
OPERATIONS_EMPTY_ERROR	11051	Operations cannot be empty
INVALID_CEILLEDGERSEQ_E RROR	11052	CeilLedgerSeq must be equal or bigger than 0
OPERATIONS_ONE_ERROR	11053	One of the operations cannot be resolved
REQUEST_NULL_ERROR	12001	Request parameter cannot be null

表 1 – 续上页

异常	错误码	描述
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化变量
String senderAddresss = "buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea";
String destAddress = "buQsurH1M4rjLkfjzkxR9KXJ6jSu2r9xBNEw";
Long buAmount = ToBaseUnit.BU2MO("10.9");
Long gasPrice = 1000L;
Long feeLimit = ToBaseUnit.BU2MO("0.01");
Long nonce = 1L;

// 构建sendBU操作
BUSendOperation operation = new BUSendOperation();
operation.setSourceAddress(senderAddresss);
operation.setDestAddress(destAddress);
operation.setAmount(buAmount);

// 初始化请求参数
TransactionBuildBlobRequest request = new TransactionBuildBlobRequest();
request.setSourceAddress(senderAddresss);
request.setNonce(nonce);
request.setFeeLimit(feeLimit);
request.setGasPrice(gasPrice);
request.addOperation(operation);

// 调用buildBlob接口
String transactionBlob = null;
TransactionBuildBlobResponse response = sdk.getTransactionService().
↪buildBlob(request);
if (response.getErrorCode() == 0) {
TransactionBuildBlobResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.8.1.1 BaseOperation

BaseOperation是buildBlob接口中所有操作的基类。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
metadata	String	选填，备注

7.8.1.2 AccountActivateOperation

AccountActivateOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.01 BU。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
destAddress	String	必填，目标账户地址
initBalance	Long	必填，初始化资产，单位MO，1 BU = 10^8 MO, 大小(0,Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.3 AccountSetMetadataOperation

AccountSetMetadataOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.01 BU。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
key	String	必填，metadata的关键词，长度限制[1, 1024]
value	String	必填，metadata的内容，长度限制[0, 256000]
version	Long	选填，metadata的版本
deleteFlag	Boolean	选填，是否删除metadata
metadata	String	选填，备注

7.8.1.4 AccountSetPrivilegeOperation

AccountSetPrivilegeOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.01 BU。

sourceAddress	String	选填，操作源账户地址
masterWeight	String	选填，账户自身权重，大小限制[0,(Integer.MAX_VALUE *2L + 1)]
signers	*Signer* []	选填，签名者权重列表
txThreshold	String	选填，交易门限，大小限制[0,Long.MAX_VALUE]
typeThreshold	*TypeThreshold* []	选填，指定类型交易门限
metadata	String	选填，备注

7.8.1.5 AssetIssueOperation

AssetIssueOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是50.01 BU。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
code	String	必填，资产编码，长度限制[1, 64]
assetAmount	Long	必填，资产发行数量，大小限制[0, Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.6 AssetSendOperation

AssetSendOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.01 BU。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
destAddress	String	必填，目标账户地址
code	String	必填，资产编码，长度限制[1, 64]
issuer	String	必填，资产发行账户地址
assetAmount	Long	必填，资产数量，大小限制[0, Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.7 BUSendOperation

BUSendOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.01 BU。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
destAddress	String	必填，目标账户地址
buAmount	Long	必填，资产发行数量，大小限制[0, Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.8 Ctp10TokenIssueOperation

Ctp10TokenIssueOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是10.08 BU。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
initBalance	Long	必填，给合约账户的初始化资产，单位MO，1必填，给合约账户的初始化资产，单位MO，1大小限制[1, max(64)]
name	String	必填，ctp10Token名称，长度限制[1,1024]
symbol	String	必填，ctp10Token符号，长度限制[1,1024]
decimals	Integer	必填，ctp10Token数量的精度，大小限制[0,8]
supply	String	必填，ctp10Token发行的总供应量(不带精度)，大小限制[1,Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.9 Ctp10TokenTransferOperation

Ctp10TokenTransferOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.02 BU

成员变量	类型	描述
sourceAddress	String	选填，合约token的持有者账户地址
contractAddress	String	必填，合约账户地址
destAddress	String	必填，待转移的目标账户地址
tokenAmount	String	必填，待转移的token数量，大小限制[1,Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.10 TokenTransferFromOperation

TokenTransferFromOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.02 BU。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
contractAddress	String	必填，合约账户地址
fromAddress	String	必填，待转移的源账户地址
destAddress	String	必填，待转移的目标账户地址
tokenAmount	String	必填，待转移的ctp10Token数量，大小限制[1,Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.11 Ctp10TokenApproveOperation

Ctp10TokenApproveOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.02 BU。

成员变量	类型	描述
sourceAddress	String	选填，合约token的持有者账户地址
contractAddress	String	必填，合约账户地址
spender	String	必填，授权的账户地址
tokenAmount	String	必填，被授权的待转移的ctp10Token数量，大小限制[1,Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.12 Ctp10TokenAssignOperation

Ctp10TokenAssignOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.02 BU。

成员变量	类型	描述
sourceAddress	String	选填，合约token的拥有者账户地址
contractAddress	String	必填，合约账户地址
destAddress	String	必填，待分配的目标账户地址
tokenAmount	String	必填，待分配的ctp10Token数量，大小限制[1, Long.MAX_VALUE]
metadata	String	选填，备注

7.8.1.13 Ctp10TokenChangeOwnerOperation

Ctp10TokenChangeOwnerOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是0.02 BU。

成员变量	类型	描述
sourceAddress	String	选填，合约token的拥有者账户地址
contractAddress	String	必填，合约账户地址
tokenOwner	String	必填，待转移的目标账户地址
metadata	String	选填，备注

7.8.1.14 ContractCreateOperation

ContractCreateOperation继承于BaseOperation，feeLimit目前(2018.07.26)固定是10.01 BU。

成员变量	类型	描述
sourceAddress	String	选填，合约token的拥有者账户地址
initBalance	Long	必填，给合约账户的初始化资产，单位MO，1BU = 10^8 MO，大小限制[1,Long.MAX_VALUE]
type	Integer	选填，合约的语种，默认是0
payload	String	必填，对应语种的合约代码
initInput	String	选填，合约代码中init方法的入参
metadata	String	选填，备注

7.8.1.15 ContractInvokeByAssetOperation

ContractInvokeByAssetOperation继承于BaseOperation，feeLimit要根据合约中执行交易来做添加手续费，首先发起交易手续费，目前(2018.07.26)是0.01BU，然后合约中的交易也需要交易发起者添加相应交易的手续费。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
contractAddress	String	必填，合约账户地址
code	String	选填，资产编码，长度限制[0,1024];当为空时，仅触发合约;
issuer	String	选填，资产发行账户地址，当为null时，仅触发合约
assetAmount	Long	选填，资产数量，大小限制[0,Long.MAX_VALUE]，当为0时，仅触发合约
input	String	选填，待触发的合约的main()入参
metadata	String	选填，备注

7.8.1.16 ContractInvokeByBUOperation

ContractInvokeByBUOperation继承于BaseOperation，feeLimit要根据合约中执行交易来添加手续费，首先发起交易手续费，目前(2018.07.26)是0.01BU，然后合约中的交易也需要交易发起者添加相应交易的手续费。

成员变量	类型	描述
sourceAddress	String	选填，操作源账户地址
contractAddress	String	必填，合约账户地址
buAmount	Long	选填，资产发行数量，大小限制[0,Long.MAX_VALUE]，当为0时仅触发合约
input	String	选填，待触发的合约的main()入参
metadata	String	选填，备注

7.8.2 evaluateFee

`evaluateFee` 接口实现交易的费用评估。

调用方法如下所示：

```
TransactionEvaluateFeeResponse evaluateFee (TransactionEvaluateFeeRequest);
```

请求参数如下表所示：

参数	类型 描述	
sourceAddress	String	必填，发起该操作的源账户地址
nonce	Long	必填，待发起的交易序列号，大小限制[1,Long.MAX_VALUE]
operation	*BaseOperation* []	必填，待提交的操作列表，不能为空
signtureNumber	Integer	选填，待签名者的数量，默认是1，大小限制[1,Integer.MAX_VALUE]
ceilLedgerSeq	Long	选填，距离当前区块高度指定差值的区块内执行的限制，当区块超出当时区块高度与所设差值的和后，交易执行失败。必须大于等于0，是0时不限制
metadata	String	选填，备注

响应数据如下表所示：

参数	类型	描述
txs	*TestTx*[]	评估交易集

错误码如下表所示：

异常	错误码	描述
INVALID_SOURCEADDRESS_ERROR	11002	Invalid sourceAddress
INVALID_NONCE_ERROR	11045	Nonce must be between 1 and Long.MAX_VALUE
OPERATIONS_EMPTY_ERROR	11051	Operations cannot be empty
OPERATIONS_ONE_ERROR	11053	One of operations cannot be resolved
INVALID_SIGNATURENUMBER _ERROR	11054	SignagureNumber must be between 1 and Integer.MAX_VALU E
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化变量
String senderAddresss = "buQnnUEBREw2hB6pWHGPzwanX7d28xk6KVcp";
String destAddress = "buQfnVYgXuMo3rvCEpKA6SfRrDpaz8D8A9Ea";
Long buAmount = ToBaseUnit.BU2MO("10.9");
Long gasPrice = 1000L;
Long feeLimit = ToBaseUnit.BU2MO("0.01");
Long nonce = 51L;

// 构建sendBU操作
BUSendOperation buSendOperation = new BUSendOperation();
buSendOperation.setSourceAddress(senderAddresss);
buSendOperation.setDestAddress(destAddress);
buSendOperation.setAmount(buAmount);

// 初始化评估交易请求参数
```

(continues on next page)

(续上页)

```
TransactionEvaluateFeeRequest request = new TransactionEvaluateFeeRequest();
request.addOperation(buSendOperation);
request.setSourceAddress(senderAddresss);
request.setNonce(nonce);
request.setSignatureNumber(1);
request.setMetadata(HexFormat.byteToHex("evaluate fees".getBytes()));

// 调用evaluateFee接口
TransactionEvaluateFeeResponse response = sdk.getTransactionService().
↪evaluateFee(request);
if (response.getErrorCode() == 0) {
TransactionEvaluateFeeResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.8.2.1 TestTx

TestTx的具体信息如下表所示：

成员变量	类型	描述
transactionEnv	*TestTransactionFees*	评估交易费用

7.8.2.2 TestTransactionFees

TestTransactionFees的具体信息如下表所示：

成员变量	类型	描述
transactionFees	*TransactionFees*	交易费用

7.8.2.3 TransactionFees

TransactionFees的具体信息如下表所示：

成员变量	类型	描述
feeLimit	Long	交易要求的最低费用
gasPrice	Long	交易燃料单价

7.8.3 sign

`sign` 接口用于实现交易的签名。

调用方法如下所示：

```
TransactionSignResponse sign(TransactionSignRequest);
```

请求参数如下表所示：

参数	类型	描述
blob	String	必填，待签名的交易Blob
privateKeys	String[]	必填，私钥列表

响应数据如下表所示：

参数	类型	描述
signatures	Signature	签名后的数据列表

错误码如下表所示：

异常	错误码	描述
INVALID_BLOB_ERROR	11056	Invalid blob
PRIVATEKEY_NULL_ERROR	11057	PrivateKeys cannot be empty
PRIVATEKEY_ONE_ERROR	11058	One of privateKeys is invalid
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String issuePrivateKey = "privbyQCRp7DLqKtRFCqKQJr81TurTqG6UKXMMtGAmPG3abcM9XHjWvq";
String []signerPrivateKeyArr = {issuePrivateKey};
String transactionBlob =
↪"0A246275516E6E5545425245773268423670574847507A77616E5837643238786B364B566370102118C0843D20E8073A5
↪";
TransactionSignRequest request = new TransactionSignRequest();
request.setBlob(transactionBlob);
for (int i = 0; i < signerPrivateKeyArr.length; i++) {
request.addPrivateKey(signerPrivateKeyArr[i]);
}
TransactionSignResponse response = sdk.getTransactionService().sign(request);
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

7.8.3.1 Signature

Signature的具体信息如下表所示：

成员变量	类型	描述
signData	Long	签名后数据
publicKey	Long	公钥

7.8.4 submit

`submit` 接口用于实现交易的提交。

调用方法如下所示：

```
TransactionSubmitResponse submit(TransactionSubmitRequest);
```

请求参数如下表所示：

参数	类型	描述
blob	String	必填，交易blob
signature	*Signature* []	必填，签名列表

响应数据如下表所示：

参数	类型	描述
hash	String	交易hash

错误码如下表所示：

异常	错误码	描述
INVALID_BLOB_ERROR	11056	Invalid blob
SIGNATURE_EMPTY_ERROR	11067	The signatures cannot be empty
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String transactionBlob =
↪"0A246275516E6E554542524577326842367057484750 7A77616E5837643238786B364B566370102118C0843D20E8073A5
↪";
Signature signature = new Signature();
signature.setSignData(
↪"D2B5E3045F2C1B7D363D4F58C1858C30ABBBB0F41E4B2E18AF680553CA9C3689078E215C097086E47A4393BCA715C7A5D2
↪");
signature.setPublicKey(
↪"b0011765082a9352e04678ef38d38046dc01306edef676547456c0c23e270aaed7ffe9e31477");
TransactionSubmitRequest request = new TransactionSubmitRequest();
request.setTransactionBlob(transactionBlob);
request.addSignature(signature);

// 调用submit接口
TransactionSubmitResponse response = sdk.getTransactionService().submit(request);
if (0 == response.getErrorCode()) { // 交易提交成功
System.out.println(JSON.toJSONString(response.getResult(), true));
} else{
System.out.println("error: " + response.getErrorDesc());
}
```

7.8.5 getInfo

`getInfo` 接口用于实现根据交易hash查询交易。

调用方法如下所示：

```
TransactionGetInfoResponse getInfo (TransactionGetInfoRequest);
```

请求参数如下表所示：

参数	类型	描述
hash	String	交易hash

响应数据如下表所示：

参数	类型	描述
totalCount	Long	返回的总交易数
transactions	*TransactionHistory* []	交易内容

错误码如下表所示：

异常	错误码	描述
INVALID_HASH_ERROR	11055	Invalid transaction hash
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
String txHash = "1653f54fbba1134f7e35acee49592a7c29384da10f2f629c9a214f6e54747705";
TransactionGetInfoRequest request = new TransactionGetInfoRequest();
request.setHash(txHash);

// 调用getInfo接口
TransactionGetInfoResponse response = sdk.getTransactionService().getInfo(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.8.5.1 TransactionHistory

TransactionHistory的具体信息如下表所示：

成员变量	类型	描述
actualFee	String	交易实际费用
closeTime	Long	交易关闭时间
errorCode	Long	交易错误码
errorDesc	String	交易描述
hash	String	交易hash
ledgerSeq	Long	区块序列号
transaction	*TransactionInfo*	交易内容列表
signatures	*Signature* []	签名列表
txSize	Long	交易大小

7.9 区块服务

区块服务提供区块相关的接口，目前有11个接口：getNumber、checkStatus、getTransactions、getInfo、getLa

7.9.1 getNumber

getNumber 接口用于查询最新的区块高度。

调用方法如下所示：

```
BlockGetNumberResponse getNumber();
```

响应数据如下表所示：

参数	类型	描述
header	BlockHeader	区块头
blockNumber	Long	最新的区块高度，对应底层字段seq

错误码如下表所示：

异常	错误码	描述
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 调用getNumber接口
BlockGetNumberResponse response = sdk.getBlockService().getNumber();
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.2 checkStatus

checkStatus 接口用于检查本地节点区块是否同步完成。

调用方法如下所示：

```
BlockCheckStatusResponse checkStatus();
```

响应数据如下表所示：

参数	类型	描述
isSynchronous	Boolean	区块是否同步

错误码如下表所示：

异常	错误码	描述
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 调用checkStatus
BlockCheckStatusResponse response = sdk.getBlockService().checkStatus();
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.3 getTransactions

getTransactions 接口用于查询指定区块高度下的所有交易。

调用方法如下所示：

```
BlockGetTransactionsResponse getTransactions(BlockGetTransactionsRequest);
```

请求参数如下表所示：

参数	类型	描述
blockNumber	Long	必填，待查询的区块高度，必须大于0

响应数据如下表所示：

参数	类型	描述
totalCount	Long	返回的总交易数
transactions	*TransactionHistory* []	交易内容

错误码如下表所示：

异常	错误码	描述
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
Long blockNumber = 617247L;// 第617247区块
BlockGetTransactionsRequest request = new BlockGetTransactionsRequest();
request.setBlockNumber(blockNumber);

// 调用getTransactions接口
BlockGetTransactionsResponse response = sdk.getBlockService().
↪getTransactions(request);
if(0 == response.getErrorCode()){
System.out.println(JSON.toJSONString(response.getResult(), true));
}else{
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.4 getInfo

getInfo 接口用于获取区块信息。

调用方法如下所示：

```
BlockGetInfoResponse getInfo(BlockGetInfoRequest);
```

请求参数如下表所示：

参数	类型	描述
blockNumber	Long	必填，待查询的区块高度

响应数据如下表所示：

参数	类型	描述
closeTime	Long	区块关闭时间
number	Long	区块高度
txCount	Long	交易总量
version	String	区块版本

错误码如下表所示：

异常	错误码	描述
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
BlockGetInfoRequest request = new BlockGetInfoRequest();
request.setBlockNumber(629743L);

// 调用getInfo接口
BlockGetInfoResponse response = sdk.getBlockService().getInfo(request);
if (response.getErrorCode() == 0) {
BlockGetInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.5 getLatestInfo

getLatestInfo 接口用于获取最新区块信息。

调用方法如下所示：

```
BlockGetLatestInfoResponse getLatestInfo();
```

响应数据如下表所示：

参数	类型	描述
closeTime	Long	区块关闭时间
number	Long	区块高度，对应底层字段seq
txCount	Long	交易总量
version	String	区块版本

错误码如下表所示：

异常	错误码	描述
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 调用getLatestInfo接口
BlockGetLatestInfoResponse response = sdk.getBlockService().getLatestInfo();
if (response.getErrorCode() == 0) {
BlockGetLatestInfoResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.5.1 getValidators

`getValidators` 接口用于获取指定区块中所有验证节点数。

调用方法如下所示：

```
BlockGetValidatorsResponse getValidators(BlockGetValidatorsRequest);
```

请求参数如下表所示：

参数	类型	描述
blockNumber	Long	必填，待查询的区块高度，必须大于0

响应数据如下表所示：

参数	类型	描述
validators	*ValidatorInfo* []	验证节点列表

错误码如下表所示：

异常	错误码	描述
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
BlockGetValidatorsRequest request = new BlockGetValidatorsRequest();
request.setBlockNumber(629743L);

// 调用getValidators接口
BlockGetValidatorsResponse response = sdk.getBlockService().getValidators(request);
if (response.getErrorCode() == 0) {
BlockGetValidatorsResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.5.2 ValidatorInfo

ValidatorInfo的具体信息如下表所示：

成员变量	类型	描述
address	String	共识节点地址
plegeCoinAmount	Long	验证节点押金

7.9.6 getLatestValidators

`getLatestValidators` 接口用于获取最新区块中所有验证节点数。

调用方法如下所示：

```
BlockGetLatestValidatorsResponse getLatestValidators();
```

响应数据如下表所示：

参数	类型	描述
validators	*ValidatorInfo* []	验证节点列表

错误码如下表所示：

异常	错误码	描述
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 调用getLatestValidators接口
BlockGetLatestValidatorsResponse response = sdk.getBlockService().
↪getLatestValidators();
if (response.getErrorCode() == 0) {
BlockGetLatestValidatorsResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.7 getReward

getReward 接口用于获取指定区块中的区块奖励和验证节点奖励。

调用方法如下所示：

```
BlockGetRewardResponse getReward(BlockGetRewardRequest);
```

请求参数如下表所示：

参数	类型	描述
blockNumber	Long	必填，待查询的区块高度，必须大于0

响应数据如下表所示：

参数	类型	描述
blockReward	Long	区块奖励数
validatorsReward	*ValidatorReward* []	验证节点奖励情况

错误码如下表所示：

异常	错误码	描述
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
BlockGetRewardRequest request = new BlockGetRewardRequest();
request.setBlockNumber(629743L);

// 调用getReward接口
BlockGetRewardResponse response = sdk.getBlockService().getReward(request);
if (response.getErrorCode() == 0) {
BlockGetRewardResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.7.1 ValidatorReward

ValidatorReward的具体信息如下表所示：

成员变量	类型	描述
validator	String	验证节点地址
reward	Long	验证节点奖励

7.9.8 getLatestReward

getLatestReward 接口获取最新区块中的区块奖励和验证节点奖励。

调用方法如下所示：

```
BlockGetLatestRewardResponse getLatestReward();
```

响应数据如下表所示：

参数	类型	描述
blockReward	Long	区块奖励数
validatorsReward	*ValidatorReward* []	验证节点奖励情况

错误码如下表所示：

异常	错误码	描述
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 调用getLatestReward接口
BlockGetLatestRewardResponse response = sdk.getBlockService().getLatestReward();
if (response.getErrorCode() == 0) {
BlockGetLatestRewardResult result = response.getResult();
System.out.println(JSON.toJSONString(result, true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.9 getFees

getFees 接口获取指定区块中的账户最低资产限制和燃料单价。

调用方法如下所示：

```
BlockGetFeesResponse getFees(BlockGetFeesRequest);
```

请求参数如下表所示：

参数	类型	描述
blockNumber	Long	必填，待查询的区块高度，必须大于0

响应数据如下表所示：

参数	类型	描述
fees	*Fees*	费用

错误码如下表所示：

异常	错误码	描述
INVALID_BLOCKNUMBER_ERROR	11060	BlockNumber must be greater than 0
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 初始化请求参数
BlockGetFeesRequest request = new BlockGetFeesRequest();
request.setBlockNumber(629743L);

// 调用getFees接口
BlockGetFeesResponse response = sdk.getBlockService().getFees(request);
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.9.9.1 Fees

Fees的具体信息如下表所示：

成员变量	类型	描述
baseReserve	Long	账户最低资产限制
gasPrice	Long	交易燃料单价，单位MO，1 BU = 10^8 MO

7.9.10 getLatestFees

`getLatestFees` 接口用于获取最新区块中的账户最低资产限制和燃料单价。

调用方法如下所示：

```
BlockGetLatestFeesResponse getLatestFees();
```

响应数据如下表所示：

参数	类型	描述
fees	*Fees*	费用

错误码如下表所示：

异常	错误码	描述
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
SYSTEM_ERROR	20000	System error

具体示例如下所示：

```
// 调用getLatestFees接口
BlockGetLatestFeesResponse response = sdk.getBlockService().getLatestFees();
if (response.getErrorCode() == 0) {
System.out.println(JSON.toJSONString(response.getResult(), true));
} else {
System.out.println("error: " + response.getErrorDesc());
}
```

7.10 错误码

下表对可能出现的错误信息进行了详细描述。

异常	错误码	描述
ACCOUNT_CREATE_ERROR	11001	Failed to create the account
INVALID_SOURCEADDRESS _ERROR	11002	Invalid sourceAddress
INVALID_DESTADDRESS_E RROR	11003	Invalid destAddress
INVALID_INITBALANCE_E RROR	11004	InitBalance must be between 1 and Long.MAX_VAL
SOURCEADDRESS_EQUAL_D ESTADDRESS_ERROR	11005	SourceAddress cannot be equal to destAddress
INVALID_ADDRESS_ERROR	11006	Invalid address
CONNECTNETWORK_ERROR	11007	Failed to connect to the network
INVALID_ISSUE_AMOUNT _ERROR	11008	Amount of the token to be issued must be between 1
NO_ASSET_ERROR	11009	The account does not have the asset
NO_METADATA_ERROR	11010	The account does not have the metadata
INVALID_DATAKEY_ERROR	11011	The length of key must be between 1 and 1024
INVALID_DATAVALUE_ERR OR	11012	The length of value must be between 0 and 256000
INVALID_DATAVERSION_E RROR	11013	The version must be equal to or greater than 0
INVALID_MASTERWEIGHT _ERROR	11015	MasterWeight must be between 0 and (Integer.MAX_
INVALID_SIGNER_ADDRES S_ERROR	11016	Invalid signer address
INVALID_SIGNER_WEIGHT _ERROR	11017	Signer weight must be between 0 and (Integer.MAX_
INVALID_TX_THRESHOLD _ERROR	11018	TxThreshold must be between 0 and Long.MAX_VA
INVALID_OPERATION_TYP E_ERROR	11019	Operation type must be between 1 and 100
INVALID_TYPE_THRESHOL D_ERROR	11020	TypeThreshold must be between 0 and Long.MAX_V
INVALID_ASSET_CODE_ER ROR	11023	The length of key must be between 1 and 64
INVALID_ASSET_AMOUNT _ERROR	11024	AssetAmount must be between 0 and Long.MAX_VA
INVALID_BU_AMOUNT_ERR OR	11026	BuAmount must be between 0 and Long.MAX_VAL
INVALID_ISSUER_ADDRES S_ERROR	11027	Invalid issuer address
NO_SUCH_TOKEN_ERROR	11030	No such token
INVALID_TOKEN_NAME_ER ROR	11031	The length of token name must be between 1 and 102
INVALID_TOKEN_SIMBOL _ERROR	11032	The length of symbol must be between 1 and 1024
INVALID_TOKEN_DECIMAL S_ERROR	11033	Decimals must be between 0 and 8
INVALID_TOKEN_TOTALSU PPLY_ERROR	11034	TotalSupply must be between 1 and Long.MAX_VAL
INVALID_TOKENOWNER_ER RPR	11035	Invalid token owner
INVALID_CONTRACTADDRE SS_ERROR	11037	Invalid contract address
CONTRACTADDRESS_NOT_C ONTRACTACCOUNT_ERROR	11038	contractAddress is not a contract account
INVALID_TOKEN_AMOUNT _ERROR	11039	TokenAmount must be between 1 and Long.MAX_V
SOURCEADDRESS_EQUAL_C ONTRACTADDRESS_ERROR	11040	SourceAddress cannot be equal to contractAddress
INVALID_FROMADDRESS_E RROR	11041	Invalid fromAddress
FROMADDRESS_EQUAL_DES TADDRESS_ERROR	11042	FromAddress cannot be equal to destAddress
INVALID_SPENDER_ERROR	11043	Invalid spender

表 2 – 续上页

异常	错误码	描述
PAYLOAD_EMPTY_ERROR	11044	Payload cannot be empty
INVALID_LOG_TOPIC_ERR OR	11045	The length of one log topic must be between 1 and 1
INVALID_LOG_DATA_ERRO R	11046	The length of one piece of log data must be between
INVALID_CONTRACT_TYPE _ERROR	11047	Invalid contract type
INVALID_NONCE_ERROR	11048	Nonce must be between 1 and Long.MAX_VALUE
INVALID_GASPRICE_ERRO R	11049	GasPrice must be between 1000 and Long.MAX_VA
INVALID_FEELIMIT_ERRO R	11050	FeeLimit must be between 1 and Long.MAX_VALU
OPERATIONS_EMPTY_ERRO R	11051	Operations cannot be empty
INVALID_CEILLEDGERSEQ _ERROR	11052	CeilLedgerSeq must be equal to or greater than 0
OPERATIONS_ONE_ERROR	11053	One of the operations cannot be resolved
INVALID_SIGNATURENUMB ER_ERROR	11054	SignagureNumber must be between 1 and Integer.MA
INVALID_HASH_ERROR	11055	Invalid transaction hash
INVALID_BLOB_ERROR	11056	Invalid blob
PRIVATEKEY_NULL_ERROR	11057	PrivateKeys cannot be empty
PRIVATEKEY_ONE_ERROR	11058	One of privateKeys is invalid
PUBLICKEY_NULL_ERROR	11061	PublicKey cannot be empty
URL_EMPTY_ERROR	11062	Url cannot be empty
CONTRACTADDRESS_CODE _BOTH_NULL_ERROR	11063	ContractAddress and code cannot be empty at the sa
INVALID_OPTTYPE_ERROR	11064	OptType must be between 0 and 2
GET_ALLOWANCE_ERROR	11065	Failed to get allowance
GET_TOKEN_INFO_ERROR	11066	Failed to get the token info
SIGNATURE_EMPTY_ERROR	11067	The signatures cannot be empty
REQUEST_NULL_ERROR	12001	Request parameter cannot be null
CONNECTN_BLOCKCHAIN_E RROR	19999	Failed to connect to the blockchain
SYSTEM_ERROR	20000	System error

CHAPTER 8

BUMO节点安装运维指南

8.1 概要

本文档将指导您如何在Linux环境和MacOS环境下安装并配置BUMO节点。

8.2 系统要求

在安装BUMO节点之前需要确保您的系统满足以下条件。

硬件要求

硬件要求至少满足以下配置:

- **推荐配置**：CPU 8 核，内存 32G，带宽 20M， SSD 磁盘500G
- **最低配置**：CPU 4 核，内存 16G，带宽 10M， SSD 磁盘500G

软件要求

系统软件可选择Ubuntu、Centos或者MacOS。

- Ubuntu 14.04
- Centos 7
- Mac OS X 10.11.4

8.3 在linux下安装BUMO节点

在本文档中，Linux中的BUMO节点安装以 Ubuntu 14.04 为例。 在Linux系统中支持两种安装方式：编译安装和 安装包安装。

注解:

- 本安装文档中使用root账号下的根目录作为安装目录。用户可选择自己的安装目录。
- 在安装BUMO节点之前需要确保设备的网络连接正常。

8.3.1 编译安装

编译安装是指先将BUMO节点的源代码编译成计算机能识别的机器码然后再进行安装。编译安装由三部分构成：安装依赖，编译*BUMO*源代码，安装*BUMO*节点。

8.3.1.1 安装依赖

在编译BUMO节点的源代码之前需要安装系统所需的依赖。安装依赖需要完成以下步骤:

1. 输入以下命令安装 `automake`。

```
sudo apt-get install automake
```

2. 输入以下命令安装 `autoconf`。

```
sudo apt-get install autoconf
```

3. 输入以下命令安装 `libtool`。

```
sudo apt-get install libtool
```

4. 输入以下命令安装 `g++`。

```
sudo apt-get install g++
```

5. 输入以下命令安装 `libssl-dev`。

```
sudo apt-get install libssl-dev
```

6. 输入以下命令安装 `cmake`。

```
sudo apt-get install cmake
```

7. 输入以下命令安装 `libbz2-dev`。

```
sudo apt-get install libbz2-dev
```

8. 输入以下命令安装 `python`。

```
sudo apt-get install python
```

9. 输入以下命令安装 `unzip`。

```
sudo apt-get install unzip
```

8.3.1.2 编译BUMO源代码

在成功安装依赖后才能编译BUMO的源代码。编译BUMO节点的源代码需要完成以下步骤:

1. 在根目录下输入以下命令下载BUMO的源代码文件。如果没有安装 `git`，可以通过 `sudo apt-get install git` 命令来安装 `git`。

```
git clone https://github.com/bumoproject/bumo.git
```

注解: 在BUMO的源代码下载过程中将自动创建bumo/目录，源代码文件将存放到该目录下。

2. 输入以下命令进入到源代码的文件目录。

```
cd /bumo/build/
```

3. 输入以下命令下载依赖并初始化开发环境。

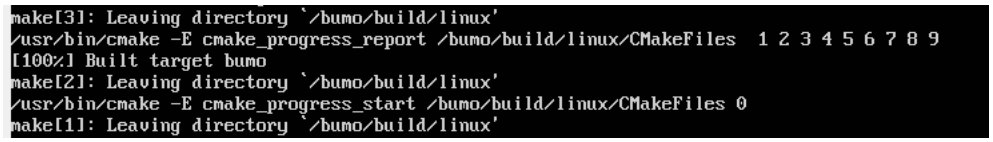
```
./install-build-deps-linux.sh
```

4. 输入以下命令回到bumo/目录下。

```
cd ../
```

5. 输入以下命令完成BUMO源代码的编译。出现下图所示信息则表示编译成功。

```
make
```



注解: 编译完成后生成的可执行文件 **bumo** 和 **bumod** 存放在/bumo/bin目录下。

8.3.1.3 安装BUMO节点

在编译完成后才能安装BUMO节点。安装BUMO节点需要完成以下步骤:

1. 输入以下命令进入到安装目录。

```
cd /bumo/
```

2. 输入以下命令完成安装。出现下图所示信息则表示安装成功。

```
make install
```

```
sudo mkdir -p /usr/local/buchain/coredump;
make[1]: Leaving directory `/bumo/build/linux'
```

注解:

- 默认情况下服务安装在/usr/local/buchain/目录下。
- 安装完成后无需其他配置即可通过 `service bumo start` 命令来启动bumo服务。
- 安装完BUMO节点后在buchain/目录下有如下目录结构：

目录	说明
bin	存放可执行文件（编译后的bumo可执行程序）
jslib	存放第三方js库
config	配置文件目录包含：bumo.json
data	数据库目录，存放账本数据
scripts	启停脚本目录
log	运行日志存储目录（该目录在运行BUMO节点后才会出现）

8.3.2 安装包安装

安装包安装是指以安装包的方式来安装BUMO节点。通过安装包安装BUMO节点由五部分构成：获取安装包并解压 、注册服务 、修改服务启动路径 、设置开机启动 、选择运行环境的配置文件。

8.3.2.1 获取安装包并解压

获取BUMO的安装包并解压安装文件需要完成以下步骤。

1. 输入以下命令下载BUMO的安装包。

```
wget https://github.com/bumoproject/bumo/releases/download/1.0.0.7/buchain-1.0.0.7-
↪linux-x64.tar.gz
```

注解:

- 如果您没有安装wget，可以用 `apt-get install wget` 命令来装 `wget`。
- 您可以在 https://github.com/bumoproject/bumo/releases 链接上找到需要的版本，然后右键单击该版本复制下载链接。
- 在本示例中文件下载到根目录下。

2. 输入以下命令把安装包拷贝到/usr/local/目录下。

```
cp buchain-1.0.0.7-linux-x64.tar.gz /usr/local/
```

注解: 以上拷贝操作是在文件下载目录下完成的。您需根据具体的下载目录来拷贝文件。

3. 输入以下命令进入到 /usr/local/目录下。

```
cd /usr/local/
```

4. 输入以下命令解压文件。

```
tar -zxvf buchain-1.0.0.7-linux-x64.tar.gz
```

注解： 解压完成后得到buchain/目录。

8.3.2.2 注册服务

文件解压后需要注册bumo和bumod的服务。注册服务需要完成以下步骤:

1. 输入以下命令注册bumo的服务。

```
ln -s /usr/local/buchain/scripts/bumo /etc/init.d/bumo
```

2. 输入以下命令注册bumod的服务。

```
ln -s /usr/local/buchain/scripts/bumod /etc/init.d/bumod
```

8.3.2.3 修改服务启动路径

修改bumo和bumod的启动路径需要完成以下步骤:

1. 在local/目录下输入以下命令打开bumo文件。

```
vim buchain/scripts/bumo
```

2. 找到install_dir并更改bumo的安装目录。

```
install_dir=/usr/local/buchain
```

```
#!/bin/bash
install_dir=/usr/local/buchain
script_dir=/usr/local/buchain/scripts
```

注解： 默认情况下install_dir的目录在/usr/local/buchain下；您可以根据bumo的具体安装目录来修改。

3. 单击 `Esc` 键退出编辑。
4. 输入 `:wq` 保存文件。
5. 在local/目录下输入以下命令打开bumod文件。

```
vim /buchain/scripts/bumod
```

6. 找到install_dir并更改bumod的安装目录。

```
install_dir=/usr/local/buchain
```

注解： 默认情况下install_dir的目录在/usr/local/buchain下；您可以根据bumod的具体安装目录来修改。

7. 单击 Esc 键退出编辑。
8. 输入 :wq 保存文件。

8.3.2.4 设置开机启动

设置开机启动包括设置启动级别，添加启动命令和修改文件权限。设置开机启动需要完成以下步骤:

1. 输入以下命令设置1级。

```
ln -s -f /etc/init.d/bumod /etc/rc1.d/S99bumod
```

2. 输入以下命令设置2级。

```
ln -s -f /etc/init.d/bumod /etc/rc2.d/S99bumod
```

3. 输入以下命令设置3级。

```
ln -s -f /etc/init.d/bumod /etc/rc3.d/S99bumod
```

4. 输入以下命令设置4级。

```
ln -s -f /etc/init.d/bumod /etc/rc4.d/S99bumod
```

5. 输入以下命令设置5级。

```
ln -s -f /etc/init.d/bumod /etc/rc5.d/S99bumod
```

6. 输入以下命令打开rc.local文件。

```
vim /etc/rc.local
```

7. 在rc.local文件末尾追加以下命令。

```
/etc/init.d/bumod start
```

```
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
/etc/init.d/bumod start
exit 0
~
~
```

8. 单击 Esc 键退出编辑。
9. 输入 :wq 命令保存文件。

10. 执行以下命令设置rc.local文件的权限。

```
chmod +x /etc/rc.local
```

注解： 至此就完成了BUMO节点的安装。在启动bumo服务之前还需要选择运行环境的配置文件。

8.3.2.5 选择运行环境的配置文件

在安装完BUMO节点后需要选择运行环境的配置文件才能启动bumo服务。选择运行环境的配置文件需要完成以下步骤：

1. 输入以下命令进入到配置文件目录。

```
cd /usr/local/buchain/config/
```

注解：

在该目录下提供了以下运行环境的配置文件。

- bumo-mainnet.json：该文件是主网环境的配置文件应用在生产环境中
- bumo-testnet.json：该文件是测试网环境的配置文件
- bumo-single.json：该文件是单节点调试环境的配置文件

2. 输入以下命令重命名运行环境的配置文件。

```
mv bumo-testnet.json bumo.json
```

注解：

- 本示例中选取了测试网环境作为运行环境。您也可以根据自己的需要选取其他文件作为运行环境。
- 重命名文件完成后可以通过 `service start bumo` 来启动bumo服务。
- 安装完BUMO节点后可以在buchain/目录下查看安装文件的目录结构。

8.4 在MacOS下安装BUMO节点

在MacOS下安装BUMO节点包括编译安装和安装包安装。

8.4.1 MacOS中的编译安装

编译安装是指先将BUMO节点的源代码编译成计算机能识别的机器码然后再进行安装。编译安装由三部分构成：安装*Xcode*、安装*Command Line Tools*、安装*Homebrew*、*MacOS*中安装依赖、*MacOS*中编译*BUMO*源代码、*MacOS*中安装*BUMO*节点。

8.4.1.1 安装Xcode

安装Xcode需要完成以下步骤:

1. 单击 登录苹果软件下载官网 。
2. 输入 Apple ID 和 Password。
3. 单击 Sign in，进入下载页面。
4. 单击 Xcode 9.4.1，开始下载 Xcode 。
5. 解压 Xcode_9.4.1.xip。
6. 双击解压出来的文件 Xcode 完成安装。

注解: 在选择 Xcode 的版本时，需要根据自己的MacOS系统版本来确定。

8.4.1.2 安装Command Line Tools

安装 Command Line Tools 需要完成以下步骤:

1. 单击 登录苹果软件下载官网 。
2. 输入 Apple ID 和 Password。
3. 单击 Sign in，进入下载页面。
4. 单击 Command Line Tools(macOS 10.14)for Xcode 10 Beta 6，开始下载 Command Line Tools。
5. 双击 Command_Line_Tools_macOS_10.14_for_Xcode_10Beta_6.dmg。
6. 单击 Command Line Tools 图标。
7. 单击 **继续**
8. 选择语言，然后单击 **继续**。
9. 单击 **同意**。
10. 单击 **安装**。
11. 输入密码并单击 **安装软件**。

注解: 在选择 Command Line Tools 的版本时，需要根据自己的MacOS系统版本来确定。

8.4.1.3 安装Homebrew

安装Homebrew需完成以下步骤:

1. 打开mac的终端。
2. 在终端中输入以下代码:

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/
↪master/install)"
```

3. 按下 Enter 键，进行安装。

8.4.1.4 MacOS中安装依赖

1. 输入以下命令设置 Homebrew 无自动更新。

```
export HOMEBREW_NO_AUTO_UPDATE=true
```

2. 输入以下命令安装 autoconf。

```
brew install autoconf
```

3. 输入以下命令安装 automake。

```
brew install automake
```

4. 输入以下命令安装 libtool。

```
brew install libtool
```

5. 输入以下命令安装 cmake。

```
brew install cmake
```

6. 输入以下命令安装 python。

```
brew install python
```

7. 输入以下命令安装 m4。

```
brew install m4
```

8. 输入以下命令安装 wget。

```
brew install wget
```

8.4.1.5 MacOS中编译BUMO源代码

1. 在根目录下输入以下命令下载BUMO的源代码文件。如果没有安装 git，可以通过 sudo apt-get install git 命令来安装 git。

```
sudo git clone https://github.com/bumoproject/bumo.git
```

注解： 在BUMO的源代码下载过程中将自动创建bumo/目录，源代码文件将存放到该目录下。

2. 输入以下命令进入到源代码的文件目录。

```
cd /bumo/build/
```

3. 输入以下命令下载依赖并初始化开发环境。

```
sudo ./install-build-deps-mac.sh
```

4. 输入以下命令回到bumo/目录下。

```
cd ../
```

5. 输入以下命令完成BUMO源代码的编译。

```
sudo make
```

注解： 编译完成后生成的可执行文件 **bumo** 和 **bumod** 存放在/bumo/bin目录下。

8.4.1.6 MacOS中安装BUMO节点

在编译完成后才能安装BUMO节点。安装BUMO节点需要完成以下步骤:

1. 输入以下命令进入到安装目录。

```
cd /bumo/
```

2. 输入以下命令完成安装。

```
sudo make install
```

注解:

- 默认情况下服务安装在/usr/local/buchain/目录下。
- 安装完BUMO节点后在buchain/目录下有如下目录结构:

目录	说明
bin	存放可执行文件（编译后的bumo可执行程序）
config	配置文件目录包含：bumo.json
data	数据库目录，存放账本数据
jslib	存放第三方js库
log	运行日志存储目录（该目录在运行BUMO节点后才会出现）

8.4.2 MacOS中安装包安装

安装包安装是指以安装包的方式来安装BUMO节点。以安装包的方式来安装BUMO节点包括两个步骤：*MacOS*中获取安装包并解压 、*MacOS*中选择运行环境的配置文件 。

8.4.2.1 MacOS中获取安装包并解压

1. 从以下地址下载需要的安装包。

```
sudo wget https://github.com/bumoproject/bumo/releases/download/1.0.0.7/buchain-1.0.0.
↪7-macOS-x64.tar.gz
```

注解：

- 如果您没有安装wget，可以用 `apt-get install wget` 命令来装 `wget`。
- 您可以在 https://github.com/bumoproject/bumo/releases 链接上找到需要的版本，然后右键单击该版本复制下载链接。
- 在本示例中文件下载到根目录下。

2. 输入以下命令把安装包拷贝到/usr/local/目录下。

```
sudo cp buchain-1.0.0.7-linux-x64.tar.gz /usr/local/
```

注解： 以上拷贝操作是在文件下载目录下完成的。您需根据具体的下载目录来拷贝文件。

3. 输入以下命令进入到 /usr/local/目录下。

```
cd /usr/local/
```

4. 输入以下命令解压文件。

```
sudo tar -zxvf buchain-1.0.0.7-linux-x64.tar.gz
```

注解： 解压完成后得到buchain/目录。

目录	说明
bin	存放可执行文件（编译后的bumo可执行程序）
config	配置文件目录包含：bumo.json
data	数据库目录，存放账本数据
jslib	存放第三方js库
log	运行日志存储目录（该目录在运行BUMO节点后才会出现）

8.4.2.2 MacOS中选择运行环境的配置文件

在安装完BUMO节点后需要选择运行环境的配置文件才能启动bumo服务。选择运行环境的配置文件需要完成以下步骤:

1. 输入以下命令进入到配置文件目录。

```
cd /usr/local/buchain/config/
```

注解：

在该目录下提供了以下运行环境的配置文件。

- bumo-mainnet.json：该文件是主网环境的配置文件应用在生产环境中

- bumo-testnet.json：该文件是测试网环境的配置文件
- bumo-single.json：该文件是单节点调试环境的配置文件

2. 输入以下命令重命名运行环境的配置文件。

```
mv bumo-testnet.json bumo.json
```

注解:

- 本示例中选取了测试网环境作为运行环境。您也可以根据自己的需要选取其他文件作为运行环境。
- 重命名文件完成后进入到 /usr/local/buchain/bin 目录下，通过 `./bumo` 命令来启动bumo服务。
- 安装完BUMO节点后可以在buchain/目录下查看安装文件的目录结构。

8.5 配置

配置分为 通用配置 和 多节点配置示例 。

8.5.1 通用配置

普通配置包括了存储数据、节点间通信、WEB API、WebSocket API、区块、创世区块（genesis）以及日志的配置。通用配置在/usr/local/buchain/config目录下的bumo.json文件中进行配置。

存储数据

```
"db":{
"account_path": "data/account.db", //存储账号数据
"ledger_path": "data/ledger.db", //存储区块数据
"keyvalue_path": "data/keyvalue.db" //存储共识数据
}
```

节点间网络通信

```
"p2p":
{
"network_id":30000,//网络 ID
//共识网络
"consensus_network":
{
"heartbeat_interval":60, //心跳周期，秒
"listen_port":36001,//已监听的端口
"target_peer_connection":50, //最大主动连接节点数
"known_peers":
[
"127.0.0.1:36001"//连接其他节点
]
}
}
```

WEB API 配置

```
"webserver":{
"listen_addresses":"0.0.0.0:16002"
}
```

WebSocket API 配置

```
"wsserver":
{
"listen_address":"0.0.0.0:36003"
}
```

区块配置

```
"ledger":
{
"validation_address":"buQmtDED9nFcCfRkwAF4TVhg6SL1FupDNhZY",//验证节点地址，同步节点或者钱
包不需要配置
"validation_private_key":
↪"e174929ecec818c0861aeb168ebb800f6317dae1d439ec85ac0ce4ccdb88487487c3b74a316ee777a3a7a77e5b12efd724
↪", //验证节点私钥，同步节点或者钱包不需要配置
"max_trans_per_ledger":1000, //单个区块最大交易个数
"tx_pool": //交易池配置
{
"queue_limit":10240, //交易池总量限制
"queue_per_account_txs_limit":64 //单个账号的交易缓冲最大值
}
}
```

注解： validation_address 和 validation_private_key 可以通过 bumo 程序命令行工具获得，请妥善保存该账号信息，一旦丢失将无法找回。

```
[root@bumo ~]# cd /usr/local/buchain/bin
[root@bumo bin]#./bumo --create-account

{
"address" : "buQmtDED9nFcCfRkwAF4TVhg6SL1FupDNhZY", //地址
"private_key" : "privbsZozNs3q9aixZWEUzL9ft8AYph5DixN1sQccYvLs2zPsPhPK1Pt", //私钥
"private_key_aes" :
↪"e174929ecec818c0861aeb168ebb800f6317dae1d439ec85ac0ce4ccdb88487487c3b74a316ee777a3a7a77e5b12efd724
↪", //AES 加密的私钥
"public_key" :
↪"b00108d329d5ff69a70177a60bf1b68972576b35a22d99d0b9a61541ab568521db5ee817fea6", //公
钥
"public_key_raw" : "08d329d5ff69a70177a60bf1b68972576b35a22d99d0b9a61541ab568521db5e",
↪ //原始公钥
"sign_type" : "ed25519" //ed25519 加密方式
}
```

创世区块

```
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3", //创世区块地址
"slogan" : "a new era of value", //存储在创世区块中的标语
"fees":
```

(continues on next page)

(续上页)

```
{
"base_reserve": 10000000, //账号最低预留费
"gas_price": 1000 //字节费
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY"] //验证节点区块列表
}
```

注解： 同一个区块链上的 genesis 配置，必须保持一致。account 可以通过 bumo 程序命令行工具 ./bumo --create-account 获取，请妥善保存该账号信息，一旦丢失将无法找回。

日志配置

```
"logger":
{
"path":"log/buchain.log", // 日志目录
"dest":"FILE|STDOUT|STDERR", //输出文件分类
"level":"TRACE|INFO|WARNING|ERROR|FATAL",//日志级别
"time_capacity":1, //时间容量，天
"size_capacity":10, //大小容量，兆
"expire_days":10 //清理日志周期，天
}
```

8.5.2 多节点配置示例

本章节以两个验证节点和一个同步节点为例，介绍多节点在同一条区块链的配置，其中需要修改p2p、ledger和genesis 这三个模块。

p2p模块配置

p2p 的 known_peers 必须为其他已知节点的 IP 和端口，用于节点之间相互连接。

```
验证节点一：
"p2p":
{
"network_id":30000,
"consensus_network":
{
"heartbeat_interval":60,
"listen_port":36001,
"target_peer_connection":50,
"known_peers":
[
"192.168.1.102:36001", //节点二的 IP 和端口
"192.168.1.103:36001" //节点三的 IP 和端口
]
}
}

验证节点二：
"p2p":
{
"network_id":30000,
"consensus_network":
```

(continues on next page)

(续上页)

```
{
"heartbeat_interval":60,
"listen_port":36001,
"target_peer_connection":50,
"known_peers":
[
"192.168.1.101:36001", //节点一的 IP 和端口
"192.168.1.103:36001" //节点三的 IP 和端口
]
}
}

同步节点三:
"p2p":
{
"network_id":30000,
"consensus_network":
{
"heartbeat_interval":60,
"listen_port":36001,
"target_peer_connection":50,
"known_peers":
[
"192.168.1.101:36001", //节点一的 IP 和端口
"192.168.1.102:36001" //节点二的 IP 和端口
]
}
}
```

leger模块配置

验证节点的 ledger 的 `validation_address` 和 `validation_private_key` 必须要匹配。并且需要把所有验证节点的 `validation_address` 填写到 `genesis.validators` 里。

```
验证节点一:
"ledger":
{
"validation_address":"buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",//验证节点一的地址，同步节点或者钱包不需要配置
"validation_private_key":
↪"66932f19d5be465ea9e7cfcb3ea7326d81953b9f99bc39ddb437b5367937f234b866695e1aae9be4bae27317c9987f80be
↪", //验证节点二的私钥，同步节点或者钱包不需要配置
"max_trans_per_ledger":1000,
"tx_pool":
{
"queue_limit":10240,
"queue_per_account_txs_limit":64
}
}

验证节点二:
"ledger":
{
"validation_address":"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm",//验证节点二的地址，同步节点或者钱包不需要配置
"validation_private_key":
↪"1cb0151ec2b23cb97bf94d86ee1100582f9f5fbfdfe40a69edae2d2b8711395c40c1da859ac0bc93240a8a70c4a06779ec
↪", //验证节点二的私钥，同步节点或者钱包不需要配置
```

(continues on next page)

(续上页)

```
"max_trans_per_ledger":1000,
"tx_pool":
{
"queue_limit":10240,
"queue_per_account_txs_limit":64
}
}

同步节点三：
"ledger":
{
"max_trans_per_ledger":1000,
"tx_pool":
{
"queue_limit":10240,
"queue_per_account_txs_limit":64
}
}
```

genesis模块配置

同一个区块链上的 genesis 配置，必须保持一致。

```
验证节点一：
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
"slogan" : "a new era of value",
"fees":
{
"base_reserve": 10000000,
"gas_price": 1000
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",
↪"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm"] //需要配置所有的验证节点地址，如果有两个验证节点，则
配置两个地址。
}

验证节点二：
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
"slogan" : "a new era of value",
"fees":
{
"base_reserve": 10000000,
"gas_price": 1000
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",
↪"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm"] //需要配置所有的验证节点地址，如果有两个验证节点，则
配置两个地址。
}

同步节点三：
"genesis":
{
"account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
```

(continues on next page)

(续上页)

```
"slogan" : "a new era of value",
"fees":
{
"base_reserve": 10000000,
"gas_price": 1000
},
"validators": ["buQBwe7LZYCYHfxiEGb1RE9XC9kN2qrGXWCY",
↪"buQqkp5SDcsxpwWXQ2QFQbvHKnZ199HY3dHm"] //需要配置所有的验证节点地址，如果有两个验证节点，则
配置两个地址。
}
```

注解:

- 运行前请确保每个节点的初始数据一致，否则无法达成共识产生区块。
- `account` 、`validation_address` 可以通过 bumo 程序命令行工具 `./bumo --create-account` 获取，请妥善保存该账号信息，一旦丢失将无法找回。

8.6 运维服务

在运维服务中对BUMO服务的启动、关闭、状态查询、系统详情查询、清空数据库、创建硬分叉、更改运行环境进行了详细说明。

启动BUMO服务

输入以下命令启动bumo服务。

```
service bumo start
```

注解: 在MacOS中启动bumo服务需要进入到/usr/local/buchain/bin目录下，然后通过 `./bumo` 命令在启动bumo服务。

关闭BUMO服务

输入以下命令关闭bumo服务。

```
service bumo stop
```

注解: 在MacOS中关闭bumo服务可以通过 `control+c` 键来完成。

查询BUMO服务状态

输入以下命令查询bumo服务。

```
service bumo status
```

注解: 在MacOS中没有service服务。

查询系统详细状态

输入以下命令查询系统详细状态:

```
curl 127.0.0.1:19333/getModulesStatus
```

得到如下结果:

```
{
 "glue_manager":{
     "cache_topic_size":0,
     "ledger_upgrade":{
         "current_states":null,
         "local_state":null
     },
     "system":{
         "current_time":"2017-07-20 10:32:22", //当前系统时间
         "process_uptime":"2017-07-20 09:35:06", //bumo启动时间
         "uptime":"2017-05-14 23:51:04"
     },
     "time":"0 ms",
     "transaction_size":0
 },
 "keyvalue_db":Object{...},
 "ledger_db":Object{...},
 "ledger_manager":{
     "account_count":2316,  //账户数
     "hash_type":"sha256",
     "ledger_sequence":12187,
     "time":"0 ms",
     "tx_count":1185   //交易数
 },
 "peer_manager":Object{...},
 "web server":Object{...},
```

注解: 在MacOS中没有service服务。

清空数据库

在清空数据之前需要停止BUMO服务。清空数据库需要完成以下步骤:

1. 输入以下命令进入bumo的服务目录。

```
cd /usr/local/buchain/bin
```

2. 输入以下命令清空数据库。

```
./bumo --dropdb
```

注解: 数据库成功清空后能看到如下所示的信息。

```
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(153):Initialize db succes
sful
[2018-07-18 18:02:08.440 - INF] <7F6CC18C18C0> main.cpp(156):Drop db successfully
```

创建硬分叉

创建硬分叉需要完成以下步骤:

1. 在/usr/local目录下输入以下命令创建硬分叉。

```
buchain/bin/bumo --create-hardfork
```

2. 在提示界面上输入 y 然后单击 Enter 键。创建成功后将出现以下界面。

```
[2018-07-19 11:45:40.464 - INF] <7F2A786AA8C0> ledger_manager.cpp(324):Are you su
re to create hardfork ledger? Press y to continue.
y
[2018-07-19 11:47:05.599 - INF] <7F2A786AA8C0> ledger_manager.cpp(341):Max closed ledger seq=290524
[2018-07-19 11:47:05.599 - INF] <7F2A786AA8C0> ledger_frm.cpp(551):total reward(800000000) = total fee(0) + block reward(800000000) in ledger(290525)
[2018-07-19 11:47:05.605 - INF] <7F2A786AA8C0> ledger_manager.cpp(438):Create hard fork ledger successful, seq(290525), consensus value hash(4b9ad78065c
65aaf1280edf6129ab2da93c99c42f2bcd380b5966750ccd5d80d)
```

注解:

- 执行完上面的命令后，新的区块链网络只有一个验证节点即本节点。
- 执行完创建硬分叉命令后将获取如下Hash值:

```
4b9ad78065c65aaf1280edf6129ab2da93c99c42f2bcd380b5966750ccd5d80d
```

3. 输入以下命令清除共识状态数据。清除共识状态数据时需要确保bumo服务没有运行，否则无法清除。

```
buchain/bin/bumo --clear-consensus-status
```

4. 把Hash值配置到本节点或同步节点/usr/local/buchain/config目录下的bumo.json文件中。

```
"ledger": {
"genesis_account": "buQs9npaCq9mNFZG18qu88ZcmXYqd6bqpTU3",
"max_trans_per_ledger": 1000,
"hardfork_points" :
[
"4b9ad78065c65aaf1280edf6129ab2da93c99c42f2bcd380b5966750ccd5d80d
"
]
},
```

5. 启动节点服务，让配置生效。

更改运行环境

在更改运行环境前，需要确保BUMO服务已经关闭。如果您想更改BUMO节点的运行环境，可按照以下步骤进行修改。

1. 输入以下命令进入到配置文件目录。

```
cd /usr/local/buchain/config/
```

注解:

在该目录下提供了以下运行环境的配置文件。

- bumo-mainnet.json：该文件是主网环境的配置文件，应用在生成环境中
- bumo-testnet.json：该文件是测试网环境的配置文件
- bumo-single.json：该文件是单节点调试环境的配置文件

2. 把当前运行环境的配置文件（bumo.json）更改为其他名称，例如：

```
mv bumo.json bumoprevious.json
```

3. 把要运行的环境配置文件更改为bumo.json，例如：

```
mv bumo-mainnet.json bumo.json
```

注解：

- 本示例中把主网环境设置成了运行环境。
- 更改运行环境后需要清空数据库才能重启bumo服务。

8.7 卸载BUMO节点

卸载BUMO节点分为两类，一类是针对编译安装的卸载，另一类是针对安装包安装的卸载。

8.7.1 针对编译安装的卸载

在安装完BUMO节点之后可以对安装文件进行卸载。如果是利用编译安装的BUMO节点，则可以按照以下步骤完成卸载：

1. 输入以下命令进入BUMO的安装目录。

```
cd /bumo
```

2. 输入以下命令删除BUMO节点。

```
make uninstall
```

注解： 至此就完成了BUMO节点的卸载。

8.7.2 针对安装包安装的卸载

在安装完BUMO节点之后可以对安装文件进行卸载。如果是利用安装包安装的BUMO节点，则可以按照以下步骤完成卸载：

1. 输入以下命令删除buchain的目录。

```
sudo rm -rf /usr/local/buchain/
```

2. 输入以下命令删除bumo的软连接。

```
sudo rm -rf /etc/init.d/bumo
```

3. 输入以下命令删除bumod的软连接。

```
sudo rm -rf /etc/init.d/bumod
```

注解： 至此就完成了BUMO节点的卸载。