**WormHole Spec**

Description of WormHole protocol

The WormHole protocol uses the Bitcoin Cash transaction as a carrier; the special op codes OP\_RETURN in the Bitcoin Cash script is used to append the WormHole protocol to the op codes. In general: The WormHole transaction is a special Bitcoin Cash transaction that uses the same security and validation model as the Bitcoin Cash transaction.

The [WormHole full-node client](https://github.com/copernet/wormhole) is a superset of the [bitcoin-abc core client](https://github.com/Bitcoin-ABC/bitcoin-abc) and it has two functions: Bitcoin Cash protocol and WormHole protocol.

The processing flow of the WormHole transaction in the core client:

Block received---》Bitcoin Cash module confirms the validity of the transaction---》WormHole module confirms the validity of the WormHole transaction.

The WormHole protocol uses an account model. Each Bitcoin Cash address is an account, and each account address can contain multiple tokens.

Definition of protocol field

Field: Transaction version

Description: The processing version of the transaction type; the version of each transaction type is monotonically increasing and independent to each other.

Byte: uint16\_t; 2 bytes

Field: Transaction type

Description: The available function fields provided by the **WormHole** protocol

Byte: uint16\_t; 2 bytes

Valid values:

0: token transfer

3: token airdrop

4: transfer all tokens

50: create fixed-amount token

51: create crowdfunding token

53: stop crowdfunding

54: create manageable token

55: issue additional token

56: destroy token

68: get WormHole system currency

70: change the issuer of token

Field: Ecosystem

Description: The ecosystem in which the token is located;

Byte: uint8\_t; 1 byte

Valid value: The current valid value is 1;

Field: Number of coins

Description: The number of tokens in the transaction data that are affected by this field, for example:

• For a divisible token, if you divide the value of this field by 100,000,000, you can get the token amount affected by the identifier; (e.g. if 1 identifies as 0.00000001LSH, then 100000000 identifies as 1LSH); WormHole tokens has the same unit structure as Bitcoin Cash.

• For an indivisible token, the value of this field indicates the amount of token affected; (e.g. 1 identifies as 1 unsplittable token).

Byte: int64\_t; 8 bytes

Valid values: 1-9, 223, 372, 036, 854, 775, 807

• For the amount range of divisible tokens: 0.00000001 - 92,233,720,368.54775807

• For amount range of indivisible tokens: 1 - 9, 223, 372, 036, 854, 775, 807

Note: The current WormHole protocol only supports indivisible tokens.

Field: Property type

Description: Identifies whether the token created in the WormHole protocol is divisible, and whether the created token will replace or append any existing token.

Byte: uint16\_t; 2 bytes

Valid values: 1, 2, 65, 66, 129, 130

* 1: New indivisible token
* 2: New divisible token
* 65: Create a new indivisible token that replaces the previous token
* 66: Create a new divisible token that replaces the previous token
* 129: Create a new indivisible token that appends the previous token
* 130: Create a new divisible token that appends the value of the previous token

Field: Currency identifier

Description: tokenID field in the system

Byte: uint32\_t; 4 bytes

Valid values:

1, 3-2,147,483,647;

0: BCH; 1: WHC

Field: Integer-one byte

Description: Used as a multiplier field or as a parameter field for other calculations

Byte: uint8\_t; 1 byte

Valid values: 0-255

Field: UTC Datetime

Description: Unix timestamp field

Bytes: uint64\_t; 8 bytes

Specifications of the WormHole protocol

All of the following available solution 1: require a wallet that can be used on the server side.

Solution 2: It can be used to develop applications such as wallets, and generate unsigned transactions by calling the RPC service on the server side; then the wallet signs transaction and sends the signed transaction.

**WormHole** protocol requirements for transaction input and output

In the **WormHole** system, the order requirements are mainly divided into two categories:

|  |  |  |
| --- | --- | --- |
| **Category** | **txin** | **txout** |
| WHC’s issuance transaction (68) | The first transaction input (index 0) is the sender | The first transaction output (index 0) is the burn output, and the second output (index 1) is the WormHole protocol. |
| Other types of transactions | The first transaction input (index 0) is the sender | The last transaction output (the largest index) is the receiver |

Issuance of WHC (68)

The WHC is obtained by burning the BCH, and the exchange ratio of BCH to WHC is 1:100; the WHC can be obtained by sending the BCH to a specific burning address.

Burning address:

Main network: bitcoincash:qqqqqqqqqqqqqqqqqqqqqqqqqqqqqu08dsyxz98whc Test network: bchtest:qqqqqqqqqqqqqqqqqqqqqqqqqqqqdmwgvnjkt8whc

In the main-net, WHC will be credited into account after confirmation from 1000 blocks; in the test-net, WHC will be credited into account after confirmation from 3 blocks.

1BCH = 100WHC; 1WHC = 100,000,000C;

Specification for obtaining WHC by burning BCH:

The condition for transaction input: The first transaction input (i.e., the index is 0) is the default account address for obtaining the WHC in the **WormHole** engine.

The conditions for transaction output: WormHole engine default: the first transaction output must be the burning output (i.e. index 0); the second output is the data of **WormHole** protocol; and the third output is the redundant BCH redemption address output (this output is not necessary);

The condition of burning amount: 1BCH minimum.

The transaction specification mainly contains: the order of transaction input and output, the amount of burning, \*\*WormHole payload data\*\*

**WormHole’s** Issuance of WHC transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 68 |

In addition to the field restrictions of the WormHole protocol, WormHole transactions are also invalid in the following cases:

• The output of the burning of BCH is not at the position where the index is 0.

• The position where the index is 1 is not filled with the data of WormHole’s burning protocol

• The amount of BCH burned is less than 1BCH

An operational example:

There are two ways to get the equivalent amount of WHC by burning.

Solution 1: wormholed-cli whc\_burnbchgetwhc 1 ""

Solution 2: The user specifies the transaction input as the address to accept WHC.

Add transaction input: wormholed-cli whc\_createrawtx\_input "" "address1" index (This step specifies the address to receive the WHC: address1)

Add the transaction output and send the tokens to the specific burning address: wormholed-cli whc\_createrawtx\_reference "rawtx" "address2" 1 (Burning address: address2)

Create **WormHole** payload data: wormholed-cli whc\_createpayload\_burnbch

Add transaction output, add **WormHole** payload data to the transaction output: wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"

Add transaction output, get change (optional): wormholed -cli whc\_createrawtx\_reference "rawtx" "address3" 0.5 (change address: address3)

Sign the created transaction: wormholed-cli signrawtransaction "rawtx"

Send transaction: wormholed-cli sendrawtransaction "hextx"

Return value: generated transaction hash

**WormHole’s** valid payload data: 6a080877686300000044

Explanation as follows:

6a: OP\_RETURN; 08: The data length of **WormHole** protocol;

08776863: Magicof **WormHole** protocol

0000: Version of **WormHole** protocol

0044: Transaction type

Example of transaction: 53d2d2701bdca225ae4486e72854369a09499a0555c22ef1936c02924fe901cb

Token transfer (0)

Transferring a specified amount of tokens from one account address to another account address.

Note: The last output must be the recipient of the token. The first input must be the sender of the token.

Specification of the transfer transaction:

Transaction input: The first transaction input (index 0)] is the default sender’s account address in the **WormHole** engine;

Transaction output: The last transaction output (maximum index) is the default recipient’s account address in the **WormHole** engine;

Transfer amount: The amount of token transferred from the sender to the recipient, and the sender’s account must have sufficient balance;

Transfer tokenID: The token must exist in the system.

The transaction specification mainly contains: the order of transaction input and output, the amount of token being transferred, tokenID

**WormHole’s** token transfer transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 68 |
| tokenID (Property ID) | Currency identifier | 1 (WHC) |
| Transfer amount (Amount to transfer) | Number of Coins | 1000 (1000C) |

Except for the field restrictions of the **WormHole** protocol, **WormHole** transactions are also invalid in the following cases：

• The available balance for the specified token is 0 on the sender's address

• The amount of the transfer exceeds the token balance on the sender’s account

• Transfer token does not exist

• The transfer's tokenID is 0 (BCH); the transaction type does not support BCH transfers.

An operational example:

There are 2 ways to transfer 1WHC.

* Solution 1:

wormholed-cli whc\_send fromaddress toaddress propertyid amount (redeemaddress) (referenceamount)

referenceamount: toaddress’s output, the minimum amount of output is the default

redeemaddress: the redeem address of the excess BCH input, the sender's address is the default address

* Solution 2:

Add transaction input: wormholed-cli whc\_createrawtx\_input "" txid index

Create payload data for sending tokens: wormholed-cli whc\_createpayload\_simplesend

Add transaction output, add the created **WormHole** payload data to the transaction output: wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload "

Add transaction output, get change: wormholed-cli whc\_createrawtx\_reference "rawtx" destination amount

Add transaction output, transfer token to the address: wormholed-cli whc\_createrawtx\_reference "rawtx" destination amount

Sign the created transaction: wormholed-cli signrawtransaction

Send Transaction: wormholed-cli sendrawtransaction

**WormHole’s** valid payload data: 6a140877686300000000000000010000000005f5e100

Explanation as follows:

6a: OP\_RETURN; 14: The data length of **WormHole** protocol;

08776863: Magic of **WormHole** protocol

0000: Version of **WormHole** protocol

0000: Transaction type

00000001: tokenID

0000000005f5e100: Transfer amount (1WHC)

Example of transaction: d9dd78f140691dd946175696bd312dfcfac4c5cc5c4f5eb1143dfde0db83c25e

Token airdrop (3)

Airdropping will send a specific amount of tokens (ID1) to the holders of that token (ID2) (except the sender), and the airdrop (ID1) is distributed by the ratio of recipients’ token (ID2) held in balance.

Note: The first input must be the sender of the airdrop

Airdrops will charge a certain amount of gas in WHC. The transaction fee for each airdrop is: the number of accounts receiving airdrops\* system unit price (1C);

**WormHole’s** airdrop transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 3 |
| Airdrop tokenID（Property ID） | Currency identifier | 1 |
| Airdrop amount (Amount to transfer) | Amount to transfer | 15,000,000,000 |
| Target tokenID (Property ID) | Currency identifier | 1 |

Except for the field restrictions of the **WormHole** protocol, **WormHole** transactions are also invalid in the following cases:

• The balance available on the sender's address is less than the amount of the airdrop

• The specified token does not exist, or the tokenID is 0

• The sender’s address cannot afford WHC gas

• The sender’s address has the total amount of target tokens

An operational example:

There are 2 ways to transfer 1WHC.

* Solution 1:

wormholed-cli whc\_sendsto fromaddress propertyid amount redeemaddress distributionproperty

redeemaddress: BCH’s change address, the sender address is the default

distributionproperty: the tokenID of airdrop’s participants, the tokenID of airdrop recipients is the default

* Solution 2:

Add transaction input: wormholed-cli whc\_createrawtx\_input "" txid index

Create airdrop payload data: wormholed-cli whc\_createpayload\_sto propertyid amount distributionproperty

Add transaction output, add the created WormHole payload data to the transaction output: wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"

Add transaction output, get change: wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount

Sign the created transaction: wormholed-cli signrawtransaction "rawtx"

Send transaction: wormholed-cli sendrawtransaction "tx"

**WormHole’s** valid payload data: 6a18087768630000000300000001000000037e11d600

Explanation as follows:

6a: OP\_RETURN; 18: The data length of **WormHole** protocol;

08776863: Magic of **WormHole** protocol

0000: Version of **WormHole** protocol

0003: Transaction type

00000001: tokenID

000000037e11d600: Airdrop amount (1.5WHC)

00000001: tokenID

Example of transaction: 9bb8eb45e5b3ba4059c38409a225216f3aaae990f07d2541f1c09571862b99d2

Transfer all tokens (4)

Transferring all tokens of an account address (A) to another account address (B), BCH is not included in the transfer.

Note: The last output must be the recipient of these tokens, the first input must be the sender of these tokens

**WormHole’s** transfer all tokens transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 4 |
| Ecosystem | Ecosystem | 1 |

An operational example:

There are 2 ways to transfer 1WHC.

* Solution 1:

wormholed-cli whc\_sendall fromaddress toaddress ecosystem (redeemaddress) (referenceamount)

redeemaddress: BCH’s redeem address, the sender's address is the default; referenceamount: the amount of BCH output to the token receiver, the minimum output amount is the default

* Solution 2:

Add transaction input: wormholed-cli whc\_createrawtx\_input "" "txid" index

Create **WormHole** payload data: wormholed-cli whc\_createpayload\_sendall ecosystem

Add transaction output, add the created **WormHole** payload data to the transaction output: wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"

Add transaction output, get change: wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount

Add transaction output, transfer all tokens to this address: wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount

Sign the created transaction: wormholed-cli signrawtransaction "rawtx"

Send transaction: wormholed-cli sendrawtransaction "tx"

**WormHole’s** valid payload data: 6a09087768630000000401

Explanation as follows:

6a: OP\_RETURN; 09: The data length of **WormHole** protocol;

08776863: Magic of **WormHole** protocol

0000: Version

0004: Transaction type

01: Ecosystem

Example of transaction: 8f4a11bb724139a43494b08da35beac1a53c34aec13eab22d188540f5cc0c164

Create fixed-amount token (50)

Creating a fixed amount of tokens, once the transaction has been confirmed valid by **WormHole**, the issuer’s address will hold all tokens.

Except for the field restrictions of the **WormHole** protocol, the transaction is considered a valid **WormHole** transaction only if the following conditions are met:

* When Property Type field identifies a new token, Previous Property ID field must be 0
* Property Name field cannot be NULL
* Ecosystem field must be 1
* Property Type field must be 1

**WormHole’s** create fixed-amount token transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Filed** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 50 |
| Ecosystem | Ecosystem | 1 |
| Token type (Property Type) | Property Type | 1 |
| Previous tokenID (Previous Property ID) | Currency identifier | 0 |
| Token category (Property Category) | String null-terminated | "" |
| Token subcategory (Property Subcategory) | String null-terminated | "" |
| Token name (Property Name) | String null-terminated | "ludete" |
| Token URL (Property URL) | String null-terminated | "" |
| Token description data (Property Data) | String null-terminated | "" |
| **Wormhole** (Number Properties) | Number of coins | 10000 |

An operational example:

There are 2 ways to transfer 1WHC.

* Solution 1:   
  wormholed-cli whc\_sendissuancefixed fromaddress ecosystem type previousid category subcategory name url data amount
* Solution 2:  
  Add transaction input：wormholed-cli whc\_createrawtx\_input "" txid index   
  Create payload data for creating fixed tokens：wormholed-cli whc\_createpayload\_issuancefixed ecosystem type previousid category subcategory name url data amount  
  Add transaction output, add created **WormHole** payload data to the transaction output:   
  wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"   
  Add transaction output, get change：wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount  
  Sign created transaction：wormholed-cli signrawtransaction "rawtx"   
  Send transaction：wormholed-cli sendrawtransaction "tx"

**WormHole’s** valid payload data：6a4108776863000000320100010000000074657374007465737432006669

Explanation as follows:

6a: OP\_RETURN; 41: The data length of the **WormHole** protocol   
08776863: The magic of **WormHole** protocol

0000: Version  
0032: Transaction type  
01: Ecosystem  
0001: Token property type   
.... : Customized data  
0000000005f5e100: The amount of token issued

Example of transaction: bf8a2e94eba3ee0c42f14fb2e3a91d2b42e0f11222db9ba7804054b614f3ea1

Create crowdfunding token (51)

Creating a crowdfunding token. Once the transaction is confirmed by a **WormHole** node, the sender’s address will get the tokens.

Note: the current crowdfunding only accepts WHC. The total amount of crowdfunding is fixed, the maximum number of int64 is：4611686018427387904

The crowdfunding will be closed permanently when the followed situations occur:

* When the Tip block timestamp of the Blockchain is greater than or equal to the crowdfunding deadline time, the crowdfunding will be permanently closed
* The crowdfunding is shut down manually
* The total amount of crowdfunding is sold out

Each account address can only host one active crowdfunding event at a specified time. It prevents crowdfunding participants from specifying which crowdfunding to participate in the issuer’s account.

The crowdfunding token will be immediately added to the available balance of participants and the issuer. These tokens can be spent or used in other ways. The raised fund will be added to the available balance of the issuer’s address and it is available for all kinds of usages.

We are planning to encourage users to participate in crowdfunding by giving them early bird bonus. This early bird bonus will decrease linearly since the beginning of crowdfunding, and the bonus will end up reaching 0 when the crowdfunding is completed.

The equation of early bird bonus is**: (the deadline time of the crowdfunding - the block time of the transaction) / the number of seconds per week + (the deadline time of the crowdfunding - the block time of the transaction) % the number of second per week) \* early bird bonus**

An operational example:

The exchange ratio WHC to HLT is: WHC=100HLT; the early bird bonus is 10%;

Purchaser who participate in crowdfunding two weeks prior to the deadline would get an additional 20% of token;

Purchaser who participate in the crowdfunding 1.5 weeks prior to the deadline would get an additional 15% of token. For example, if you buy 1 WHC of crowdfunding, you can get 115 HLT.

The **WormHole** client guarantees that the sum of tokens from the participant and the tokens in the issuer's account does not exceed the total issuance amount. If the last crowdfunding purchase breaches the total amount issued, the protocol will cut down the amount of tokens available for both participants and issuer, ensuring that the purchase can only buy the remaining unissued tokens.

Percentage for issuer: For each crowdfunding transaction, the available balance of the issuer's address will receive corresponding amount of tokens.

The consensus is: the amount of tokens purchased by the crowdfunding\* ((Percentage for issuer) / 100)

**WormHole’s** create crowdfunding token transaction protocol field

|  |  |  |
| --- | --- | --- |
| **Filed** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 50 |
| Ecosystem | Ecosystem | 1 |
| Token type (Property Type) | Property Type | 1 |
| Previous tokenID (Previous Property ID) | Currency identifier | 0 |
| Token category (Property Category) | String null-terminated | "" |
| Token subcategory (Property Subcategory) | String null-terminated | "" |
| Token name (Property Name) | String null-terminated | "ludete" |
| Token URL (Property URL) | String null-terminated | "" |
| Token description data (Property Data) | String null-terminated | "" |
| Crowdfunding tokenID (Currency Identifier Desired) | Currency identifier | 1 |
| Exchange ratio (Number Properties per Unit Invested) | Number of Coins | 100(raise token: fund raising token) |
| Deadline time of crowdfunding (Deadline) | Deadline | Unix time stamp |
| Percentage for timestamp early bird bonus (Early Bird Bonus %/Week ) | Integer one-byte | 10 |
| Percentage for issuer | Integer one-byte | 12 |

Except for the field restrictions of the **WormHole** protocol, the transaction is considered as a valid **WormHole** transaction only if the following conditions are met.

• When the Property Type field identifies a new token, the Previous Property ID field must be 0.

• Property Name field must not be NULL

• Ecosystem field must be 1

• Property Type field must be 1

• Currency Identifier Desired crowdfunding token must be WHC (1)

• Deadline crowdfunding deadline must be greater than crowdfunding start time

An operational example:

There are 2 ways to transfer 1WHC.

* Solution 1:

wormholed-cli whc\_sendissuancecrowdsale fromaddress ecosystem type previousid category subcategory name url data propertyiddesired tokensperunit deadline earlybonus issuerpercentage

* Solution 2:

Add transaction input: wormholed-cli whc\_createrawtx\_input "" txid index

Create payload data for generating crowdfunding tokens: wormholed-cli whc\_createpayload\_issuancecrowdsale ecosystem type previousid category subcategory name url data propertyiddesired tokensperunit deadline earlybonus issuerpercentage

Add transaction output, add created **WormHole** payload data to the transaction output: wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"

Add transaction output, get change: wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount

Sign the created transaction: wormholed-cli signrawtransaction "rawtx"

Send transaction: wormholed-cli sendrawtransaction "tx"

**WormHole’s** valid payload data**:**

6a4c5f087768630000003301000100000000416374697669746965730041637469766974696573206f66006f6d6e696361736800687474703a2f2f7777772e676f6f676c652e636100222200 00000001 00000002540be400 000000005b574d00 0a0a

Explanation as follows:

6a: OP\_RETURN; 41: The data length of **WormHole** protocol;

08776863: Magic of **WormHole** protocol

0000: Transaction version  
0033: Transaction type  
01: Ecosystem

0001: token property type  
00000000: previous tokenID   
.... : Customized data

0000000005f5e100: the amount of token issued  
00000001: the collected tokenID of crowdfunding  
00000002540be400: exchange ratio  
000000005b574d00: crowdfunding deadline

0a: Percentage for early bird bonus

0a: Percentage for issuer

Example of Transaction：

4037086b5d1c16e97bc791f242db4996241202a2ac3e9700b6098547b552ffc2

Stop crowdfunding (53)

When the crowdfunding is active (i.e. raising funds), and the project team needs to stop the crowdfunding in order to keep the value of the token (or have other reasons to do so); it can be done by creating such a transaction and broadcast it to the Blockchain. The shutdown of crowdfunding will not affect the early bird bonus who already purchased the token.

Except for the field restrictions of the **WormHole** protocol, **WormHole** transactions are also invalid in the following cases:

* Shutting down a crowdfunding that has been closed
* The sender of this transaction is not the issuer of the crowdfunding token

**WormHole’s** stop crowdfunding transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Filed** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 53 |
| TokenID (Property ID) | Currency identifier | 9 |

Note: Participate in the closed crowdfunding is invalid, the participant will not acquire any token, and the invested amount of WHC will be transferred to the address of the issuer. The redemption of such investment requires negotiation with the issuer.

Create manageable token (54)

Creating a token that can be managed by the issuer. The address of the first transaction input (index 0) is the issuer of the token. At this time, the issuer's address has 0 available balance of the token, and two additional transaction types are used to issue a specific amount of the token (issue additional token), or destroy a specific amount of the token (destroy token).

Except for the field restrictions of the **WormHole** protocol, the **WormHole** transaction is considered valid only if the following conditions are met:

• When the token type field identifies a new token, the previous tokenID field must be 0.

• Property Name field must not be NULL

• Ecosystem field must be 1

• Property Type field must be 1

**WormHole’s** create manageable token transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Filed** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 50 |
| Ecosystem | Ecosystem | 1 |
| Token type (Property Type) | Property Type | 1 |
| Previous tokenID (Previous Property ID) | Currency identifier | 0 |
| Toke category (Property Category) | String null-terminated | "" |
| Token subcategory (Property Subcategory) | String null-terminated | "" |
| Token name (Property Name) | String null-terminated | "ludete" |
| Token URL (Property URL) | String null-terminated | "" |
| Token description data (Property Data) | String null-terminated | "" |

An operational example: There are 2 ways to transfer 1WHC.

* Solution 1:  
  wormholed-cli whc\_sendissuancemanaged fromaddress ecosystem type previousid category subcategory name url data
* Solution 2:  
  Add transaction input：wormholed-cli whc\_createrawtx\_input “” txid index   
  Create payload data of generating manageable token：wormholed-cli whc\_createpayload\_issuancemanaged ecosystem type previousid category subcategory name url data propertyiddesired tokensperunit deadline earlybonus issuerpercentage  
  Add transaction output, add the created payload data **of WormHole** to the transaction output：   
  wormholed-cli whc\_createrawtx\_opreturn “rawtx” “payload”   
  Add transaction output, get change：wormholed-cli whc\_createrawtx\_reference “rawtx” “destination” amount  
  Sign the created transaction：wormholed-cli signrawtransaction “rawtx”   
  Send transaction：wormholed-cli sendrawtransaction “tx”

**WormHole’s** valid payload data: 6a3608776863000000360100010000000048656c6c6f00576f726c64006c7564657465007777772e6c75646574652e636f6d00796f6e6700

Explanation as follows:

6a: OP\_RETURN; 36: The data length of **WormHole** protocol

08776863: Magic of **WormHole** protocol

0000: Version  
0036: Transaction type  
01: Ecosystem  
0001: Token type  
00000000: Previous token ID   
.... : Customized data

Example of transaction：b85ffb2d38339b4432cd8dfe50861e05376665f1ffe08c4824dea934aadb20e1

Issue additional token (55)

This transaction type is used to service the manageable token by issuing additional token amounts. After the issuance transaction is confirmed, the additional token amount will be added to the recipient's available balance. Note: After the token is created, the initial amount is 0; the maximum legal amount is: (1 << 63) -1

Except for the field restrictions of the **WormHole** protocol, **WormHole** transactions are also invalid in the following cases:

• The additional token category is not a manageable type of token

• The sender of the transaction is not the token issuer

• The sum of the cumulative tokens exceeds the maximum legal amount.

**WormHole’s** issue additional token transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Filed** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 55 |
| tokenID (Property ID) | Currency identifier | 5 |
| Additional issued amount (Number Properties) | Number of coins | 1000 |
| Additional issuance message (Memo) | String null-terminated | "reward" |

An operational example:

* Solution 1:   
  wormholed-cli whc\_sendgrant fromaddress toaddress propertyid amount ("data")
* Solution 2：   
  Add transaction input: wormholed-cli whc\_createrawtx\_input "" txid index   
  Create payload data of token: wormholed-cli whc\_createpayload\_grant propertyid amount ("data")   
  Add transaction output, add the created **WormHole** payload data to the transaction output:  
  wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"   
  Add transaction output, get change: wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount  
  Sign the created transaction：wormholed-cli signrawtransaction "rawtx"   
  Send transaction：wormholed-cli sendrawtransaction "tx"

**WormHole’s** valid payload data：6a1d08776863000000370000000c00000000000027107061792062696c6c00

Explanation as follows：

6a: OP\_RETURN; 1d: The data length of **WormHole** protocol;

08776863: Magic of **WormHole** protocol   
0000: Transaction version  
0037: Transaction type   
01: Ecosystem  
0001: token type  
0000000c: tokenID   
0000000000002710: Amount of additional issued token  
.... : Customized data

Example of transaction：6a1d08776863000000370000000c00000000000027107061792062696c6c00

Destroy token (56)

This transaction type is used to service the manageable token by destroying token amounts. After the destroying transaction is confirmed, and the total token amount will also be deducted from the system.

Except for the field restrictions of the **WormHole** protocol, **WormHole** transactions are also invalid in the following cases:

• The destroyed token is not a manageable type of token

• The sender of the transaction is not the token issuer

• The number of destroyed tokens exceeds the amount of tokens available for the current account

**WormHole’s** destroy token transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Filed** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 55 |
| tokenID (Property ID) | Currency identifier | 5 |
| Destroyed amount of tokens (Number Properties) | Number of coins | 1000 |
| Destroy message (Memo) | String null-terminated | "spend" |

An operational example:

* Solution 1:  
  wormholed-cli whc\_sendgrant fromaddress toaddress propertyid amount ("data")
* Solution 2:  
  Add transaction input: wormholed-cli whc\_createrawtx\_input "" txid index   
  Create payload data of generating manageable token: wormholed-cli whc\_createpayload\_revoke propertyid amount ("data")   
  Add transaction output, add the created **WormHole** payload data to the transaction output:   
  wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"   
  Add transaction output, get change: wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount  
  Sign the created transaction: wormholed-cli signrawtransaction "rawtx"   
  Send transaction: wormholed-cli sendrawtransaction "tx"

**WormHole’s valid payload data**：6a1508776863000000380000000c0000000000000bb800

Explanation as follows：

6a: OP\_RETURN; 15: The data length of **WormHole** protocol；   
08776863: Magic of **WormHole** protocol   
0000: Transaction version  
0038: Transaction type  
01: Ecosystem  
0001: Token type  
0000000c: tokenID

0000000000000bb8: Destroyed amount   
.... : Customized data

Example of transaction：d5966fd7568b29b19ba5400d94e4fa8adbc91040fd0261f1209db163518a0327

Change the issuer of the token (70)

This transaction type is used to change the issuer of a token which is already issued in the system.

Conditions for transaction input: The first transaction input is the original issuer of the token

Conditions for transaction output: The last transaction output is the new token issuer

Except for the field restrictions of the **WormHole** protocol, **WormHole** transactions are also invalid in the following cases:

• The sender of the transaction is not the original issuer of the token

• No new issuer address is specified.

**WormHole’s** *change issuer of token* transaction protocol field:

|  |  |  |
| --- | --- | --- |
| **Filed** | **Type** | **Example** |
| Transaction version | Transaction version | 0 |
| Transaction type | Transaction type | 55 |
| tokenID (Property ID) | Currency identifier | 5 |

An operational example:

* Solution 1:   
  wormholed-cli whc\_sendchangeissuer fromaddress toaddress propertyid
* Solution 2:   
  Add transaction input: wormholed-cli whc\_createrawtx\_input "" txid index   
  Create payload data of **Wormhole**: wormholed-cli whc\_createpayload\_changeissuer propertyid   
  Add transaction output, add the created **WormHole** payload datato transaction output:  wormholed-cli whc\_createrawtx\_opreturn "rawtx" "payload"   
  Add transaction output, get change：wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount(this step is not necessary)   
  Add transaction output as the new token issuer: wormholed-cli whc\_createrawtx\_reference "rawtx" "destination" amount  
  Sign the created transaction: wormholed-cli signrawtransaction "rawtx"   
  Send transaction: wormholed-cli sendrawtransaction "tx"

**WormHole’s** valid payload data:：6a0c08776863000000460000000c

Explanation as follows：

6a: OP\_RETURN; 15: The data length of **WormHole** protocol；   
08776863: Magic of **WormHole** protocol   
0000: Transaction version  
0038: Transaction type

0000000c: tokenID

Example of transaction：ce6877f4da9e627741cf087aa777917f5ecf5fb8ff47037b211140b68155d9eb