Conflux101

    - Week9 - 8.29
    - Week10-NFT
    - Week11-9.13
    - Week12 Contract address
    - Week13 - 9.26
    - Week14-
    - Week15 - 10.17
    - Week16: 10.24 - 10.28 python-sdk
    - Week 17 - 10.31
    - Week18-Conflux
    - Week19- 11.14
    - Week 20 11.28

- Gas
- EN Translate
    - Gas
    - Accounts

- Accounts
- Token Standard
    - ERC4626-Tokenized Vault Standard

- Core Space eSpace

Overview

Conflux Network 3000 TPS EVM ,

Fluent

Defi NFT

SDKs

- js-conflux-sdk
- go-conflux-sdk
- java-conflux-sdk
- python-conflux-sdk

Other Libs

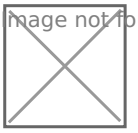
- cip-23

Tools

- Conflux studio
- chainIDE
- Conflux-truffle
- hardhat-conflux

Glossary

TreeGraph



Epoch

Pivot chain block Epoch Epoch block EpochNumber 0

Base32

base32 CIP-37

Era

20000 Epoch Era

Checkpoint

era Checkpoint checkpoint consensus fullnode

Snapshot

Snapshot state 2000 epoch merge state snapshot

AdminControl

`AdminControl` admin

`admin` `setAdmin(address contractAddr, address newAdmin)`

`destroy(address contractAddr)` `suicide()` SponsorWhitelist

ConfluxScan

`AdminControl` `getAdmin(address contractAddr)`

1. `A` `B` `C` `C`
2. `A` `B` `B` `C` `D` `C` `A` `C` `B`
3. Conflux `D` 2

`contract_addr` `AdminControl.setAdmin(contract_addr, new_admin)`
`AdminControl.destroy(contract_addr)`

```
const PRIVATE_KEY = '0xxxxxxx';
const cfx = new Conflux({
  url: 'http://test.confluxrpc.org',
  logger: console,
});
const account = cfx.wallet.addPrivateKey(PRIVATE_KEY); // create account instance

const admin_contract = cfx.InternalContract('AdminControl')
// to change administrator
```

```
admin_contract.setAdmin(contract_addr, new_admin).sendTransaction({
  from: account,
}).confirmed();

// to kill the contract
admin_contract.destroy(contract_addr).sendTransaction({
  from: account,
}).confirmed();
```

SponsorWhitelistControl

Conflux Dapps Sponso

Message Call `A` `B` fl `B` `C`, Conflux `A` `B` `A` `B` `B` `C` `B` /

SponsorControl

- `sponsor_for_gas`
- `sponsor_for_collateral`
- `sponsor_balance_for_gas`
- `sponsor_balance_for_collateral`
- `sponsor_limit_for_gas_fee`
- `whitelist`

- *: `sponsor_for_gas` `whitelist` `sponsor_limit_for_gas_fee` `sponsor_balance_for_gas` `sponsor_balance_for_gas`
- : `sponsor_balance_for_collateral` `whitelist` `sponsor_balance_for_collateral`

`sponsor_for_gas` `sponsor_for_collateral` `sponsor_limit_for_gas_fee` SponsorCo

`sponsor_for_gas` `setSponsorForGas(address contractAddr, uint upperBound)`

1. `sponsor_balance_for_gas`
2. `sponsor_limit_for_gas_fee` `upperBound` `sponsor_balance_for_gas`
3. `1000` 1000

`sponsor_balance_for_gas` `sponsor_for_gas` `sponsor_balance_for_gas` `sponsor_for_gas` `sponsor_limit_for_gas_fee`

`sponsor_for_collateral` `setSponsorForCollateral(address contractAddr)` `sponsor_for_collateral` `sponsor_balance_for_collateral`

`setSponsorForGas(address contractAddr, uint upperBound)` `setSponsorForCollateral(address contractAddr)` `sponsor_limit_for_gas_fee`

`addPrivilege(address[] memory)` `sponsor_for_gas` `0x0000000000000000000000000000000000000000` `removePrivilege(address[] memory)`

1.

`addPrivilegeByAdmin(address contractAddr, address[] memory addresses)` `removePrivilegeByAdmin(address contractAddr, address[] memory addresses)`

```
pragma solidity >=0.4.15;

import "https://github.com/Conflux-Chain/conflux-rust/blob/master/internal_contract/contracts/SponsorWhitelistControl.sol";

contract CommissionPrivilegeTest {
    mapping(uint => uint) public ss;

    function add(address account) public payable {
        SponsorWhitelistControl cpc = SponsorWhitelistControl(0x0888000000000000000000000000000000000001);
        address[] memory a = new address[](1);
        a[0] = account;
```

```
        cpc.addPrivilege(a);
    }

    function remove(address account) public payable {
        SponsorWhitelistControl cpc =
SponsorWhitelistControl(0x0888000000000000000000000000000000000001);
        address[] memory a = new address[](1);
        a[0] = account;
        cpc.removePrivilege(a);
    }

    function foo() public payable {
    }

    function par_add(uint start, uint end) public payable {
        for (uint i = start; i < end; i++) {
            ss[i] = 1;
        }
    }
}
```

`contract_addr` /

```
const PRIVATE_KEY = '0xxxxxxx';
const cfx = new Conflux({
  url: 'http://test.confluxrpc.org',
  logger: console,
});
const account = cfx.wallet.addPrivateKey(PRIVATE_KEY); // create account instance

const sponsor_contract = cfx.InternalContract('SponsorWhitelistControl');
sponsor_contract.setSponsorForGas(contract_addr, your_upper_bound).sendTransaction({
  from: account,
  value: your_sponsor_value
}).confirmed();
```

`setSponsorForGas(contract_addr, your_upper_bound)` `setSponsorForCollateral(contract_addr)`

`addPrivilege` `removePrivilege` `whitelist` `0x0000000000000000000000000000000000000000` `whitelist`

```
you_contract.add(white_list_addr).sendTransaction({
  from: account,
})

you_contract.remove(white_list_addr).sendTransaction({
  from: account,
})
```

`whitelist` `you_contract.foo()` `you_contract.par_add(1, 10)`

Staking

Conflux “ ”

Con **Staking** /

`deposit(uint amount)` `amount` `balance` `stakingBalance` . `payable`

`withdraw(uint amount)` Conflux

`voteLock(uint amount, uint unlock_block_number)` `stakingBalance` `unlock_block_number` `amount`
” `stakingBalance` X 5CFX ” ** **

`x` `y` Cc y l `2 * 60 * 60 * 24 * 365` `x`

1. `stakingBalance` 10CF `voteLock(100 * 10^18, x)` 1 `stakingBalance`
2. `voteLock(8 * 10^18, x)`
3. `voteLock(6 * 10^18, x+y)` 2CFX 6CFX
4. `voteLock(0, x)` 2 3
5. `voteLock(9 * 10^18, x+y)` “ 9CFX ”

Conflux Protocol Specification 8.3.2

```
const PRIVATE_KEY = '0xxxxxxx';
const cfx = new Conflux({
  url: 'http://test.confluxrpc.org',
```

```
  logger: console,
});
const account = cfx.wallet.addPrivateKey(PRIVATE_KEY); // create account instance

const staking_contract = cfx.InternalContract('Staking');
// deposit some amount of tokens
staking_contract.deposit(your_number_of_tokens).sendTransaction({
  from: account,
}).confirmed();

// withdraw some amount of tokens
staking_contract.withdraw(your_number_of_tokens).sendTransaction({
  from: account,
}).confirmed();

// lock some tokens until some block number
staking_contract.voteLock(your_number_of_tokens, your_unlock_block_number).sendTransaction({
  from: account,
}).confirmed();
```

ConfluxContext

ConfluxContext `epochNumber`, `posHeight`, `finalizedEpochNumber`

```
pragma solidity >=0.4.15;


contract ConfluxContext {
    /*** Query Functions ***/
    /**
     * @dev get the current epoch number
     * @return the current epoch number
     */
    function epochNumber() public view returns (uint256) {}
    /**
     * @dev get the height of the referred PoS block in the last epoch
`    * @return the current PoS block height
     */
    function posHeight() public view returns (uint256) {}
    /**
     * @dev get the epoch number of the finalized pivot block.
     * @return the finalized epoch number
     */
    function finalizedEpochNumber() public view returns (uint256) {}
}
```

PoSRegister

PoSRegister PoW PoS PoW PoS

1. conflux PoW PoS PoW
2. CFX 1000
3. 1 PoS = 1000 CFX
4. vote power
5. `voteLock()` ng

- `staking`
- 1000CFX

register

PoW PoS , PoW 1000CFX PoS PoW 1000CFX 1 vote power

- PoS
- `conflux rpc local pos register --power 1`, `register`

- indentifier: PoS
- votePower
- blsPubKey: PoS blsPubKey
- vrfPubKey PoS vrfPubKey
- blsPubKeyProof: PoS blsPubKeyProof

increaseStake

PoW PoS , , CFX

- PoW PoS
- CFX 1000CFX

- votePower:

retire

PoW PoS , PoW PoS

- PoW PoS

- votePower:

getVotes

token votes token votes

- PoW PoS

- identifier: PoS

- totalStakedVotes token votes
- totalUnlockedVotes token votes

identifierToAddress

PoS PoW

- PoW PoS

- identifier: PoS

- address PoW

addressToIdentifier

PoW PoS

- PoW PoS

- address: PoW

- identifier PoS

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0;

interface PoSRegister {
    /**
     * @dev Register PoS account
     * @param indentifier - PoS account address to register
     * @param votePower - votes count
     * @param blsPubKey - BLS public key
     * @param vrfPubKey - VRF public key
     * @param blsPubKeyProof - BLS public key's proof of legality, used to against some
attack, generated by conflux-rust fullnode
     */
    function register(
        bytes32 indentifier,
        uint64 votePower,
        bytes calldata blsPubKey,
        bytes calldata vrfPubKey,
        bytes[2] calldata blsPubKeyProof
    ) external;

    /**
     * @dev Increase specified number votes for msg.sender
     * @param votePower - count of votes to increase
     */
    function increaseStake(uint64 votePower) external;

    /**
     * @dev Retire specified number votes for msg.sender
     * @param votePower - count of votes to retire
     */
```

```
    function retire(uint64 votePower) external;

    /**
     * @dev Query PoS account's lock info. Include "totalStakedVotes" and "totalUnlockedVotes"
     * @param identifier - PoS address
     */
    function getVotes(bytes32 identifier) external view returns (uint256, uint256);

    /**
     * @dev Query the PoW address binding with specified PoS address
     * @param identifier - PoS address
     */
    function identifierToAddress(bytes32 identifier) external view returns (address);

    /**
     * @dev Query the PoS address binding with specified PoW address
     * @param addr - PoW address
     */
    function addressToIdentifier(address addr) external view returns (bytes32);

    /**
     * @dev Emitted when register method executed successfully
     */
    event Register(bytes32 indexed identifier, bytes blsPubKey, bytes vrfPubKey);

    /**
     * @dev Emitted when increaseStake method executed successfully
     */
    event IncreaseStake(bytes32 indexed identifier, uint64 votePower);

    /**
     * @dev Emitted when retire method executed successfully
     */
    event Retire(bytes32 indexed identifier, uint64 votePower);
}
```

CrossSpaceCall

CrossSpaceCall conflux coreSpace ESpace coreSpace ESpace

1. conlux CoreSpace ESpace conflux
2. CoreSpace ESpace

- Conflux

createEVM

espace call

call `createEVM`nce+1 call

- init
-

transferEVM

PoW Espace

- conflux
- to: PoW Espace
-

callEVM

ESpace call

- to: PoW Espace

- data: encode

-

staticCallEVM

ESpace

- to: CroeSpace Espace
- data: encode

-

withdrawFromMapped

ESpace CoreSpace

- value:

mappedBalance

CoreSpace CoreSpace

- address: CoreSpace
-

mappedNonce

CoreSpace CoreSpace nonce

- address: CoreSpace
- nonce

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0;
```

```
interface CrossSpaceCall {

    event Call(bytes20 indexed sender, bytes20 indexed receiver, uint256 value, uint256 nonce,
bytes data);

    event Create(bytes20 indexed sender, bytes20 indexed contract_address, uint256 value,
uint256 nonce, bytes init);

    event Withdraw(bytes20 indexed sender, address indexed receiver, uint256 value, uint256
nonce);

    event Outcome(bool success);

    function createEVM(bytes calldata init) external payable returns (bytes20);

    function transferEVM(bytes20 to) external payable returns (bytes memory output);

    function callEVM(bytes20 to, bytes calldata data) external payable returns (bytes memory
output);

    function staticCallEVM(bytes20 to, bytes calldata data) external view returns (bytes
memory output);

    function withdrawFromMapped(uint256 value) external;

    function mappedBalance(address addr) external view returns (uint256);

    function mappedNonce(address addr) external view returns (uint256);
}
```

ParamsControl

`ParamsControl` DAO Conflux `ParamsControl` PoW CIP-94 PoS

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.8.0;

interface ParamsControl {
    struct Vote {
        uint16 topic_index;
        uint256[3] votes;
    }

    /*** Query Functions ***/
    /**
     * @dev cast vote for parameters
     * @param vote_round The round to vote for
     * @param vote_data The list of votes to cast
     */
    function castVote(uint64 vote_round, Vote[] calldata vote_data) external;

    /**
     * @dev read the vote data of an account
     * @param addr The address of the account to read
     */
    function readVote(address addr) external view returns (Vote[] memory);

    /**
     * @dev Current vote round
     */
    function currentRound() external view returns (uint64);
```

```
    /**
     * @dev read the total votes of given round
     * @param vote_round The vote number
     */
    function totalVotes(uint64 vote_round) external view returns (Vote[] memory);

    /**
     * @dev read the PoS stake for the round.
     */
    function posStakeForVotes(uint64) external view returns (uint256);

    event CastVote(uint64 indexed vote_round, address indexed addr, uint16 indexed
topic_index, uint256[3] votes);
    event RevokeVote(uint64 indexed vote_round, address indexed addr, uint16 indexed
topic_index, uint256[3] votes);
}
```

Misc

Tx Pool GC Rule

After receiving a valid new transaction from an account:

1. If there is an old transaction from the same sender with the same nonce, replace the old transaction if any of the conditions holds:
   - the `epoch_height` of the new transaction is 200000 more than the old one
   - the new transaction has a higher gas price
   - the new transaction has the same gas price and a higher `epoch_height`
2. If the transaction pool is full, try to garbage collect an old transaction:
   1. Try to garbage collect a transaction that has been packed (packed but not executed transactions will be inserted back to the tx pool later). This is the normal case when we receive a new transaction.
   2. If all transactions have not been packed, try to garbage a transaction that is not ready to be packed (with a non-consecutive nonce or not enough balance).
   3. If all transactions can be packed, choose a transaction (if an account has many ready transaction with consecutive nonces, only the one with the least nonce is considered here) with the least gas price. If the gas price is less than that of the newly received transaction, garbage collect this chosen old transaction.
3. If the transaction pool is not full now, insert the new transaction. Otherwise, discard the new transaction.

Misc

cfx_estimate

1. `from` `balance`
2. `gas_price` Sponsor Gas Upper Bound
3. `nonce` `nonce`nce. nonce nonce
4. `value` 0
5. `data`
6. `to`

Estimate gas storage collateral, gas storage collateral. “ ” estim

gas

1. `gas` gas `OutOfGas`lion
2. `from` `gas_price`

storage limit

1. `storage_limit` storage limit

Estimate gas storage limit

1. Storage Sponsor storage_limit Conflux

gas storage

1. Gas sponsor gas limit ( )
2. Storage sponsor sponsor sponsor.

Misc

PoS

Conflux PoS Finality

EVM

1. Truffle
2. hardhat
3. Foundry
4. Remix
5. https://tenderly.co/

DevTips

Conflux Tip

DevTips

Week 1 - 7.4

Day1 - CFX

CFX Conflux `Drip` `GDrip`

- Drip `1 CFX = 10^18 Drip`
- Gdrip `1 CFX = 10^9 GDrip`, `1 GDrip = 10^9 Drip`

`1 uCFX = 10^12 Drip` ( ) uCFX

`GDrip` `1-100 GDrip`

Day2 - EpochNumber Tag

Epoch RPC hex number EpochNumber Epoch Tag

- `earliest` Epoch, Epoch 0
- `latest_checkpoint` checkpoint Epoch
- `latest_finalized` PoS finalized Epoch Epoch Epoch
- `latest_confirmed` PoW confirmed Epoch Epoch Epoch , !
- `latest_state` Epoch, RPC epochNumber tag
- `latest_mined` Epoch

Conflux 5 epoch

latest_state - latest_confirmed 40-50 Epoch. latest_state - latest_finalized 400-600 Epoch.

Day3

Conflux `3000w gas`, 1420 CFX

Day4 Transaction

Conflux gas 1500w gas data 200k

Day5 Conflux

Conflux Core 0.5s Epoch 1s 5-10s 40-50s

DevTips

Week2-7.11

Day1 Conflux Core 155

Conflux Core 155

- storageLimit
- epochHeight

(RLP)

status Conflux Core receipt status

- 0 -
- 1 -
- 2 or null -

Day2 Conflux Core

Conflux

- user -
- contract -
- builtin -
- null -

base32 verbose

CFX:`TYPE. USER`:AATD0WZV4F7F6J33KH5E182Z4NSCSP59VYE32H4YZ6

CIP-37

Day3 Gas Explained

Gas https://ethgas.io/cn/index.html

Day4 Conflux Gas

Conflux Gas

Day5 Web3

Library of web3 Alex Phan Web3

DevTips

Week3-7.18

Day1 - Java-solidity

java-conflux-sdk web3j. , java solidity . :
https://github.com/web3j/web3j/blob/master/abi/src/main/java/org/web3j/abi/datatypes/AbiTypes.java#:~:text=public%20static%20Class%3C%3F%20extends%20Type%3E%20getType(String%20type%2C%20boolean%20primitives)

Day2 - Java-conflux-sdk hexstring byte[]

java-conflux-sdk call hexstring byte[]
web3j Numeric https://github.com/web3j/web3j/blob/master/utils/src/main/java/org/web3j/utils/Numeric.java#:~:text=public%20final%20class%20Numeric

```
        String hexstring = "xxxxxx"; //hexstring
        byte[] test = Numeric.hexStringToByteArray(hexstring);

        String tmp = Numeric.toHexString(test);
```

Day3 - account.call client.call

java-conflux-sdk client call nonce account.call

Day4 - sdk rpc_url

sdk client rpc_url rpc_url

URL		
http://127.0.0.1:12539		port toml
https://test.confluxrpc.com		
https://main.confluxrpc.com		
https://test.confluxrpc.org		
https://main.confluxrpc.org		

Day5 - pos

conflux pos pos pos java-conflux-sdk raw

https://wiki.conflux123.xyz/link/32#bkmrk-%E6%9C%89%E4%B8%80%E4%B8%AA%E6%AD%A3%E5%9C%A8%E8%BF%90%E8%A1%8C%E7%9A%84pos%E6%9C%AC%E5%9C%B0%E8%8A%82%E7%82%B9-%E5%9C%A8%E6%9C%AC%E5%9C%B0%E8%B0%83

DevTips

Week4 - 7.25

Day1 - PoW

Conflux `cfx_getConfirmationRiskByHash` **15%** `1e-8` `0.000001%` RPC

> 6 1 10% 6 0.1%

```
// Request
curl -X POST --data
'{"jsonrpc":"2.0","method":"cfx_getConfirmationRiskByHash","params":["0x3912275cf09f8982a69735
a876c14584dae95078762090c5d32fdf0dbec0647c"],"id":1}' -H "Content-Type: application/json"
localhost:12539

// fix64 2**256-1 1e-8
{
  "jsonrpc": "2.0",
  "result": "0x2af31dc4611873bf3f70834acdae9f0f4f534f5d60585a5f1c1a3ced1b",
  "id": 1
}
```

Day2 -

`cfx_getConfirmationRiskByHash` **51%** PoWPoS

RP`cfx_getStatus` `latestFinalized` PoS Epoch Epoch

> PoS 400 Conflux Scan

```
// Request
curl -X POST --data '{"jsonrpc":"2.0","method":"cfx_getStatus","id":1}' -H "Content-Type:
application/json" localhost:12539
```

```
// Result
{
  "jsonrpc": "2.0",
  "result": {
    "bestHash": "0x7bbb518ec0b8671a60e9c98619137b0d52522a9ef9490c9b4c23c30f178312f8",
    "chainId": "0x405",
    "ethereumSpaceChainId": "0x406",
    "networkId": "0x405",
    "epochNumber": "0x2fa1a24",
    "blockNumber": "0x70a7fda",
    "pendingTxNumber": "0x83b",
    "latestCheckpoint": "0x2f91bc0",
    "latestConfirmed": "0x2fa19e1",
    "latestState": "0x2fa1a20",
    "latestFinalized": "0x2fa1854" //    PoS       Epoch
  },
  "id": 1
}
```

Day3 - EpochNumber

`cfx_getStatus` R`EpochNumber` `ConfluxContext` `ConfluxContext` `epochNumber(), posHeight(), finalizedEpochNumber()`

> `ConfluxContext` `CFX:TYPE.BUILTIN:AAEJUAAAAAAAAAAAAAAAAAAAAAAAAAAU5XA6TK73 / CFXTEST:TYPE.BUILTIN:AAEJUAAAAAAAAAAAAAAAAAAAAAAAAAAUV2XPKD3X` abi metadata

```
pragma solidity >=0.4.15;


contract ConfluxContext {
    /*** Query Functions ***/
    /**
     * @dev get the current epoch number
     * @return the current epoch number
     */
    function epochNumber() public view returns (uint256) {}
    /**
     * @dev get the height of the referred PoS block in the last epoch
`    * @return the current PoS block height
```

```
     */
    function posHeight() public view returns (uint256) {}
    /**
     * @dev get the epoch number of the finalized pivot block.
     * @return the finalized epoch number
     */
    function finalizedEpochNumber() public view returns (uint256) {}
}
```

Day4 - ga gasUsed gasLimit

SDK R cfx_estimateGasAndCollateral estimate estimate.gasUsed estimate.gasLimit estimate.gasUsed 63/ tx tx.gasLimit estimate.gasUsed estimate.gasLimit tx.gasLimit estimate.gasUsed 4/3 gasUsed

> gasUsed tx.gasLimit * 1/4 tx.gasLimit gasUsed * 4/3 gasUsed Gas

```
// Request
curl -X POST --data
'{"method":"cfx_estimateGasAndCollateral","id":1,"jsonrpc":"2.0","params":[{"from":"cfx:type.user:aarc9abycue0hhzgyrr53m6cxedgccrmmyybjgh4xg","to":"cfx:type.contract:acc7uawf5ubtnmezvhu9dhc6sghea0403y2dgpyfjp","data":"0x","gasPrice":"0x2540be400","nonce":"0x0"}]}' -H "Content-Type:application/json" localhost:12539

// Result
{
  "jsonrpc": "2.0",
  "result": {
    "gasLimit": "0x6d60", // 21000 * 4/3
    "gasUsed": "0x5208", // 21000
    "storageCollateralized": "0x80"
  },
  "id": 1
}
```

Day5 - EpochHeight

Conflux `EpochHeight` `-100000 < currentEpochNumber - tx.epochHeight < 100000` `EpochHeight` `EpochNumber` `EpochNumber`

```
    100000  Epoch   100000     28
```

Week5 - 8.1

Day1

Conflux Core Space

Conflux Core Space

Conflux Core Space

1. Hex `0x1defad05b632ba2cef7ea20731021657e20a7596`
2. Base32 Hex `CFX:TYPE.USER:AAKPBX01FZM1XP89CB7URF6YVYXH3GGX5E9HZ07B38`

- CIP-37
- Conflux

Day2

Base32 Base32

Base32

Base32

Base32 5bit Base32 `0b00000"(16 "0x00") "a"

0x00~0x1f <==> abcdefghjkmnprstuvwxyz0123456789

Base32

1. Network-prefix : cfx cfxtest

2. Address type : `TYPE. XXX`
3. Payload : Hex base32
4. Checksum : 8

```
CFX: TYPE. USER: AAKPBX01FZM1XP89CB7URF6YVYXH3GGX5E9HZ07B38
|   |        |                                  |--- Checksum
|   |        |--- Payload
|   |---
|---
```

Base32 Hex

Base32 Hex

Base32

Hex conflux Base32

- CIP-37
- Conflux

Day3

eSpace

Conflux v2.0 eSpace Core Space

Metamask eSpace

espce

Image not found or type unknown

- eSpace
-

Day4

Core Space eSpace

Core Space eSpace eSpace

1. Core Space base32 decode bytes
2. bytes keccak
3. 20 bytes hex

CrossSpaceCall eSpace eSpace

- Core Space eSpace

Day5

Core Space

Conflux `AdminControl` `SponsorWhitelistControl` `Staking` `ConfluxContext`, `PoSRegister`, `ConfluxContext` `CALL` `STATICCALL` `CALLCODE` `DELEGATECALL`

Hex

1. AdminControl: 0x0888000000000000000000000000000000000000
2. SponsorWhitelistControl: 0x0888000000000000000000000000000000000001
3. Staking: 0x0888000000000000000000000000000000000002
4. ConfluxContext: 0x0888000000000000000000000000000000000004
5. PoSRegister: 0x0888000000000000000000000000000000000005
6. CrossSpaceCall: 0x0888000000000000000000000000000000000006

Week6- PoS

Day1 PoS

Conflux PoS 15%

1. Conflux Staking CFX PoSRegister PoS
2. PoS PoolCFX

Day2 PoS

- PoS 1
- PoS 256 hash
- PoS = 1000 CFX PoS CFX 1000
- PoS Stake PoS 14
- PoS 300
- PoS 6

Day3 PoS PoW finalized

PoS `pivotDecision` PoW pivot chain hash PoS Revert

PoW `posReference` PoW PoS

PoW finalized

1. cfx_getStatus `latestFinalized` finalized
2. cfx_epochNumbe `latest_finalized` number finalized

Day4 ForceRetire

PoS PoS PoS 1 PoS

hardfork 2.1.0 forceRetire PoS

Day5 PoS

Conflux PoS Conflux 1.3 16.6%

1. 4%
2. (totalCirculate) - - - 0
3. PoS (totalStake)

$$0.04 * \sqrt{totalCirculate/totalStake}$$

PoS PoS PoW PoS

DevTips

Week7-8.15

Day1 CIP23 EIP712

hex EIP712 conflu

- message conf`\x19Conflux Signed Message:\n` `\x19Ethereum Signed Message:\n`
- `EIP712domain` `CIP23domain`
- CIP23domai`chainId`

Day2 CIP23

- :
  encode(message : [8n]) = "\x19Conflux Signed Message:\n" ‖ len(message) ‖ message where len(message) is the non-zero-padded ascii-decimal encoding of the number of bytes in message.
- encode(domainSeparator : [256], message :) = "\x19\x01" ‖ domainSeparator ‖ hashStruct(message) where domainSeparator and hashStruct(message) are defined below.

conflux-sdk sdk java-sdk

```
package conflux.web3j.crypto;


import org.junit.jupiter.api.Test;
import org.web3j.utils.Numeric;


import java.io.IOException;


import static org.junit.jupiter.api.Assertions.assertEquals;


public class SignDataTests {
```

```
    //cfx_signTypedData_v4
    @Test
    public void testSignValidStructure() throws IOException {
        StructuredDataTests t = new StructuredDataTests();

      //  TypedData
        String msg = t.getResource(
                "build/resources/test/"
                        + "structured_data_json_files/ValidStructuredData.json");
      //  msg
        StructuredDataEncoder dataEncoder = new StructuredDataEncoder(msg);
      //        msghash
        org.web3j.crypto.Sign.SignatureData sign =
org.web3j.crypto.Sign.signMessage(dataEncoder.hashStructuredData(), SampleKeys.KEY_PAIR,
false);
        assertEquals(

"0x371ef48d63082d3875fee13b392c5b6a7449aa638921cb9f3d419f5b6a817ba754d085965fb3a041c3b178d3ae3
798ea322ae74cb687dd699b5f6045c7fe47a91c",
                Numeric.toHexString(sign.getR()) + Numeric.toHexStringNoPrefix(sign.getS()) +
Numeric.toHexStringNoPrefix(sign.getV()));
    }

    //personal_sign
    @Test
    public void testSignAnyMessage() throws IOException {
        String message = "v0G9u7huK4mJb2K1";
      // msg          msghash
        org.web3j.crypto.Sign.SignatureData sign =
Sign.signPrefixedMessage(message.getBytes(), SampleKeys.KEY_PAIR);
        assertEquals(

"0xbb0ee8492623f2ef6ed461ea638f8b5060b191a1c8830c93d84245f3fb27e20a755e24ff60fe76482dd4377a0ae
f036937ef88537b2d0fdd834a54e76ecafadc1c",
                Numeric.toHexString(sign.getR()) + Numeric.toHexStringNoPrefix(sign.getS()) +
Numeric.toHexStringNoPrefix(sign.getV()));
    }
}
```

Day3 cip23

cip23 sdk

java-sdk sdk

```
package conflux.web3j.crypto;

import org.junit.jupiter.api.Test;
import org.web3j.crypto.*;
import org.web3j.crypto.Sign;
import org.web3j.utils.Numeric;

import java.io.IOException;
import java.util.Arrays;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class ECRecoverTest {
    private String getAddress() {
        return Numeric.prependHexPrefix(Keys.getAddress(getPubKey()));
    }

    private String getPubKey() {
        return SampleKeys.KEY_PAIR.getPublicKey().toString();
    }



    @Test
    public void testSignAndRecoverMessage() {
        String message = "v0G9u7huK4mJb2K1";

        byte[] msgHash = conflux.web3j.crypto.Sign.getConfluxMessageHash(message.getBytes());

        Sign.SignatureData sign =
conflux.web3j.crypto.Sign.signPrefixedMessage(message.getBytes(), SampleKeys.KEY_PAIR);
□□□//       recover,
        String recoverAddress = conflux.web3j.crypto.Sign.recoverSignature(sign, msgHash,
getAddress());
```

```
        assertEquals(recoverAddress, getAddress());
    }

  // fluent                  recover
    @Test
    public void testRecoverTyped() throws IOException {
        StructuredDataTests t = new StructuredDataTests();
        String msg = t.getResource(
                "build/resources/test/"
                        + "structured_data_json_files/ValidStructuredData.json");
        StructuredDataEncoder dataEncoder = new StructuredDataEncoder(msg);

        //           fluent
        String signature =

"0x371ef48d63082d3875fee13b392c5b6a7449aa638921cb9f3d419f5b6a817ba754d085965fb3a041c3b178d3ae3
798ea322ae74cb687dd699b5f6045c7fe47a91c";

      □ //
        byte[] signatureBytes = Numeric.hexStringToByteArray(signature);
        byte v = signatureBytes[64];
        if (v < 27) {
            v += 27;
        }

        Sign.SignatureData sd =
                new Sign.SignatureData(
                        v,
                        (byte[]) Arrays.copyOfRange(signatureBytes, 0, 32),
                        (byte[]) Arrays.copyOfRange(signatureBytes, 32, 64));
□□□//getAddress()       fluent
        String recoverAddress = conflux.web3j.crypto.Sign.recoverSignature(sd,
dataEncoder.hashStructuredData(), getAddress());
        assertEquals(recoverAddress, getAddress());
    }

  //    testSignAndRecoverMessage()
    @Test
    public void testSignAndRecoverTyped() throws IOException {
        StructuredDataTests t = new StructuredDataTests();
```

```
        String msg = t.getResource(
                "build/resources/test/"
                        + "structured_data_json_files/ValidStructuredData.json");
        StructuredDataEncoder dataEncoder = new StructuredDataEncoder(msg);
        Sign.SignatureData sign = Sign.signMessage(dataEncoder.hashStructuredData(),
SampleKeys.KEY_PAIR, false);

        String recoverAddress = conflux.web3j.crypto.Sign.recoverSignature(sign,
dataEncoder.hashStructuredData(), getAddress());
        assertEquals(recoverAddress, getAddress());
    }
}
```

recover()

```
    public static String recoverSignature(SignatureData sd, byte[] data, String address) {
        String addressRecovered = null;

        // Iterate for each possible key to recover
        for (int i = 0; i < 4; i++) {
            BigInteger publicKey =
                    org.web3j.crypto.Sign.recoverFromSignature(
                            (byte) i,
                            new ECDSASignature(
                                    new BigInteger(1, sd.getR()), new BigInteger(1,
sd.getS())),
                            data);

            if (publicKey != null) {
                addressRecovered =
Numeric.prependHexPrefix(Keys.getAddress(publicKey.toString()));

                if (addressRecovered.equals(address)) {
                    break;
                }
            }
        }

        return addressRecovered;
    }
```

https://github.com/web3j/web3j/blob/7dea3d99c5bdbfcc03aaeaa8575fb0c9a9a771ab/crypto/src/main/java/org/web3j/crypto/Sign.java#:~:text=public%20static%20BigInteger%20recoverFromSignature(int%20recId%2C%20ECDSASignature%20sig%2C%20byte%5B%5D%20message)%20%7B

https://wiki.conflux123.xyz/link/60#bkmrk-%E5%90%88%E7%BA%A6%E9%AA%8C%E7%AD%BE

Day4 fluent

java-sdk test sdk provide sdk recover fluent cfx eth fluent

`personal_sign` console

```
conflux
  .request({
    method: 'personal_sign',
    params: [
 'v0G9u7huK4mJb2K1', '<your_address>']})
```

fluent

```
> conflux
    .request({
      method: 'personal_sign',
      params: [
  'v0G9u7huK4mJb2K1', 'cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49']})
< ▼Promise {<pending>}
    ▶[[Prototype]]: Promise
     [[PromiseState]]: "fulfilled"
     [[PromiseResult]]: "0x7e4216720b40f8d7f2cda70433d4a94f3926635517cd5691c778e38eaed87236759ddf7c53ec55c5463f8e966ebd5a32c2dbe1061e95afcbe64ef3a8187badcb00"
```

`cfx_signTypedData_v4` console

```
conflux
  .request({
    method: 'cfx_signTypedData_v4',
    params: [
 '<your_address>',
       `{
    "types": {
        "CIP23Domain": [
            {
                "name": "name",
                "type": "string"
            },
            {
                "name": "version",
                "type": "string"
            },
            {
                "name": "chainId",
                "type": "uint256"
            },
            {
                "name": "verifyingContract",
                "type": "address"
            }
        ],
        "Person": [
            {
                "name": "name",
                "type": "string"
            },
            {
```

```
                "name": "wallet",
                "type": "address"
            }
        ],
        "Mail": [
            {
                "name": "from",
                "type": "Person"
            },
            {
                "name": "to",
                "type": "Person"
            },
            {
                "name": "contents",
                "type": "string"
            }
        ]
    },
    "primaryType": "Mail",
    "domain": {
        "name": "Ether Mail",
        "version": "1",
        "chainId": 1,
        "verifyingContract": "0xCcCCccccCCCCcCCCCCCcCcCccCcCCCcCccccccccC"
    },
    "message": {
        "from": {
            "name": "Cow",
            "wallet": "0xCD2a3d9F938E13CD947Ec05AbC7FE734Df8DD826"
        },
        "to": {
            "name": "Bob",
            "wallet": "0xbBbBBBBbbBBBbbbBbbBbbbbBBbBbbbbBbBbbBBbB"
        },
        "contents": "Hello, Bob!"
    }
}`]
  })
```

message

Day5 cip23 sdk

java-sdk js-sdk cip23 java-sdk cip23 js-sdk cip23

go-sdk python-sdk cip23

DevTips

Week8-8.21

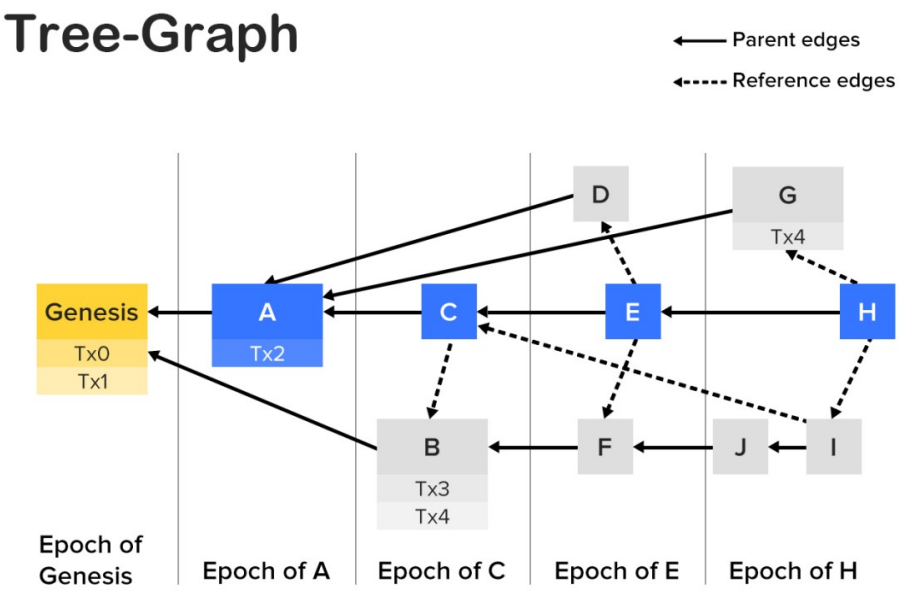
8.22 Epoch Number, Block Number Height Epoch Height

Conflux `epoch number` `block number` `height` `epoch height`

1. Epoch Number Con `epoch` `epoch` pivot `epoch` `epoch number` `epoch number`
2. Block Number `epoch` `block number` `block number` `block numebr`

> `epoch number` `block number`

3. Height `height` `height` `epoch number` t C D G Height
4. Epoch Height `epoch height` `epoch number` EpochHeight



8.23

Conflux	Collateral for storage	
Conflux	64	1/16 CFX

Conflux

8.24

Conflux `setSponsorForCollateral(address contractAddr)` CFX CFX

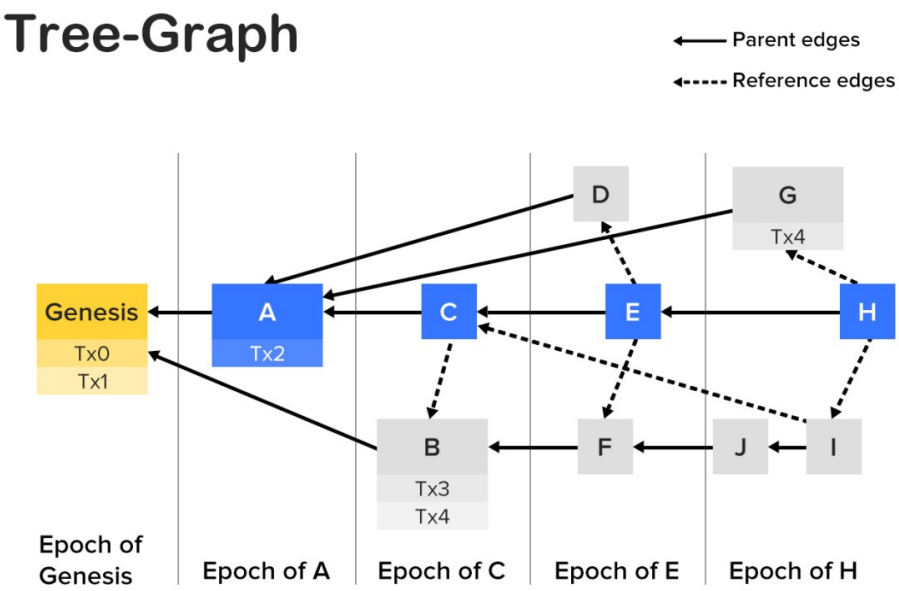
8.25 Staking CFX

Conflux Staking CFX CFX CFX

1. POSPOS
2. CIP-94 On-chain DAO Vote for Chain Parameters

8.26 Pivot RPC

Pivot Conflux Conflux RPC Pivot



1. `cfx_getStatus` `cfx_getBestBlockHash` `cfx_getStatus` `bestHash` epoch Pivot `cfx_getBestBlockHash`

```
// getStatus
{
  "jsonrpc": "2.0",
  "result": {
    "bestHash": "0xe4bf02ad95ad5452c7676d3dfc2e57fde2a70806c2e68231c58c77cdda5b7c6c",
```

```
    "chainId": "0x1",
    "networkId": "0x1",
    "blockNumber": "0x1a80325",
    "epochNumber": "0xaf28ab",
    "latestCheckpoint": "0xada520",
    "latestConfirmed": "0xaf2885",
    "latestState": "0xaf28a7",
    "latestFinalized": "0x2a420c",
    "ethereumSpaceChainId": "0x22b9",
    "pendingTxNumber": "0x0"
  },
  "id": 1
}
```

2. `cfx_getBlocksByEpoch` RPC Epoch blockNumber Pivot .

```
// Result
{
    "jsonrpc": "2.0",
    "result": [
        "0x618e813ed93f1020bab13a1ab77e1550da6c89d9c69de837033512e91ac46bd0",
        "0x0f6ac81dcbc612e72e0019681bcec32254a34bd29a6bbab91e5e8dc37ecb64d5",
        "0xad3238c00456adfbf847d251b004c1e306fe637227bb1b9917d77bd5b207af68",
        "0x0f92c2e796be7b016d8b74c6c270fb1851e47fabaca3e464d407544286d6cd34",
        "0x5bcc2b8d2493797fcadf7b80228ef5b713eb9ff65f7cdd86562db629d0caf721",
        "0x7fcdc6fff506b19a2bd72cd3430310915f19a59b046759bb790ba4eeb95e9956",
        "0xf4f33ed08e1c625f4dde608eeb92991d77fff26122bab28a6b3a2037511dcc83",
        "0xa3762adc7f066d5cb62c683c2655be3bc3405ff1397f77d2e1dbeff2d8522e00",
        "0xba7588476a5ec7e0ade00f060180cadb7430fd1be48940414baac48c0d39556d",
        "0xe4dc4541d07118b598b2ec67bbdaa219eb1d649471fe7b5667a0001d83b1e9b6",
        "0x93a15564544c57d6cb68dbdf60133b318a94439e1f0a9ccb331b0f5a0aaf8049" // pivot
    ],
    "id": 1
}
```

3. `cfx_getBlockByHashWithPivotAssumption` RPC `cfx_getBlockByHash` RPC
1 blockHash 2 assumedPivotHash 3 epochNumber blockHash RPC

DevTips

Week9 - 8.29

Day1 conflux

openzepplin conflux-contracts conflux

- library
- utils
- token

Day2 cfx

conflux erc18 `0x1820a4B7618BdE71Dce8cdc73aAB6C95905faD24` conflux core space `0x88887eD889e776bCBe2f0f9932EcFaBcDfCd1820` conflux-contracts

Day3

dapp rpc rpc node multicall

multicall static call abi

`cfxtest:aanpu16mtgc7dke5xhuktyfyef8f00pz8a2z5mc14g`
`cfxtest:aaskvgxcfej371g4ecepx9an78ngrke5ay9f8jtbgg` `usdt`

- data
  - `balanceOf("cfxtest:aanpu16mtgc7dke5xhuktyfyef8f00pz8a2z5mc14g")` data `0x70a08231000000000000000000000000016c85f8a7985d1a49b99e097d0b4217c5b5995f0`
  - `balanceOf("cfxtest:aaskvgxcfej371g4ecepx9an78ngrke5ay9f8jtbgg")` data `0x70a08231000000000000000000000001c989a6229119edcda2088c9fc0bef9666a49b05`
- `muticall` `function aggregate(address target, bytes[] memory calldatas) public view virtual returns (bytes[] memory)`

```
results =
muticall.aggregate(usdt_address,["0x70a08231000000000000000000000000016c85f8a7985d1a49b99e097d0
b4217c5b5995f0","0x70a08231000000000000000000000000001c989a6229119edcda2088c9fc0bef9666a49b05"])
```

- abi

> ABI SDK sdk

Day4 Fluent - Fluent

1. Fluent

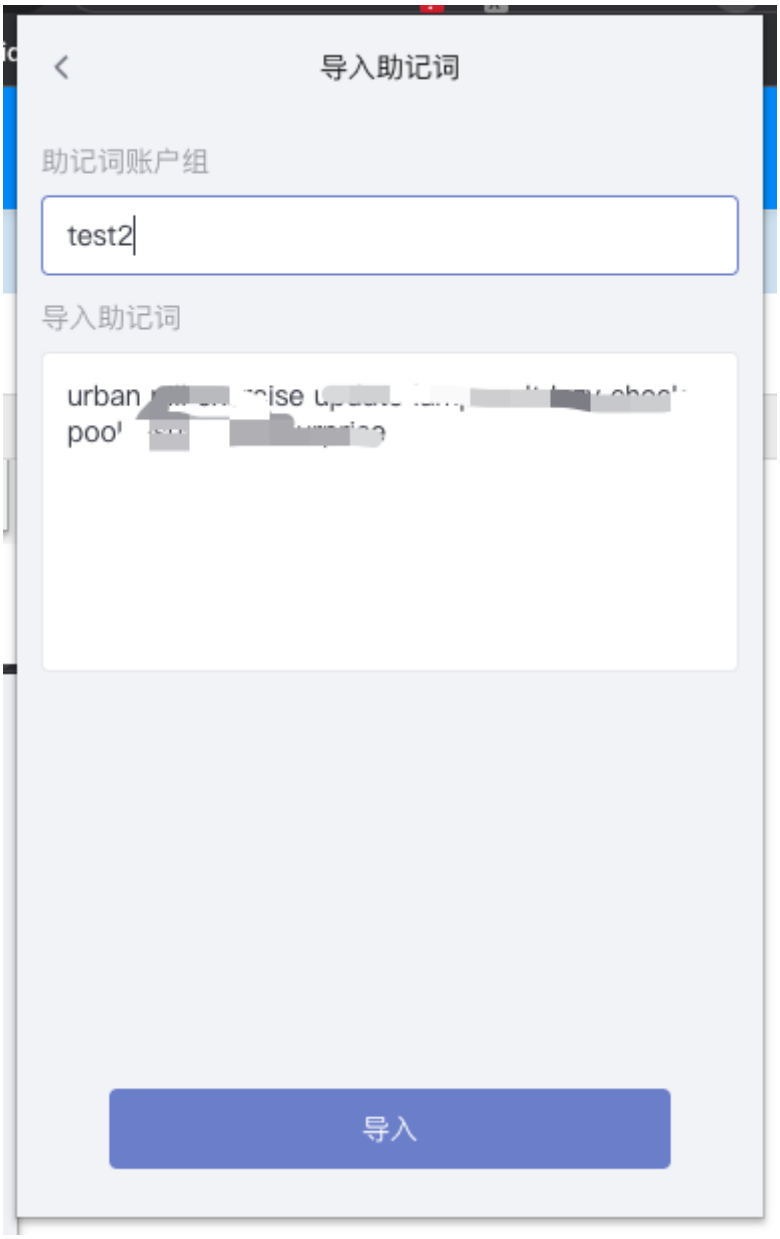
Fluent


Fluent Wallet
test1
Conflux Testnet
cfxtest:aam...gzz8
发送
历史记录
资产
CFX
Conflux
354.924106
未连接至当前网站

"+"


Fluent Wallet
test1
Conflux Testnet
cfxtest:aam...gzz8
我的账户
搜索
test1
354.924106 CFX


新增账户
创建账户
新的助记词
生成一组新的助记词并需备份
导入账户
导入助记词
助记词是一组由空格分隔的词组
导入私钥
填写明文私钥
硬件钱包
连接你的 Ledger 钱包


导入助记词
助记词账户组
test2
导入助记词
urban
pool
导入

Fluent Conflux

Day5 Fluent

1. Fluent conflux ethereum derive path

   - Conflux dervie `m’/44’/60’/0’/0`
   - Ethereum dervie `m’/44’/503’/0’/0`

2. Fluent Fluent

Fluent


返回
账户管理
网络管理
已授权的网站
高级选项
关于
锁定


高级选项
优先连接
在连接至 DApp 时，Fluent 将会作为默认钱包
显示测试网络
开启后，可在网络列表中展示测试网络

DevTips

Week10-NFT

NFT NFT NFT NFT EIP

Day1 - ERC721

ERC-721 NFT , CryptoKit `token ID` `token`

```
interface ERC721 /* is ERC165 */ {
    // events
    event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId);
    event Approval(address indexed _owner, address indexed _approved, uint256 indexed
_tokenId);
    event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);

    function balanceOf(address _owner) external view returns (uint256);
function ownerOf(uint256 _tokenId) external view returns (address);
function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external
payable;
function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;
function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
function approve(address _approved, uint256 _tokenId) external payable;
function setApprovalForAll(address _operator, bool _approved) external;
function getApproved(uint256 _tokenId) external view returns (address);
function isApprovedForAll(address _owner, address _operator) external view returns (bool);
}
```

1. transferFrom
2. safeTransferFrom
3. approve
4. setApprovalForAll

1. balanceOf
2. ownerOf
3. getApproved
4. isApprovedForAll

ERC721

1. metadata
2.
3.
4.

Day2 ERC1155

ERC1155 `ERC20` , token ERC721 tokenId

ERC1155 mint

```
interface ERC1155 /* is ERC165 */ {
    event TransferSingle(address indexed _operator, address indexed _from, address indexed
_to, uint256 _id, uint256 _value);
    event TransferBatch(address indexed _operator, address indexed _from, address indexed _to,
uint256[] _ids, uint256[] _values);
    event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);
    event URI(string _value, uint256 indexed _id);

    function safeTransferFrom(address _from, address _to, uint256 _id, uint256 _value, bytes
calldata _data) external;
    function safeBatchTransferFrom(address _from, address _to, uint256[] calldata _ids,
uint256[] calldata _values, bytes calldata _data) external;
    function balanceOf(address _owner, uint256 _id) external view returns (uint256);
    function balanceOfBatch(address[] calldata _owners, uint256[] calldata _ids) external view
returns (uint256[] memory);
    function setApprovalForAll(address _operator, bool _approved) external;
    function isApprovedForAll(address _owner, address _operator) external view returns (bool);
}
```

Day3 ERC2981

ERC2981 NFT NFT NFT ERC721 ERC1155 token.

`royaltyInfo` tokenId

```
interface IERC2981 is IERC165 {
    function royaltyInfo(
        uint256 _tokenId,
        uint256 _salePrice
    ) external view returns (
        address receiver,
        uint256 royaltyAmount
    );
}
```

Day4 SBT

SBT Token NFT KYC

SBT EIP EIP-4973 EIP-5114 ERC721S Solv Protocol EIP-3525

SBT -

Day5 ERC3525

“ ”

ERC-3525 ERC-1155 -

NFT

DevTips

Week11-9.13

conflux \`ConfluxContext`, `PoSRegister`, `CrossSpaceCall`, `ParamsControl`

Day1 ConfluxContext

`epochNumber`,u`posHeight` `0x0888000000000000000000000000000000000004`.

abi :

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.15;

interface ConfluxContext {
    /*** Query Functions ***/
    /**
     * @dev get the current epoch number
     * @return the current epoch number
     */
    function epochNumber() external view returns (uint256);
    /**
     * @dev get the height of the referred PoS block in the last epoch
`    * @return the current PoS block height
     */
    function posHeight() external view returns (uint256);
    /**
     * @dev get the epoch number of the finalized pivot block.
     * @return the finalized epoch number
     */
    function finalizedEpochNumber() external view returns (uint256);
}
```

java sdk

```
    public static void test(String contractAddr, String caller) throws Exception {
        Cfx cfx = Cfx.create("https://test.confluxrpc.com");
        Account acc = Account.create(cfx, caller);
        Address address = new Address(contractAddr);
        String hash = acc.call(address, "epochNumber");
        cfx.waitForReceipt(hash);
        Optional<Receipt> receipt = cfx.getTransactionReceipt(hash).sendAndGet();
        if (receipt.isPresent()) {
□□□□.....
        } else {
           .......
        }

    }
```

Day2 PoSRegister

PoSRegister PoS

- `register` - pos pos
- `increaseStake` - pos
- `retire` - pos
- `getVotes` - vote
- `identifierToAddress` - pos pow
- `addressToIdentifier` - pow pos

`0x0888000000000000000000000000000000000005`

abi

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0;

interface PoSRegister {
    /**
```

```
     * @dev Register PoS account
     * @param indentifier - PoS account address to register
     * @param votePower - votes count
     * @param blsPubKey - BLS public key
     * @param vrfPubKey - VRF public key
     * @param blsPubKeyProof - BLS public key's proof of legality, used to against some
attack, generated by conflux-rust fullnode
     */
    function register(
        bytes32 indentifier,
        uint64 votePower,
        bytes calldata blsPubKey,
        bytes calldata vrfPubKey,
        bytes[2] calldata blsPubKeyProof
    ) external;

    /**
     * @dev Increase specified number votes for msg.sender
     * @param votePower - count of votes to increase
     */
    function increaseStake(uint64 votePower) external;

    /**
     * @dev Retire specified number votes for msg.sender
     * @param votePower - count of votes to retire
     */
    function retire(uint64 votePower) external;

    /**
     * @dev Query PoS account's lock info. Include "totalStakedVotes" and "totalUnlockedVotes"
     * @param identifier - PoS address
     */
    function getVotes(bytes32 identifier) external view returns (uint256, uint256);

    /**
     * @dev Query the PoW address binding with specified PoS address
     * @param identifier - PoS address
     */
    function identifierToAddress(bytes32 identifier) external view returns (address);
```

```
    /**
     * @dev Query the PoS address binding with specified PoW address
     * @param addr - PoW address
     */
    function addressToIdentifier(address addr) external view returns (bytes32);

    /**
     * @dev Emitted when register method executed successfully
     */
    event Register(bytes32 indexed identifier, bytes blsPubKey, bytes vrfPubKey);

    /**
     * @dev Emitted when increaseStake method executed successfully
     */
    event IncreaseStake(bytes32 indexed identifier, uint64 votePower);

    /**
     * @dev Emitted when retire method executed successfully
     */
    event Retire(bytes32 indexed identifier, uint64 votePower);
}
```

posRegister

java-examples

Day3 CrossSpaceCall

Conflux core space esapce core spa`CrossSpaceCall` space

- `createEVM` - espace
- `transferEVM` - espace
- `callEVM` - espace
- `staticCallEVM` - espace
- `withdrawFromMapped` - espace token
- `mappedBalance` -
- `mappedNonce` - nonce

`0x0888000000000000000000000000000000000006`

abi

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0;

interface CrossSpaceCall {

    event Call(bytes20 indexed sender, bytes20 indexed receiver, uint256 value, uint256 nonce, bytes data);

    event Create(bytes20 indexed sender, bytes20 indexed contract_address, uint256 value, uint256 nonce, bytes init);

    event Withdraw(bytes20 indexed sender, address indexed receiver, uint256 value, uint256 nonce);

    event Outcome(bool success);

    function createEVM(bytes calldata init) external payable returns (bytes20);

    function transferEVM(bytes20 to) external payable returns (bytes memory output);

    function callEVM(bytes20 to, bytes calldata data) external payable returns (bytes memory output);

    function staticCallEVM(bytes20 to, bytes calldata data) external view returns (bytes memory output);

    function withdrawFromMapped(uint256 value) external;

    function mappedBalance(address addr) external view returns (uint256);

    function mappedNonce(address addr) external view returns (uint256);
}
```

CrossSpaceCall

java-examples

core space espace java-sdk

```
    public String getMappedEVMSpaceAddress() {
        String hexAddr =  this.getHexAddress();
        hexAddr = hexAddr.substring(2, hexAddr.length());
        byte[] t = Hash.sha3(Numeric.hexStringToByteArray(hexAddr));

        byte[] mappedBuf = new byte[20];
        System.arraycopy(t, t.length - 20, mappedBuf, 0, 20);

        return Keys.toChecksumAddress("0x" + BaseEncoding.base16().encode(mappedBuf));
    }
```

Day4 ParamsControl

`ParamsControl` CIP-94

- `readVote` -
- `castVote` -
- `readVote` - espace token
- `currentRound` -
- `totalVotes` -

`0x0888000000000000000000000000000000000006`

abi

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.8.0;

interface ParamsControl {
    struct Vote {
        uint16 topic_index;
        uint256[3] votes;
    }

    /*** Query Functions ***/
```

```
    /**
     * @dev cast vote for parameters
     * @param vote_round The round to vote for
     * @param vote_data The list of votes to cast
     */
    function castVote(uint64 vote_round, Vote[] calldata vote_data) external;

    /**
     * @dev read the vote data of an account
     * @param addr The address of the account to read
     */
    function readVote(address addr) external view returns (Vote[] memory);

    /**
     * @dev Current vote round
     */
    function currentRound() external view returns (uint64);

    /**
     * @dev read the total votes of given round
     * @param vote_round The vote number
     */
    function totalVotes(uint64 vote_round) external view returns (Vote[] memory);

    event CastVote(uint64 indexed vote_round, address indexed addr, uint16 indexed
topic_index, uint256[3] votes);
    event RevokeVote(uint64 indexed vote_round, address indexed addr, uint16 indexed
topic_index, uint256[3] votes);
}
```

JS

v2 ~ V1

Week12 Contract address

9.20 Conflux create2

C`create2` Conflux create2Conflux hex

```
#      Web3.py
# from web3 import Web3
from conflux_web3 import Web3

#    salt bytes32
def compute_address_using_salt(salt: bytes, bytecode_hash: bytes, deployer: HexAddress):
    core_part = Web3.solidityKeccak(
        ["bytes1", "address", "bytes32", "bytes32"],
        ["0xff", deployer, salt, bytecode_hash]
    )
    return "0x8"+ core_part.hex()[-39:]
```

9.21 Conflux

Conflux `create2` `nonce` `create2` Confluhex

```
#      Web3.py
# from web3 import Web3
from conflux_web3 import Web3

def compute_address_using_nonce(nonce: int, bytecode_hash: bytes, deployer: HexAddress):
    core_part = Web3.solidityKeccak(
        ["bytes1", "address", "bytes32", "bytes32"],
        ["0x00", deployer, nonce.to_bytes(32, "little"), bytecode_hash]
    )
    return "0x8"+ core_part.hex()[-39:]
```

9.22 Create2Factory

Conflux CIP-31 `Create2Factory` `0x8a3a92281df6497105513b18543fd3b60c778e40` `create2`

ConfluxScan

9.23 Conflux OppenZeppelin Clones, Create2

OppenZeppelin `Clones` `Create2` `minimal proxy` create2 config OppenZeppelin Clones.sol, Create2.sol

DevTips

Week13 - 9.26

Day1

Conflux Core Space (CFS) CFS

= * 1CFX /1024 1KB 1CFX

mint ERC20

```
pragma solidity ^0.8.0;
contract MinimalERC20 {
    mapping(address => uint256) public balances;

    function mint(address account, uint256 amount) public {
        balances[account] += amount;
    }
}
```

a balance 0 bala a: mint(a,10) a balance balances a valu key 10 address uint256 32 64*1CFX/1024=0.0625CFX

Day2 Keystore

`keystore` keystore APP ZIP RAR

keystore 4

- KDF Scrypt 123456 AES-128-CTR S
- S AES-128-CTR
- 2 cyphertext
- cyphertext S SHA3

keystore

- 123456 KDF Scrypt S’
- S = S’
- S’ AES-128-CTR

keystore

```
{
  "version": 3,
  "id": "7d5d99c8-f455-49aa-8b89-6c795a7cdd46",
  "address": "7c52e508c07558c287d5a453475954f6a547ec41",
  "crypto": {
      "kdf": "scrypt",
      "kdfparams": {
          "dklen": 32,
          "salt": "a4f9677eaf6f72394da51e16695899ad3e9b4f2228ad4eca5ef2a5c36093fe12",
          "n": 262144,
          "r": 8,
          "p": 1
      },
      "cipher": "aes-128-ctr",
      "ciphertext": "d89df5ef74f51ae485308e6dce8991dd80674e111f8073f9efa52cb2dd6eca3f",
      "cipherparams": {
          "iv": "6b064c5b09a154d9877d3a07e610a567"
      },
      "mac": "30949eb085ce342a6a488fd51fa5e3231e45f7515efa10c19ea0d46270c73f06"
  }
}
```

JSON keystore App 123456 keystore

keystore

id	uuid
address	keystore
crypto	
kdf	Key Derivation Function , Scrypt
kdfparams	Scrypt
cypher	AES-128-CTR
cyphertext	
cypherparams	AES-128-CTR
mac	cyphertext keystore

go-conflux-sdkjs-conflux-sdkjava-conflux-sdk keystore sdk

1. https://ethbook.abyteahead.com/ch2/keystore.html

Day3 Conflux

Dapp Conflux

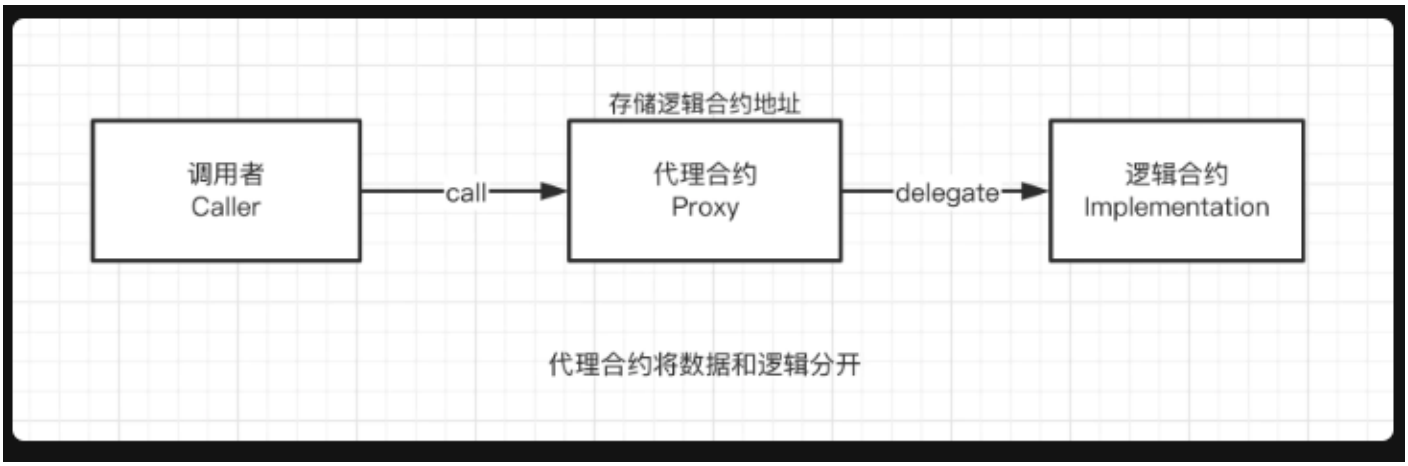
- hardhat-conflux
- Conflux-Truffle
- Conflux ChainIDE
- Conflux studio
- Conflux Studio Web

Tools

Day4

Solidity immutable

-
- bug gas



Proxy delegate

-
- gas

Day5 -EIP1167

EIP1167 gas

openzepplin clone

1. https://eips.ethereum.org/EIPS/eip-1167
2. https://mirror.xyz/xyyme.eth/mmUAYWFLfcHGCEFg8903SweY3Sl-xIACZNDXOJ3twz8

Week14-

Day1 -

- 4C 16G (fullnode 700G archivenode 1.5T)
- Linux
- 100-500G

Day2 -

`Catch-up mode` catch-up mode false

```
cfxcore::syn - Catch-up mode: false, latest epoch: 56218274 missing_bodies: 0
```

Catch-up mode latest epoch

`Statistics: StatisticsInner` `inserted_block_count` `inserted_header_count` `inserted_block_count` `activated_block_count` `processed_block_count`

```
2022-10-11T22:00:32.918557596+08:00 INFO  IO Worker #1         cfxcore::sta - Statistics:
StatisticsInner { sync_graph: SyncGraphStatistics { inserted_block_count: 198117,
inserted_header_count: 371367 }, consensus_graph: ConsensusGraphStatistics {
inserted_block_count: 136125, activated_block_count: 371361, processed_block_count: 371366 } }
```

Day3 -

1.
2.

3. bootnode peer
4. ,
5. pow pos

Day4 -

Err value: PKCS#8 cryptographic error

pos_key , pos_key

failed to start full client: Os { code: 6, kind: Uncategorized, message: "No such device or address" }

Docker daemon `dev_pos_private_key_encryption_password` “

Day5 - pos_key

p`pos_config/pos_key`, `pos_db/secure_storage.json`

DevTips

Week15 - 10.17

java-sdk ~

```
    public static void pubsub() throws ConnectException {
        WebSocketService wsService = new WebSocketService("wss://test.confluxrpc.com/ws",
false);
        wsService.connect();
        Cfx cfx = Cfx.create(wsService);

        // add the filter
        BigInteger cur = cfx.getEpochNumber().sendAndGet();
        // construct the filter parameter
        LogFilter filter = new LogFilter();

        // filter details
//        filter.setFromEpoch(Epoch.numberOf(cur));
//        filter.setToEpoch(Epoch.numberOf(cur.add(new BigInteger("20"))));
//        // To filter address
//        List<Address> toFilterAddress = new ArrayList<Address>();
//        toFilterAddress.add(new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"));
//        filter.setAddress(toFilterAddress);

        // subscribe epoch events
//             Flowable<EpochNotification> events1 = cfx.subscribeEpochs();
//             Disposable disposable1 = events1.subscribe(event -> {
//                // You can get the detail through getters
//                  System.out.println(event.getParams().getResult().getEpochNumber());
//                 System.out.println("epoch");
//            });
//
//            disposable1.dispose();
```

```
        // subscribe newheads events
        cfx.subscribeNewHeads().subscribe(event -> {
            // You can get the detail through getters
            System.out.println(event.getParams().getResult().getEpochNumber());
        }, error -> {
            error.printStackTrace();
        });

        // subscribe log events
//        cfx.subscribeLogs(filter).subscribe(event -> {
//            // You can get the detail through getters
//            System.out.println(event.getParams().getResult().getLogIndex());
//        }, error -> {
//            error.printStackTrace();
//        });
    }
```

conflux-java sdk web3j batch request Batch RPC tip

```
    public static void test() throws Exception{
        Cfx t = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));
        Account acc = Account.create(t, "fjkaldsdjfasxjvzlkxjczxlkjfas"); //replace with your
private key or to export the account from the keystore.
        Account.Option option = new Account.Option();
        RawTransaction tx = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"), acc.getPoolNonce(), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);
        RawTransaction tx1 = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.ONE), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);
        RawTransaction tx2 = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.TWO), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);
```

```
        RawTransaction tx3 = option.buildTx(t, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.valueOf(3)), new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd"), null);

        tx.setValue(BigInteger.valueOf(100));
        String signedTx = acc.sign(tx);
        String signedTx1 = acc.sign(tx1);
        String signedTx2 = acc.sign(tx2);
        String signedTx3 = acc.sign(tx3);

        BatchResponse resp = client.newBatch()
                .add(t.sendRawTransaction(signedTx))
                .add(t.sendRawTransaction(signedTx1))
                .add(t.sendRawTransaction(signedTx2))
                .add(t.sendRawTransaction(signedTx3))
                .send();

        System.out.println(resp.getResponses().get(0).getResult());
    }
```

web3j client -> rawtransaction -> -> web3jclient batch ->

java-sdk nonce

web3j client -> rawtransaction -> -> web3jclient batch

ERC20

```
    public static void batchTx() throws Exception {
        String addr = "cfxtest:acffj2hwbrwbsxuk56jne9913xvmwj5g4u7syhbfr2";
        Cfx cfx = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));
        Account acc = Account.create(cfx, "fjkaldsdjfasxjvzlkxjczxlkjfas"); //replace with
your own private key.
        BigInteger amount = BigInteger.valueOf(100);
```

```
        String data = call(new Address(addr), "transfer", new
Address("cfxtest:aar9up0wsbgtw7f0g5tyc4hbwb2wa5wf7emmk94znd").getABIAddress(), new
Uint256(amount));

        Account.Option option = new Account.Option();
        RawTransaction tx = option.buildTx(cfx, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"), acc.getPoolNonce(), new
Address(addr), data);
        RawTransaction tx1 = option.buildTx(cfx, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.ONE), new Address(addr), data);
        RawTransaction tx2 = option.buildTx(cfx, new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49"),
acc.getPoolNonce().add(BigInteger.TWO), new Address(addr), data);

        String signedTx = acc.sign(tx);
        String signedTx1 = acc.sign(tx1);
        String signedTx2 = acc.sign(tx2);



        BatchResponse resp = client.newBatch()
        .add(cfx.sendRawTransaction(signedTx))
        .add(cfx.sendRawTransaction(signedTx1))
        .add(cfx.sendRawTransaction(signedTx2))
        .send();

        System.out.println(resp.getResponses().get(0).getResult());

        }

    public static String call(String method, Type<?>... inputs) throws Exception {

            Function function = new Function(method, Arrays.asList(inputs),
Collections.emptyList());
            String data = FunctionEncoder.encode(function);



        return data ;
    }
```

&

call reque`DecodeUtil.decode`tch rawdata

```
    public static void queryInfo() throws Exception{
        Cfx t = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));

        BatchResponse resp = client.newBatch()
                .add(t.getBestBlockHash())
                .add(t.getBalance(new
Address("cfxtest:aajb342mw5kzad6pjjkdz0wxx0tr54nfwpbu6yaj49")))
                .add(t.getEpochNumber())
                .send();


System.out.println(Numeric.decodeQuantity(resp.getResponses().get(2).getResult().toString()));
        System.out.println(resp.getResponses().get(0).getResult().toString());

System.out.println(Numeric.decodeQuantity(resp.getResponses().get(1).getResult().toString()));
    }

//  ERC20
    public static void batchQueryContract() throws Exception {
        String addr = "cfxtest:acffj2hwbrwbsxuk56jne9913xvmwj5g4u7syhbfr2";
        Cfx cfx = Cfx.create("https://test.confluxrpc.com");
        Web3j client = Web3j.build(new HttpService("https://test.confluxrpc.com"));
        ContractCall call = new ContractCall(cfx, new Address(addr));


        ContractCall call1 = new ContractCall(cfx, new Address(addr));
        ContractCall call2 = new ContractCall(cfx, new Address(addr));
        ContractCall call3 = new ContractCall(cfx, new Address(addr));

        BatchResponse resp = client.newBatch()
                .add(call.call("name"))
                .add(call1.call("symbol"))
                .add(call2.call("totalSupply"))
```

```
                .add(call3.call("decimals"))
                .send();
        String name = DecodeUtil.decode(resp.getResponses().get(0).getResult().toString(),
Utf8String.class);
        String symbol = DecodeUtil.decode(resp.getResponses().get(1).getResult().toString(),
Utf8String.class);
        BigInteger decimals =
DecodeUtil.decode(resp.getResponses().get(3).getResult().toString(), Uint8.class);
        BigInteger totalSupply =
DecodeUtil.decode(resp.getResponses().get(2).getResult().toString(), Uint256.class);

        System.out.println(name);
        System.out.println(symbol);
        System.out.println(decimals);
        System.out.println(totalSupply);
        }
```

ERC4907

ERC4907 ERC721 ERC721 NFT

NFT ERC4907

ERC4907

ERC4907

User owner owner NFTERC490 owner user

ERC49(user N expire

1. A A NFT
2. A NFT
3. B B NFT A NFT B
4. A NFT

```
interface IERC4907 {

    // Logged when the user of an NFT is changed or expires is changed
    /// @notice Emitted when the `user` of an NFT or the `expires` of the `user` is changed
    /// The zero address for user indicates that there is no user address
    event UpdateUser(uint256 indexed tokenId, address indexed user, uint64 expires);

    /// @notice set the user and expires of an NFT
    /// @dev The zero address indicates there is no user
    /// Throws if `tokenId` is not valid NFT
    /// @param user  The new user of the NFT
    /// @param expires  UNIX timestamp, The new user could use the NFT before expires
    function setUser(uint256 tokenId, address user, uint64 expires) external;

    /// @notice Get the user address of an NFT
    /// @dev The zero address indicates that there is no user or the user is expired
    /// @param tokenId The NFT to get the user address for
    /// @return The user address for this NFT
    function userOf(uint256 tokenId) external view returns(address);

    /// @notice Get the user expires of an NFT
    /// @dev The zero value indicates that there is no user
    /// @param tokenId The NFT to get the user expires for
    /// @return The user expires for this NFT
    function userExpires(uint256 tokenId) external view returns(uint256);
}
```

The userOf(uint256 tokenId) function MAY be implemented as pure or view.

The userExpires(uint256 tokenId) function MAY be implemented as pure or view.

The setUser(uint256 tokenId, address user, uint64 expires) function MAY be implemented as public or external.

The UpdateUser event MUST be emitted when a user address is changed or the user expires is changed.

The supportsInterface method MUST return true when called with 0xad092b5c.

EIP-4907

DevTips

Week16: 10.24 - 10.28 python-sdk

python-conflu `python-conflux-sdk` `web3.py` API `3.8 <= python version <= 3.10`

```
python -m venv venv
source ./venv/bin/activate
pip install conflux-web3
```

10.24 python-conflux-sdk: Base32Address

Conflux Base32 `Base32Address` k pyth `str` `==` `__eq__`) Base32Address

```
>>> from cfx_address import Base32Address
>>> address = Base32Address("0x1ecde7223747601823f7535d7968ba98b4881e09", network_id=1)
'cfxtest:aatp533cg7d0agbd87kz48nj1mpnkca8be1rz695j4'
>>> address_ = Base32Address(address, network_id=1029, verbose=True)
>>> address_
'CFX:TYPE.USER:AATP533CG7D0AGBD87KZ48NJ1MPNKCA8BE7GGP3VPU'
>>> isinstance(address_, str)
True
>>> address_ == 'cfxtest:aatp533cg7d0agbd87kz48nj1mpnkca8be1rz695j4'
True
>>> [
...     address.address_type,
...     address.network_id,
...     address.hex_address,
...     address.verbose_address,
...     address.abbr,
...     address.mapped_evm_space_address,
...     address.eth_checksum_address,
... ]
```

```
['user', 1, '0x1ecde7223747601823f7535d7968ba98b4881e09',
'CFXTEST:TYPE.USER:AATP533CG7D0AGBD87KZ48NJ1MPNKCA8BE1RZ695J4', 'cfxtest:aat...95j4',
'0x349f086998cF4a0C5a00b853a0E93239D81A97f6', '0x1ECdE7223747601823f7535d7968Ba98b4881E09']
```

python-conflux-sdk(conflux-web3) `Base32Address` `str` / `Base32Address`

```
>>> from conflux_web3 import Web3
>>> w3 = Web3(Web3.HTTPProvider("https://test.confluxrpc.com"))
>>> miner = w3.cfx.get_block_by_epoch_number("latest_mined")["miner"]
>>> miner.address_type
'user'
```

10.25 python-conflux-sdk: wallet

`web3.py` `construct_sign_and_send_raw_middleware` `send_transaction`

```
# `web3.py`      `construct_sign_and_send_raw_middleware`
from web3 import Web3, EthereumTesterProvider
w3 = Web3(EthereumTesterProvider)
from web3.middleware import construct_sign_and_send_raw_middleware
from eth_account import Account
acct = Account.create('random string')
w3.middleware_onion.add(construct_sign_and_send_raw_middleware(acct))
w3.eth.default_account = acct.address
legacy_transaction = {
    'to': Account.create('another random string'),
    'value': 22,
    'gasPrice': 123456,  # optional - if not provided, gas_price_strategy (if exists) or
eth_gasPrice is used
}
w3.eth.send_transaction(legacy_transaction)
```

python-conflux-sdk `conflux_web3.middleware` `construct_sign_and_send_raw_middleware`)

```
from conflux_web3 import Web3
from cfx_account import LocalAccount
w3 = Web3(Web3.HTTPProvider("https://test.confluxrpc.com"))
acct:LocalAccount = w3.account.create("random string")

w3.cfx.default_account = acct
# default_account     LocalAccount
```

```
# w3.wallet.add_account(acct)
# w3.cfx.default_account = acct.address

#                from
#   from        w3.cfx.default_account      from
transaction = {
    'to': w3.address.zero_address(network_id=1),
    'value': 22,
    'gasPrice': 10**9,
}
w3.cfx.send_transaction(transaction).executed()
```

```
# wallet        account
assert acct.address in w3.wallet
acct_ = w3.account.from_key("FILL YOUR SECRET KEY HERE")
w3.wallet.add_account(acct_)
assert (acct.address in w3.wallet) and (acct_ in w3.wallet)

transaction = {
    'to': w3.address.zero_address(network_id=1),
    'value': 22,
    'gasPrice': 10**9,
  from': acct_.address
}
w3.cfx.send_transaction(transaction).executed()
```

10.26 python-conflux-sdk:

`web3.py` `wait_for_transaction_receipt` Conflux

1. pending
2. mined
3. executed 5 epoch
4. confirmed PoW revert
5. finalized PoS revert 5-10

python-conflux-sdk `wait_till_transaction_mined`, `wait_till_transaction_executed`, `wait_till_transaction_executed`, `wait_till_transaction_confirmed`, `wait_till_transaction_finalized` API sdk API

```
from conflux_web3 import Web3
w3 = Web3(Web3.HTTPProvider("https://test.confluxrpc.com"))
```

```
acct = w3.account.create("random string")
w3.cfx.default_account = acct

transaction = {
    'to': w3.address.zero_address(network_id=1),
    'value': 22,
    'gasPrice': 10**9,
}
tx_hash = w3.cfx.send_transaction(transaction)

tx_hash.mined()
tx_hash.executed()
tx_hash.confirmed()
tx_hash.finalized()
```

10.27 python-conflux-sdk: contract name

python-conflux-sdk `name` `abi`,`bytecode`,`address`

```
from conflux_web3 import Web3
w3 = Web3(Web3.HTTPProvider("https://test.confluxrpc.com"))
acct = w3.account.create()
w3.default_account = acct

#            CFX
faucet = w3.cfx.contract(name="Faucet")
assert faucet.address == "cfxtest:acejjfa80vj06j2jgtz9pngkv423fhkuxj786kjr61"
faucet.functions.claimCfx().transact().executed()

#  ERC20
erc20_factory = w3.cfx.contract(name="ERC20")
contract_address = (erc20_factory.constructor(name="Coin", symbol="C", initialSupply=10**18)
                    .transact()
                    .executed()["contractCreated"])

#     ERC20
erc20_instance = erc20_factory(contract_address)
erc20_instance.functions.transfer(w3.cfx.address.zero_address(network_id=1),
100).transact().executed()
```

DevTips

Week 17 - 10.31

Day1 - EIP1967

delegatcall

EIP-1967

```
=>
bytes32(uint256(keccak256("eip1967.proxy.implementation") - 1))

event Upgraded(address indexed implementation);

=> beacon
bytes32(uint256(keccak256("eip1967.proxy.beacon") - 1))

event BeaconUpgraded(address indexed beacon);

=> admin
bytes32(uint256(keccak256("eip1967.proxy.admin") - 1))

event AdminChanged(address indexed previousAdmin, address newAdmin);
```

hash slot slot

EIP1967 https://eips.ethereum.org/EIPS/eip-1967

Day2 - ERC1967Proxy

ERC1967Proxy EIP1967 openzepplin EIP1967 SLOT OpenZeppln ERC1967Proxy

FUNCTIONS

```
constructor(_logic, _data)

_implementation()
```

ERC1967UPGRADE

```
_getImplementation()

_upgradeTo(newImplementation)

_upgradeToAndCall(newImplementation, data, forceCall)

_upgradeToAndCallUUPS(newImplementation, data, forceCall)

_getAdmin()

_changeAdmin(newAdmin)

_getBeacon()

_upgradeBeaconToAndCall(newBeacon, data, forceCall)
```

PROXY

```
_delegate(implementation)

_fallback()

fallback()

receive()

_beforeFallback()
```

EVENTS

```
ERC1967UPGRADE
Upgraded(implementation)

AdminChanged(previousAdmin, newAdmin)
```

```
BeaconUpgraded(beacon)
```

Day3 - Transparent Proxy

Openzepplin

Transparent Proxy EIP1967Proxy

1.
2. “admin cannot fallback to proxy t

ProxyAdmin

https://docs.openzeppelin.com/contracts/4.x/api/proxy#transparent_proxy

Day4 - UUPS Proxy

`upgradeTo(address newImpl)`

Box

```
contract Box {
    uint256 private _value;

    function store(uint256 value) public { /*..*/ }

    function retrieve() public view returns (uint256) { /*..*/ }
}

contract BoxProxy {

     function _delegate(address implementation) internal virtual { /*..*/ }

     function getImplementationAddress() public view returns (address) { /*..*/ }
```

```
     fallback() external { /*..*/ }

     // Upgrade logic in Proxy contract
     upgradeTo(address newImpl) external {
         // Changes stored address of implementation of contract
         // at its slot in storage
     }
}
```

UUPS Proxy

UUPS EIP1822 UUPS Transparent Pro

UUPS OpenZeppelin UUPSUpgradea

```
contract Box is UUPSUpgradeable {
    uint256 private _value;

    function store(uint256 value) public { /*..*/ }

    function retrieve() public view returns (uint256) { /*..*/ }

     // Upgrade logic in Implementation contract
     upgradeTo(address newImpl) external {
         // Changes stored address of implementation of contract
         // at its slot in storage
     }
}

contract BoxProxy {

     function _delegate(address implementation) internal virtual { /*..*/ }

     function getImplementationAddress() public view returns (address) { /*..*/ }

     fallback() external { /*..*/ }

}
```

UUPS UUPS

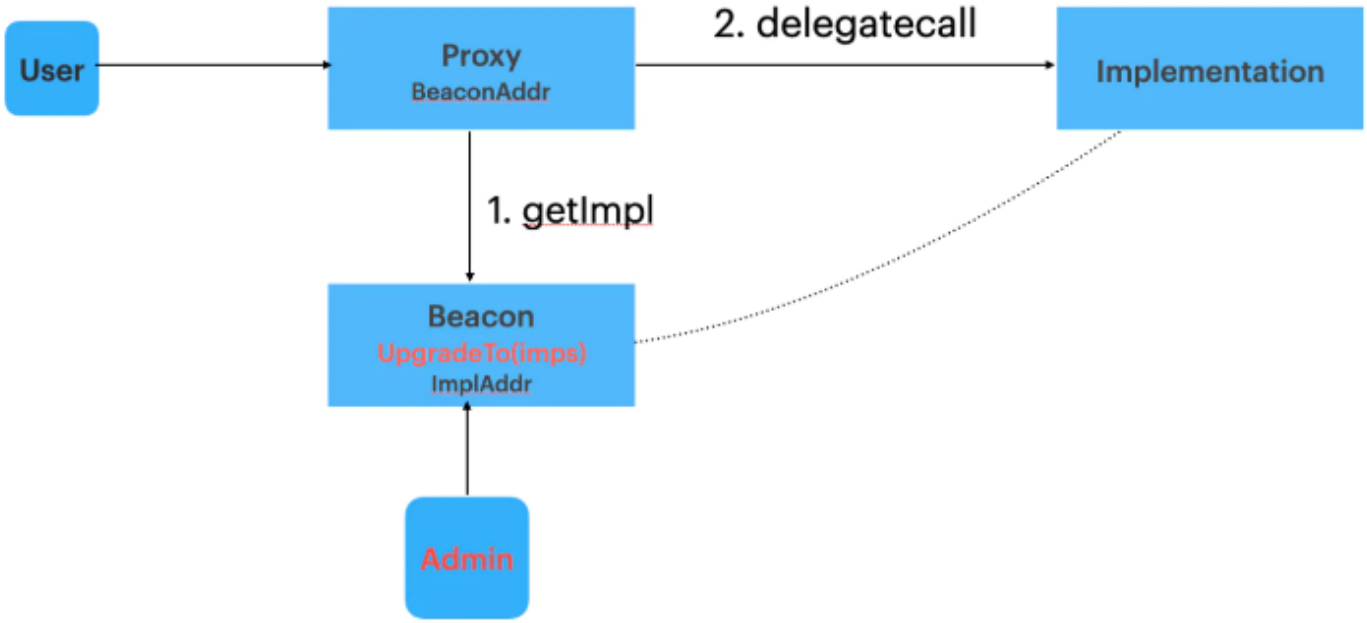
1. https://learnblockchain.cn/article/4257
2. https://docs.openzeppelin.com/contracts/4.x/api/proxy#UUPSUpgradeable

Day5 - Beacon

2 Beacon Implementation Proxy Beacon Proxy Beacon

Proxy Proxy Implementation Beacon Implementation

Proxy Beacon Beacon Implementation



implementation Beacon Beacon Implementation Proxy

1. https://docs.openzeppelin.com/contracts/4.x/api/proxy#beacon
2. https://mirror.xyz/rbtree.eth/qDSQvenBZ_TWqZLUTlxiXVlhKQzPygRyv88EMFWxJro

DevTips

Week18-Conflux

Day1 Shuttleflow

Shuttleflow Conflux Conflux Core EVM ETH BSC, Bitcoin, Huobi Heco

Day2 ConfluxHub Space Bridge

Space Bridge Conflux Space Space bridge, CFX ERC20 Bridge Core C

Day3 ConfluxHub BSC-eSpace Bridge

Bridge CFX BSC eSpace

Day4 multichain

Day5 celer bridge

Day6 meson

meson Core eSpace

DevTips

Week19- 11.14

Day1 Java-Solidity-

The basic types in `solidity` such as the `uint256`, `bytes`, `byte32[]` can be shown in the following.

Solidity Type	web3j.abi type	conflux-Java-sdk Type	Example
uintxxx	Uintxxx	BigInteger	new Uint256(new BigInteger("111"))
address	Address	Address(cfx:xxxx)	new org.web3j.abi.Address(new Address("xxxxx"))
bool	Bool	Boolean	new Bool(true)
string	Utf8String	String	new Utf8String("heyman")
bytesxx	Bytesxx	byte[]	new Bytes32("111".getBytes())
bytes	DynamicBytes	byte[]	new DynamicBytes("111".getBytes())
Static Array (address[2])	StaticArray2		new StaticArray2<>(org.web3j.abi.datatypes.Address.class, xxx)
Dynamic Array (address[])	DynamicArray		new DynamicArray<>(org.web3j.abi.datatypes.Address.class, xxx)

```
struct Foo {
    Address[] addresses;
}
```

```
    public static class BasicAddressesStruct extends DynamicStruct {
        public List<org.web3j.abi.datatypes.Address> addr;

        public BasicAddressesStruct(List<org.web3j.abi.datatypes.Address> addr) {
            super(new
org.web3j.abi.datatypes.DynamicArray<org.web3j.abi.datatypes.Address>(addr));
            this.addr = addr;
        }

        public BasicAddressesStruct(DynamicArray<org.web3j.abi.datatypes.Address> addr) {
            super(addr);
            this.addr = addr.getValue();
        }
    }
```

`bytes[]` , `string` DynamicStruct StaticStruct

Day2 Decode

hexstring hexstring

```
contractCall.callAndGet(Utf8String.class,"text", node, key)
```

java-sdk posRegist `getVotes`

TupleDecoder hexstring decode

```
        BigInteger[] res = new BigInteger[2];

        String rawData = this.call("getVotes", new
Bytes32(Numeric.hexStringToByteArray(identifier))).sendAndGet();
        rawData = Numeric.cleanHexPrefix(rawData);
        TupleDecoder decoder = new TupleDecoder(rawData);
        res[0] = decoder.nextUint256();
```

```
        res[1] = decoder.nextUint256();
```

decoder address bytes decode

web3j FunctionReturnDecoder decode

```
        String rawData = this.call("getVotes", new
Bytes32(Numeric.hexStringToByteArray(identifier))).sendAndGet();

List outputParameters = new ArrayList<TypeReference<Type>>();
outputParameters.add(new TypeReference<Uint256>() {});
outputParameters.add(new TypeReference<Uint256>() {});

List<Type> list = FunctionReturnDecoder.decode(rawData, outputParameters);
```

Day3 decode

```
struct price{
   Uint256 base
   Uint256 premium
}
```

```
    public static class Price extends StaticStruct {
        public BigInteger base;
        public BigInteger premium;

        public Price(Uint256 base, Uint256 premium) {
            super(base,premium);
            this.base = base.getValue();
            this.premium = premium.getValue();
        }
    }
```

price Uint256 StaticStruct DynamicStruct

```
static void decodeStruct() {
  String structData =
"0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000006000000000000000000000000000000000000000000000000000000000000000a000000000000000000000000008f03f1a3f10c05e7cccf75c1fd10168e06659be7000000000000000000000000000000000000000000000000000000000000000568656c6c6f000000000000000000000000000000000000000000000000000000000000";
  TypeReference ts = new TypeReference<Price>() {};
  List<Type> list = DefaultFunctionReturnDecoder.decode(structData, Arrays.asList(ts));
  System.out.println(((Price)list.get(0)).base);
}
```

Day4

Web3j `DynamicArray` `StaticArray`

```
//struct BasicAddresses{
//address[] addr;
//}

String hash = account.call(opt, contract_address, "write", basicAddresses);
```

`MethodSignature` signature methodid encode rawdata

, methodid methodid rawdata

web3j DefaultFunctionEncoder.java encodeFunction

```
    @Override
    public String encodeFunction(final Function function, String signature) {
      String methodId = buildMethodId(signature);
        final StringBuilder result = new StringBuilder(methodId);

        return encodeParameters(parameters, result);
    }
// methodSignature ( ) (xxx) writeBasicAddressesStruct((address[]))

    public static String buildMethodId(final String methodSignature) {
        final byte[] input = methodSignature.getBytes();
        final byte[] hash = Hash.sha3(input);
```

```
        return Numeric.toHexString(hash).substring(0, 10);
    }
```

Function

```
        Function f = new Function(
                "writeBasicAddressesStruct",
                Arrays.asList(struct),
                Collections.emptyList()
        );
```

rawdata

```
String hash = acc.callWithData(new Address(addr), t);
```

Day5 web3j

web3j web3j cc `java.lang.UnsupportedOperationException: Array types must be wrapped in a TypeReference`

https://github.com/web3j/web3j/issues/1726

web3j bug

Week 20 11.28

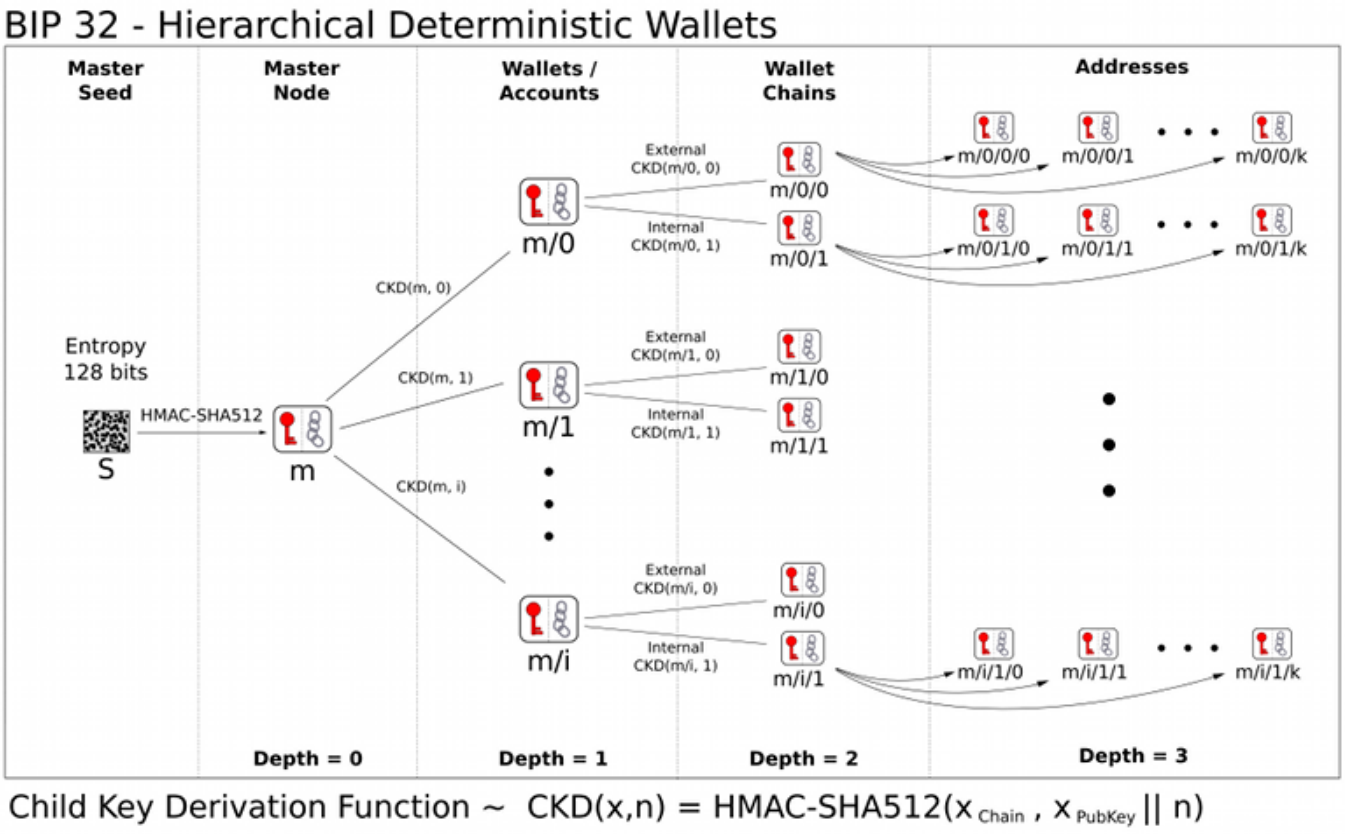
11.28 HD Wallet

BIP-32 HD Wallet / Hierarchical Deterministic BIP-44

```
m / purpose' / coin_type' / account' / change / address_index
```

`coin_type` `coin_type` 0 60 Conflux 503

HD Wallet BIP-32 Master Seed 128bit 512bit



11.29 HD Wallet

HD HD Master Seed “ ” fluent, meta

BIP-39 HD Master Seed PBKDF2 password-based key derivation funciton 2

```
DK = PBKDF2(PRF, Password, Salt, c, dkLen)
```

5 BIP39

- `PRF` BIP-39 HMAC-SHA512
- `Password` PBKDF2 BIP-39 Master Seed utf-8
- `Salt` `mnemonic` `123456` `mnemonic` + `mnemonic123456`
- `c` BIP-39 2048.
- `dkLen` BIP-39 512

HD Wallet Master Seed HD Master Seed

11.30

BIP-39

- 2048
- 12

```
nurse silk fiber machine jelly reduce coffee language fox forum cause team
```

204 `nurse` `1212` `10010111100` 12 (12*11=)132bit 132 128

1. `ENT` bit E `128<=ENT<=256` `os.urandom` python
2. S `ENT` 256 `ENT/32` `ENT=128` 4
3. `3/32*ENT`

12.1 Keystore

Keystore Keystore

```
{
    "version": 3,
    "id": "db029583-f1bd-41cc-aeb5-b2ed5b33227b",
    "address": "1cad0b19bb29d4674531d6f115237e16afce377c",
    "crypto": {
        "ciphertext": "3198706577b0880234ecbb5233012a8ca0495bf2cfa2e45121b4f09434187aba",
        "cipherparams": {"iv": "a9a1f9565fd9831e669e8a9a0ec68818"},
```

```
        "cipher": "aes-128-ctr",
        "kdf": "scrypt",
        "kdfparams": {
            "dklen": 32,
            "salt": "3ce2d51bed702f2f31545be66fa73d1467d24686059776430df9508407b74231",
            "n": 8192,
            "r": 8,
            "p": 1,
        },
        "mac": "cf73832f328f3d5d1e0ec7b0f9c220facf951e8bba86c9f26e706d2df1e34890",
    },
}
```

`ciphertext` `cipher` `cipherparams` `kdf` `kdfparams` keystore

SDK Keystore Keystore

12.2

`random` pyt`random.random` jav`java.util.random` win~~ter~~mute

SDK `/dev/urandom` `/dev/random` Linux SDK

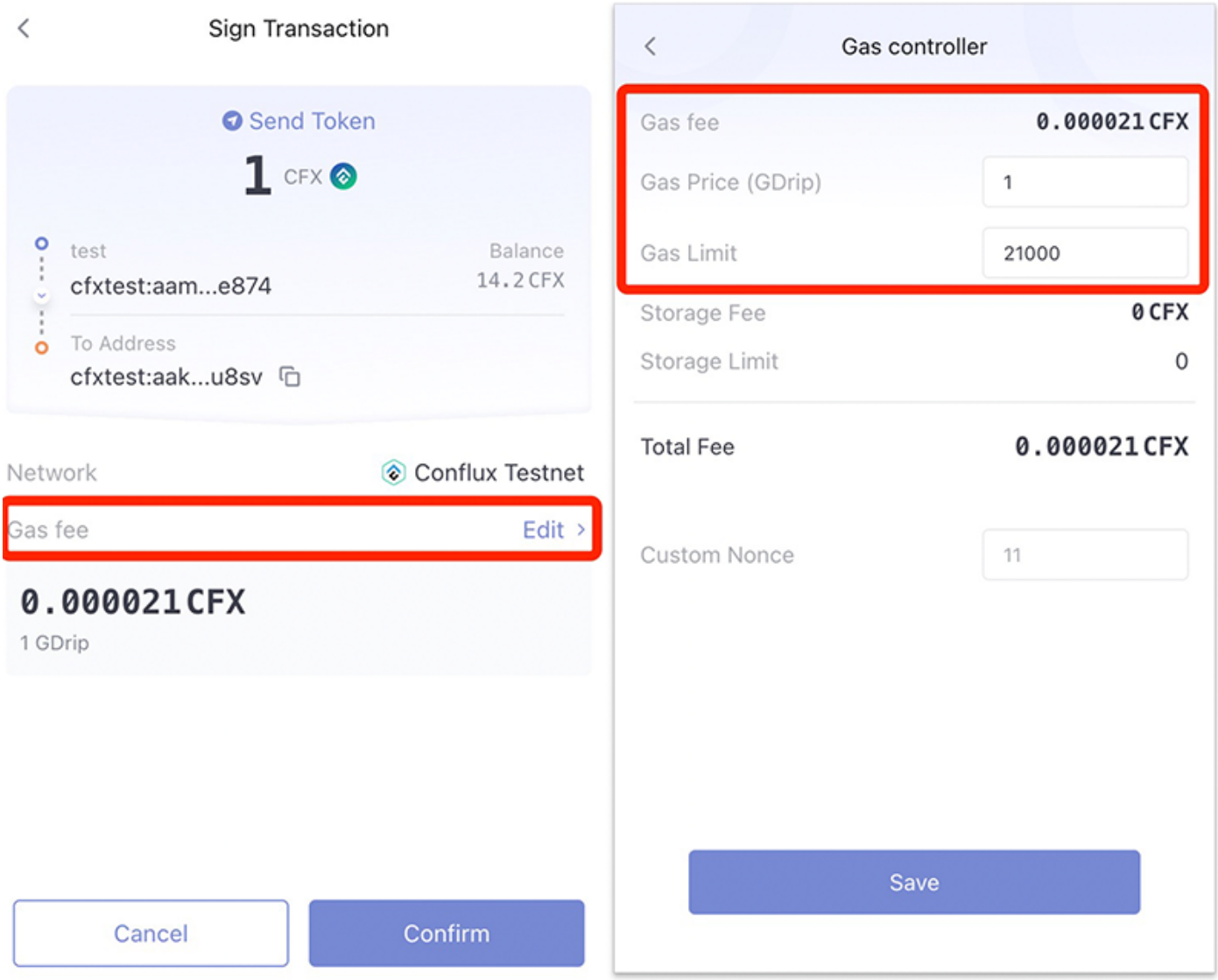
```
extra_key_bytes = text_if_str(to_bytes, extra_entropy)
key_bytes = keccak(os.urandom(32) + extra_key_bytes)
```

`/dev/urandom` urandom u "u`/dev/random` `/dev/urandom` `/dev/urandom` `/dev/random` `/dev/urandom` `/dev/random`

```
with open("/dev/random", 'rb') as f:
    print(f.read(32).hex())
```

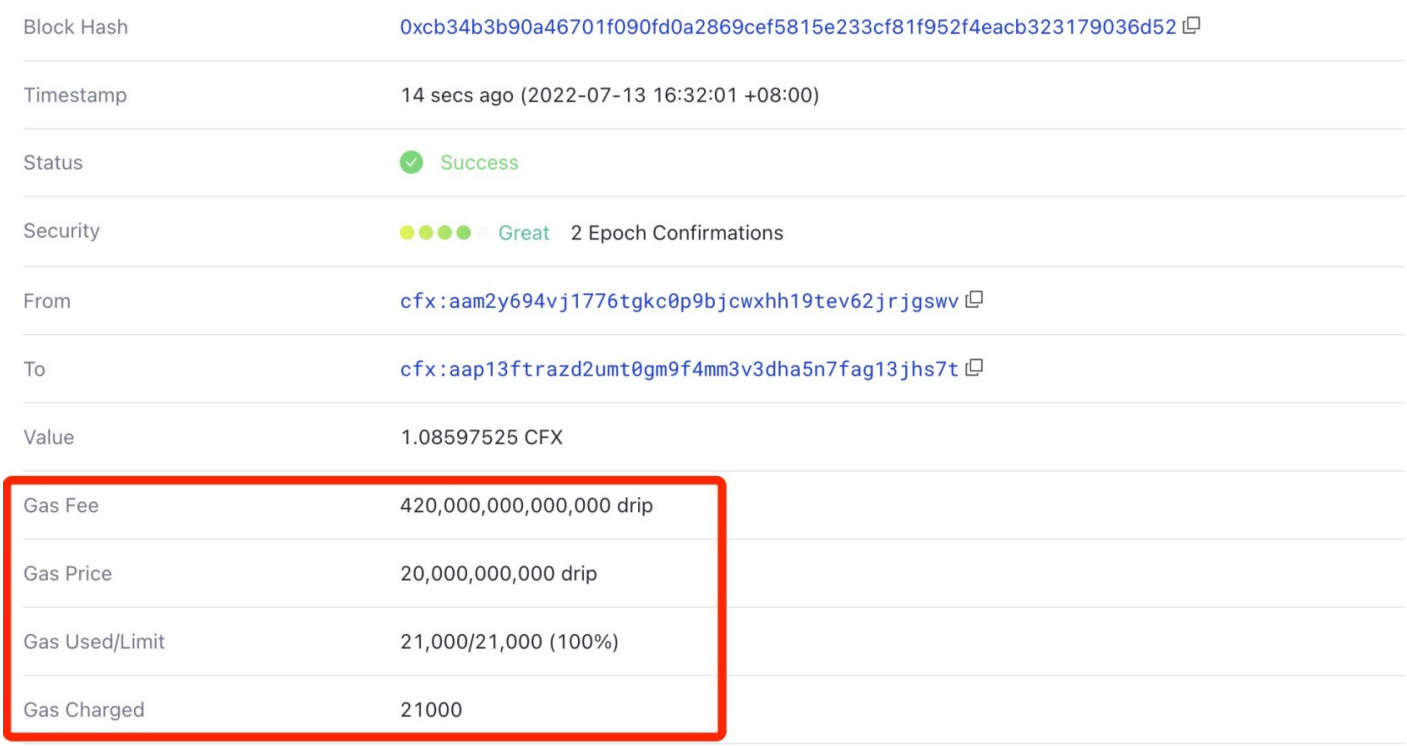
Gas

Conflux Flue`gasFee( ,)gas` `gasPrice` gas gasPrice



gasFee

(Conflux) gasFee ,



( )

gas

gas EVM Gas

Conflux EVM CFX OPCode OPCode OPC

CFX `21000` gas gas

gas

`gas`

gas gas gas

Conflux 1500w, EVM

gasPrice

gasPrice GDrip 1 GDrip 0.000000001 CFX (10^-9 CFX)

gasPrice `1Gdrip` gasPrice gasPrice gasPrice

gasPrice Conflux 10G-1000G gasPrice

NOTE gasPrice gasPrice 1CFX 21000 CFX

gasFee

gasFee `gasFee = gasUsed * gasPrice` gasFee CFX Drip

1CFX gas `1 + 21000 * 0.000000001 = 1.000021 CFX`. 1CFX 0

Confl `x gasUsed`) `gas`

CFX gas 10w `0.000075 CFX`. 000 25000

`0.000021 CFX` 25000, 4000

gas gasPrice

gasPrice

gasPrice 1-10000G 1G gasPrice 10G 100G

gas

CFX gas 21000

`cfx_estimateGasAndCollateral` `gasUsed` gas 63/`gasLimit` `gasUsed` gas
`cfx_estimateGasAndCollateral` `gasLimit` `gasUsed` `4/3` gas

FAQs

1. Conflux 1559 ?

Conflux 155

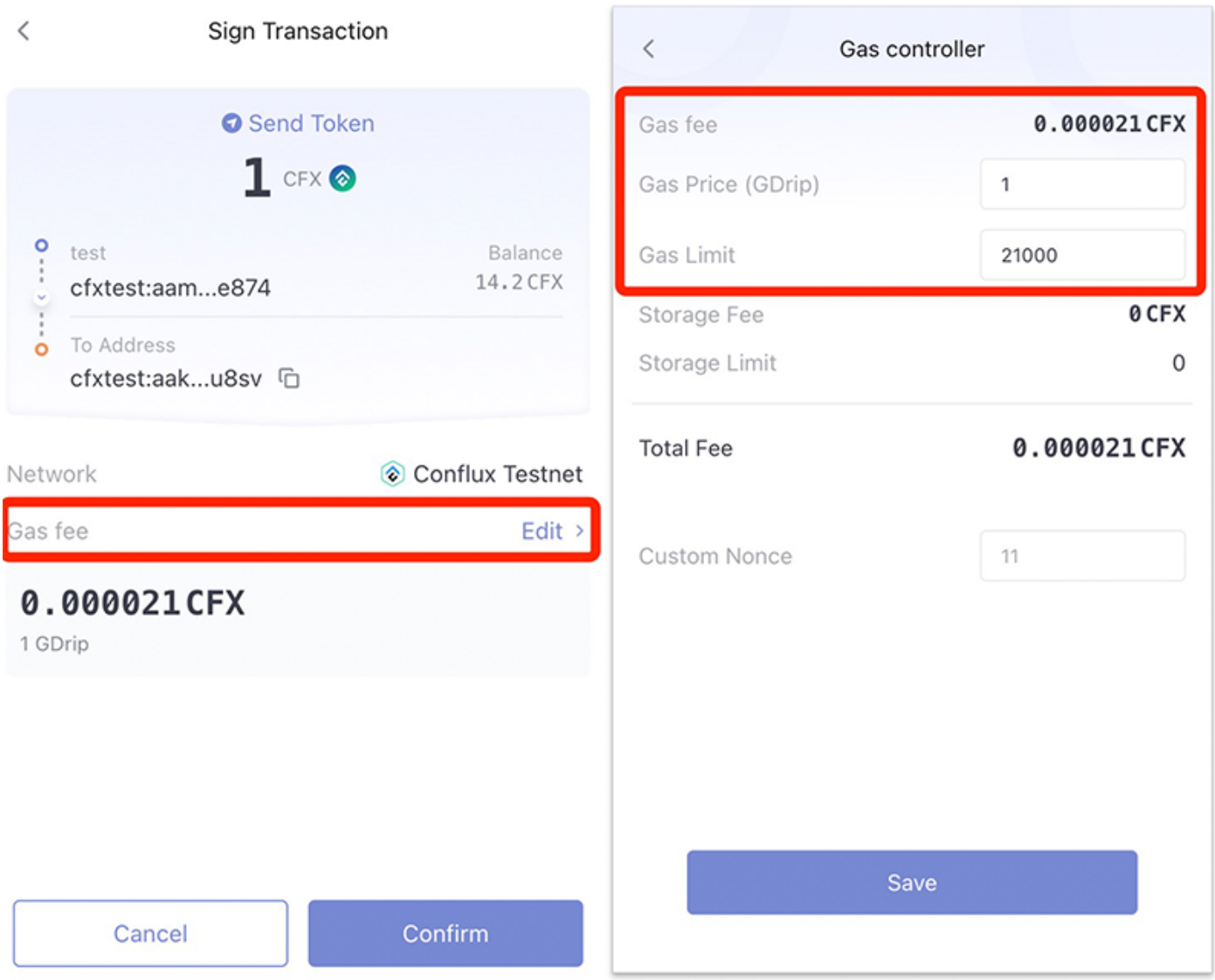
- : Gas and Fees
- Ethereum Gas Explained
- gas

EN Translate

EN Translate

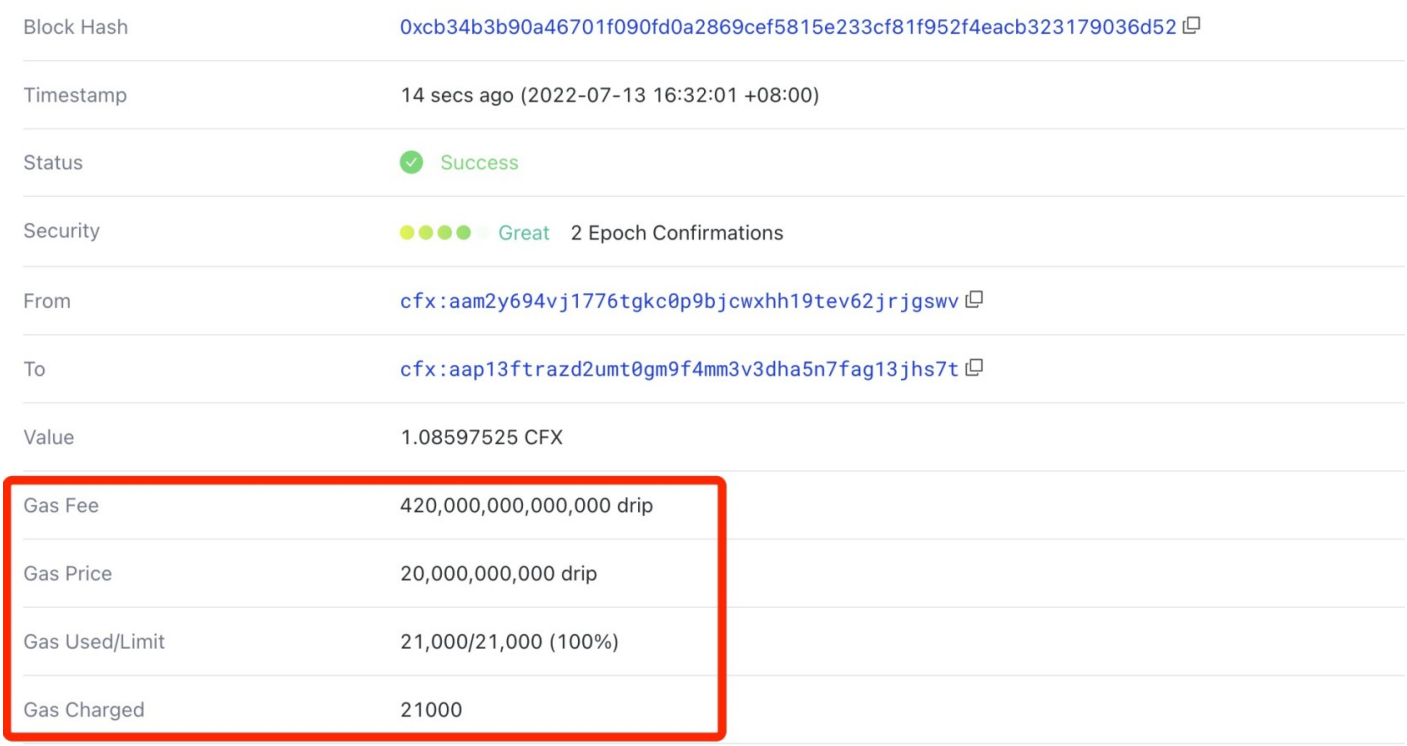
Gas

Conflux users usually see fields like `gasFee`, `gas`, and `gasPrice` when they are sending transactions using their wallets (Fluent) or SDK. This article is going to explain in detail about what these concepts mean and how to set the values properly.



gasFee

In real life, when we send money to someone else at a bank, we usually have to pay transaction fees. It is the same for sending transactions in blockchains (Bitcoin, Ethereum, Conflux). `gasFee` is the fee for sending a transaction. Usually, it needs to be paid by the native token of the chain. Take Conflux as an example, CFX is needed for paying the transaction fee (gas fee).



Why Pay Fees

As we all know, blockchain is in fact a decentralized ledger, which is maintained by miners. There is a cost for miners to store data and generate blocks (calculating hashes). Therefore, in order to motivate miners to actively participate in the chain maintenance and protect the network security, the blockchain consensus system is designed to include a reward mechanism for miners, and the transaction fee is one of the rewards for miners, which will be paid to miners who participate in generating blocks. This mechanism can ensure the sustainability of the entire decentralized network.

In addition, the gas fee mechanism can also prevent abusive transactions and thus improve the efficiency of blockchain utilization.

What is Gas

The concept of gas is borrowed from the real 'gas', petrol. Vehicles consume gasoline to drive. The further a car travels, the more gasoline it consumes. In EVM blockchains, gas represents the work amount required to execute a transaction. Therefore, it is a unit that measures the amount of computation required to perform certain operations.

To provide more details, all Conflux transactions are executed by EVM, including regular CFX transfers and smart contract method calls. When these operations are executed, they are compiled into individual OPCodes. The amount of work required to execute each OPCode varies. More information for OPCode gas fees can be found here.

Usually, the gas consumed for a regular CFX transfer is `21000`. A smart contract transaction usually needs more, depending on the complexity of the contract execution.

Gas Limit

When constructing a transaction, the `gas` field is very important, as the field itself means the `maximum amount of gas` that can be consumed by the execution of the transaction.

It is very important to fill the gas field correctly. If the gas limit is set to a value less than the actual amount of gas needed, the transaction will fail. If the gas limit is set too high, you may pay more gas than you actually need to.

The maximum gas limit for a single transaction in the Conflux network is 15M. This ensures that the transactions will not overconsume EVM resources.

gasPrice

The gasPrice of a transaction is specified by the sender of the transaction and represents the fee that the person is willing to pay per unit of gas. The unit of gasPrice is GDrip, where 1 GDrip is equal to 0.000000001 CFX (10^{-9} CFX).

A transaction's gasPrice value affects how fast the transaction is packaged by miners, as miners will prioritize packaging transactions with higher gasPrice in order to make more profits. When the network is not congested, setting gasPrice to `1Gdrip` is enough to be packed normally. However, when the network is congested, more transactions are waiting to be packed. At this time, if the gasPrice is set to be less than most other transactions, it will not be packed but keep waiting.

Therefore, if you want the transaction to be packaged quickly, you can set the gasPrice higher. Usually setting it to 10G-1000G is high enough in Conflux to ensure it is executed quickly.

NOTE: Do not set gasPrice too high. It may lead to sky-high transaction fees. If gasPrice is set to 1CFX, then the fee for a regular transfer is 21000 CFX, which is quite a lot for a transaction.

How gasFee is Calculated

gasFee is the total gas fee paid for a transaction. It is calculated as `gasFee = gasUsed * gasPrice`. gasFee takes the smallest unit of CFX, Drip.

Suppose there is a regular transfer of 1 CFX, the gas limit can be set to 21,000. If the gasPrice is set to 1GDrip, then the total cost of the transaction is `1 + 21000 * 0.000000001 = 1.000021 CFX`, where 1 CFX is transferred to the recipient's account, and 0.000021 CFX is the reward for the miner.

In addition, in a Conflux transaction, if the `gas limit` is more than the actual amount of gas consumed (`gasUsed`), the exceeding part will be returned. The returning amount of gas can only be up to a quarter of the `gas limit`.

Suppose the gas limit for a regular CFX transfer is set to 100k and the actual execution consumed 21,000, since the gas limit for the transaction is set too high, at most 25,000 of the gas fee will be returned(25% of the gas limit). The gas used for the transaction will be `0.000075 CFX`.

If the gas limit setting of the transaction is not that high, take the same example as above but set the gas limit to 25000, which is 4000 more than the actual amount used, the exceeding part is not more than a quarter of the gas limit. This part will be returned fully, and the final amount of fees charged will still be `0.000021 CFX`.

How to Set gas and gasPrice Properly

gasPrice

Usually, gasPrice can be set within the range of 1G to 10,000G. If the network is not congested, 1G would be enough. If the waiting time after sending the transaction is too long, you can set the gasPrice to 10G or 100G.

gas

For regular CFX transfers, setting the gas to 21,000 is enough.

For contract interactions, you can estimate the gas value by using the method `cfx_estimateGasAndCollateral`, which simulates the execution of the transaction and returns the actual amount of gas used for the transaction. In most cases the value returned by this method is accurate, but sometimes there could be underestimations. A safe way of dealing with this is to multiply the result of the estimation by a factor `1.3`. This ensures that the gas limit is sufficient for the transaction, and any excessive gas fee will be refunded.

FAQs

1. Are there any EIP-1559 transactions in the Conflux network?

Currently, in the Conflux network, there are only transactions that correspond to the EIP-155 standard.

Further Readings

- Ethereum Developer Documentation: Gas and Fees
- Ethereum Gas Explained

Accounts

`Account` is a very important object entity in the Conflux network. It is used to store CFX (every account has its CFX balance) and send Conflux transactions. Accounts and account balances are stored in a huge table in the Conflux VM, and they are part of the full state of the Conflux ledger.

Types of Accounts

Conflux has two types of accounts.

- External accounts (private key accounts) - are controlled by the holder of the private key
- Smart Contracts - are the ones deployed in the network and controlled by the contract codes

Note: There is a special type of smart contract in the Conflux network - the internal contracts. They are created automatically when the network is started or upgraded, but not by deploying contract codes. There are currently 6 internal contracts.

Similarities of Accounts

- Both of them can accept, hold, and send CFX and tokens
- Both of them can interact with smart contracts in the network

Differences of Accounts

External Accounts

- Creating external accounts does not have costs, such as CFX or other resources
- They can send transactions to others
- Transactions between external accounts can only be CFX or token transactions

Smart Contracts

- Creating smart contracts does have costs, as it uses the network's storage and computational resources
- Transactions can only be sent to other contracts as a response to a received transaction
- Transactions sent from external accounts to contract accounts can trigger the smart contract's codes to perform many different operations, such as token transfers, creating new contracts, etc.

External Accounts and Public-private Key Pairs

An external account is generated by a public-private key pair that can be used to prove that a transaction was indeed signed by the account, in order to prevent transaction falsification. The private key is used by the user to sign a transaction. It gives you the right to operate the assets on the account corresponding to the private key. Essentially, the user does not hold the CFX or the tokens but the private key. CFX is always present in Conflux's ledger.

This mechanism prevents malicious users from broadcasting fake transactions, as we can verify the address that sends the transaction at any time.

For example, if Alice wants to send CFX from her account to Bob's account, she needs to create a transaction and send it to the network for verification. Conflux uses the public key encryption mechanism to ensure that Alice can prove that the transaction is sent by herself. Suppose now, Eve, a malicious user, directly broadcasts a transaction, say, "send 5 CFX from Alice's account to Eve's account". Without the above-mentioned mechanism, no one would be able to verify that the transaction was sent by Alice.

External Account Creation

When you want to create an external account, you can use a wallet, like FluentWallet, or any language library, where essentially both of them generate a random private key.

A private key contains 64 hexadecimal characters and can be encrypted using a password.

```
ffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd036415f
```

The public key is calculated from the private key by the Elliptic Curve Cryptography Algorithm. Then, Keccak-256 hashing is performed on the public key, and the Conflux address is yielded by encoding the last 20 bytes (the first 4 bit will be set to 0001) of the result with base 32 formats.

```
// Mainnet address
cfx: aatktb2te25ub7dmyag3p8bbdgr31vrbeackztm2rj
// Testnet address
cfxtest: aatktb2te25ub7dmyag3p8bbdgr31vrbeajcg9pwkc
```

The public key can be calculated from the private key, but the private key cannot be calculated from the public key. The private key has to be kept safe by the user.

Smart Contract Account

Smart contracts also have base32 encoded addresses

```
cfx: acf2rcsh8payyxpg6xj7b0ztswwh81ute60tsw35j7
```

This address is determined when the contract is deployed and is calculated by the deployed transaction's `sender address`, `nonce`, and the smart contract's `code`.

Note: The addresses of internal contracts are special - they are assigned by the network itself.

Details of Accounts

The global state of Conflux is composed of individual account states, each of which is an address-state pair (key pair).

A Conflux account state includes five parts:

- `Basic state` is the basic state of the account.
- `Storage state` is a key/value database or storage space that can be used to store custom states or data of smart contracts.
- `Code information` is the code information of the smart contract account. It includes the `contract codes` and the `address` of the account that paid the fee for the storage space occupied by the codes.
- `Staking deposit list` is the list of Staking operations of the accounts (it will be removed in the next Hardfork).
- `Staking vote lock list` is the list of lock operations performed by the account to participate in DAO voting.

The basic status of the account consists of eight fields as follows:

- `nonce` is a counter to keep track of the number of transactions sent by an account. It is also used to ensure that each transaction can only be executed once. For contract accounts, this value indicates the number of `contracts created by this contract`.
- `balance` is the number of CFX of the address in Drip. Drip is the smallest unit of CFX, where 1CFX=10^{18} Drip.
- `codeHash` is the hash of the code of the contract account. The user can reference the contract code, the code cannot be modified after the contract is created. The code will be executed when the contract receives a message call. For external accounts, codeHash is a hash of an empty string.
- `stakingBalance` is the balance of the staked amount. Similarly, the unit is Drip.
- `admin` is the administrator address of the `contract account` recorded in the AdminControl internal contract. In default, the contract administrator is set to the account which deployed this contract when it is created. The administrator can destroy the contract it

manages through the internal contract AdminControl, or give the administrator role to another account. The administrator address of an external account is itself.

- `sponsorInfo` is the information of the contract sponsor. It contains `sponsor for gas`, `sponsor for collateral`, `sponsor gas limit`, `sponsor balance for gas`, and `sponsor balance for collateral`.
- `storageCollateral` is the amount of Drip staked to use the storage spaces.
- `accumulatedInterestReturn` is the amount of cumulative reward of the account from Staking. The unit of it is Drip. Starting with Conflux 2.0, users must also participate in PoS in order to receive the reward.

For more details about accounts, please refer to the `Accounts` section in Conflux Protocol Specification.

Accounts

Conflux CFX (CFX) Conflux Conflux VM

Conflux

- -
- -

Conflux -- , Conflux 6

- CFX tokens
-

- CFX
-
- CFX tokens

-
-
- tokens

CFX t

Alice CFX Bob Alice Conflux Alice

FluentWallet

64 hex

```
ffffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd036415f
```

Keccak-256 20 base32 Conflux

```
//
cfx: aatktb2te25ub7dmyag3p8bbdgr31vrbeackztm2rj
//
cfxtest: aatktb2te25ub7dmyag3p8bbdgr31vrbeajcg9pwkc
```

base32

```
cfx: acf2rcsh8payyxpg6xj7b0ztswwh81ute60tsw35j7
```

`nonce`

Conflux - key pair

Conflux

-
- key/value
-
- `Staking deposit` Staking Hardfork
- `Staking vote lock` DAO

- `nonce`

- `balance` Drip Drip CFX 1CFX=1e+18 Drip
- `codeHash` hash message call codeHash
- `stakingBalance` Staking Drip
- `admin` AdminControl AdminControl
- `sponsorInfo` `sponsor for gas`, `sponsor for collateral`, `sponsor gas limit`, `sponsor balance for gas` `sponsor balance for collateral`
- `storageCollateral` Drip
- `accumulatedInterestReturn` Staking Drip Conflux 2.0 PoS

Conflux Accounts

Token Standard

ERC4626-Tokenized Vault Standard

Core Space eSpace

Conflux 2.0 `eSpace` EVM `CrossSpaceCall` CFX Conflux

(mirror address)

CrossSpaceCall

eSpace `0xc655a016f2ebe30f0dbe1a957d439104e05451e9` Core CrossSpaceCal

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

contract Counter {
    uint256 public number;

    function setNumber(uint256 newNumber) public {
        number = newNumber;
    }

    function increment() public {
        number++;
    }
}
```

Core CrossSpaceCall eSpace Counter

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import "@confluxfans/contracts/InternalContracts/CrossSpaceCall.sol";

contract CrossSpaceDemo {
    CrossSpaceCall crossSpaceCall =
```

```
CrossSpaceCall(0x0888000000000000000000000000000000000006);

    address eSpaceCounterAddr = 0xc655A016f2eBE30f0DBe1A957D439104E05451e9;

    function setNumber(uint256 newNumber) public {
        // use abi.encodeWithSignature method to build the method call data
        crossSpaceCall.callEVM(bytes20(eSpaceCounterAddr),
abi.encodeWithSignature("setNumber(uint256 newNumber)", newNumber));
    }

    function getNumber() public returns (uint256) {
        bytes memory num = crossSpaceCall.callEVM(bytes20(eSpaceCounterAddr),
abi.encodeWithSignature("number()"));
        // use abi.decode to decode data
        return abi.decode(num, (uint256));
    }
}
```

ConfluxHub Space Bridge App