

Tutorials

- [Conflux](#)
- [ConfluxScan](#)
- [Conflux](#)
- [PersonalMessage & CIP23](#)
- [MetaMask](#) [Conflux eSpace](#)
- [Conflux](#) [v2.1.0](#)
- [Conflux Core](#)
- [Conflux](#)

Conflux

- :
- <https://developer.confluxnetwork.org>
 - <https://docs.confluxnetwork.org/go-conflux-sdk> sdk
 - <https://zhuanlan.zhihu.com/p/400583419> <https://zhuanlan.zhihu.com/p/393935101> NFT

Image not found or type unknown

, , IoTa (DAG) , .

CFX

CFX conflux token, Ether. conflux , Drip GDrip, uCFX, CFX. :

Image not found or type unknown

CFX

1. -> CFX ,
2. : ,
3. (staking mechanism):

CFX

1. : .
 1. 2CFX
 - 2.
 3. 4% (PoS)
2. : , 4%
3. :

ChainID NetworkID

,

- ChainID 1029, 1
- ChainID (transaction replay attacks)
- NetworkID ChainID

conflux base32 , cip317(,)

:

```
16 0x1386b4185a223ef49592233b69291bbe5a80c527
```

```
base32 cfx: aak2rra2njvd77ezwjvx04kkds9fzagfe6ku8scz91
```

```
common. Address 16 GetHexString , , conflux
```

```
, -> nonce, balance, codeHash( ) :
```

CONFLUX stakingBalance, storageCollateral, accumulatedInterestReturn, admin, sponsorInfo
(https://developer.confluxnetwork.org/introduction/en/conflux_basics)

fields: from, to, nonce, gasPrice, gas, value, chainId, data, v, r, s

confluxEpochHeight storageLimit, , confluxEpoch

- 1.
2. , encode(keccak256), (ECDSA), rlp encode -> hexString -> rawtransaction
3. rpc , rawtransaction
- 4.
5. 5 epoch -> executed, ()
6. 50 epoch -> confirmed, (,) , epoch
7. -> Finalized, ()

ETH

https://developer.confluxnetwork.org/sending-tx/en/transaction_explain#:~:text=no%20details%20provided.-,Differences%20between%20Conflux%20and%20Ethereum,-%23

, storage token, token ;

X B/64 *(1/16)

64 storage entry

, token , token , (,)

AdminControl contract, SponsorWhitelistControl contract, Staking Contract

AdminControl contract

admin, destroy, , admin

:

1. A B, C -> C admin
2. A B, B C, C D admin. -> admin , A C , B
3. 2 , D , , C admin

SponsorWhitelistControl contract

1. sponsor (sponsor)
2. (admin , sponsor)
3. sponsor

Staking Contract

token , storage

: , , 1CFX, 2CFX . (token)

Linux OSX , Windows vs

, <https://developer.confluxnetwork.org/conflux-doc/docs/installation>

```
run

cd run

toml toml

cp hydra.toml development.toml

development.toml
```

```

mode = "dev"
genesis_secrets = "key.txt" ##
dev_block_interval_ms = 250
jsonrpc_http_port=12537 ## jsonrpc_local_http_port=12539 , localhost 12537

```

keystore, github keystore

<https://github.com/conflux-fans/conflux-abigen-example/tree/main/keystore>
<https://wallet.confluxscan.io/login>

go-sdk

```
go get github.com/Conflux-Chain/go-conflux-sdk
```

sdk keystore ,

```

func createAcc(client *conflux.Client){
    cclient, err := conflux.NewClient(local_url, conflux.ClientOption{
        KeystorePath: ".keystore"}) //local_url = "http://127.0.0.1:12537"
    if err != nil {
        panic(err)
    }
    defer client.Close()
    acc1, err := client.AccountManager.Create("test")
    if err != nil {
        log.Fatalln(err)
    }
    fmt.Println(acc1.String())

    acc2, err := client.AccountManager.Create("test")
    if err != nil {
        log.Fatalln(err)
    }
    fmt.Println(acc2.String())
}

func exportPriKeys(client *conflux.Client){
    list := client.GetAccountManager().List()
    priv1, err := client.GetAccountManager().Export(list[0], "test")
    if err != nil {

```

```

    panic(err)
}
priv2, err := client.GetAccountManager().Export(list[1], "test")
if err != nil {
    panic(err)
}
fmt.Println(priv2)
fmt.Println(priv1)
}

```

run key.txt, 1000cfx

```
vim key.txt
```

,

```
sh clear_state.sh
```

```
sh test.sh
```

```

test.sh :
export RUST_BACKTRACE=1
export RUST_BACKTRACE=full
../target/release/conflux --config development.toml

```

sdk

```

func sendTx(client *conflux.Client){
    list := client.GetAccountManager().List()

    var utx types.UnsignedTransaction
    utx.From = &list[0] //use default account if not set
    utx.To = &list[1]

    // unlock account
    err = client.AccountManager.Unlock(acc1, "test")
}

```

```

if err != nil {
    log.Fatal(err)
}
txhash, err := client.SendTransaction(utx)
if err != nil {
    log.Fatal(err)
}
fmt.Println(txhash)
}

```

cfxabigen, <https://docs.confluxnetwork.org/go-conflux-sdk/cfxabigen>

```
git clone https://github.com/Conflux-Chain/go-conflux-sdk.git
```

```
go install ./cmd/cfxabigen
```

ERC20

go

```
cfxabigen --sol token.sol --pkg main --out token.go >> token.go
```

<https://github.com/conflux-fans/conflux-abigen-example/blob/main/token/main.go>

ERC721

ERC20 , <https://zhuanlan.zhihu.com/p/400583419> <https://zhuanlan.zhihu.com/p/393935>

, Openzeppelin

npm

```
sudo apt install npm
```

Openzeppelin

```
npm install @openzeppelin/contracts
```

```
remix          abi,          nft.bin nft.abi ( : ipfs url          ,          )
```

cfxabigen

```
cfxabigen --abi nft.abi --bin nft.bin --pkg main --out nft.go >> nft.go
```

nft.go sdk

```
func deployNFT(client *conflux.Client){
    err := client.AccountManager.UnlockDefault("test1")
    if err != nil {
        log.Fatal(err)
    }

    //tx, hash, t, err := DeployMain(nil, client)

    tx, hash, _, err := DeployMain(nil, client)
    if err != nil {
        panic(err)
    }
    fmt.Println(tx)
    fmt.Println(hash)

    receipt, err := client.WaitForTransactionReceipt(*hash, time.Second)
    if err != nil {
        panic(err)
    }

    logrus.WithFields(logrus.Fields{
        "tx":          tx,
```

```

    "hash":          hash,
    "contract address": receipt.ContractCreated,
  }).Info("deploy token done")
}

```

NFT

```

func mintNFT(client *conflux.Client) {
    acc, _ := client.GetAccountManager().GetDefault()
    err := client.AccountManager.UnlockDefault("test")
    if err != nil {
        panic(err)
    }

    contractAddr := cfxaddress.MustNew("cfx: acdujjy8mrjm3913cnztf0zbgrp5h3fbby7jcw2pc")

    instance, err := NewMain(contractAddr, client)
    if err != nil {
        panic(err)
    }

    err = client.AccountManager.UnlockDefault("test")
    if err != nil {
        panic(err)
    }

    //to := cfxaddress.MustNew("cfx: aap05tb7b9bsxdn5rn365y3peta8pr6mxefp7m682a")

    comacc, _, _ := acc.ToCommon()
    tx, hash, err := instance.Mint(nil, comacc)
    if err != nil {
        panic(err)
    }

    logrus.WithField("tx", tx).WithField("hash", hash).Info("transfer")
    receipt, err := client.WaitForTransationReceipt(*hash, time.Second)
    if err != nil {
        panic(err)
    }
}

```

```
}

```

```
logrus.WithField("transfer receipt", receipt).Info()
}
```

nft

```
func queryNFT(client *conflux.Client){
    acc, _ := client.GetAccountManager().GetDefault()
    err := client.AccountManager.UnlockDefault("test1")
    if err != nil {
        panic(err)
    }

    contractAddr := cfxaddress.MustNew("cfx: acdujjy8mrjm3913cnztf0zbgpx5h3fbby7jcwp2pc")

    instance, err := NewMain(contractAddr, client)
    if err != nil {
        panic(err)
    }

    comacc, _, _ := acc.ToCommon()
    res, err := instance.MainCaller.BalanceOf(nil, comacc)
    if err != nil {
        panic(err)
    }

    res1, _ := instance.MainCaller.Symbol(nil)
    fmt.Println(res)
    fmt.Println(res1)
}
```

nft nft symbol

nft

```

func transferNFT(client *conflux.Client){
    acc, _ := client.GetAccountManager().GetDefault()
    err := client.AccountManager.UnlockDefault("test1")
    if err != nil {
        panic(err)
    }

    contractAddr := cfxaddress.MustNew("cfx: acdujjy8mrjm3913cnztf0zbgp5h3fbby7jcwp2pc")

    instance, err := NewMain(contractAddr, client)
    if err != nil {
        panic(err)
    }

    comacc, _, _ := acc.ToCommon()
    list := client.GetAccountManager().List()
    toacc, _, _ := list[len(list)-1].ToCommon()

    err = client.AccountManager.Unlock(list[len(list)-1], "test")
    if err != nil {
        panic(err)
    }

    id := big.NewInt(0)

    tx, hash, err := instance.TransferFrom(nil, comacc, toacc, id)
    if err != nil {
        panic(err)
    }

    receipt, err := client.WaitForTransationReceipt(*hash, time.Second)
    if err != nil {
        panic(err)
    }

    logrus.WithFields(logrus.Fields{
        "tx": tx,
        "hash": hash,
        "contract address": receipt.ContractCreated,
    }).Info("deploy token done")
}

```

```
res, err := instance.MainCaller.BalanceOf(nil, comacc)
if err != nil {
    panic(err)
}

res1, err := instance.MainCaller.BalanceOf(nil, toacc)
if err != nil {
    panic(err)
}

fmt.Println(res)
fmt.Println(res1)
}
```

0, 1,

ConfluxScan

ConfluxScan Scan

1. Scan Solidity ABI
2. Scan Fluent
3. Scan

Contract

cfx:achcuvuasx3t8zcumtwuf35y51sksewvca0h0hj71a

More

Balance

0

Token

2

Storage Collateral

26.149K

Nonce

1

Contract Name Tag

MOON

Token Tracker

conflux MOON (cMOON)

Contract Creator

cfx:aas_6r0shr6g at txn 0xfd6e...1772

Contract Admin

Zero Address

Storage Sponsor

Sponsor faucet

Gas Fee Sponsor

cfx:aas_6r0shr6g

Transactions

CFX Txns

CRC20 Txns

Analysis

Contract

Code

Read Contract

Write Contract

Contract Source Code Verified

Contract Name: TokenBase

Compiler Version: v0.5.11+commit.c082d0b4

Optimization Enabled: yes with 1000 runs

Other Settings: None

Source Code

```
// Sources flattened with hardhat v2.6.4 https://hardhat.org
// File @openzeppelin/contracts/GSN/Context.sol@v2.4.0

pragma solidity ^0.5.0;

/*
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with GSN meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
contract Context {
    // Empty internal constructor, to prevent people from mistakenly deploying
    // an instance of this contract, which should be used via inheritance.
}
```

Contract

cfx:achcuvuasx3t8zcumtwuf35y51sksewvca0h0hj71a

More

Balance

0

Token

2

Storage Collateral

26.149K

Nonce

1

Contract Name Tag	MOON		Contract Admin	Zero Address	
Token Tracker	conflux MOON (cMOON)		Storage Sponsor	Sponsor faucet	
Contract Creator	cfx:aas...6r0shr6g at txn 0xfd6e...1772		Gas Fee Sponsor	cfx:aas...6r0shr6g	

Transactions CFX Txns CRC20 Txns Analysis Contract

Code

Read Contract

Write Contract

Read Contract Information

Collapse All

1. defaultOperators

2. name

>> string conflux MOON

3. totalSupply

4. SPONSOR

5. decimals

6. _account_list

7. isPauser

8. granularity

9. paused

10. balanceOf

Scan

Contract Tab

Contract Verification

* Contract Address	* Contract Name	* License
<input type="text" value="Please enter a contract address"/>	<input type="text" value="Please enter the contract name"/>	<input type="text" value="Please select a license"/>
* Compiler	* Optimization	* Runs
<input type="text" value="Please select a compiler"/>	<input type="text" value="No"/>	<input type="text" value="0"/>
* Contract Source Code		Upload Contract File
<div></div>		
<input type="button" value="Submit"/>		

Scan

- : Base32
- :
- License:
- Solidity
- :
- : flatten

Flatten Solidity

ConfluxScan flatten Solidity flatten :

- [truffle-flattener](#)
- `npx hardhat flatten`
- [chainIDE flatten](#)

npx hardhat flatten

hardhat merge .

```
$ npx hardhat flatten ./contracts/ERC20Token.sol > ERC20TokenMerged.sol
```

ERC20TokenMerged.sol ConfluxScan

`npx hardhat flatten` [How to Flatten a Solidity file using Hardhat.](#)

chainIDE flatten

ChainIDE

Solidity@chainide/solidity-flattener

Flattener

PLUGIN MANAGER

input plugin URL

LOAD

ACTIVE MODULES

@chainide/solidity-compiler

web3SolidityPlugin

web3SolidityPlugin

ConfluxCoreWallet

use conflux fluent wallet to provide service

ChainIDE Conflux Wallet

ChainIDE Conflux Wallet

ChainIDE Conflux Tools

ChainIDE Conflux Tools

INACTIVE MODULES

EvmWalletPlugin

EvmWalletPlugin

activate

@chainide/solidity-flattener

extensionDescription

activate

ChainIDE Debugger

extensionDescription

activate

ALL PLUGINS

readme.md (preview)

HelloWorld.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract HelloWorld {
5     // Event declaration
6     // Up to 3 parameters can be indexed.
7     // Indexed parameters helps you filter the logs by the indexed parameters
8     event Log(address indexed sender, string message);
9
10    address owner;
11
12    constructor() {
13        owner = msg.sender;
14    }
15
16    function test() external {
17        emit Log(msg.sender, "Hello EVM1!");
18    }
19
20    function get() external view returns(string memory greeting){
21        if(owner == msg.sender) {
22            return greeting = "Hello, world!";
23        }
24    }
25 }
```

Output

Console

Terminal

```
[error] [14:52:49] No wallet connected, please select one at the top to load
[info] [14:53:07] Creation of contract HelloWorld transaction hash: 0xf38594ee710bd12abada1c8a6b1cd4d14428859617948a66e76ab498323a2109
[info] [14:53:07] https://confluxscan.io/transaction/0xf38594ee710bd12abada1c8a6b1cd4d14428859617948a66e76ab498323a2109
[info] [14:53:07] Waiting for confirmation ...
[info] [14:53:16] Creation of contract HelloWorld transaction have confirmed, 0xf38594ee710bd12abada1c8a6b1cd4d14428859617948a66e76ab498323a2109
[info] [14:53:16] upload file [.build/HelloWorld.HelloWorld.cfx:acgxedpz6r99.e
[info] [14:53:18] upload file [.build/HelloWorld.HelloWorld.cfx:acgxedpz6r99.e
```

Flatten

Flatten

Flattener

✓ FLATTEN AND SAVE


Select a contract, compile it, then get the flattened version by pressing the button. Flattened source code will be copied to the clipboard.

Flatten HelloWorld.sol

You can save the flattened version to the file inside ChainIDE.

Save HelloWorld_flat.sol

 Compile

 Deploy & Interaction

 Transaction

 Tools

 Flattener

hardhat-conflux

[EtherScan](#)

[ConfluxScan](#)

[Open](#) [hardhat-etherscan](#)

[EtherScan](#)

[hardhat-conflux](#)

[Hardhat](#)

[Conflux](#)

[verifyCfxContract](#) [hardhat.config.js](#)

```
# npx hardhat verifyCfxContract CONTRACT_NAME DEPLOYED_CONTRACT_ADDRESS
$ npx hardhat verifyCfxContract Greeter cfxtest: acba7cvb1k6bhctzsfshybg5zgch39gnpuc8teem53
```

FAQs

Conflux eSpace

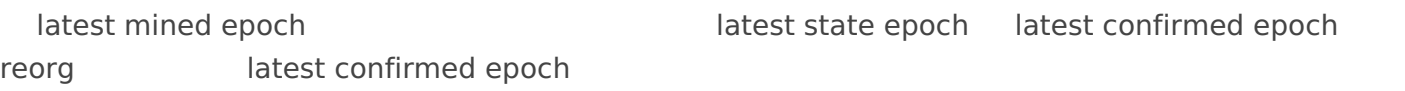
[hardhat-etherscan](#)

[OpenAPI](#)

Conflux



Target



epoch

RPC

epoch

RPC

:

- epoch < target , catch-up mode,
- target 1s

epoch RPC

a. epoch block hash

[cfx_getBlocksByEpoch](#) epoch block hash pivot block hash

b. block hash block/transaction

[cfx_getBlockByHashWithPivotAssumption](#) reorg pivot block hash

* block epoch block hash

c. epoch transaction receipts

- transaction receipts:
- block [cfx_getTransactionReceipt](#) receipts

- pivot block hash `sh_getEpochReceipts` epoch receipts, `fullnode debug`
`public_rpc_api` pivot block hash pivot chain switch
 hash txn hash

d.

Reorg

pivot chain switch reorg epoch epoch pi

- `Pubsub epochs` latest state epoch epoch epoch
- epoch parent hash epoch pivot hash epoch number

PersonalMessage & CIP23


conflux-sdk fluency EIP712 CIP23 CIP23 CIP23

(typedData)

Signature Request

Account:

Account 1



Balance:

0.101587

ETH

Your signature is being requested

Signing this message can have dangerous side effects. Only sign messages from sites you fully trust with your entire account. This dangerous method will be removed in a future version. [Learn more](#)

Message:

Ox

CANCEL

SIGN

Signature Request

Account:

 Account 1 ▼


Balance:

0 ETH

Your signature is being
requested

You are signing:

Domain

```

domain: Object {name: "Decentralised Exchange", ve...
  name: "Decentralised Exchange"
  verifyingContract: "0x4b56356cd2a2bf3202f771f50d...
  version: "1"
  salt: "aa07ca11cc0cd5e0cbd94719c78d230f5d2bf2d2d...

```

Message

```

message: Object {orderHash: "0xf46bd6143937478a8be...
  orderHash: "0xf46bd6143937478a8be2db6d85d4dd95f4...
  amount: 560
  address: "0xbcd24a6b4ccb1b6faa2625fe562bdd9a2326...
  nonce: 1

```

```

{
  "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": "0xCcCCccccCCCCcCCCCCCcCcCccCcCCCcCcccccccC"
},
"message": {
  "from": {
    "name": "Cow",
    "wallet": "0xCD2a3d9F938E13CD947Ec05AbC7FE734Df8DD826"
  },
  "to": {
    "name": "Bob",
    "wallet": "0xbBbBBBBbbBBBbbbBbbBbbbbBBbBbbbbBbBbbBBbB"
  },
  "contents": "Hello, Bob! "
}
}

```

typed

<https://github.com/Conflux-Chain/CIPs/blob/master/CIPs/cip-23.md#:~:text=include%20chainId%20field.-,Encoding%20method,-The%20set%20of>

msg \x19Conflux Signed Message: \n hash

fluent

fluent web3 <https://conflux-chain.github.io/fluent-wallet-doc/docs/provider-rpc/>

fluent <https://fluent-wallet.zendesk.com/hc/zh-cn> fluent



console

```
> conflux
```

```
> conflux.isFluent
```

fluent js

```
> conflux
< ▼ 0e {#o: f, #i: f, #s: f, #r: Ee, #n: {...}, ...} ⓘ
  isFluent: true
  isMetaMask: true
  ▶ #c: f #c(e={})
  ▶ #e: Array(10)
    #e: true
  ▶ #i: f #i(e)
    #i: undefined
  ▶ #n: Object
  ▶ #n: send(e){let n=new Promise(r=> {...}
  ▶ #o: f #o(e)
    #o: "1"
  ▶ #r: Ee
  ▶ #r: ge
  ▶ #s: f #s({eventType:e,listener:n}={})
  ▶ #s: #s(){return this.request({method:"wallet_isLocked"}).then(e=> {...}
  ▶ #t: Map(3)
    #t: "0x1"
    chainId: (...)
    networkVersion: (...)
    selectedAddress: (...)
    _fluent: (...)
    _metamask: (...)
  ▶ [[Prototype]]: g

> conflux.isFluent
< true
>
```

fluent

personal_sign

console

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

fluent

Fluent

—

□

×

文本签名

英语

中文 (简体)

⋮

×

Google Translate

web3-3
cfxtest

签署这个文本?

v0G9u7huK4mJb2K1

取消

签名

```
> conflux
  .request({
    method: 'personal_sign',
    params: [
      'v0G9u7huK4mJb2K1', 'cfxtest:aajb342mw5kzad6pjkdz0wxx0tr54nfwpbu6yaj49'
    ]
  })
< ▼Promise {<pending>} ⓘ
  ► [[Prototype]]: Promise
    [[PromiseState]]: "fulfilled"
    [[PromiseResult]]: "0x7e4216720b40f8d7f2cda70433d4a94f3926635517cd5691c778e38eae87236759ddf7c53ec55c5463f8e966ebd5a32c2dbe1061e95afcbe64ef3a8187badcb00"
```

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": "0xCcCCccccCCCCcCCCCcCcCcCcCCCCcCCCCccC"
  },
  "message": {
    "from": {
      "name": "Cow",
      "wallet": "0xCD2a3d9F938E13CD947Ec05AbC7FE734Df8DD826"
    },
    "to": {
      "name": "Bob",
      "wallet": "0xbBbBBBBbbBBBbbbBbbBbbbbBBbBbbbbBbBbbBBbB"
    },
    "contents": "Hello, Bob!"
  }
}
}
})

```

签署这个信息?

Ether Mail

信息

from:

name: Cow

wallet: 0xCD2a3d9F938E13CD947Ec05AbC7FE7
34Df8DD826

to:

name: Bob

wallet: 0xbBbBBBBbbBBBbbbBbbBbbbbBBbBbb
bbBbBbbBBbB

contents: Hello, Bob!

取消

签名

```

        "name": "contents",
        "type": "string"
    }
}
},
"primaryType": "Mail",
"domain": {
    "name": "Ether Mail",
    "version": "1",
    "chainId": 1,
    "verifyingContract": "0xCcCCccccCCCCcCCCCcCCCCcCCCCcCCCCcCCCCc"
},
"message": {
    "from": {
        "name": "Cow",
        "wallet": "0xCD2a3d9F938E13CD947Ec05AbC7FE734Df8DD826"
    },
    "to": {
        "name": "Bob",
        "wallet": "0xbBbBBBBbbBBBbbbBbbBbbbbBBbBbbbbBbBbb8BBB8"
    },
    "contents": "Hello, Bob!"
}
}
})

```

◀ ▼ Promise {<pending>} ⓘ
 ▶ [[Prototype]]: Promise
 [[PromiseState]]: "fulfilled"
 [[PromiseResult]]: "0x7f28d98e75cdcaee68354d6ad0b9a2e8c4a3d365fb10fb70a1bc03a72bdb70de5b6d6587c7af57994c494ca3a1672e17d3f8f013e20641a7299f0d427a39a39001"

Java-conflux-sdk

```
java-conflux-sdk personal_sign cfx_signTypedData_v4
```

```

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()));
}
}

```

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 ECTest {

    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());
    }
}

```

```

// file: CIP23DomainExample.sol
pragma solidity ^0.4.24;

contract Example {
    struct Person {
        string name;
        address wallet;
    }

    struct Mail {
        Person from;
        Person to;
        string contents;
    }

    bytes32 constant PERSON_TYPEHASH = keccak256(
        "Person(string name,address wallet)"
    );

    bytes32 constant MAIL_TYPEHASH = keccak256(
        "Mail(Person from,Person to,string contents)Person(string name,address wallet)"
    );

    struct CIP23Domain {
        string name;
        string version;
    }
}

```

```

        uint256 chainId;
    }

    struct VerifyClaim{
        address userAddress;
        uint256 randNo;
        uint256 amount;
    }

    bytes32 constant CIP23DOMAIN_TYPEHASH = keccak256(
        "CIP23Domain(string name,string version,uint256 chainId)"
    );

    bytes32 constant VERIFYCLAIM_TYPEHASH = keccak256(
        "VerifyClaim(address userAddress,uint256 randNo,uint256 amount)"
    );

    bytes32 DOMAIN_SEPARATOR;

    constructor () public {
        DOMAIN_SEPARATOR = hash(CIP23Domain({
            name: "VerifyClaim",
            version: '1',
            chainId: 97
        }));
    }

    function hash(Person person) internal pure returns (bytes32) {
        return keccak256(abi.encode(
            PERSON_TYPEHASH,
            keccak256(bytes(person.name)),
            person.wallet
        ));
    }

    function hash(Mail mail) internal pure returns (bytes32) {
        return keccak256(abi.encode(
            MAIL_TYPEHASH,
            hash(mail.from),
            hash(mail.to),

```

```

        keccak256(bytes(mail.contents))
    ));
}

function hash(CIP23Domain cip23Domain) internal pure returns (bytes32) {
    return keccak256(abi.encode(
        CIP23DOMAIN_TYPEHASH,
        keccak256(bytes(cip23Domain.name)),
        keccak256(bytes(cip23Domain.version)),
        cip23Domain.chainId
    ));
}

function hash(VerifyClaim verifyclaim) internal pure returns (bytes32) {
    return keccak256(abi.encode(
        VERIFYCLAIM_TYPEHASH,
        verifyclaim.userAddress,
        verifyclaim.randNo,
        verifyclaim.amount
    ));
}

function verify(VerifyClaim verifyclaim, uint8 v, bytes32 r, bytes32 s) internal view
returns (bool) {
    // Note: we need to use `encodePacked` here instead of `encode`.
    bytes32 digest = keccak256(abi.encodePacked(
        "\x19\x01",
        DOMAIN_SEPARATOR,
        hash(verifyclaim)
    ));
    return ecrecover(digest, v, r, s) == 0x53dE6A872435F5286BEFd0b6fB3bC06742aF8C8F;
}

function test(address _userAddress, uint256 _randNO, uint256 _amount, uint8 _v, bytes32
_r, bytes32 _s) public view returns (bool) {
    // Example signed message
    VerifyClaim memory verifyclaim = VerifyClaim({
        userAddress: _userAddress,
        randNo: _randNO,
        amount: _amount
    });

```

```

    });
    assert(verify(verifyclaim, _v, _r, _s));
    return true;
  }
}

```

sdk sdk

1. fluent

```
> conflux
```

```
✖ ▶ Uncaught ReferenceError: conflux is not defined
   at <anonymous>:1:1
```

:

2. fluent

```

> conflux
  .request({
    method: 'personal_sign',
    params: [
      'v0G9u7huK4mJb2K1', 'cfxtest:aajb342mw5kzad6pjkdz0wxx0tr54nfwpbu6yaj49']])
< ▶ Promise {<pending>}

```

```
✖ ▶ - [wallet_validateAppPermissions] [Unauthorized 4100]
```

```

RPC Stack:
-> wallet_validateAppPermissions
-> personal_sign
  ▶ {code: 4100, message: ' - [wallet_validateAppPermissions] [Unauthorized 4... wallet_validateAppPermissions\n-> personal_sign\n', data: {...}}

```

```
✖ ▶ Uncaught (in promise) ▶ {code: 4100, message: ' - [wallet_validateAppPermissions] [Unauthorized 4... wallet_validateAppPermissions\n-> personal_sign\n', data: {...}} ECDSA
```

fluent

<https://fluent-wallet.zendesk.com/hc/zh-cn>,

java-conflux-sdk-tests

MetaMask Conflux eSpace

Conflux eSpace

Conflux eSpace

Metamask

eSpace

Conflux Core [FluentWallet](#)

MetaMask

MetaMask eSpace :

1. MetaMask
- 2.

image not found or type unknown



- 3.

eSpace

- Network Name: Conflux eSpace
- New RPC URL: <https://evm.confluxrpc.com>
- Chain ID: 1030
- Currency Symbol: CFX
- Block Explorer URL: <https://evm.confluxscan.net>
- For the eSpace testnet, please use the following configuration values:

eSpace

- Network Name: Conflux eSpace (Testnet)
- New RPC URL: <https://evmtestnet.confluxrpc.com>
- Chain ID: 71
- Currency Symbol: CFX
- Block Explorer URL: <https://evmtestnet.confluxscan.net>

image not found or type unknown



eSpace <https://efaucet.confluxnetwork.org/>

chainlist
MetaMask eSpace

chainlist :

- <https://chainlist.org>.
- "Conflux eSpace".
- "Conflux eSpace" "Connect Wallet" MetaMask
- "Conflux eSpace" "Add to Metamask" .
- MetaMask "Allow this site to add a network?" "Approve".
- MetaMask "Allow this site to switch the network?" "Approve".

MetaMask Conflux eSpace

TokenPocket

Conflux v2.1.0

Conflux

v2.1.0

 For [English check here](#)

CIP CIP94, CIP99 bug .

CIP-94

[CIP-94](#) Conflux DAO CFX

- PoW base reward
- PoS interest rate

Conflux

CIP

[DAO](#) Dapp Fluent DAO

CIP-99

[CIP-99](#) PoS CIP-99 1

PoS CIP99

- down
- 7 1
- PoS unlock 7 1 7 13

FullState

v2.1.0 FullState FullState balance no
Conflux-Rust snapshot FullState snapshot G sn
FullState FullState FullState

- enable_single_mpt_storage
- single_mpt_space = "evm" eSpace

FullState eSpace fullstate eSpace hardfork

FullState 21-25G FullState

Conflux Core

Conflux

Conflux

Conflux

- [Compatibility with the EVM](#)

Conflux

1. `0x00`
2. `cfx: cfxtest: Base cfx: aaejuaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa2mhju8k`

Base: `cfx: ... cfxtest:`

1. `.sol`
2. Base32 `.sol`

Conflux

1. EOA
- 2.

`.sol`

Conflux

EOA

EOA

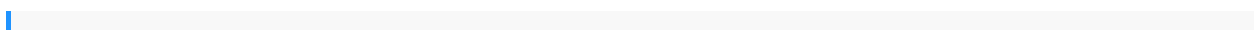
Conflux

```
from cfx_address.utils import eth_eoa_address_to_cfx_hex
eth_address = "0xcfffde169afbd51f081d2e82acca0f19cadcbbe1"
print(eth_eoa_address_to_cfx_hex(eth_address))
```

Conflux

ERC777

ERC1820



ConfluxScan

Create2Cloneconfluxcontractconflux

opcode(NUMBERBLOCKHASH)

ConfluxOpcode

NUMBER

ConfluxNUMBERopcodeepoch numberblock number

BLOCKHASH

BLOCKHASHopcodeBLOCKHASHopcodeBLOCK-1BLOCK-256) ConfluxBLOCKHASHNUMBER-1

EIP-712

EIP-712: Typed structured data hashing and signingConfluxCIP-23EIP-712

- \x19Ethereum Signed Message: \n\x19Conflux Signed Message: \n
- typed structured dataEIP712domainCIP23domain
- CIP23domainchainId