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)

fie `from` , `to` , `nonce` , `gasPrice` , `gas` , `value` , `chainId` , `data` , `v, r, s` field

conflux `epochHeight` `storageLimit` , , conflux epoch

1.
2. , encode(keccak256), (ECDSA), rlp encode -> hexString -> rawtransa
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

, sponsor . :

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 develpoment.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){
	client, 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.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")
}
```

NFT

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

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

    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:acdujjy8mrjm3913cnztf0zbgpx5h3fbby7jcwp2pc")

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,

Revision #7
Created 10 June 2022 13:18:01 by Xianqi
Updated 10 June 2022 13:49:31 by Xianqi