Isotop 跨链钱包API调用接口

入门指引

提供了简单易用的API接口，通过API可以创建账户查询资产等

创建api-key

联系管理员获取APIKEY

> **请不要泄露API Key 与 Secret Key信息，以免造成资产损失**

接口调用方式说明

- REST API
  提供账户创建，资产查询等操作

接入 URL

- http://35.175.145.216:8087/
- http://isotop.top:8087/

请求交互

介绍

REST API 提供创建账户、查询账户、资产查询、合约是否支持某接口功能

请求头信息中content-type需要统一设置为表单格式:

- **content-type:application/x-www-form-urlencoded**

状态码

状态码	说明	备注
0	成功	code=0 成功， code >0 失败
403	No permission	没有权限
405	invalid apiKey	API Key 无效
406	signature error	签名错误
408	expire time	请求过期
409	invalid nonce	nonce 无效
301	请求参数错误	请求参数错误

签名认证

签名说明

API 请求在通过网络传输的过程中极有可能被篡改，为了确保请求未被更改，私有接口均必须使用您的 API Key 做签名认证，以校验参数或参数值在传输途中是否发生了更改。

- 1.所有接口都需要进行鉴权，header参数为apiKey, timestamp, nonce, sign
- 2.timestamp为当前时间戳（秒级），与服务器时间差正负10分钟会被拒绝，nonce为随机字符串（16位），不能与上次请求所使用相同
- 3.签名方法, apiKey, timestamp, nonce，接口参数进行排序连接，使用md5方法进行签名

签名步骤

以获取账户地址为例

- 接口
  - GET /api/v1/chain/queryUser
- 示例API秘钥
  - apiKey = 7956ca03fe44238ef1d254799de1b556
  - apiSecret = bd09139024cdd3136a4f6cf60038c1194e6641063e413c47f517a579fbb158ba

1. 按照ASCII码的顺序对参数名进行排序

- 原始参数顺序为:
  - api_key = 7956ca03fe44238ef1d254799de1b556
  - nonce = 1659936393439541
  - timestamp =1659936393
  - id = 15866665555
  - chainid=1
- 按照ASCII码顺序对参数名进行排序：
  - api_key = 0816016bb06417f50327e2b557d39aaa
  - chainid = 1
  - id = 15866665555
  - nonce = 1659936589419161
  - timestamp= 1659936589

2. 所有参数按"参数名参数值"格式拼接在一起组成要签名计算的字符串

- `apiKey7956ca03fe44238ef1d254799de1b556chainid1id15866665555nonce1659936589419161t`

3. 签名计算的字符串与秘钥(Secret Key)拼接形成最终计算的字符串，使用32位MD5算法进行计算生成数字签名

- MD5(第二步字符串+秘钥)
- MD5(apiKey7956ca03fe44238ef1d254799de1b556chainid1id15866665555nonce1659936589419161timestamp1659936589bd09139024cdd3136a4f6cf60038c1194e6641063e413c47f517a579f

bb158ba)

- 签名结果中字母全部小写:
  - sign = bca925c9e774baf35288dc993c160df2

4. 将生成的数字签名加入header参数里

5. header参数

- api_key = 7956ca03fe44238ef1d254799de1b556
- nonce = 1659936393439541
- timestamp =1659936393
- sign = bca925c9e774baf35288dc993c160df2

6. body业务参数

- chainid = 1
- id = 15866665555

REST API列表

API	接口类型	说明
POST /api/v1/chain/create	POST	创建账户
GET /api/v1/chain/queryUser	GET	查询用户在链上账户地址
GET /api/v1/chain/queryAsset	GET	查询用户链上资产
GET /api/v1/chain/supportsInterface	GET	查询合约地址接口是否支持某个标准
POST /api/v1/chain/writeCall	POST	动态写入合约方法
GET /api/v1/chain/readCall	GET	动态写入合约方法
GET /api/v1/chain/getTransactionByHash	GET	根据交易Hash查询交易信息
POST /api/v1/chain/importAddress	POST	用户导入地址

创建账户

POST [/api/v1/chain/create]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
id	true	string	会员手机号	1586666555
chainid	true	int	链id	1,1029

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败
data	string	新的地址

返回示例:

```
{
  "success":true,
  "code":"0",
  "data":"cfxtest:aas4n7d0f4484ety7p9b399kffd3u8p3cajkdsr4tn"
}
```

查询用户在链上账户地址

GET [/api/v1/chain/queryUser]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
id	true	string	会员手机号	1586666555
chainid	true	int	链id	1,1029

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败

参数名称	数据类型	描述
data	数组	地址集合

返回示例:


```
{
  "success":true,
  "code":"0",
  "data":["cfxtest:aardvffhv7mgvpbm3naaycbejr7vvjtc5epsjt22bv","cfxtest:aat050mxjwuxz5wg
}
```


查询用户链上资产

GET [/api/v1/chain/queryAsset]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
tokenId	true	string	token id	1
chainid	true	int	链id	1,1029
contract	true	string	合约地址	conflux合约地址

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败
data	string	tokenURI

返回示例:


```
{
  "success":true,
  "code":"0",
  "data":"https://nftstorage.link/ipfs/bafybeicsfqe2q4rwea7pnn3tpymfayoumbfgclbhtfza7f2e
```

查询合约地址接口是否支持某个标准

GET [/api/v1/chain/supportsInterface]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
interfaceID	true	string	interface ID	十六进制字符串 0x01ffc9a7
chainid	true	int	链id	1,1029
contract	true	string	合约地址	conflux合约地址

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败
data	bool	true:支持，false:不支持

返回示例:

```
{
  "success":true,
  "code":"0",
  "data":true
}
```

动态写入合约方法

POST [/api/v1/chain/writeCall]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
data	true	string	data	
chainid	true	int	链id	1,1029

参数名称	是否必须	数据类型	描述	取值范围
contract	true	string	合约地址	conflux合约地址
id	true	string	会员手机号	会员手机号
fromAddress	true	string	msgSender	会员系统内的个人地址

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败
data	string	交易hash

返回示例:

```
{
  "success":true,
  "code":"0",
  "data":"0x81065643975146780aea7ed79b393bd3f97d3dd8145f0c78441f1dbfe5510681"
}
```

动态读取合约方法

GET [/api/v1/chain/readCall]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
data	true	string	data	
chainid	true	int	链id	1,1029
contract	true	string	合约地址	conflux合约地址
id	true	string	会员手机号	会员手机号

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败
data	string	原生数据，需自己转码

返回示例:


```
{
  "success":true,
  "code":"0",
  "data":"0x0000000000000000000000000000000000000000000000000000000000000020000000000000
}
```


根据交易查询交易信息

GET [/api/v1/chain/getTransactionByHash]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
chainid	true	int	链id	1,1029
hash	true	string	交易hash id	

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败
data	json	交易信息

返回示例:

```
{
  "success":true,
  "code":"0",
  "data":{
    "hash":"0x2b3c532af6ce2b15039b6aead1b8e4c59a453ed4b1f6b4e0f5fc04e97e176c90",
    "nonce":32,
    "blockHash":"0xa52f65da9dd716c0598ff01cfc53f78155f409b067a02084875372dcf281a734",
    "transactionIndex":0,
    "from":"cfxtest:aaj3u3efxtt0yk9jv4hp7egf8r9tee08gy9777z73a",
    "to":"cfxtest:acfpbb9kn2b2z3bev2435dk6j236gzc0kjjj4hwm45",
    "value":0,
    "gasPrice":1000000000,
    "gas":150000,
    "contractCreated":null,
    "data":"0xe4452100000000000000000000000000000000000000000000000000000000000000007b0(
    "storageLimit":0,
    "epochHeight":87255643,
    "chainId":1,
    "status":0,
    "v":0,
    "r":48244449714512018821650909603302023667601838446831864372152677559202493356686,
    "s":76874595916858495443730534422053520028790519188778582761750460123024696307 2}
}
```


用户导入地址

POST [/api/v1/chain/importAddress]

输入参数:

参数名称	是否必须	数据类型	描述	取值范围
chainid	true	int	链id	1,1029
privateKey	true	string	私钥	地址私钥
id	true	string	会员手机号	会员手机号

返回参数:

参数名称	数据类型	描述
code	string	code=0 成功， code >0 失败
success	bool	true：成功 false:失败

参数名称	数据类型	描述
data	Bool	True

返回示例:

```
{
  "success":true,
  "code":"0",
  "data":true
}
```

Python

```python
from csv import reader
from datetime import datetime
import hashlib
import requests

from rich import print
from rich import pretty
pretty.install()
from rich.console import Console
from pprint import pprint
console = Console(style="white on black", stderr=True)

apiKey = "7956ca03fe44238ef1d254799de1b556"
apiSecret = "bd09139024cdd3136a4f6cf60038c1194e6641063e413c47f517a579fbb158ba"
contract_add="cfxtest:acdeym6gccnx752abhpupmmtar5e635uu6xcv2cfgy"
# contract_add=='cfxtest:acdk44u31uwr42hy4h6ux03r5kw4ffx9ausk8k53kg'

def makeHeader(body):
    sortArgs={}
    header={}
    hash = hashlib.md5()

    # 当前日期和时间
    now = datetime.timestamp(datetime.now())
    header['timestamp']= str(int(now))
    header['nonce']= str(int(now*1000000))
    header['apiKey']=apiKey

    sortArgs.update(header)
    sortArgs.update(body)
    # print(f"{args=}")

    content=''
    for item in sorted(sortArgs):
        content+= item+sortArgs[item]

    content+= apiSecret
    # print(content)

    hash.update(content.encode(encoding='utf-8'))

    # print(hash.hexdigest())

    # print(response.json())

    header["content-type"]="application/x-www-form-urlencoded"
    header['sign']= hash.hexdigest()

    # print(header)
    # print(body)
```

```
    # print(hash.hexdigest())
    # print(header)
    # console.print(Panel(header))
    return header

def getTransactionByHash(hash):
    body={}
    body['chainid']='1'
    body['hash']= hash
    body['id']='13911024683'

    api_url = "http://35.175.145.216:8087/api/v1/chain/getTransactionByHash"

    header= makeHeader(body)
    response = requests.get(api_url, params= body, headers=header)

    json= response.json()
    if json['success']==True:
        return json['data']
    else:
        print(json)

def writeCall(_from, data):
    body={}
    body['chainid']='1'
    body['data']= data
    body['fromAddress']=_from
    body['contract']=contract_add
    body['id']='13911024683'

    api_url = "http://35.175.145.216:8087/api/v1/chain/writeCall"

    header= makeHeader(body)
    console.print(body, style="bold yellow")

    response = requests.post(api_url, params= body, headers=header)

    json= response.json()
    if json['success']==True:
        return json['data']
    else:
        console.print(json, style="bold red")

def readCall(data):
    body={}
    body['chainid']='1'
    body['data']= data
    body['contract']=contract_add
    body['id']='13911024683'

    api_url = "http://35.175.145.216:8087/api/v1/chain/readCall"
```

```
    header= makeHeader(body)
    # print(body)
    response = requests.get(api_url, params= body, headers=header)

    json= response.json()
    if json['success']==True:
        return json['data']
    else:
        console.print(json, style="bold red")

def supportsInterface(selector):
    body={}
    body['chainid']='1'
    body['interfaceID']= selector
    body['contract']=contract_add

    api_url = "http://35.175.145.216:8087/api/v1/chain/supportsInterface"

    header= makeHeader(body)
    response = requests.get(api_url, params= body, headers=header)

    json= response.json()
    if json['success']==True:
        return json['data']
    else:
        console.print(json, style="bold red")


def queryAsset(tokenId):
    body={}
    body['chainid']='1'
    body['tokenId']= tokenId
    body['contract']=contract_add

    api_url = "http://35.175.145.216:8087/api/v1/chain/queryAsset"

    header= makeHeader(body)
    response = requests.get(api_url, params= body, headers=header)

    json= response.json()
    if json['success']==True:
        return json['data']
    else:
        console.print(json, style="bold red")

def queryUser():
    body={}
    body['chainid']='1'
    body['id']='13911024683'
```

```
    header= makeHeader(body)
    api_url = "http://35.175.145.216:8087/api/v1/chain/queryUser"
    response = requests.get(api_url, params= body, headers=header)

    json= response.json()
    if json['success']==True:
        return json['data']
    else:
        console.print(json, style="bold red")

def createUser():
    body={}
    body['chainid']='1'
    body['id']='13911024683'

    header= makeHeader(body)
    api_url = "http://35.175.145.216:8087/api/v1/chain/create"
    # print(f"{header} {body} {api_url}")
    response = requests.post(api_url, params= body, headers=header)

    # print(response)
    json= response.json()
    if json['success']==True:
        return json['data']
    else:
        console.print(json, style="bold red")

if __name__=="__main__":
    while True:
        choice = input("1) createUser\n2) queryUser\n3) queryAsset\n4) supportsInterface

        commands= choice.split()
        if len(commands)== 0 or commands[0] == "q":
            break
        if commands[0] == '1':
            ret= createUser()
        if commands[0] == '2':
            ret= queryUser()
        if commands[0] == '3':
            ret= queryAsset(commands[1])
        if commands[0] == '4':
            ret= supportsInterface(commands[1])
        if commands[0] == '5':
            ret= readCall(commands[1])
        if commands[0] == '6':
            ret= writeCall(commands[1], commands[2])
        if commands[0] == '7':
            ret= getTransactionByHash(commands[1])
        console.print(ret, style="white on black")
```

Java

```java
package com.wallet.admin;

import com.wallet.common.utils.StringUtils;
import org.apache.commons.codec.Charsets;
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.math.BigDecimal;
import java.util.*;

/**
 * @ClassName: test
 * @Description:
 * @Author
 * @Date 2022/8/8
 * @Version 1.0
 */
public class test {
    public static void main(String[] args) {
        String result=null;
/*
        Map<String, String> params = new HashMap<>();
        params.put("apiKey","7956ca03fe44238ef1d254799de1b556");
        params.put("nonce", System.currentTimeMillis()/1000+"_"+ StringUtils.getRandom(
        params.put("timestamp",String.valueOf(System.currentTimeMillis()/1000));
*/

        Map<String, String> params = new HashMap<>();
        params.put("chainid","1");
        params.put("id","1");
        //params.put("data","0xe4452100000000000000000000000000000000000000000000000000000
```

```
        //params.put("contact","cfxtest:acfpbb9kn2b2z3bev2435dk6j236gzc0kjjj4hwm45");



        params.put("contract","cfxtest:acdk44u31uwr42hy4h6ux03r5kw4ffx9ausk8k53kg");
        //params.put("tokenId","1");

        //params.put("interfaceID","0x01ffc9a7");
        params.put("hash","0xf736bbc8b9c807a57c211f2c6235ebccdd9f425f73497d769555550edc
        //params.put("fromAddress","cfxtest:aajpefpv3b528myyvv1p6fy8ey5j6ve63ezjb7ebxv"
        //params.put("contract","0xc54b96b04AA8828b63Cf250408E1084E9F6Ac6c8");
        //params.put("data","0xc87b56dd0000000000000000000000000000000000000000000000000
        Map<String, String> headerMap = makeHeaders(params);
        System.out.println("header参数》》"+headerMap);
        System.out.println("body参数》》"+params);

        String rsc= getRequest("http://localhost:8088/api/v1/chain/getTransactionByHash'
        System.out.println(rsc);
        String rsc1= getRequest("http://35.175.145.216:8087/api/v1/chain/getTransactionB
        System.out.println(rsc1);



/*
        String rs1= getRequest("http://35.175.145.216:8087/api/v1/chain/queryUser",heade
        System.out.println(rs1);
*/

/*
        String rs1= postRequest("http://localhost:8088/api/v1/chain/create",headerMap,pa
        System.out.println(rs1);
*/

/*
        String rsc= getRequest("http://localhost:8088/api/v1/chain/queryUser",headerMap,
        System.out.println(rsc);
*/

/*
        String rs= getRequest("http://localhost:8088/api/v1/chain/supportsInterface",hea
        System.out.println(rs);
*/

    }
    private static Map<String, String> makeHeaders(Map<String, String> data){
        Map<String, String> rsMap = new HashMap<>();
        String nonce = System.currentTimeMillis()/1000+""+ (int)((Math.random()*9+1)*100
        String timestamp = System.currentTimeMillis()/1000+"";
/*
        timestamp="1660278066";
        nonce="1660278066380482";
*/
```

```
        rsMap.put("apiKey","7956ca03fe44238ef1d254799de1b556");
        rsMap.put("timestamp",timestamp);
        rsMap.put("nonce",nonce);
        SortedMap<String,String> sortedMap = new TreeMap<>();
        sortedMap.putAll(rsMap);
        sortedMap.putAll(data);
        StringBuilder sbd = new StringBuilder();
        for (Map.Entry<String, String> entry : sortedMap.entrySet()) {
            // 排除空val的参数
            if (StringUtils.isEmpty(entry.getValue())){
                continue;
            }
            sbd.append(entry.getKey()).append(entry.getValue());
        }
        System.out.println("ASCII排序字符串"+sbd.toString());
        String apiSecret="bd09139024cdd3136a4f6cf60038c1194e6641063e413c47f517a579fbb158
        sbd.append(apiSecret);
        rsMap.put("sign",DigestUtils.md5Hex(sbd.toString()));
/*
        System.out.println(nonce);
        System.out.println(timestamp);
        System.out.println(rsMap.get("sign"));
*/
        return rsMap;
    }
    private static String sign(String nonce, Map<String, String> data) {
        List paramArr = new ArrayList<>();
        for (String key : data.keySet()) {
            paramArr.add(key + "=" + data.get(key));
        }
        Collections.sort(paramArr);
        System.out.println(paramArr);
        String paramStr = String.join("", paramArr);

        String signature = DigestUtils.md5Hex(paramStr);
        return signature;
    }
    public static String postRequest(String url, Map<String,String> headerMap, Map<Strin
        String result = null;
        CloseableHttpClient httpClient = HttpClients.createDefault();
        HttpPost post = new HttpPost(url);
        List<NameValuePair> content = new ArrayList<NameValuePair>();
        Iterator iterator = paramsMap.entrySet().iterator();           //将content生成ent
        while(iterator.hasNext()){
            Map.Entry<String,String> elem = (Map.Entry<String, String>) iterator.next()
            content.add(new BasicNameValuePair(elem.getKey(),elem.getValue()));
        }
        CloseableHttpResponse response = null;
        try {
            Iterator headerIterator = headerMap.entrySet().iterator();          //循环增
            while(headerIterator.hasNext()){
```

```
            Map.Entry<String,String> elem = (Map.Entry<String, String>) headerIterat
            post.addHeader(elem.getKey(),elem.getValue());
        }
        if(content.size() > 0){
            UrlEncodedFormEntity entity = new UrlEncodedFormEntity(content,"UTF-8");
            post.setEntity(entity);
        }
        response = httpClient.execute(post);            //发送请求并接收返回数据
        if(response != null && response.getStatusLine().getStatusCode() == 200)
        {
            HttpEntity entity = response.getEntity();       //获取response的body部分
            result = EntityUtils.toString(entity);          //读取reponse的body部分并
        }
        return result;
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            httpClient.close();
            if(response != null)
            {
                response.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    return null;
}

public static String getRequest(String url,  Map<String, String> headerMap,Map<Strin
    String result = "";
    BufferedReader in = null;
    List<NameValuePair> formparams = setHttpParams(paramMap);
    String param = URLEncodedUtils.format(formparams, "UTF-8");

    String reqUrl = url + "?" + param;
    try {
        RequestConfig config = RequestConfig.custom().setConnectTimeout(3000)
                .setConnectionRequestTimeout(3000).build();
        HttpClient client = HttpClientBuilder.create().setDefaultRequestConfig(confi
        HttpGet htGet = new HttpGet(reqUrl);
        // 添加http headers
        if (headerMap != null && headerMap.size() > 0) {
            for (String key : headerMap.keySet()) {
                htGet.addHeader(key, headerMap.get(key));
```

```
                }
            }
            HttpResponse r = client.execute(htGet);
            in = new BufferedReader(new InputStreamReader(r.getEntity().getContent(), Cl
            String line;
            while ((line = in.readLine()) != null) {
                result += line;
            }
        } catch (Exception e) {
            System.out.println("发送GET请求出现异常! " + e);
            e.printStackTrace();
        } finally {
            try {
                if (in != null) {
                    in = null;
                }
            } catch (Exception e2) {
                e2.printStackTrace();
            }
        }
        return result;
    }
    private static List<NameValuePair> setHttpParams(Map<String, String> paramMap) {
        List<NameValuePair> formparams = new ArrayList<NameValuePair>();
        Set<Map.Entry<String, String>> set = paramMap.entrySet();
        for (Map.Entry<String, String> entry : set) {
            formparams.add(new BasicNameValuePair(entry.getKey(), entry.getValue()));
        }
        return formparams;
    }
}
```