智能合约对接开发说明

同位素网络科技/HEYF00 钱包

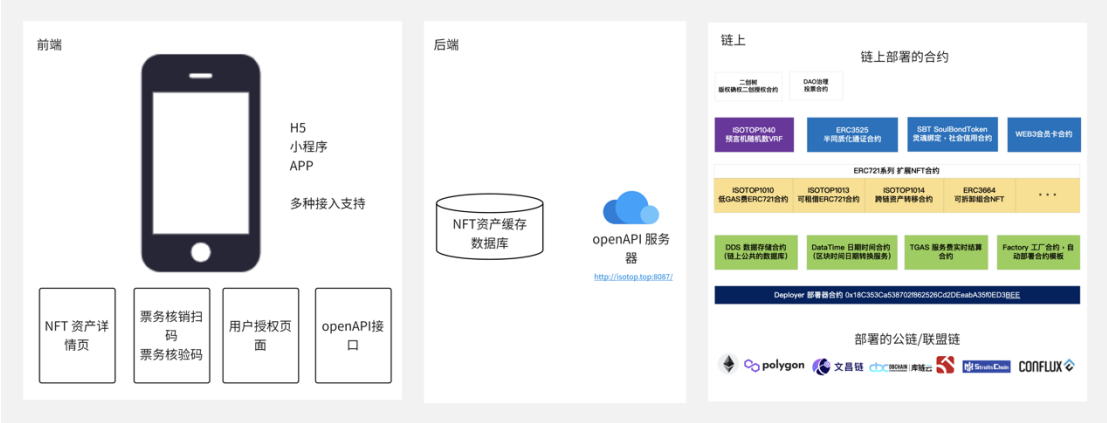
2023 年 3 月

概述

本文档说明了升级 NFT721 数字藏品为带有权益的数字资产（如 SBT，乡村卡，数字权益票等等）的对接接口。同位素提供了链上部署的智能合约，以及配合调用合约的 heyfoo 钱包 openAPI，钱包授权页面。所有用户交互部分由合作方自行设计和开发。

系统架构

同位素提供的智能合约已经在指定的链上部署好，通过 heyfoo 钱包 openAPI 或者 H5 授权页面可以直接访问，也可以嵌入到成熟的数藏平台的 APP 或 H5 之内。同位素提供智能合约和相关的技术接口。



图中，前端完全由合作方定制开发，可以选择使用带有 H5 授权页面的钱包对接方式，或者自己管理用户手机号注册的 heyfoo 钱包 openAPI 方式。
后端服务器部署在亚马逊云上，提供 GET/POST 两种方式的 heyfoo 钱包 openAPI 调用方式，私钥采用分片存储方式，最大可能保护用户资产安全性。
NFT 资产的缓存可以选择使用。
链上部分针对不同的国内外公链，联盟链，合约部署在统一的地址，可以通过合约地址访问。

POAP（类型一）合约调用

这里介绍最简单的 POAP 类型，单一图片的 POAP 的发行。ISOTOP1015.此类型 POAP 只需要把图片存储在 IPFS 或者云盘，获取图片 URL 作为 baseURL 即可铸造。

1. 合约概述

合约名称	描述	部署地址
Factory	工厂合约	Conflux core test: 0x88089689F057F28fffC01D7A97c8a123c7C457fE cfxtest:aceavfyk8bn9fd992as1zf8jyev6tvc192tdk8t Wenchang test: 0xc53eCBa8f91b6E36E1E108B02DB6efB408b9c7bD
ISOTOP1015	POAP（类型一）合约	工厂合约后台

2. 使用流程

2.1. 平台管理员通过链上的 Factory 地址，并且创建一个属于自己的 ISOTOP1015 合约。

合约	ABI	参数
Factory	deployContract	字符串”ISOTOP1015”
Factory	getLastDeployed	字符串”ISOTOP1015” 返回最后部署的合约地址
Factory	getContractsDeployed	字符串”ISOTOP1015” 返回部署的合约地址列表

例如：

```
factory.deployContract("ISOTOP1015, addr(consumer))
sbtop = ISOTOP1013.at(factory.getLastDeployed(
   "ISOTOP1013", addr(consumer)))
```

2.2. 初始化合约

```
function init(
      string memory name_,
      string memory symbol_,
      string memory base_,
      string memory details_
   )
```

初始化的参数分别是：
POAP 名称，
代号，
图片 URL，

介绍内容的 URL

2.3. 发行 POAP

```
function mint(address _to, uint256 quantity)
```

发行 POAP 的参数是：
用户地址，
发行数量

数字藏品智能合约调用

智能合约是部署在链上的可调用代码。 ABI 是智能合约的接口文档，定义了智能合约的调用方式和参数类型。本文选择重要的 ABI 参数和使用流程进行了详细的解释。

3. 合约概述

合约名称	描述	部署地址
Factory	工厂合约	Conflux core test: 0x845b52016Afc1651AA7401E2B6750a7e47d3e817 cfxtest:aamf0yubrn8bpyrmsua8frxzbk9ety9jc6uf455 Wenchang test: 0xc53eCBa8f91b6E36E1E108B02DB6efB408b9c7bD
ISOTOP1013	数字藏品合约	工厂合约后台

4. 使用流程

4.1. 平台管理员通过链上的 Factory 地址，并且创建一个属于自己的 ISOTOP1013 合约。

合约	ABI	参数
Factory	deployContract	字符串”ISOTOP1013”
Factory	getLastDeployed	字符串”ISOTOP1013” 返回最后部署的合约地址
Factory	getContractsDeployed	字符串”ISOTOP1013” 返回部署的合约地址列表

例如：

```
factory.deployContract("ISOTOP1013, addr(consumer))
```

```
sbtop = ISOTOP1013.at(factory.getLastDeployed(
    "ISOTOP1013", addr(consumer)))
```

4.2. 定义数藏的的 metadata

数藏的内容定义都在 metadata 中，以下是一个例子：
1. json

```
{
    "name": "数字藏品 #1",
    "symbol": "SHC",
    "description": "数字藏品",
    "image": "https://...jpg ",
    "designer": "isotop.top"
}
```

需要创建一批数藏，可以把各自的 json 文件准备好，放在服务器，顺序命名，并且将目录名字作为参数传递，这样铸造好的数藏产品就可以正常显示了。

4.3. 调用 mint

调用合约的 mint，提供两个参数：用户地址和数量，即可铸造上链。

SBT 智能合约调用

SBT 智能合约是针对社会信用资产的发行制定的。SBT Token 领取后，无法转增和销毁，属于用户终身的资产

5. 合约概述

合约名称	描述	部署地址
Factory	工厂合约	Conflux core test: 0x845b52016Afc1651AA7401E2B6750a7e47d3e817 cfxtest:aamf0yubrn8bpyrmsua8frxzbk9ety9jc6uf4553bw Wenchang test: 0xc53eCBa8f91b6E36E1E108B02DB6efB408b9c7bD
ISOTOP1040	SBT 平台	工厂合约后台
SoulBondToken	SBT token 生成器	
DDS	DID 注册表	Conflux core test: 0x89212dcE4dEaa27a25AAea1140a589D18aeBf362

		cfxtest:aapwcnssk1zme8vfznzbcuffvhj2z49xpjmkgf3gx0 文昌链测试链： 0xCFfFDE169Afbd51F081d2e82aCcA0f19cAdCbbeE

6. 角色说明

A：平台方。 可以创建 Soul，可以把某个 SBT 赋予某个 Soul。平台方是 SBT 的管理者
B：证书发行方。 代表某个官方发行者。需要在 DDS 进行注册，证明自己的官方身份，才能发行官方的 SBT。
C：个人。 个人可以创建 Soul，也可以领取某个 SBT。

7. 使用流程

7.1. 平台管理员通过链上的 Factory 地址，并且创建一个属于自己的 ISOTOP1040 合约。

合约	ABI	参数
Factory	deployContract	字符串”ISOTOP1040”
Factory	getLastDeployed	字符串”ISOTOP1040” 返回最后部署的合约地址
Factory	getContractsDeployed	字符串”ISOTOP1040” 返回部署的合约地址列表

例如：

```
factory.deployContract("ISOTOP1040", addr(consumer))
sbtop = ISOTOP1040.at(factory.getLastDeployed(
   "ISOTOP1040", addr(consumer)))
```

7.2. 平台管理员通过 ISOTOP1040 合约，创建一个 Soul。

合约	ABI	参数
ISOTOP1040	createSoul(address _to, uint256 soulId)	地址和 soul 编号（必须唯一）

例如：

```
sbtop.createSoul(iwan, IWAN, addr(consumer))
```

7.3. 证书发行方在 DID 注册自己。

合约	ABI	参数
DDS	put(string _domain, string _key, bytes _data)	域名， 证书名， 值(SBT 的 JSON 文件目录地址)

例如：

```
dds.put('BUAA', 'CGC', str_to_bytes('http://buaa.org/nft/'), addr(creator))
```

7.4. 证书发行方发行 SBT。

合约	ABI	参数
soulBondToken	issue(string _issuer, string _certification, uint256 validDate_, address[]_allows)	发行者， 证书， 有效期 允许领取者列表

例如：

```
sbt.issue('BUAA', 'CGC', chain.time()+10000, [], addr(creator))
```

7.5. 平台方指派 SBT 给用户 soul。

合约	ABI	参数
ISOTOP1040	assign(address _to, uint256 _soulId, string _issuer, string _certification)	分配地址 分配者编号 SBT 编号 发行者， 证书

例如：

```
sbtop.assign(iwan, IWAN, 'BUAA', 'CGC', addr(consumer))
```

8. SBT 的 metadata

SBT 的内容定义都在 metadata 中，以下是一个例子：

1. json

```
{
    "name": "创世徽章 #1",
    "symbol": "CSH",
    "description": "创世徽章 SBT",
    "image": "https://...jpg ",
```

```
    "designer": "isotop.top"
}
```

需要创建一批 SBT，可以把各自的 json 文件准备好，放在服务器，顺序命名，并且将目录名字在铸造 SBT 时作为参数传递，这样铸造好的 SBT 就可以正常显示了。

数字权益门票智能合约调用

数字权益门票是动态的 NFT，兼容 ERC721 标准。

1. 合约地址

合约名称	描述	部署地址
Factory	工厂合约	Conflux core test: 0x845b52016Afc1651AA7401E2B6750a7e47d3e817 cfxtest:aamf0yubrn8bpyrmsua8frxzbk9ety9jc6uf455 Wenchang test: 0xc53eCBa8f91b6E36E1E108B02DB6efB408b9c7bD
ISOTOP1052	数字权益票务	工厂注册合约

2. 使用流程

2.1. 平台管理员通过链上的 Factory 地址，并且创建一个属于自己的 ISOTOP1052 合约。

合约	ABI	参数
Factory	deployContract	字符串”ISOTOP1052”
Factory	getLastDeployed	字符串”ISOTOP1052” 返回最后部署的合约地址
Factory	getContractsDeployed	字符串”ISOTOP1052” 返回部署的合约地址列表

例如：

```
factory.deployContract("ISOTOP1052", addr(consumer))
sbtop = ISOTOP1040.at(factory.getLastDeployed(
   "ISOTOP1040", addr(consumer)))
```

2.2. 初始化票务。

初始化票务需要提供：
Name：名称
Symbol：编号
Base：图片目录的地址
Details：管理网站
Sites：打卡点，优惠券的名称
Suffix：图片类型后缀（如 JPG，PNG）

```
init(
      string memory name_,
      string memory symbol_,
      string memory base_,
      string memory details_,
      string[] memory sites_,
      string memory suffix_
   )
```

2.3. 创建一张票。

```
mint(
      address _to,
      uint256 quantity
   )
```

给一个地址创建 N 张票

2.4. 打卡一张票。

```
check(
      uint256 tokenId,
      uint8 _site,
      address _who
   )
```

heyfoo 钱包 openAPI

对于前端完全自己定制的 APP 或小程序，可以直接使用同位素的 heyfoo 钱包 openAPI 创建链账户和访问合约。heyfoo 钱包 openAPI 符合 HTTP 请求调用标准，可以兼容各种开发语言。

3. 创建 api-key

签署合同后，联系同位素客户经理获取 APIKEY

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

4. 接口调用方式说明

- REST API

  提供账户创建，资产查询等操作

5. 接入 URL

- http://isotop.top:8087/

6. 请求交互

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

7. 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。

- 所有接口都需要进行鉴权，header 参数为 apiKey, timestamp, nonce, sign
- imestamp 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
- 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 = aaaaa

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

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

- ```
  apiKey7956ca03fe44238ef1d254799de1b556chainid1id15866665555nonce1659
  936589419161timestamp1659936589
  ```

-

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(apiKey7956ca03fe44238ef1d254799de1b556chainid1id15866665555nonce1659936589419161timestamp1659936589bd09139024cdd3136a4f6cf60038c1194e6641063e413c47f517a579fbb158ba)
- 签名结果中字母全部小写:
    - sign = bca925c9e774baf35288dc993c160df2

将生成的数字签名加入 header 参数里

header 参数

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

body 业务参数
  ```

- chainid = 1
- id = 15866665555

8. REST API 功能介绍

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

8.1. 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"
}
```


8.2. 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
data	数组	地址集合

返回示例:


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

"data":["cfxtest:aardvffhv7mgvpbm3naaycbejr7vvjtc5epsjt22bv","cfxtest:aat05
0mxjwuxz5wge6r3xhw224y8zd5s1jrm94b8d1","cfxtest:aas4n7d0f4484ety7p9b399kffd
3u8p3cajkdsr4tn"]
}
```

8.3. 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: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/bafybeicsfqe2q4rwea7pnn3tpymfayoumbfgclbhtfza7f2eza7sarjqrm/1.json"}
```

8.4. 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

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
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
}
```

8.5. 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
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

参数名称	数据类型	描述
data	string	交易 hash

返回示例:


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

"data":"0x81065643975146780aea7ed79b393bd3f97d3dd8145f0c78441f1dbfe5510681"
}
```


8.6. 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":"0x0000000000000000000000000000000000000000000000000000000000000020
0000000000000000000000000000000000000000000000000000000064e4654373231
0000000000000000000000000000000000000000000000000000"
}
```


8.7. 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":"0xa52f65da9dd716c0598ff01cfc53f78155f409b067a02084875372dcf281
a734",
    "transactionIndex":0,
    "from":"cfxtest:aaj3u3efxtt0yk9jv4hp7egf8r9tee08gy9777z73a",
```

```
    "to":"cfxtest:acfpbb9kn2b2z3bev2435dk6j236gzc0kjjj4hwm45",
    "value":0,
    "gasPrice":1000000000,
    "gas":150000,
    "contractCreated":null,

"data":"0xe4452100000000000000000000000000000000000000000000000000000000000000000000000000000000000000007b0000000000000000000000000000000000000000000000000000000000000001",
    "storageLimit":0,
    "epochHeight":87255643,
    "chainId":1,
    "status":0,
    "v":0,

"r":48244449714512018821650909603302023667601838446831864372152677559202493356686,

"s":76874595916858495443730534422053520028790519188778582761750460123024696 3072}
}
```


8.8. 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
data	Bool	True

返回示例:

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

案例代码

1. Java 代码示例

```
```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(5));

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","0xe4452100000000000000000000000000000000000000000000000
0000000000000007b0000000000000000000000000000000000000000000000000000000000
000000001");

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




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

        //params.put("interfaceID","0x01ffc9a7");

params.put("hash","0xf736bbc8b9c807a57c211f2c6235ebccdd9f425f73497d76955555
0edcf11c89");
```

```
//params.put("fromAddress","cfxtest:aajpefpv3b528myyvv1p6fy8ey5j6ve63ezjb7ebxv");

//params.put("contract","0xc54b96b04AA8828b63Cf250408E1084E9F6Ac6c8");

//params.put("data","0xc87b56dd0000000000000000000000000000000000000000000000000000000000000000");
        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",headerMap,params);
        System.out.println(rsc);
        String rsc1= getRequest("http://35.175.145.216:8087/api/v1/chain/getTransactionByHash",headerMap,params);
        System.out.println(rsc1);


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

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

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

/*
```

```
        String rs=
getRequest("http://localhost:8088/api/v1/chain/supportsInterface",headerMap
,params);
        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)*100000);
        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="aaa";
        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<String, String> paramsMap) {
        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 entity
        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();
//循环增加 header
            while(headerIterator.hasNext()){
                Map.Entry<String,String> elem = (Map.Entry<String, String>)
headerIterator.next();
                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 部分并转化成字符串
            }
            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<String, String> paramMap) {
        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(config).build();
            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(), Charsets.UTF_8));
            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;
    }
}

```

2. Python 代码示例

```python
```

```python
from rich.console import Console
from pprint import pprint
from datetime import datetime
import hashlib
import requests
import sys

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

apiKey = "7956ca03fe44238ef1d254799de1b556"
apiSecret =
"bd09139024cdd3136a4f6cf60038c1194e6641063e413c47f517a579fbb158ba"
# contract_add="cfxtest:acdeym6gccnx752abhpupmmtar5e635uu6xcv2cfgy"
# contract_add='0x05271BB6F2387fbFf5cacdDBCBD5a7C1021A4b11'
# contract_add=='cfxtest:acdk44u31uwr42hy4h6ux03r5kw4ffx9ausk8k53kg'
# contract_add = 'cfx:acak9mwwemm4tgvs7j798je832mrptyrpa9d6ea78s'
contract_add = 'cfx:accm4vkvec1y96jajw0gw6k6k39gbm5bha9b8knzt5'
# chain_id='5555' # BSN wuhan
chain_id = '1029'  # Conflux


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 importAccount(private_key):
    body = {}
    body['chainid'] = chain_id
    body['privateKey'] = private_key
    body['id'] = '13911024683'

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

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

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


def exportAccount(address):
    body = {}
    body['chainid'] = chain_id
    body['address'] = address
    body['id'] = '13911024683'

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

    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 getTransactionByHash(hash):
    body = {}
    body['chainid'] = chain_id
    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'] = chain_id
    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'] = chain_id
    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(erc):
    body = {}
    body['chainid'] = chain_id
    body['interfaceID'] = erc
    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'] = chain_id
```

```
    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'] = chain_id
    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'] = chain_id
    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__":
    if len(sys.argv) == 3:
        contract_add = sys.argv[1]
        chain_id = sys.argv[2]
    print(f"{contract_add=} {chain_id=}")

    while True:
        choice = input("1) createUser\n2) queryUser\n3) queryAsset\n4)
supportsInterface\n5) readCall\n6) writeCall\n7)
getTransactionReceiptByHash\n8) importAccount\n9) exportAccount\nq) to
exit:\n\n")

        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])
        if commands[0] == '8':
            ret = importAccount(commands[1])
        if commands[0] == '9':
            ret = exportAccount(commands[1])
        console.print(ret, style="white on black")
```

```
```

3. JS 代码示例

```
//npm i,New terminal,run: node index.js
const abi = require('ethjs-abi');
//这块更换为自己合约的 abi 数组
const SimpleStoreABI = [{ "inputs": [{ "internalType": ...}]}]
console.log('SimpleStoreABI.length:'+SimpleStoreABI.length);

/**
 * 下方是如何编码举例,该编码用于 api 文档中的 writeCall 和 readCall 中的 data 入参
 * ownerOf(查询拥有者)
 * mint(铸造 NFT)
 */

//此方法为 ownerOf(查询拥有者),18 是上方 abi 数组的下标,入参 name 是 tokenId,类型是
uint256
const tokenId = 1;
const encodeData1 = abi.encodeMethod(SimpleStoreABI[18], [tokenId]);
console.log('encodeData1:'+encodeData1);

//此方法为 mint(铸造 NFT),15 是上方 abi 数组的下标,两个入参   1:name 是 to(接收者),类型是
address   2:name 是 tokenId,类型是 uint256
const address = '0xd27bF27652D12d765A053bf742fd89f2c155cc67';
const encodeData2 = abi.encodeMethod(SimpleStoreABI[15], [address,
tokenId]);
console.log('encodeData2:'+encodeData2);
```