复杂查询与审计模块API

1. 公用说明

1.1. 关于ApiUrl

请求参数中出现了ApiUrl 的，均可参考本部分

`ApiUrl` 是如何访问 ipfs、合约的重要内容。 `ipfsServiceUrl` 请填写部署的 IPFS 后端端口； `contractName` 请填写在长安链上部署的合约名称。

```
"apiUrl": {
    "ipfsServiceUrl": "http://服务器IP:5001",
    "chainServiceUrl": "",
    "contractName": "合约名称"
  }
```

2. 上传

2.1. 文件批量上载 IPFS 并上载数字信封

`POST` /upload/uploadFileBatch

2.1.1. 接口说明

根据前端传来的文档内容和AES密钥，进行加密后上传IPFS，随后生成数字信封，上传到区块链上。返回存储的IPFS地址。

本业务接口的基本流程：

1 **Step 1**

接收你传入的文本内容、AES密钥

2 **Step 2**

使用AES密钥对文本内容进行加密

3 **Step 3**

将加密后的文本内容上传IPFS，保存上传的位置pos

4 **Step 4**

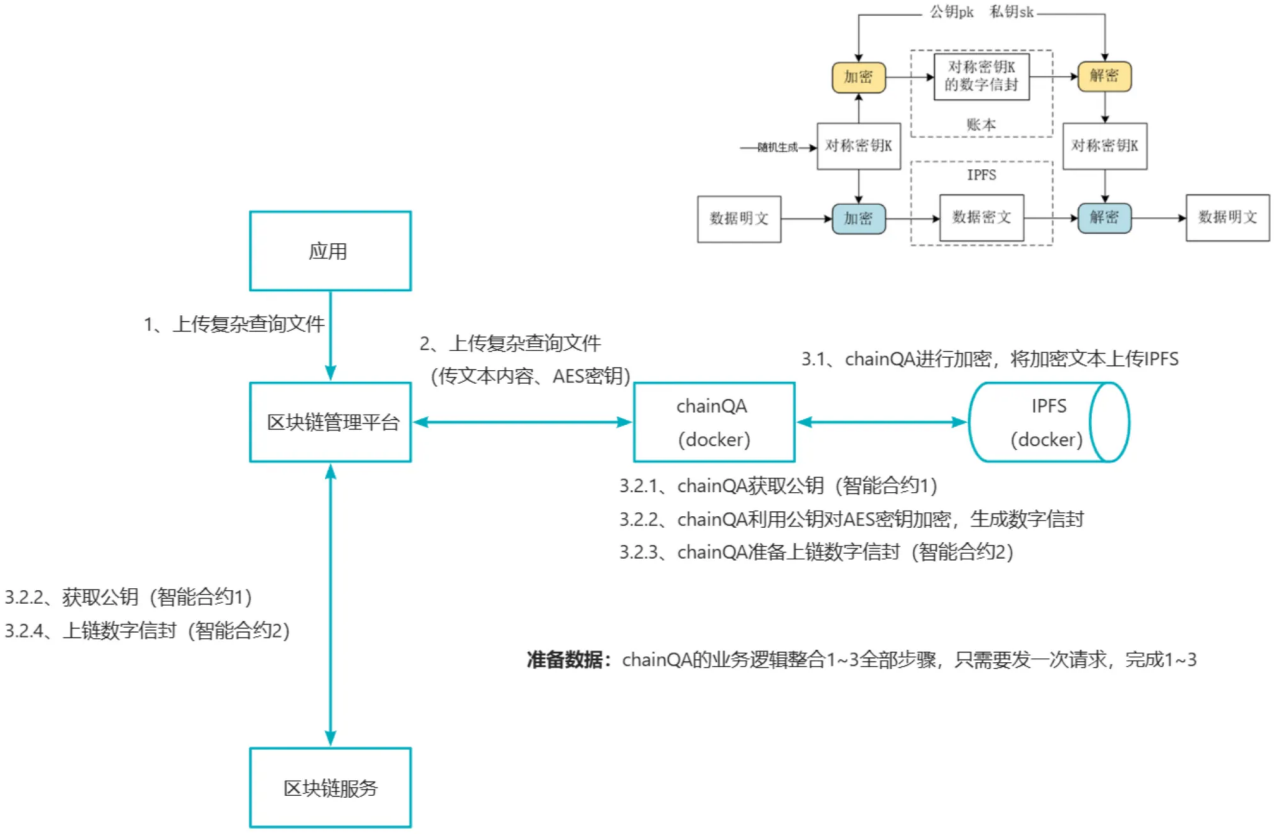
调用区块链智能合约，获取区块链节点公钥PK

5 **Step 5**

用公钥PK对AES密钥加密，生成数字信封。数字信封上传区块链。

6 **Step 6**

返回pos



参数中AesKey的说明

为了保证安全性，chainQA业务逻辑本身不涉及生成AES密钥。

注意：发送的aes密钥必须是16位/32位字符，例如 `"b1bdd2f304a1ac6f1629051b229a44c1"` 。

可使用前端的AES密钥生成库完成

```
const generateSecureAesKey = (length = 16) => {
  const array = new Uint8Array(length);
  window.crypto.getRandomValues(array);
  return Array.from(array, (byte) => ("0" + byte.toString(16)).slice(-2)).join(
    ""
  );
};
```

参数中fileContent的说明

上传IPFS的FileContent要符合ChainQA规定的一般格式。简而言之就是：

- 第一行是列名，列名之间用一个 `空格` 分割，列名中不包含空格，不能仅为“*”
- 后续为数据，每个数据（单元格）与对应的列名一一对应。单元格数据之间用一个空格分割。单元格值不允许出现空格。每行单元格个数严格等于列个数。
- 行之间用\n换行，最后一行结尾无需\n

例如——

```
id name score
1 HELLO 45
2 WORLE 56
3 TEST 56
4 TEST2 36
```

上述要求是针对解密后的文本信息。实际上传 IPFS 的是对称加密后的密文文本

> 链下版网页端已支持 excel 的便捷转换

2.1.2. 请求参数

> Body 请求参数

```
{
  "fileContent": [
    "string"
  ],
  "fileName": [
    "string"
  ],
  "aesKeys": [
    "string"
  ],
  "uId": "string",
  "apiUrl": {
    "ipfsServiceUrl": "string",
    "chainServiceUrl": "string",
    "contractName": "string"
  }
}
```


具体内容如下表：

名称	位置	类型	必选	说明
body	body	object	否	none
» fileContent	body	[string]	是	文件内容 请参见**接口说明**：fileContent
» fileName	body	[string]	是	文件名（主要用于前端标识，后端不处理，原样返回
» aesKeys	body	[string]	是	请参见**接口说明**：Aeskey
» uId	body	string	是	上传者 id 或名称（仅限英文字母/数字）
» apiUrl	body	object	是	

»» ipfsServiceUrl	body	string	是	IPFS地址
»» chainServiceUrl	body	string	是	区块链SDK（或者说区块链管理平台）地址。
»» contractName	body	string	是	合约名称

2.1.3. 结果返回

状态码：

状态码	状态码含义	说明	数据模型
200	OK	none	Inline

返回数据结构为：

名称	类型	必选	约束	中文名	说明
» code	integer	true	none		none
» data	object	true	none		none
»» fileName	string	true	none		none
»» pos	string	true	none		none
» msg	string	true	none		none

例子：

> 200 Response

```
{
  "code": 0,
  "data": {
    "fileName": "文件名",
    "pos": "IPFS地址"
  },
  "msg": "上传文件XXX到IPFS成功"
}
```


3. 下载

3.1. 下载IPFS文件

3.1.1. 接口说明

POST /download/downloadIPFSFile

从IPFS下载文档，仅支持文本数据。如果是从ChainQA业务逻辑上传的数据，那么下载下来应该是用AES加密的。

⚠️本接口不会经过ChainQA的查询逻辑，可供调试。一般不面向应用。应用层请使用查询数据接口进行查询。

3.1.2. 请求参数

名称	位置	类型	必选	说明
body	body	object	否	none
» cid	body	string	是	IPFS地址
» apiUrl	body	object	是	
»» ipfsServiceUrl	body	string	是	IPFS地址

»» chainServiceUrl	body	string	是	区块链SDK（或者说区块链管理平台）地址。
»» contractName	body	string	是	合约名称

3.1.3. 结果返回

状态码	状态码含义	说明	数据模型
200	OK	none	Inline

```
{
    "code": 0,
    "data": "Q+QFhMOin9hKbe6WLU7EF01DFhnnoiYe+GdS2iBPK0d3Ich66ZAAeoKoQq8P7i
cgQy5LtViUxVUTJOKjs3OsJXFueWcQGF0ZS2vI33luHaA3wfK2/+Vv0ifdDzXAXJhDnrCSf8lx1
Np6e5lRJDT045Tm+k+5xpa8ep8TkjjMuxv3UI4ko2tvhF9O2uxL94oXIYWiD3+i2cmbJjCBce4p
nGF8ID6SY6XhsTzXlwKbyctdchrhYFODwTTGFgF3LmT3gEdN8aUP4898fpaHc8zGincqKi+NjL6
9a3h8nf4Yo76aHY/3BAHN2IJbofctpmL9lQSwjYl/YLHm7E/UNFw9QxKTV7LL+jUbdGeD7D0Q9M
lQy2lWk9JnGLhMMOrqq1QO6a+6Yl8hckRyZb5elqcoNDWH+gl8R4RHH4NdfycZE96SG/gr+9D+s
NnitmdIZO5huvN/I3PdIomp0ryiWqGFrj+rCjbFHgAvTciWUW9uVWfZDxQNs8Wwrsa+8jH8eNwL
56rr7yS+Ec8V2ItMiBp5gAdUum7r8UDhgtMCeeL/GYnKiWbDIzACC/bw+6Ss8CNPu2m2L47xJNO
x9Ncq1+Ex9fVfyDOZxCCodPGJNhP4dc2fHDQ08isnz357jqHLLPf3RNAfqm+pe7LW1/dauD6GUi
6wxuwX4RZhgb+hi15GzweBLf3qAicm85pONS42FoRgA/qE/ago2drJMhjh2cBZUbjEO0k9ccpyw
ZK+ncJpe3MNhzm3UrTUmLCiiw1bUJkrOvZHu2GgXQHbM4H0RyekeFe6D/SsxKMkf8PPVE4jrGjm
lboPWaUbiJi2XW6Zvj4edF8+0H/NDidl5Okw1FskzLqn5nCtwmwR1lTj98V2dc5GsTEnEAEGN58
ALrKw2kdw5eDseiJ+ynquznliDXpg4mZAamTKUTaXI050YNb1rnVluUfGGGSGjz5+BWI++TBvt5
c/fQ20GBqQgnDAYRom4foa2SfE2NUDVylHIyD/bWE4kF9I+3XLxW228kO1+uP6ehhcy/5oWWF5S
t5EnjnGIRGfaVpIvG+BTjZixmNoB6bcIQUxidrNXbuZMG/6y51ALjQW0XHg2F4o4c/jez/WKahu
y5D4S/877Ji2fWzWn5vNRAeEjsP7h4P6d7i1ydO7+gPpCRj/FQoA+pnRrwrm3zjIuDiR1UAgzfx
HnxkvQmp1SzJy6Fec8OQ/yZCl5KkUlWW6I5lm4FIZJAp1X0+9WTQiDRu2skSqcxTiC1VVsd81bp
o7l05pbVfVRggRsNTP0qq4TPdBoR1tkoRlAVTB9/xTmVZi7PjWXOutjwZ2oJ0XQYwShIVSJvkcH
lhvV1YCBjtoJYubnpb8pU4N/oMT1HAKu4Qanon+l1p6CI06Fs91zYAYQzFlVOsxmowMwZxNddeC
2uqW2nCU6zP1pDzqOWqPUQflmBHO9MD2Fcjm1JaeakkXPb40h7gvt+oeP8TA4XMcoYNAW88WYNL
M26mpzECXcumHAK5NZ9XMfXGSdjHwHC1ihndVYi3a7pJnzULAye8bvRh2PHXpw/Oe5Np6nIEyO5
3EGB5C53e3JnARmGRyrE48fS9e2Xlu3gmlNAK/ZEPjISRHw9Dd8LFCJn2lvwqMqa9xYTKt/EHZe
SqTQl2/qAPVIWcIDxIJeHgfvEUGwF3TgRl2+GBuChuhPo/tN7N5lzgoJ+iQJ9VG3uZQf3VuTuvG
32wpyxxQWl3ofNhJMRvksEBuJAzt6dDbk/rq5PhVMCcFkPh/ezHY7BseMWZy4W2bt0LXO4Ft/+U
g7GqbPXn7ubuub6KpVru5ltPLOsGiflvkBMrEfUX2Qu0mXpZraXxYCNvZT2PObaQWCqLB4WcWoF
MAsf8SMj932pZoia3jPXE8Y+323W+Awg2lCyRwQStVP3mAqnJXJlBC9nMP1ziu9+vrqCdX3GqKv
DT9AgBsetm7A/yZ1Ikuhbc2dl9XrDL7m6yVCe6fPrJrD/GCW+qUDOWGqx51XzDH0wKLdpmp12A3
KJ1MSrp9IDpmn2E/TElTJXzJwruuRzV96IBknxWfO/fnvemtS10jyrJo4MfFQcXxo1qO+zM7gRs
vusiYCNFgaDlZMo6U9VQCt4I/6BQ+nL4/xuiY+MYHY17L8n2DcKcALryt2zH8vNtNz8DbVYDZBa
wmY+4CzNwqjxXnPRuup0t/94jcnf6pH15kF4v1MV/0YeVjtHR3sbSADWbgHGGl8xo2Wsnkq4HMj
0Our0z4jn/wlbE6uPnC1YmkLL8Rx1iIP7P05oTq0AV0a0b/wOk5LTvmBAiG8H5dsJv+iulJCR6J
guQROqFzURxQLTt+ckd1wvbQQ9/SCUtGXFe9CdxHwhF0maY9AOhUDLhxOMlnVP0wmfpeKYa5B7C
A/Pcq7L3QKU/KGR4D6RRlmJDGV6MHMvlKShp8TPAWjDxEs58dE8CMwhhKtnhYXU3Ex7bd1I3Qbq
OjWuDTygnyxykmXAP0xEpjNYqqakq5YJrsDiw8QCIiGDqrsvuuJUZ3RxtSxQ28jHDQPjrXv0nrD
HkWIA9gJ/M2mkClx34qJDhi/ZS2j2mz5rypvcWp872WiKgbj5v3lb9AYEeRdozaonodyUTyCzeh
8NtD61tx5LySDRTPh/iZKsR2rFpsskXYtYAcJvcybiPcakQU48o7+546j+GHFDdW05qzFI6xP8n
2ksmkIlvTaaXj+WbVz+Uffs80Qb5u3aq5BBpoSULqTpGlENXBEJ6ege5c6WARvsc00qmmJbt2ws
1+AYqsL63jWWXex4L2/lXyfhgyccIEoq2K01JxWj/DcP145o2cZVhDF1GXzwyhPVTyj0Nb/GLNi
+41tmhatqLZDTMC8U6s5MXpsnoZJSNGW0YlVTynPb6KKSqaP1NntvXGEfpdX+LzbMO1H7BgablW
aujUK4CC2eT7UU8vVucVE8zV+7SDpQdRxx3pqOqlq9bbhyCx30hV+HXxgYnnwEbZ+nsw/VL/Mpc
=",
    "msg": "获取IPFS文件（密文）成功"
}
```

3.2. 尝试解密

3.2.1. 接口说明

`POST` /download/tryDecryptFile

基本同下载IPFS文件接口。
只是这个接口从IPFS下载数据后，会使用你提供的AES密钥进行解密。

⚠本接口不会经过ChainQA的查询逻辑，可供调试（比如观察某个数据是否上传成功，如果上传成功+解密成功，很可能是查询模块的逻辑出错了）。一般不面向应用。应用层请使用查询数据接口。

3.2.2. 请求参数

名称	位置	类型	必选	说明
body	body	object	否	none
» aesKey	body	string	是	解密密钥
» cid	body	string	是	IPFS 地址
» apiUrl	body	object	是	none
»» ipfsServiceUrl	body	string	是	IPFS地址
»» chainServiceUrl	body	string	是	区块链SDK（或者说区块链管理平台）地址。
»» contractName	body	string	是	合约名称

3.2.3. 结果返回

状态码	状态码含义	说明	数据模型
200	OK	none	Inline

```
{
    "code": 0,
    "data": "mail\nzsyu@alu.uestc.edu.cn\nlsy@alu.uestc.edu.cn\nzhouhao@alu.uestc.edu.cn\nzql@alu.uestc.edu.cn\nchenxinyi@alu.uestc.edu.cn\nzzr@alu.uestc.edu.cn\nlinchen@alu.uestc.edu.cn\nwangxuan@alu.uestc.edu.cn\nyjy@alu.uestc.edu.cn\npengbo@alu.uestc.edu.cn\nwjj@alu.uestc.edu.cn\nzhaoweichao@alu.uestc.edu.cn\nzhangzhuo@alu.uestc.edu.cn\nzhouhao@alu.uestc.edu.cn\nsyh@alu.uestc.edu.cn\njxh@alu.uestc.edu.cn\nlyh@alu.uestc.edu.cn\nccj@alu.uestc.edu.cn\nzhengyu@alu.uestc.edu.cn\nliuyang@alu.uestc.edu.cn\nhuangqian@alu.uestc.edu.cn\nlxy@alu.uestc.edu.cn\nchenxinyi@alu.uestc.edu.cn\njiangjy@alu.uestc.edu.cn\nyzx@alu.uestc.edu.cn\nyxy@alu.uestc.edu.cn\nzhouxin@alu.uestc.edu.cn\nwusijia@alu.uestc.edu.cn\nzsj@alu.uestc.edu.cn\nyangshuai@alu.uestc.edu.cn\nxujiayi@alu.uestc.edu.cn\ntiantao@alu.uestc.edu.cn\nwxl@alu.uestc.edu.cn\nruiwang@alu.uestc.edu.cn\nasuka@alu.uestc.edu.cn\nliuchang@alu.uestc.edu.cn\nhaoyunliu@alu.uestc.edu.cn\nliudn@alu.uestc.edu.cn\nlinxiong@alu.uestc.edu.cn\nyxb@alu.uestc.edu.cn\nliutao@alu.uestc.edu.cn\ncyx@alu.uestc.edu.cn\nluoyi@alu.uestc.edu.cn\ncoco@alu.uestc.edu.cn\nwzy@alu.uestc.edu.cn\nxiaozhen@alu.uestc.edu.cn\nliyan@alu.uestc.edu.cn\nzyk@alu.uestc.edu.cn\nliuyuchen@alu.uestc.edu.cn\nchenjing@alu.uestc.edu.cn\nzpf@alu.uestc.edu.cn\nnicole@alu.uestc.edu.cn\nwdj@alu.uestc.edu.cn\nxuzijing@alu.uestc.edu.cn\nluyang@alu.uestc.edu.cn\nhyj@alu.uestc.edu.cn\nyyy@alu.uestc.edu.cn\nzjp@alu.uestc.edu.cn\nlyc@alu.uestc.edu.cn\nyangrui@alu.uestc.edu.cn\nlc@alu.uestc.edu.cn\nlsy@alu.uestc.edu.cn\nzzx@alu.uestc.edu.cn\nyudong@alu.uestc.edu.cn\nwt@alu.uestc.edu.cn\n202221060430@std.uestc.edu.cn\nzxy@alu.uestc.edu.cn\nlxd@alu.uestc.edu.cn\njqli@std.uestc.edu.cn\nabby@alu.uestc.edu.cn\nzhy136@std.uestc.edu.cn\ntiansr@std.uestc.edu.cn\n202221020431@std.uestc.edu.cn\nhanfeng@std.uestc.edu.cn\nliqian@alu.uestc.edu.cn\nyanyu@alu.uestc.edu.cn\nzhujiahui@alu.uestc.edu.cn\nzhouyp@alu.uestc.edu.cn\njzy@alu.uestc.edu.cn\nwjl@alu.uestc.edu.cn\nljp@alu.uestc.edu.cn\nhuhong@alu.uestc.edu.cn\nliuzejia@alu.uestc.edu.cn\nzjy@alu.uestc.edu.cn\nlxl@alu.uestc.edu.cn\nxyl@alu.uestc.edu.cn\nwangwentao@alu.uestc.edu.cn\ngzy@alu.uestc.edu.cn\nym@alu.uestc.edu.cn\nliuyh@alu.uestc.edu.cn\nyjh@alu.uestc.edu.cn\nliukai@alu.uestc.edu.cn\nyy@alu.uestc.edu.cn\nlzr@alu.uestc.edu.cn\nwzy@alu.uestc.edu.cn",
    "msg": "解密文件（明文）成功"
}
```


4. 查询

4.1. 查询数据

POST /query/queryData

4.1.1. 接口说明

ChainQA的核心接口，用于查询数据。

基本逻辑如下

1 **Step 1**

获取参数。从QueryItem中解析要查询的所有pos

2 **Step 2**

调取所有的pos，从IPFS下载加密的数据集。

3 **Step 3**

从区块链中调取所有pos的数字信封

4 **Step 3**

用区块链节点私钥SK解密数字信封，得到AES密钥，用AES密钥解密刚刚从IPFS下载的被加密的数据集

5 **Step 4**

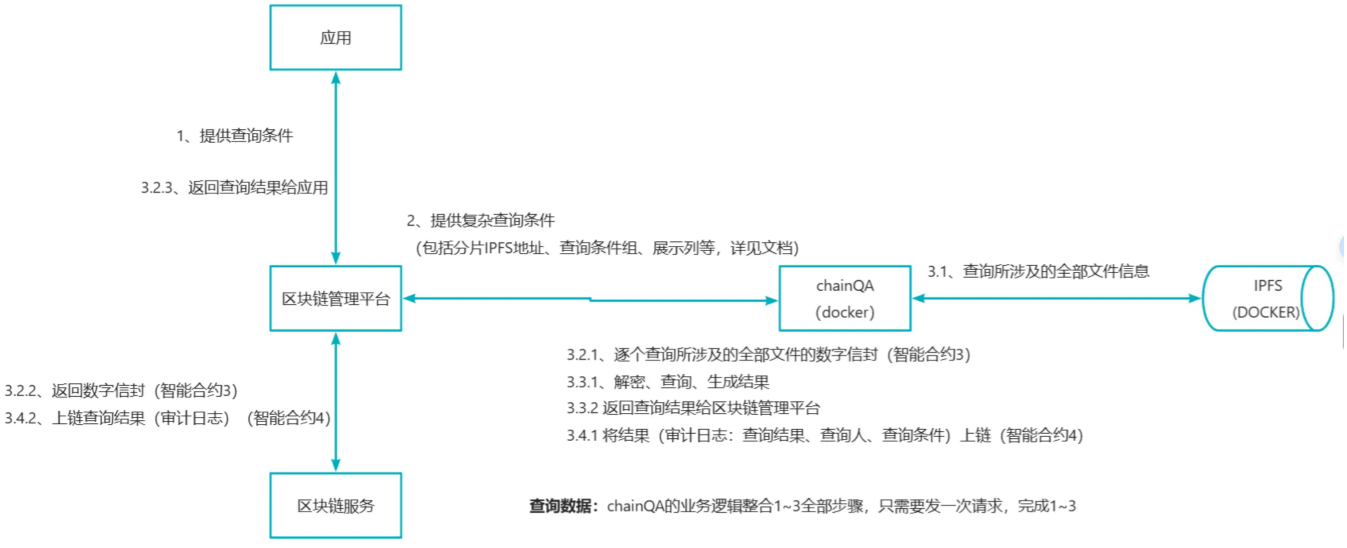
根据QueryItem进行查询

6 **Step 5**

生成查询结果，上链查询日志

7 **Step 6**

返回查询结果



2025年7月8日

关于参数QueryItem

JSON

```
{
  "queryConcatType": "single",
  "filePos": [
    [
      "QmZwbqchGGkxsGdM7MuX7yicwzVUV5V5ZHe2iqfQu8xZy5",
      "QmT93D2EtFmZpvwh8YoRwpzExCd2dFfNHrZx7H7jbArb81"
    ],
    [
        "QmT93D2EtFmZpvwh8YoRwpzExCd2dFfNHrZx7H7jbArb81"
    ]
  ],
  "returnFeild": [
    "QmZwbqchGGkxsGdM7MuX7yicwzVUV5V5ZHe2iqfQu8xZy5_Cscore",
    "QmZwbqchGGkxsGdM7MuX7yicwzVUV5V5ZHe2iqfQu8xZy5_id",
    "QmT93D2EtFmZpvwh8YoRwpzExCd2dFfNHrZx7H7jbArb81_Mscore"
  ],
  "queryConditions": [
    [
      {
        "field": "name",
        "pos": "QmT93D2EtFmZpvwh8YoRwpzExCd2dFfNHrZx7H7jbArb81",
        "compare": "prefix",
        "val": "大",
        "type": "string"
      },
      {
        "field": "id",
        "pos": "QmT93D2EtFmZpvwh8YoRwpzExCd2dFfNHrZx7H7jbArb81",
        "compare": "lt",
        "val": "12",
        "type": "float"
      }
    ]
  ],
  "jointConditions": [
    {
      "pos1": "QmZwbqchGGkxsGdM7MuX7yicwzVUV5V5ZHe2iqfQu8xZy5",
      "field1": "id",
      "pos2": "QmZwbqchGGkxsGdM7MuX7yicwzVUV5V5ZHe2iqfQu8xZy5",
      "field2": "id",
      "compare": "eq",
      "type": "string",
      "jointType": "INNER"
    }
  ]
```

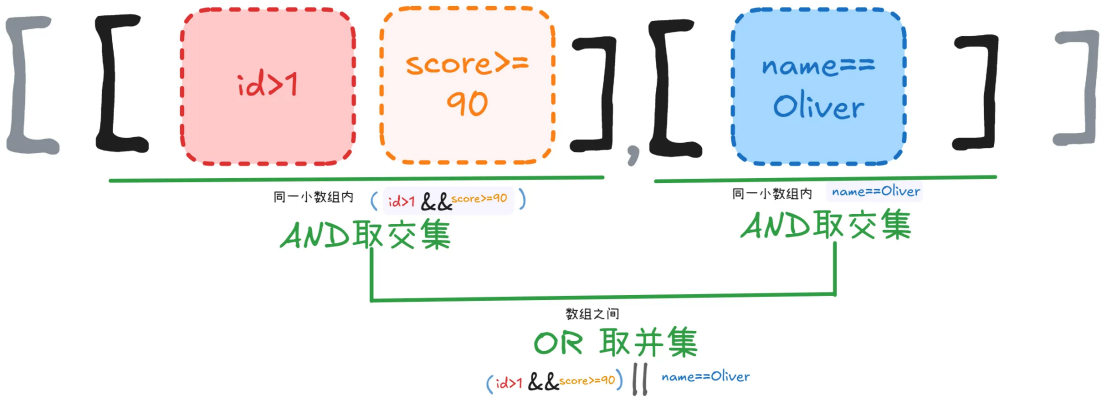
```
46    }
```

- "queryConcatType"：查询类型
  - single——单表查询
  - multi——联表查询
- "FilePos"：查询文件所在 IPFS 的哈希值。
  - 是一个二维数组。每一个维度的元素代表**同一个数据集**的**不同分片**
  - 例如 [[A1,A2],[B]]：其中 A1、A2 结构需要完全相同，合起来为一整个数据集 A
- "ReturnFeild"为元素为字符串的数组，存储了**需要返回（展示）的列名**。
  - 为了避免在联表查询时，联表有两个同名但不同意思的列。列名带前缀：**所属数据集**的 ipfs 地址。
    - 例如：`QmX798NJ5UrTwkRKNwwvg7goD4JwiXY8SA8zs8qoxLJcQP_name` 就标识 `QmX798NJ5UrTwkRKNwwvg7goD4JwiXY8SA8zs8qoxLJcQP` 的 `name` 列
    - 若同一个数据集有多个分片，以FilePos 所对应**数据集**第一个分片为前缀！
  - 若为空，则返回的 data 中不会展示数据，但是返回的counts 会记录满足条件的行数
  - 若列名为 `前缀_*`(例如 `QmX798NJ5UrTwkRKNwwvg7goD4JwiXY8SA8zs8qoxLJcQP_*`)就代表返回该数据集的**所有列**
- "QueryConditions"组成元素为一个二维数组。如下所示
  - 数组内部：多个对象内部统一采用 `{"field": "XXX", "val": "XXX", "compare": "XX", "type":"XXXX"}` 的形式
    - 标识说明
      - field 标识列名
      - val 标识基准值
      - compare 标识运算规则
        - 公共规则
          - eq：相等
          - ne：不等
          - gt：> （string 类型将按字典序进行比较）——即寻找**目标值** `大于` **基准值**的行
          - lt：< （string 类型将按字典序进行比较）
          - ge：>=
          - le：<=
        - int/float 类型独有规则

- 无
- string 类型独有规则
  - regexp：正则表达式匹配（程序会判断正则是否正确）
  - contain：是否包含某个子串
  - suffix：后缀
  - prefix：前缀
- 非对应规则（如在 int 中使用 regexp）会默认判不符合。

- type 标识列类型（按什么类型比较）【注：对于数据列的类型可灵活变更，因此类型的选择是在查询时根据需求传入，以便进行比较。例如，学号的数字列可以根据需要灵活处理：既可以作为“int”类型，按照整数的大小进行比较；也可以作为“string”类型，进行正则匹配。】
  - string（按字典序比较）
  - int（按整数值比较）
  - float（按浮点数进行比较）

- 举例
  - `{"field": "id", "val": "2", "compare": "gt","type":"int"}` 标识在 id 列寻找 id 大于基准值 2 的数据，数据以 int 存储比较

- 数组**之间**：采用 OR 进行逻辑查询，数组**内部**采用 AND 进行逻辑查询
  - 例如：`(id>1 && score >=90) || (name =="Oliver")` 的**行记录**。
  - 

> 错误处理：函数已经实现——IPFS 文件不存在、IPFS 文件解析错误、列名不存在、类型不匹配（比如 string 类型列标明为 int）、正则表达式不正确、运算符不正确 的错误处理。发生错误时，会在查询结果的 message 中提到，不会中断本次背书逻辑。

- "jointConditions"联表条件数组。每个<pos1、pos2>组只能在联表条件中出现一次。(即不允许定义 A.id=B.index，然后又定义 A.idx=B.name，**A-B 连接的条件**只能出现**一对**)

- 必须填 `pos1` 、 `pos2` 、 `feild1` 、 `feild2` 指定**联表列**
- compare 联表比较条件（判断为真，行数据进行连）、type 列类型（按什么类型比较）
- jointType 可选 `INNER` / `LEFT` / `RIGHT`

4.1.2. 请求参数

名称	位置	类型	必选	说明
body	body	object	否	none
» uId	body	string	是	查询者 id 或名称（仅限英文字母/数字）
» queryItem	body	string	是	参加本文档第一部分
» apiUrl	body	object	是	none
»» ipfsServiceUrl	body	string	是	IPFS地址
»» chainServiceUrl	body	string	是	区块链SDK（或者说区块链管理平台）地址。
»» contractName	body	string	是	合约名称

```
// 例子:
{
  "uId": "test11213123",
  "queryItem": "{\"queryConcatType\":\"single\",\"filePos\":[[\"QmTizVQtz6aGMKm1QnSsAZTxrbLU2PvqZWy4dGmXJ1G8VF\"]],\"returnField\":[],\"queryConditions\":[]}",
  "apiUrl": {
    "ipfsServiceUrl": "http://47.113.204.64:5001",
    "chainServiceUrl": "http://47.113.204.64:9001/tencent-chainapi/exec",
    "contractName": "tencentChainqaContractV221demo01"
  }
}
```

4.1.3. 返回结果

状态码	状态码含义	说明	数据模型
200	OK	none	Inline

> 返回示例

```
{
  "code": 0,
  "data": "{\"counts\":8,\"data\":[{\"Cscore\":\"9\",\"name\":\"大兰\"},{\"Cscore\":\"9\",\"name\":\"大黑\"},{\"Cscore\":\"9\",\"name\":\"大橙\"},{\"Cscore\":\"9\",\"name\":\"大黄\"},{\"Cscore\":\"9\",\"name\":\"小兰\"},{\"Cscore\":\"9\",\"name\":\"小黑\"},{\"Cscore\":\"9\",\"name\":\"小橙\"},{\"Cscore\":\"9\",\"name\":\"小黄\"}],\"message\":\"查询成功\"}",
  "msg": "查询成功"
}
```

5. 审计

5.1. 查询UID的所有日志

POST /log/logByUid

5.1.1. 接口说明

uId不能以“__”结尾

5.1.2. 请求参数

名称	位置	类型	必选	说明
body	body	object	否	none
» uId	body	string	是	要查询的查询者
» apiUrl	body	object	是	none
»» ipfsServiceUrl	body	string	是	IPFS地址
»» chainServiceUrl	body	string	是	区块链SDK（或者说区块链管理平台）地址。
»» contractName	body	string	是	合约名称

5.1.3. 结果返回

状态码	状态码含义	说明	数据模型
200	OK	none	Inline

> 返回示例

```
{
    "code": 0,
    "data": "{\"Count\":1,\"QueryLogArray\":[{\"QueryId\":\"b7e9d99a03d55f6
473d5317a3d5d0cc7\",\"Uid\":\"test11213123\",\"Timestamp\":\"1751982788
\",\"QueryItem\":\"{\\\"queryConcatType\\\":\\\"single\\\",\\\"filePos\\\":
[[\\\"QmTizVQtz6aGMKm1QnSsAZTxrbLU2PvqZWy4dGmXJ1G8VF\\\"]],\\\"returnField
\\\":[],\\\"queryConditions\\\":[]}\",\"QueryStatus\":95,\"QueryResult\":\"
{\\\"counts\\\":95,\\\"data\\\":[{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{}],\\\"message\\\":\\\"查询成功\\\"}\"}]}",
    "msg": "查询成功"
}
```


其中 data 中的 queryLogArray 是日志数组，每个元素是一个查询结果

- **QueryResult**：
  - 即返回用户的查询结果，采用 JSON 格式封装。
  - counts：结果数量，-1 标识查询错误，具体请见：QueryStatus
  - data：查询结果，数组，内部元素封装为结果，采用键值对形式（键为列名，值为单元格值）。数组每个元素（`{}` 包裹得对象）为一行
  - message：信息，主要是发生错误时供查阅

5.2. 查询时间范围的所有日志

`POST` /log/logByTimeRange

5.2.1. 接口说明

startTime和endTime必须是**秒级时间戳**格式

> 错误的时间戳会导致程序发生未知的错误。
> 什么是时间戳请参考：http://shijianchuo.wiicha.com/
> 例如：“1735742171”是一个正确的时间戳

5.2.2. 请求参数

名称	位置	类型	必选	说明
body	body	object	否	none

» apiUrl	body	object	是	none
»» ipfsServiceUrl	body	string	是	IPFS地址
»» chainServiceUrl	body	string	是	区块链SDK（或者说区块链管理平台）地址。
»» contractName	body	string	是	合约名称
» startTime	body	string	是	秒级时间戳
» endTime	body	string	是	秒级时间戳


```
{
  "startTime": "1751817600",
  "endTime": "1751990399",
  "apiUrl": {
    "ipfsServiceUrl": "http://47.113.204.64:5001",
    "chainServiceUrl": "http://47.113.204.64:9001/tencent-chainapi/exec",
    "contractName": "tencentChainqaContractV221demo01"
  }
}
```


5.2.3. 结果返回

状态码	状态码含义	说明	数据模型
200	OK	none	Inline

> 返回示例

```
{
    "code": 0,
    "data": "{\"Count\":1,\"QueryLogArray\":[{\"QueryId\":\"b7e9d99a03d55f6
473d5317a3d5d0cc7\",\"Uid\":\"test11213123\",\"Timestamp\":\"1751982788
\",\"QueryItem\":\"{\\\"queryConcatType\\\":\\\"single\\\",\\\"filePos\\\":
[[\\\"QmTizVQtz6aGMKm1QnSsAZTxrbLU2PvqZWy4dGmXJ1G8VF\\\"]],\\\"returnField
\\\":[],\\\"queryConditions\\\":[]}\",\"QueryStatus\":95,\"QueryResult\":\"
{\\\"counts\\\":95,\\\"data\\\":[{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},{},
{},{},{},{},{},{}],\\\"message\\\":\\\"查询成功\\\"}\"}]}",
    "msg": "查询成功"
}
```