八爪鱼数据导出 API 开发者文档

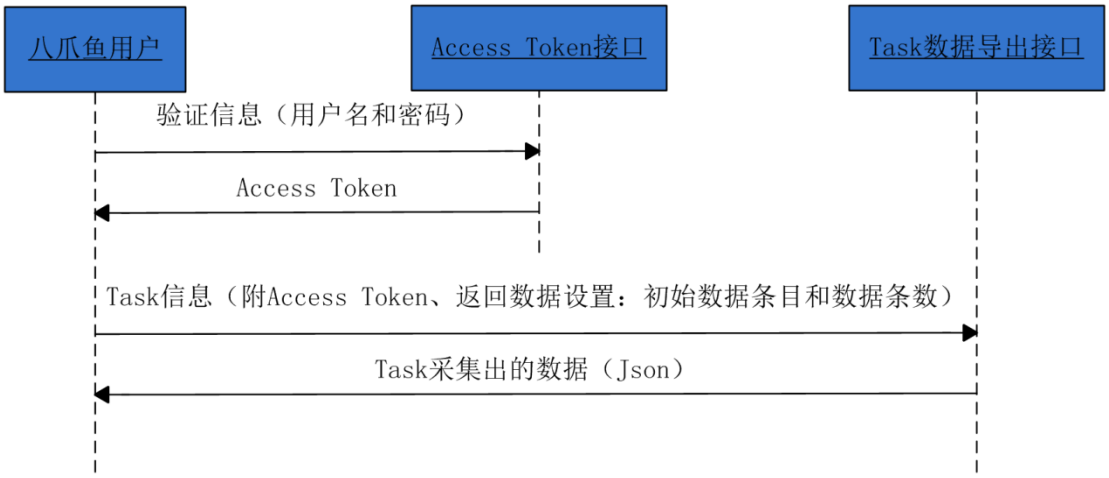
目录

1. 概述

开发者需要按以下流程来使用八爪鱼数据 API 获取采集的数据。在此之前，您应该拥有一个八爪鱼数据平台的账号，并且建立了能够正常采集数据的任务。

API 使用的基本流程如下：

- 通过用户名和密码获取 Access Token
- 在采集服务器中使用 Access Token 和任务 ID（taskId）来获取任务采集的数据



2. 获取 Access Token

使用如下接口并提供用户名和密码来获取 Access Token。

```
http 请求方式: POST
http://dataapi.bazhuayu.com/token
POST 表单数据格式: application/x-www-form-urlencoded
POST 表单参数示例:
username={username}&password={password}&grant_type=password
POST 表单数据中 username、password 的值需要进行 url 编码。
```

以上请求如果用户名和密码验证通过，Http Response 会返回类似如下的 Json 格式文本。

```
{
    "access_token": "ABCD1234",
    "token_type": "bearer",
    "expires_in": 86399,
    "refresh_token": "refresh_token"
}
```

返回结果中各个字段的说明如下：

字段	说明
access_token	访问标识 access token
token_type	access token 类型，目前为 bearer
expires_in	access token 过期时间（秒），当前 为 24 小时，即 86400 秒
refresh_token	在 access token 过期后刷新 access token 的一个标记

Access Token 是以下所有 API 访问的访问许可标志，在获取数据的 API 中务必加入到 Http 请求的头文件中。

```
Name: Authorization
Value: bearer {access token}
```

如果获取 Access Token 失败会返回若干种 Json 格式文本，每种错误原因和 Json 格式解

释如下。

1. POST 数据格式不正确，要保证格式为：

```
username={username}&password={password}&grant_type=password
```

```
{
    "error": "unsupported_grant_type"
}
```

2. POST 数据中用户名和密码不匹配

```
{
    "error": "invalid_grant",
    "error_description": "The user name or password is incorrect."
}
```

注：参考示例代码 5.1。

3. 调用数据导出 API

3.1. 分页获取任务所有数据

使用此接口来分页获取任务的数据，接口需要传入任务 ID（TaskId）及分页信息（pageindex、pagesize），同时还需要在 Header 中添加 Access Token 信息。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/alldata?taskid={taskid}&pageindex={page
Index}&pagesize={pageSize}
Http 头文件参数 1:
    Name: Authorization
    Value: bearer {access token}
Http 头文件参数 2:
    Name: Accept
    Value: application/json
HTTP URL 参数示例:
http://dataapi.bazhuayu.com/api/alldata?taskid=taskid&pageindex=1&pages
ize=2
```

根据给定的 pagesize(pagesize 最大为 1000)对当前任务的数据进行分页，然后在所有分页中返回 pageindex 这一页数据，共返回 pagesize 条数据。例如某任务有 1000 条数据，若传入 pagesize=2，pageindex=1，则将数据分为 1000/2=500 页，每页 2 条数据，然后返回第 1 页的 2 条数据。若以上请求中 Access Token 和 TaskId 均合法，将会得到类似如下 Json 格式的任务数据：

```
{
    "data": {
        "total": 1000,
        "currentTotal": 2,
        "dataList": [
```

```
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-1",
                "最高气温": "0℃",
                "天气状况": "多云",
                "风力风向": "东北风微风",
                "最低气温": ""
            },
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-2",
                "最高气温": "5℃",
                "天气状况": "多云转阴",
                "风力风向": "东北风 3～4 级",
                "最低气温": "-2℃"
            }
        ]
    },
    "error": "success"
}
```


返回的数据有如下字段：

字段	解释
total	当前任务的数据记录总数
currentTotal	此次请求的数据记录数
dataList	数据记录集
error	提示信息，若为 success 则表示成功

注：参考示例代码 5.4。

3.2. 分时分页获取任务所有数据

使用此接口来分页获取任务的数据，接口需要传入任务 ID（TaskId），采集起止时间（from、to）及分页信息（pageindex、pagesize），同时还需要在 Header 中添加 AccessToken 信息。

```
http 请求方式：GET
http://dataapi.bazhuayu.com/api/alldata/getdataoftaskbytimeandpaging?taskid={taskid}&from={from}&to={to}&pageindex={pageIndex}&pagesize={pageSize}
Http 头文件参数 1：
    Name: Authorization
    Value: bearer {access token}
Http 头文件参数 2：
    Name: Accept
    Value: application/json
```

```
HTTP URL 参数示例
http://dataapi.bazhuayu.com/api/alldata/getdataoftaskbytimeandpaging?taskid=taskid&from=2017-02-21 11:00:00&to=2017-02-21 12:00:00&pageindex=1&pagesize=2
```

根据采集起止时间 from（采集开始时间），to（采集结束时间），开始时间和结束时间两者间隔不能超过 1 小时，且必须为“yyyy-MM-dd HH:mm:ss”格式，以及给定的 pagesize(pagesize 最大为 1000)对当前任务的数据进行分页，然后返回在 from 到 to 这个时间段内采集到的所有数据中，分页为 pageindex 这一页的数据，共返回 pagesize 条数据。例如某任务有 1000 条数据，若传入 pagesize=2，pageindex=1，则将数据分为 1000/2=500 页，每页 2 条数据，然后返回第 1 页的 2 条数据。若以上请求中 Access Token 和 TaskId 均合法，将会得到类似如下 Json 格式的任务数据：


```
{
    "data": {
        "total": 1000,
        "currentTotal": 2,
        "dataList": [
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-1",
                "最高气温": "0℃",
                "天气状况": "多云",
                "风力风向": "东北风微风",
                "最低气温": ""
            },
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-2",
                "最高气温": "5℃",
                "天气状况": "多云转阴",
                "风力风向": "东北风 3～4 级",
                "最低气温": "-2℃"
            }
        ]
    },
    "error": "success"
}
```


返回的数据有如下字段：

字段	解释
total	当前任务的数据记录总数
currentTotal	此次请求的数据记录数

dataList	数据记录集
error	提示信息，若为 success 则表示成功

3.3. 分页获取任务最近一次启动时采集到的数据

使用此接口来分页获取任务的数据，接口需要传入任务 ID（TaskId）及分页信息（pageindex、pagesize），同时还需要在 Header 中添加 AccessToken 信息。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/alldata/getlastdataoftaskbypaging?taskid={taskid}&pageindex={pageIndex}&pagesize={pageSize}
Http 头文件参数 1:
    Name: Authorization
    Value: bearer {access token}
Http 头文件参数 2:
    Name: Accept
    Value: application/json
HTTP URL 参数示例:
http://dataapi.bazhuayu.com/api/alldata/getlastdataoftaskbypaging?taskid=taskid&pageindex=1&pagesize=2
```

根据给定的 pagesize(pagesize 最大为 1000)对当前任务的数据进行分页，然后在所有分页中返回 pageindex 这一页数据，共返回 pagesize 条数据。例如某任务启动云采集 3 次，分别采集了 500，800，1000 条数据。若传入 pagesize=2，pageindex=1，则将第 3 次采集到的数据分为 1000/2=500 页，每页 2 条数据，然后返回第 3 次采集中第 1 页的 2 条数据。若以上请求中 Access Token 和 TaskId 均合法，将会得到类似如下 Json 格式的任务数据：

```
{
    "data": {
        "total": 1000,
        "currentTotal": 2,
        "dataList": [
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-1",
                "最高气温": "0℃",
                "天气状况": "多云",
                "风力风向": "东北风微风",
                "最低气温": ""
            },
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-2",
                "最高气温": "5℃",
```

```
                "天气状况": "多云转阴",
                "风力风向": "东北风 3～4 级",
                "最低气温": "-2℃"
            }
        ]
    },
    "error": "success"
}
```


返回的数据有如下字段：

字段	解释
total	当前任务的数据记录总数
currentTotal	此次请求的数据记录数
dataList	数据记录集
error	提示信息，若为 success 则表示成功

3.4. 获取任务未导出的数据

使用此接口来分批次获取任务未导出的数据，接口需要传入任务 ID（TaskId）及每批返回的数据条数（size），同时还需要在 Header 中添加 Access Token 信息，接口将会返回先采集到的数据。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/notexportdata?taskid={testtaskid}&size={size}
Http 头文件参数 1:
    Name: Authorization
    Value: bearer {access token}
Http 头文件参数 2:
    Name: Accept
    Value: application/json
HTTP URL 参数示例:
http://dataapi.bazhuayu.com/api/notexportdata?taskid=testtaskid&size=100
```

接口返回 size 条未导出的数据，同时把返回的数据标识为已导出，下次调用这个接口时将会跳过这部分已导出的数据。例如某任务有 1000 条数据，若第一次调用传 size =2，返回 2 条未导出的数据，第二次调用的时候将会返回剩下 998 中的 2 条数据。若以上请求中 Access Token 和 TaskId 均合法，将会得到类似如下 Json 格式的任务数据：


```
{
    "data": {
        "total": 1000,
        "currentTotal": 2,
        "dataList": [
```

```
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-1",
                "最高气温": "0℃",
                "天气状况": "多云",
                "风力风向": "东北风微风",
                "最低气温": ""
            },
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-2",
                "最高气温": "5℃",
                "天气状况": "多云转阴",
                "风力风向": "东北风 3～4 级",
                "最低气温": "-2℃"
            }
        ]
    },
    "error": "success"
}
```


返回的数据有如下字段：

字段	解释
total	当前任务的未导出的数据记录总个数
currentTotal	此次请求的数据记录个数
dataList	数据记录集
error	提示信息，若为 success 则表示成功

注：一、参考示例代码 5.5。

二、获取任务数据时若提供的参数不合法，会返回如下错误结果，每种错误的原因和 json 格式解释如下。

1. Access Token 无效，有可能过期，请使用正确的用户名和密码重新获取 Access Token


```
{
    "error": "unauthorized",
    "error_Description": "access_token 无效"
}
```


2. 采用了 POST 方法，请使用 GET 方法


```
{
    "message": "请求的资源不支持 http 方法“POST”。"
}
```


3. taskId 错误或该任务不属于 Access Token 对应的用户，请使用正确的 taskId


```
{
```

```
    "error": "taskid_error",
    "error_Description": "taskId 错误或该 task 不属于您"
}
```

4. size 太大，size 不允许超过 MaxSize 的值，此值目前系统设置为 1000

```
{
    "error": "export_pagesize_error",
    "error_Description": "size 限制在 1 到 1000"
}
```

5. 服务器暂时不可用

```
{
    "error": "server_error",
    "error_Description": "服务器错误，请稍后再试！"
}
```

3.5. 获取任务未导出的数据安全接口 A

使用此接口用来分批次获取任务未导出的数据，但不更新导出状态，接口需要传入任务 ID（TaskId）及每批返回的数据条数（size），同时还需要在 Header 中添加 Access Token 信息，接口将会返回先采集到的数据。

此接口配合下面获取任务未导出的数据安全接口 B 一起使用，调用顺序为 AB、AB，若连续调用 AA，会出现重复数据。

```
http 请求方式：GET
http://dataapi.bazhuayu.com/api/notexportdata/gettop?taskid={testtaskid}&size={size}
Http 头文件参数 1：
    Name：Authorization
    Value：bearer {access token}
Http 头文件参数 2：
    Name：Accept
    Value：application/json
HTTP URL 参数示例：
http://dataapi.bazhuayu.com/api/notexportdata/gettop?taskid=testtaskid&size=100
```

接口返回 size 条未导出的数据，同时把返回的数据标识为已导出，下次调用这个接口时将会跳过这部分已导出的数据。例如某任务有 1000 条数据，若第一次调用传 size =2，返回 2 条未导出的数据，第二次调用的时候将会返回剩下 998 中的 2 条数据。若以上请求中 Access Token 和 TaskId 均合法，将会得到类似如下 Json 格式的任务数据：

```
{
    "data": {
        "total": 1000,
```

```
        "currentTotal": 2,
        "dataList": [
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-1",
                "最高气温": "0℃",
                "天气状况": "多云",
                "风力风向": "东北风微风",
                "最低气温": ""
            },
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-2",
                "最高气温": "5℃",
                "天气状况": "多云转阴",
                "风力风向": "东北风 3～4 级",
                "最低气温": "-2℃"
            }
        ]
    },
    "error": "success"
}
```


返回的数据有如下字段：

字段	解释
total	当前任务的未导出的数据记录总个数
currentTotal	此次请求的数据记录个数
dataList	数据记录集
error	提示信息，若为 success 则表示成功

注：

获取任务数据时若提供的参数不合法，会返回如下错误结果，每种错误的原因和 json 格式解释如下。

6. Access Token 无效，有可能过期，请使用正确的用户名和密码重新获取 Access Token


```
{
    "error": "unauthorized",
    "error_Description": "access_token 无效"
}
```


7. 采用了 POST 方法，请使用 GET 方法


```
{
    "message": "请求的资源不支持 http 方法“POST”。"
}
```

8. taskId 错误或该任务不属于 Access Token 对应的用户，请使用正确的 taskId

```
{
    "error": "taskid_error",
    "error_Description": "taskId 错误或该 task 不属于您"
}
```

9. size 太大，size 不允许超过 MaxSize 的值，此值目前系统设置为 1000

```
{
    "error": "export_pagesize_error",
    "error_Description": "size 限制在 1 到 1000"
}
```

10. 服务器暂时不可用

```
{
    "error": "server_error",
    "error_Description": "服务器错误，请稍后再试！"
}
```

3.6. 获取任务未导出的数据安全接口 B

使用 updata 接口将正在导出的数据更改为已导出状态

```
http 请求方式：POST
http://dataapi.bazhuayu.com/api/notexportdata/update?taskid={taskid}
Http 头文件参数 1：
    Name：Authorization
    Value：bearer {access token}
Http 头文件参数 2：
    Name：Accept
    Value：application/json
HTTP URL 参数示例：
http://dataapi.bazhuayu.com/api/notexportdata/update?taskid=taskid
```

在以上请求中若 access token 合法，参数传递正确将会得到类似如下的任务组列表结构体文本：

```
true
```

若没有返回正确的数据，那返回的数据有以下几种错误提示：

1. Access Token 无效，有可能过期，请使用正确的用户名和密码重新获取 Access Token

```
false
```

2. 采用了 GET 方法，请使用 POST 方法

```
{
    "message": "请求的资源不支持 http 方法“GET”。"
}
```

3. taskId 错误或该任务不属于 Access Token 对应的用户，请使用正确的 taskId

```
{
    "error": "taskid_error",
    "error_Description": "taskId 错误或该 task 不属于您"
}
```

4. 服务器暂时不可用

```
{
    "error": "server_error",
    "error_Description": "服务器错误，请稍后再试"
}
```

3.7. 清理任务数据

使用 RemoveData 接口并在 Header 中提供 Access Token 验证信息、任务 ID 来清理任务数据。

```
http 请求方式: POST
http://dataapi.bazhuayu.com/api/task/RemoveDataByTaskId?taskid={taskid}
Http 头文件参数 1:
    Name: Authorization
    Value: bearer {access token}
Http 头文件参数 2:
    Name: Accept
    Value: application/json
HTTP URL 参数示例:
http://dataapi.bazhuayu.com/api/task/RemoveDataByTaskId?taskid=taskid
```

在以上请求中若 access token 合法，参数传递正确将会得到类似如下的任务组列表结构体文本：

```
{
    "error": "Success",
    "error_Description": "该任务的数据已经清理完毕"
}
```

若没有返回正确的数据，那返回的数据有以下几种错误提示：

5. Access Token 无效，有可能过期，请使用正确的用户名和密码重新获取 Access Token

```
{
    "error": "unauthorized",
    "error_Description": "access_token 无效"
}
```

6. 采用了 GET 方法，请使用 POST 方法

```
{
    "message": "请求的资源不支持 http 方法“GET”。"
}
```

7. taskId 错误或该任务不属于 Access Token 对应的用户，请使用正确的 taskId

```
{
    "error": "taskid_error",
    "error_Description": "taskId 错误或该 task 不属于您"
}
```

8. 服务器暂时不可用

```
{
    "error": "server_error",
    "error_Description": "服务器错误，请稍后再试"
}
```

3.8. 根据偏移量获取任务数据

此接口根据偏移量来获取任务的数据，接口需要传入任务 ID（TaskId），偏移量（Offset）以及获取的数据数量（Size），同时还需要在 Header 中添加 AccessToken 信息。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/alldata/getdataoftaskbyoffset?taskid={taskid}&offset={offset}&size={size}
Http 头文件参数 1:
    Name: Authorization
    Value: bearer {access token}
Http 头文件参数 2:
    Name: Accept
    Value: application/json
HTTP URL 参数示例:
http://dataapi.bazhuayu.com/api/alldata/getdataoftaskbyoffset?taskid=taskid&offset=20&size=10
```

根据给定的 offset 作为偏移量读取当前任务的 size（size 最大为 1000）条数据， 当 offset 小等于 0 时，则按照约定从起始位置读取任务数据。每次请求将会返回下一次的偏移量，可以作为下一次读取的 offset。例如某任务有 1000 条数据， 第一次调用 offset = 0, size=100，则将会返回任务的头 100 条数据，以及下一次的偏移量 offset=x（x 不一定等于 100）。第二次调用时 offset=x（第一次请求返回的 offset）， size=100，则返回任务的 100-200 区间的数据，以及下一次偏移量 offset=x1。以此类推

注：此接口只用来读取数据，不会影响数据的导出状态

```
{
    "data": {
        "total": 1000,
        "restTotal": 900,
        "offset": 100,
        "files": [
            {
                "省份": "安徽",
                "地域": "安庆",
```

```
                "日期": "2013-1-1",
                "最高气温": "0℃",
                "天气状况": "多云",
                "风力风向": "东北风微风",
                "最低气温": ""
            },
            {
                "省份": "安徽",
                "地域": "安庆",
                "日期": "2013-1-2",
                "最高气温": "5℃",
                "天气状况": "多云转阴",
                "风力风向": "东北风 3～4 级",
                "最低气温": "-2℃"
            }
        ]
    },
    "error": "success"
}
```


返回的数据有如下字段：

字段	解释
total	当前任务的数据记录总数
restTotal	剩余的数据记录总数
offset	下一次请求的偏移量
sfiles	数据记录集
error	提示信息，若为 success 则表示成功

4. 获取 TaskId 的两种方式

4.1. 通过 API 获取 TaskId

1. 首先获取用户的任务组，使用如下接口并在 Header 中提供 Access Token 验证信息来获取任务组列表。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/taskgroup
Http 头文件参数:
    Name: Authorization
    Value: bearer {access token}
```

在以上请求中若 Access Token 合法且成功查询到数据将会得到类似如下的任务组列表结构体文本：


```
{
    "data": [
        {
```

```
            "taskGroupId": 84,
            "taskGroupName": "任务组 1"
        },
        {
            "taskGroupId": 527,
            "taskGroupName": "任务组 1"
        }]
    "error": "success"
}
```

任务组列表结构体各个字段解释如下：

字段	解释
taskGroupId	任务组唯一标示符
taskGroupName	任务组名称

2. 对于一个任务组，可以通过提供任务组 ID 来获取该任务组下的所有任务列表。

使用如下接口并在 Header 中提供 Access Token 验证信息,同时传入任务组 ID 作为参数来获取任务列表。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/task?taskgroupid={taskgroupid}
Http 头文件参数:
    Name: Authorization
    Value: bearer {access token}
HTTP URL 参数示例: http://dataapi.bazhuayu.com/api/task?taskgroupid=84
```

以上请求中若 Access Token 合法并且 taskgroupid 的值是用户拥有的某个任务组 ID，即是第 3 节中查询到的多个 ID 之一，将会得到类似如下任务列表结构体文本。

```
{
    "data": [
        {
            "taskId": "taskid1",
            "taskName": "任务 1"
        },
        {
            "taskId": "taskid2",
            "taskName": "任务 2"
        }]
    "error": "success"
}
```

任务列表结构体各个字段解释如下：

字段	解释
taskId	采集任务唯一标示符
taskName	采集任务名称

可以通过某个任务 ID 来获取该任务所采集的数据。

一般的使用场景：对于某一种采集任务，若用户因为采集量太大而将其分成了一个任务组中的多个任务来采集，此时通过本节两个 API 来获得任务组 ID 和该组下的所有任务 ID，然后编写程序使用数据 API 分别获取这些任务 ID 对应的任务采集来的数据，再进行合并。

注：一、参考示例代码 5.2 和 5.3。

二、获取任务/任务组时若提供的参数不合法，会返回如下错误结果，每种错误的原因和 json 格式解释如下。

1.Access Token 无效，有可能过期，请使用正确的用户名和密码重新获取 Access Token

```
{
    "error": "unauthorized",
    "error_Description": "access_token 无效"
}
```

2.采用了 POST 方法，请使用 GET 方法

```
{
    "message": "请求的资源不支持 http 方法“POST”。"
}
```

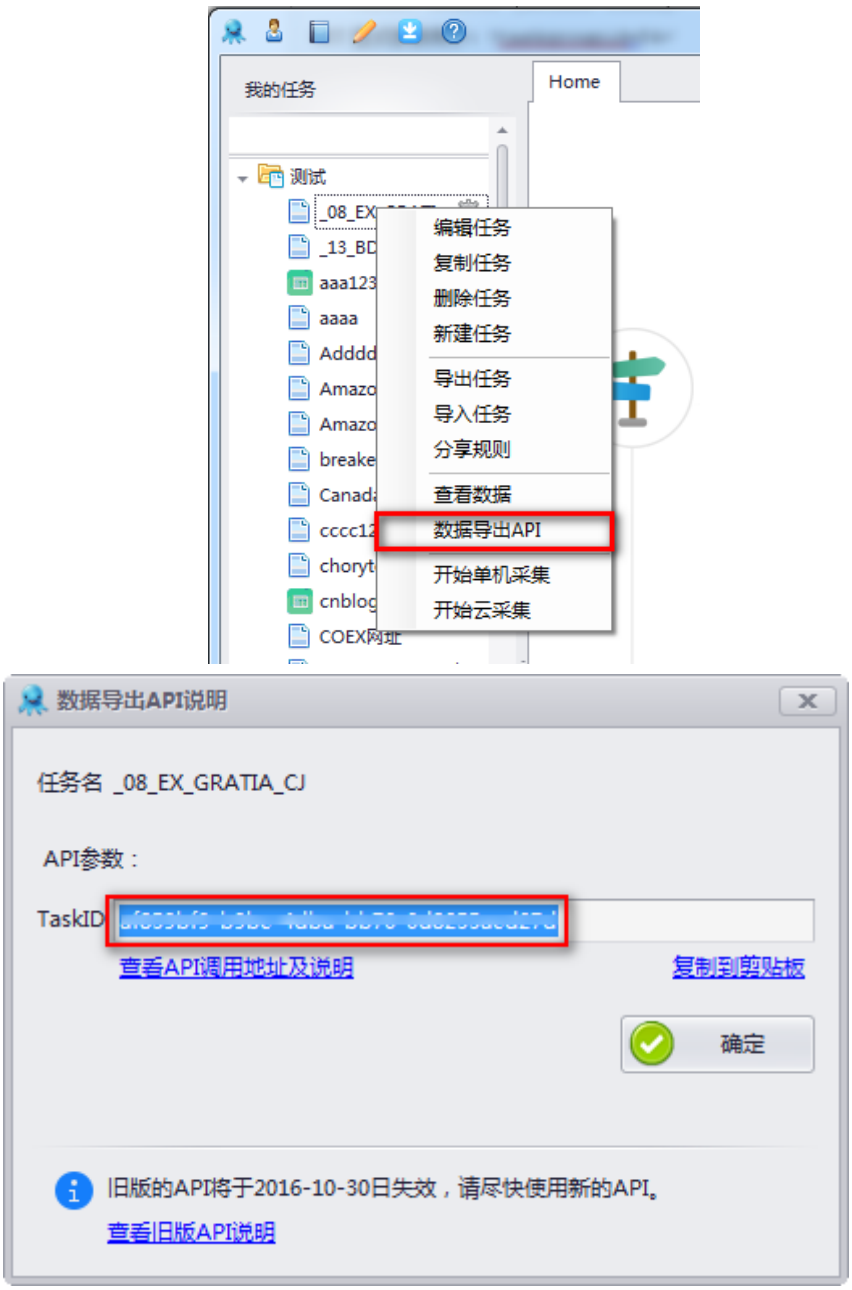
3.服务器暂时不可用

```
{
    "error": "server_error",
    "error_Description": "服务器错误，请稍后再试！"
}
```

4.2. 通过八爪鱼客户端获取 TaskId

（仅旗舰版和私有云用户可使用此功能）

登录八爪鱼客户端之后在“我的任务”中展开一个任务组，右击其中一个任务并单击“数据导出 API”，在对话框中可得到 TaskId。



5. 示例代码

完整可运行的示例代码见 GitHub：https://github.com/octopus-dev/DataExportApi

5.1. 获取用户的 Access Token

```
///<summary>
///用HTTP的POST方法提交username和password，通过token接口来获取token
///</summary>
///<param name="userName">八爪鱼用户名</param>
///<param name="password">登录密码</param>
///<returns></returns>
publicstring GetToken(string userName, string password)
{
```

```
    if (!string.IsNullOrEmpty(userName) && !string.IsNullOrEmpty(password))
    {
        string postdata =
        string.Format("username={0}&password={1}&grant_type=password", userName,
        password);
        string responseText = HttpHelper.Post("http://dataapi.bazhuayu.com/token",
        postdata);
        if (responseText.Contains("access_token"))
{
    token = JObject.Parse(responseText)["access_token"].ToString();
        }
    }
    return token;
}
```

5.2. 获取用户任务组

```
///<summary>
///通过taskgroup接口来获得某个用户的所有taskgroup
///</summary>
///<param name="token">某个用户的access token</param>
publicvoid GetTaskGroups(string token)
{
    if (!string.IsNullOrEmpty(token))
    {
        Dictionary<string, string> headers = new Dictionary<string, string>(1);
headers.Add("Authorization", string.Format("bearer {0}", user.token));
        string TaskGroupsStr =
        HttpHelper.GetWithHeaders("http://dataapi.bazhuayu.com/api/taskgroup",
        headers);
        if (!TaskGroupsStr.Contains("\"data\":"))
        {
            return;
        }
        JObject jsonTaskGroupsAll = JObject.Parse(TaskGroupsStr);
        JArray jsonTaskGroupsJarray = jsonTaskGroupsAll["data"] as JArray;
        user.taskGroups = new List<TaskGroup>(jsonTaskGroupsJarray.Count);
        foreach (JToken jtokenTaskGroup in jsonTaskGroupsJarray)
        {
            user.taskGroups.Add(new TaskGroup()
            {
taskGroupID = jtokenTaskGroup["taskGroupId"].ToString(),
taskGroupName = jtokenTaskGroup["taskGroupName"].ToString(),
tasks = GetTasks(token,jtokenTaskGroup["taskGroupId"].ToString(),taskUrl)
    });
}
```

```
    }
}
```

5.3. 获取用户任务组的任务信息

```
///<summary>
///通过 task 接口来获得某个任务组的所有 tasks
///</summary>
///<param name="token">access token</param>
///<param name="taskGroupID">任务组标识</param>
///<returns>传入任务组 ID 的任务组下的所有任务</returns>
public List<Task> GetTasks(string token,string taskGroupID)
{
    List<Task> tasks = null;
    if (!string.IsNullOrEmpty(token))
    {
        Dictionary<string, string> headers = new Dictionary<string, string>(1);
        headers.Add("Authorization", string.Format("bearer {0}", token));
        string URL = string.Format("{0}?taskgroupid={1}",
"http://dataapi.bazhuayu.com/api/task", taskGroupID);
        string taskStr = HttpHelper.GetWithHeaders(URL, headers);
        if (taskStr.Contains("\"data\":"))
        {
            JObject jsonTasks = JObject.Parse(taskStr);
            JArray jsonTasksJarray = jsonTasks["data"] as JArray;
            tasks = new List<Task>(jsonTasksJarray.Count);
            foreach (JToken jtokenTask in jsonTasksJarray)
            {
                tasks.Add(new Task()
                {
                    taskID = jtokenTask["taskId"].ToString(),
                    taskName = jtokenTask["taskName"].ToString()
                });
            }
        }
    }
    return tasks;
}
```

5.4. 获取某个任务的采集数据

```
///<summary>
///根据 TaskID 获取该任务采集的数据
///</summary>
///<param name="token">用户 Token</param>
///<param name="taskID">任务 ID</param>
```

```
///<param name="pageIndex">初始数据条目位置</param>
///<param name="pageSize">数据条目个数</param>
///<returns>任务采集的数据</returns>
public string GetDataByTask(string token,string taskID,int pageIndex,int pageSize)
{
    string taskData = "";
    if (!string.IsNullOrEmpty(token) &&!string.IsNullOrEmpty(taskID))
    {
        string URL = "";
        Dictionary<string, string> headers = new Dictionary<string, string>(1);
        headers.Add("Authorization", string.Format("bearer {0}", user.token));
        URL = string.Format("{0}?taskid={1}&pageindex={2}&pagesize={3}",
"http://dataapi.bazhuayu.com/api/alldata", taskID, pageIndex, pageSize);
        taskData = HttpHelper.GetWithHeaders(URL, headers);
    }
    return taskData;
}
```

5.5. 获取某个任务的未导出的数据

```
///<summary>
///根据 TaskID 获取该任务未导出的数据
///</summary>
///<param name="token">用户 Token</param>
///<param name="taskID">任务 ID</param>
///<param name="size">数据条目个数</param>
///<returns>任务采集到的数据</returns>
public string GetNewDataByTask(string token,string taskID,int size)
{
    string taskData = "";
    if (!string.IsNullOrEmpty(token) &&!string.IsNullOrEmpty(taskID))
    {
        string URL = "";
        Dictionary<string, string> headers = new Dictionary<string, string>(1);
        headers.Add("Authorization", string.Format("bearer {0}", user.token));
        URL =
string.Format("{0}?taskid={1}&size={2}","http://dataapi.bazhuayu.com/api/
notexportdata", taskID, size);
        taskData = HttpHelper.GetWithHeaders(URL, headers);
    }
    return taskData;
}
```