八爪鱼数据导出 API 开发者文档

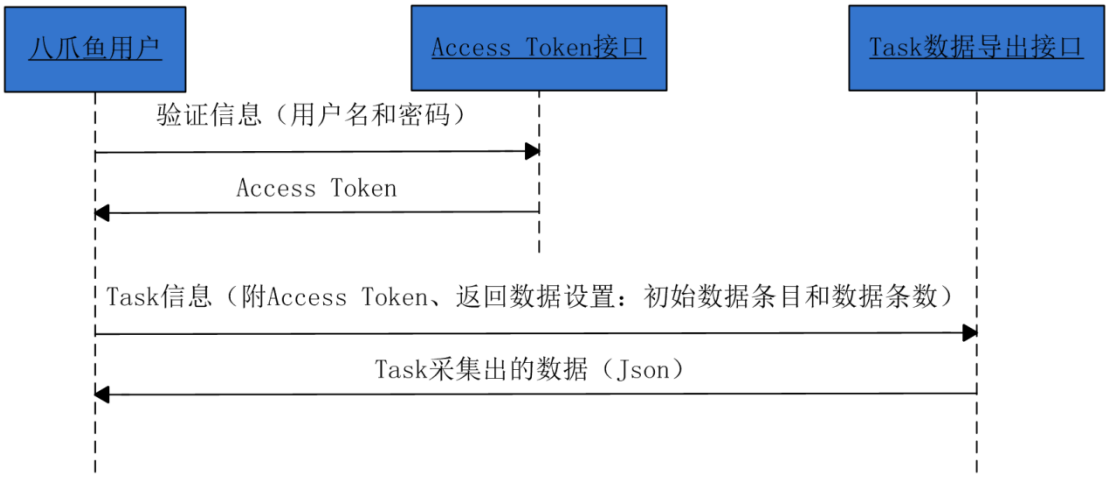
目录

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。

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

2.1. 首次获取 Access Token

使用如下接口并提供用户名和密码以获取 Access Token。

```
http 请求方式：POST
http://dataapi.bazhuayu.com/token
POST 表单数据类型：application/x-www-form-urlencoded
POST 表单数据格式：
username={username}&password=password&grant_type=password
POST 表单参数示例：
username=testusername&password=testpassword&grant_type=password
说明：请将 POST 表单数据中 testusername/testpassword 替换为你的八爪鱼用户名和密码即可，并注意将 username、password 的值进行 url 编码后再传入。
```

若参数无误，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": "1234ABCD"
}
```

Access Token 结构体各个字段的解释如下：

字段	解释
access_token	访问标识 access token
token_type	access token 类型
expires_in	access token 过期时间（秒），当前为 24 小时
refresh_token	在 access token 过期后刷新 access token 的一个标记，将在 2.2 节中讲解使用方法

Access Token 是几乎八爪鱼所有 API 的访问许可标志，在获取数据等 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=test1&password=test2&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。

2.2. 使用长有效期的 refresh_token 重新获取 Token

若你之前获取的 Access Token 过期，请使用如下接口，并提供 refresh_token（上一次返回 Access Token 的 JSON 文本中包含一个 refresh_token），同时将“grant_type”改为“refresh_token”，即可获取一个相同过期时间的新 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：refresh_token=1234ABCD&grant_type=refresh_token
```

以上请求如果 refresh_token 可用且未过期，Http Response 会返回类似如下的 JSON 格式文本。

```
{
    "access_token": "ABCD12345",
    "token_type": "bearer",
    "expires_in": 86399,
    "refresh_token": "46b94a3ead204d1b84e101364a71c6d2"
}
```

此接口返回数据各个字段的解释如下：

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

其中的 access_token 即是通过 refresh_token 获得的新 Access Token，它与使用用户名和密码获取的 Access Token 是等效的。

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）、分页信息（pageindex、pagesize），并在 Header 中添加 Access Token 信息。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/alldata?taskid={taskid}&pageindex={pageIndex}&pagesize={pageSize}
Http 头文件参数 1:
    Name: Authorization
    Value: bear [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=1234ABCD&pageindex=1&pagesize=2
```

Http 头文件数据例子：

Name	Value	说明
Authorization	bearer ABCD1234	Token 验证信息
Accept	application/json	表明接受返回数据的格式为 JSON

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 10000 条数据，若传入 pagesize=100，pageindex=1，则将数据分为 10000/100=100 页，每页 100 条数据，然后返回第 1 页的 100 条数据。若参数均合法，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. 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），并在 Header 中添加 Access Token 信息。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/notexportdata?taskid={taskId}&
size={size}
Http 头文件参数 1:
    Name: Authorization
    Value: bear [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=1234ABCD&size=100
```

Http 头文件数据例子：

Name	Value	说明
Authorization	bearer ABCD1234	Token 验证信息
Accept	application/json	表明接受返回数据的格式为 JSON

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的 998 第中 2 条数据。若参数均合法，

将会返回与 3.1 节中接口调用的结果相同 JSON 格式的任务数据（无分页）。

注：参考示例代码 5.5。

3.3. 根据入库时间获取任务数据

使用此接口来根据任务数据入库时间（即云采集到一条数据后写入数据库的时间）获取数据，接口需要传入任务 ID（TaskId）、时间区间参数（from/to）和分页参数（pagesize/pageindex），并在 Header 中添加 Access Token 信息。

```
http 请求方式: GET
http://dataapi.bazhuayu.com/api/alldata/GetDataOfTaskByTimeAndPaging?taskid={taskId}&from={ealyDateTime}&to={laterDateTime}&&pageindex={pageIndex}&pagesize={pageSize}
Http 头文件参数 1:
    Name: Authorization
    Value: bear [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=1234ABCD&from={2017-01-01 00:00:00}&to={2017-01-02 00:00:00}&&pageindex=1&pagesize=100
```

Http 头文件数据例子：

Name	Value	说明
Authorization	bearer ABCD1234	Token 验证信息
Accept	application/json	表明接受返回数据的格式为 JSON

若参数均合法，将会返回与 3.1 节中接口调用的结果相同 JSON 格式的任务数据（分页）。

3.4. 获取上一次云采集的任务数据

使用此接口获取上一次运行云采集后采集到的数据，接口需要传入任务 ID（TaskId）和分页参数（pagesize/pageindex），并在 Header 中添加 Access Token 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: bear [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=b1d51de6-f644-4a26-b659-d001fe527f4c&pageindex=1&pagesize=100
```

Http 头文件数据例子：

Name	Value	说明
Authorization	bearer ABCD1234	Token 验证信息
Accept	application/json	表明接受返回数据的格式为 JSON

若参数均合法,将会返回与 3.1 节中接口调用的结果相同 JSON 格式的任务数据(分页)。

注：若使用任一获取任务数据的接口时若提供的参数不合法，会返回错误结果，每种错误的解释如下：

1. Access Token 无效，有可能过期，请使用用户名和密码或者 refresh_token 重新获取 Access Token

```
{
    "error": "unauthorized",
    "error_Description": "access_toekn 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 不允许超过系统设置的 MaxSize 的值，目前为 1000

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

5. 服务器暂时不可用，请稍后再尝试，若频繁发生请联系客服

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

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=1234
Http 头文件参数：
    Name: Authorization
    Value: bearer [access token]
HTTP URL 参数示例：?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。

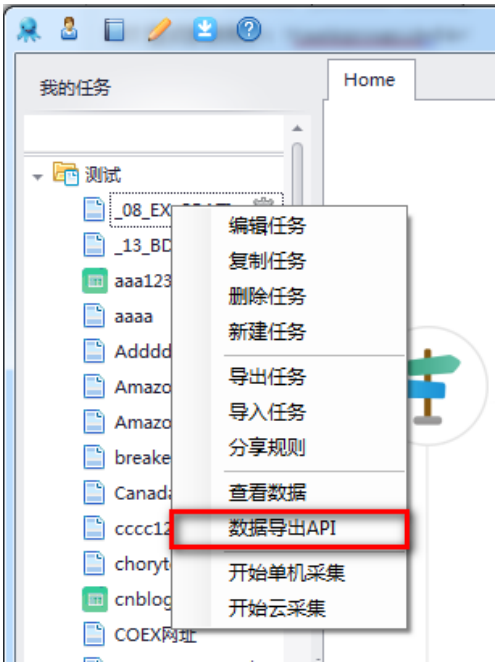
一般的使用场景：对于某一种采集任务，若用户因为采集量太大而将其分成了一个任务组中的多个任务来采集，此时通过本节两个 API 来获得任务组 ID 和该组下的所有任务 ID，然后编写程序使用数据 API 分别获取这些任务 ID 对应的任务采集来的数据，再进行合并。

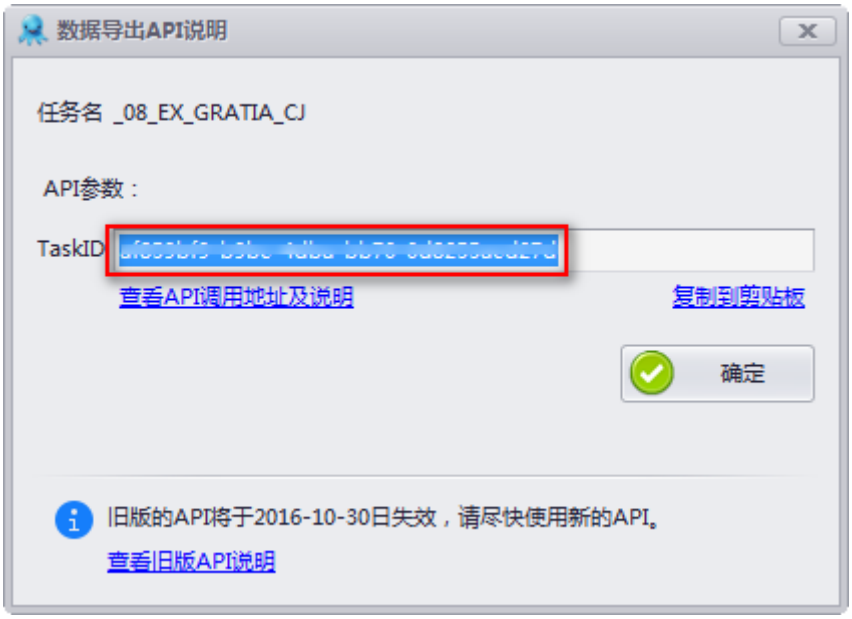
注：参考示例代码 5.2 和 5.3。

4.2. 通过八爪鱼客户端获取 Task ID

（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>
public string GetToken(string userName, string password)
{
    if (null != userName &&null != 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>
public void GetTaskGroups(string token)
{
    if (null != token)
    {
        Dictionary<string, string> headers = newDictionary<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"] asJArray;
        user.taskGroups = newList<TaskGroup>(jsonTaskGroupsJarray.Count);
        foreach (JToken jtokenTaskGroup in jsonTaskGroupsJarray)
        {
            user.taskGroups.Add(newTaskGroup()
            {
                    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(torken))
    {
        Dictionary<string, string> headers = newDictionary<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"] asJArray;
            tasks = newList<Task>(jsonTasksJarray.Count);
            foreach (JToken jtokenTask in jsonTasksJarray)
            {
                tasks.Add(newTask()
                {
                    taskID = jtokenTask["taskId"].ToString(),
                    taskName = jtokenTask["taskName"].ToString()
                });
            }
        }
        else{}
    }
    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(torken) && !string.IsNullOrEmpty(taskID))
    {
        string URL = "";
        Dictionary<string, string> headers = newDictionary<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(torken) && !string.IsNullOrEmpty(taskID))
    {
        string URL = "";
        Dictionary<string, string> headers = newDictionary<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;
}
```