ArkID Client

发布 *1.0*

2020 年 05 月 06 日

Contents

ArkID Client 为 ArkID REST API 提供了便捷的 Pythonic 接口，其中包括了 UserClient、OrgClient、NodeClient 等客户端。您可以直接实例化 ArkIDClient 客户端来调用所有底层的各类客户端的功能。REST API 完整文档详见 https://docs.arkid.longguikeji.com/ 。

CHAPTER 1

目录

1.1 安装

ArkID Client 需要 Python 2.7 + 或者 3.4 + 版本的支持。如果您的系统还未安装支持的 Python 版本，请参见 Python 安装指南 。

安装 ArkID Client 的最简单的方法是使用 `pip` 包管理器 (https://pypi.python.org/pypi/pip)，它包含于您已安装的 Python 版本中:

```
pip install arkid_client
```

执行上述命令后，将安装 ArkID Client 及其依赖项。

1.2 Client 教程

在这里为您提供 ArkID Client 的使用教程，它将指引你通过一个简单的流程来获取 ArkID 的认证信息，并使用它来访问 ArkID 的服务。

以下是我们将采取的步骤:

1. *创建认证客户端*
2. *创建认证授权器*
3. *创建服务客户端*
4. *访问 ArkID 服务*

以上的四个步骤足够让您起手开始将 ArkID Client 应用到您的项目中。考虑到安全方面的因素，我们也许会在将来使用 oauth2.0 协议或者 OIDC 协议注入到认证客户端中，让用户更加安全的使用 ArkID Client 。

1.2.1 步骤一：创建认证客户端

为了获得认证信息，您必须通过实例化一个认证客户端来访问 ArkID 服务端的认证服务。以下的代码是基于 ArkID 服务端的官方认证来获取访问凭证 `oneid_token` 。

```
from arkid_client.auth import ConfidentialAppAuthClient

client = ConfidentialAppAuthClient(base_url='<ARKID_SERVICE_URL>')
token = client.auth_to_get_token('<USERNAME>', '<PASSWORD>')
```

上述的 `ARKID_SERVICE_URL` 是您将要访问的 ArkID 服务的根地址。在获取到有效的 `token` 后，您就可以进行下一步操作来获得认证授权器。

1.2.2 步骤二：创建认证授权器

从上面的示例继续，认证客户端获取到 `token` 后，您就可以创建一个认证授权器来为后续的服务客户端所使用：

```
from arkid_client.authorizers import BasicAuthorizer

authorizer = BasicAuthorizer(oneid_token=token)
```

在步骤一中*创建认证客户端* 的流程稍显繁琐，如果您不在意认证流程的显式性，您可以直接从认证客户端中获取到所需的 `authorizer` ，让您的使用体验达到最佳效果：

```
from arkid_client.auth import ConfidentialAppAuthClient

client = ConfidentialAppAuthClient(base_url='<ARKID_SERVICE_URL>')
authorizer = client.get_authorizer('<USERNAME>', '<PASSWORD>')
```

注意：虽然可直接从 `ConfidentialAppAuthClient` 获取到 `authorizer` ，但这并不严格符合 `ArkID Client` 的认证流程规范，仅仅是为了在某种程度上更加方便用户的使用。

1.2.3 步骤三：创建服务客户端

从上面的示例继续，创建认证授权器之后，您可以开始初始化一个 ArkIDClient 的实例了。

```
from arkid_client import ArkIDClient

client = ArkIDClient(authorizer='<authorizer>', base_url='<ARKID_SERVICE_URL>')
```

上述的 ARKID_SERVICE_URL 是您将要访问的 ArkID 服务的根地址。当 client 创建成功之后，您就可以开始访问 ArkID 的服务了。

1.2.4 步骤四: 访问 ArkID 服务

从上面的示例继续，您可以开始访问任何有关 ArkID 服务端的接口，比如，您将要获取所有用户的信息：

```
from arkid_client import ArkIDClient

# 上面的示例中初始化的 ArkID Client
client = ArkIDClient(authorizer='<authorizer>', base_url='<ARKID_SERVICE_URL>')

# 访问 ArkID 服务并获取响应
users = client.query_user()
```

访问 ArkID 服务后得到的结果是 ArkIDHTTPResponse 对象，有关它的具体说明请参考*Responses*

1.3 服务客户端

ArkID Client 提供了一个封装了 ArkID 所有的 API 接口的客户端类。您无需进行任何配置，只需要在初始化客户端实例的时候，向其添加 ArkID 服务的根地址以及必要的认证授权器*ArkIDAuthorizers* ；在客户端创建成功之后，您就可以使用顶层的接口来调用 ArkID 的各项服务而无需知道 ArkID 的各端点地址或者详细的参数。

客户端类型

1.3.1 认证服务客户端

class arkid_client.**AuthClient**(*base_url*, *service=None*, *authorizer=None*, ***kwargs*)

基类：*arkid_client.base.BaseClient*

认证客户端，用于向 ArkID 服务端请求授权认证信息，并获取访问令牌。

Examples

初始化 < AuthClient > 客户端，以 Access Token 授权方式向 ArkID 服务端请求对调用的用户进行身份验证 (TODO)

```
>>> from arkid_client import AuthClient, AccessTokenAuthorizer
>>> ac = AuthClient(authorizer=AccessTokenAuthorizer('<token_string>'))
```

上述使用 oauth2.0 协议来请求授权，虽然 ArkID Client 暂时还不支持这样做。但是，这里可以使用任何其它的符合规则的授权器。

class arkid_client.ConfidentialAppAuthClient(*base_url*, ***kwargs*)

基类：arkid_client.auth.client.base.AuthClient

与 ArkID 认证服务端进行通信的 AuthClient 类型的认证客户端。此客户端必须是已受到 ArkID 官方高度信任的第三方客户端，可凭借 username 和 password 直接向 ArkID 认证端发起授权请求。最终，它将得到 ArkID 官方默认的认证凭证 oneid_token 。

Methods

- *start_auth()*
- *get_token()*
- *auth_to_get_token()*
- *revoke_token()*
- *auth_token()*
- *get_authorizer()*

start_auth(*username: str*, *password: str*)

开始进行身份认证 (POST /siteapi/v1/ucenter/login/)

Parameters

username (***string***) 用户唯一标识

password (***string***) 密码

Examples

```
>>> caac = arkid_client.ConfidentialAppAuthClient(...)
>>> caac.start_auth(username, password)
```

get_token()

获取 oneid_token

Examples

```
>>> caac = arkid_client.ConfidentialAppAuthClient(...)
>>> caac.start_auth('<username>', '<password>')
>>> token = caac.get_token()
```

auth_to_get_token(*_username: str*, *_password: str*)

简化 oneid_token 的获取流程，大多数时候比传统的获取方式更轻松。(POST /siteapi/v1/ucenter/login/)

Parameters

username (***string***) 用户唯一标识

password (***string***) 密码

Examples

```
>>> caac = arkid_client.ConfidentialAppAuthClient(...)
>>> token = caac.auth_to_get_token(_username, _password)
```

revoke_token(*authorizer: arkid_client.authorizers.basic.BasicAuthorizer*)

撤销 oneid_token (POST /siteapi/v1/revoke/token/)

Parameters

authorizer (***BasicAuthorizer***) 特指 < BasicAuthorizer > 类型的授权器

Examples

```
>>> caac = arkid_client.ConfidentialAppAuthClient(...)
>>> token = caac.revoke_token(authorizer)
```

auth_token(*authorizer: arkid_client.authorizers.basic.BasicAuthorizer*)

校验 oneid_token 所代表的用户是否有某特定权限 (GET /siteapi/v1/auth/token/)

Parameters

authorizer (***BasicAuthorizer***) 特指 < BasicAuthorizer > 类型的授权器

Examples

```
>>> caac = arkid_client.ConfidentialAppAuthClient(...)
>>> token = caac.auth_token(authorizer)
```

get_authorizer(*username: str*, *password: str*)

封装 self.auth_to_get_token 方法，通过认证客户端直接获得有效的授权器。(GET /siteapi/v1/ucenter/login/)

Parameters

username (***string***) 用户唯一标识

password (***string***) 密码

Examples

```
>>> caac = arkid_client.ConfidentialAppAuthClient(...)
>>> authorizer = caac.get_authorizer(username, password)
```

1.3.2 用户服务客户端

class arkid_client.user.**UserClient**(*base_url*, *authorizer=None*, ***kwargs*)

基类：*arkid_client.base.BaseClient*

用户管理客户端，用于与 ArkID 服务端用户管理相关接口的访问操作。

Methods

- *query_user()*
- *query_isolated_user()*
- *query_user()*
- *create_user()*
- *update_user()*
- *delete_user()*
- *query_specified_perm()*

query_user_list(***params*)

获取用户信息列表 (`GET /siteapi/v1/user/`)

Parameters:

keyword (***str***) 查询关键字，进行用户名、姓名、邮箱、手机号模糊搜索

wechat_unionid (***str***) 微信客户端 openid

page (***int***) 用于分页，*Default: 1*

page_size (***int***) 指定分页大小，*Default: 30*

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> users = uc.query_user(...)
>>> for user in users:
>>>     print(user['username'], 'id: '
>>>           ,user['id'])
```

External Documentation

关于 用户的元数据 详情请参阅 API 文档。

query_isolated_user(***params*)

获取所有独立用户 (GET /siteapi/v1/user/isolated/)

Parameters:

page (***int***) 用于分页, *Default: 1*

page_size (***int***) 指定分页大小, *Default: 30*

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> users = uc.query_isolated_user(...)
>>> for user in users:
>>>     print(user['username'], 'id: '
>>>           ,user['id'])
```

query_user(*username: str*)

获取指定用户的信息 (GET /siteapi/v1/user/<username>/)

Parameters:

username (***str***) 用户唯一标识

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> user = uc.query_user(...)
>>> print(user['username'], 'id: '
>>>       ,user['id'])
```

create_user(*json_body: dict*)

创建用户 (需要管理员权限) (POST /siteapi/v1/user/)

Parameters:

json_body (*dict*)

group_uids (***list[str]***) 用户组 uuid 的集合

dept_uids (***list[str]***) 部门 uuid 的集合

user (***dict***) 用户的元信息, 参数详情请参考接口文档

node_uids (***list[str]***) (可选的) 此字段提供时会忽略 group_uids, dept_uids

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> user_data = {
>>>   "user": {
```

(下页继续)

(续上页)

```
>>>     "password": "example",
>>>     "username": "example"
>>>     }
>>> }
>>> user = uc.create_user(user_data)
>>> print(user['username'], 'id: '
>>>           ,user['id'])
```

External Documentation

关于 用户的元数据 详情请参阅 API 文档。

`update_user`(*username: str*, *json_body: dict*)

修改指定用户的信息 (`PATCH /siteapi/v1/user/<username>`)

Parameters:

`username` (***str***) 用户唯一标识

`json_body` (***dict***) 用户的元信息, 参数详情请参考接口文档

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> query_data = {
>>>   "password": "example",
>>>   "private_email": "example@org.com"
>>> }
>>> user = uc.update_user(username, query_data)
>>> print(user['username'], 'id: '
>>>           ,user['id'])
```

External Documentation

关于 用户的元数据 详情请参阅 API 文档。

`delete_user`(*username: str*)

删除指定用户的信息 (`DELETE /siteapi/v1/user/<username>/`)

Parameters:

`username` (***str***) 用户唯一标识

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> uc.delete_user(username)
```

query_specified_perm(*username: str*, *uid: str*)

获取用户权限详情，包括权限来源 (GET /siteapi/v1/user/<username>/perm/<uid>/)

Parameters:

username (***str***) 用户唯一标识

uid (***str***) 权限唯一标识

Examples

```
>>> pc = arkid_client.UserClient(...)
>>> perm = pc.query_specified_perm(...)
>>> print('perm is', perm)
```

1.3.3 组织服务客户端

class arkid_client.org.**OrgClient**(*base_url*, *authorizer=None*, ***kwargs*)

基类：*arkid_client.base.BaseClient*

组织管理客户端，用于与 ArkID 服务端组织管理相关接口的访问操作。

Methods

- *query_own_org()*
- *query_org()*
- *create_org()*
- *delete_org()*
- *update_org()*
- *query_orguser()*
- *add_orguser()*
- *delete_orguser()*
- *query_orguser()*
- *update_orguser()*
- *get_org_invitation_key()*
- *refresh_org_invitation_key()*
- *view_org_by_invitation_key()*
- *join_org_by_invitation_key()*

query_own_org(***params*)

查询用户所在的组织 (GET /siteapi/v1/org/)

Parameters:

`role` (*str*) 在组织内的角色

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> orgs = oc.query_own_org(role='admin')
>>> for org in orgs:
>>>     print(org['oid'], 'name: '
>>>           ,org['name'])
```

`query_org`(*oid: str*)

查看指定组织的信息 (`GET /siteapi/v1/org/<oid>/`)

Parameters:

`oid` (*str*) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.query_org(oid)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

`create_org`(*json_body: dict*)

创建组织 (`POST /siteapi/v1/org/`)

Parameters:

`json_body` (*dict*) 组织的元信息，参数详情请参考接口文档

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org_data = {
>>>   "name": "example",
>>> }
>>> org = oc.create_org(org_data)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

External Documentation

关于 组织的元数据 详情请参阅 API 文档。

`delete_org`(*oid: str*)

删除指定组织的信息 (`DELETE /siteapi/v1/org/<oid>/`)

Parameters:

`oid` (***str***) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> oc.delete_org(oid)
```

`update_org`(*oid: str*, *json_body: dict*)

修改指定组织的信息 (`PATCH /siteapi/v1/org/<oid>/`)

Parameters:

`oid` (***str***) 组织的唯一标识

`json_body` (***dict***) 组织的元信息，参数详情请参考接口文档

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org_data = {
>>>   "name": "example",
>>> }
>>> org = oc.update_org(oid, org_data)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

External Documentation

关于 组织的元数据 详情请参阅 API 文档。

`query_orguser_list`(*oid: str*, ***params*)

查看特定组织的成员信息 (`GET /siteapi/v1/org/<oid>/user/`)

Parameters:

`oid` (***str***) 组织的唯一标识

`page` (***int***) 用于分页，*Default: 1*

`page_size` (***int***) 指定分页大小，*Default: 30*

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.query_orguser_list(oid)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

add_orguser(*oid: str, usernames: list*)

向指定组织中添加成员 (PATCH /siteapi/v1/org/<oid>/user/)

Parameters:

oid (***str***) 组织的唯一标识

usernames (***list***) 用户的唯一标识组成的列表

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> usernames = [
>>>     'username1',
>>>     'username2',
>>>         ...
>>>     'usernamen'
>>> ]
>>> org = oc.add_orguser(oid, usernames)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

delete_orguser(*oid: str, usernames: list*)

从指定组织中移除成员 (PATCH /siteapi/v1/org/<oid>/user/)

Parameters:

oid (***str***) 组织的唯一标识

usernames (***list***) 用户的唯一标识组成的列表

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> usernames = [
>>>     'username1',
>>>     'username2',
>>>         ...
>>>     'usernamen'
>>> ]
>>> org = oc.delete_orguser(oid, usernames)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

query_orguser(*oid: str, username: str*)

查看指定组织的指定成员的信息 (GET /siteapi/v1/org/<oid>/user/<username>/)

Parameters:

oid (*str*) 组织的唯一标识

username (*str*) 用户唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> user = oc.query_orguser(oid, username)
>>> print('user is', user)
```

update_orguser(*oid: str*, *username: str*, *json_body: dict*)

编辑指定组织的指定成员的信息 (PATCH /siteapi/v1/org/<oid>/user/<username>/)

Parameters:

oid (*str*) 组织的唯一标识

username (*str*) 用户唯一标识

json_body (*dict*)

email (*str*) 成员邮箱

employee_number (*str*) 成员工号

position (*str*) 成员职位

hiredate (*str*) 成员雇佣日期

remark (*str*) 成员备注

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> user_data = {
>>>   "email": "example@org.com",
>>> }
>>> user = oc.update_orguser(oid, username, user_data)
>>> print(user['id'], 'name: '
>>>       ,user['name'])
```

External Documentation

关于 成员的元数据 详情请参阅 API 文档。

get_org_invitation_key(*oid: str*)

获取指定组织邀请用的最新的密钥 (GET /siteapi/v1/org/<oid>/invitation/)

Parameters:

oid (*str*) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> key = oc.get_org_invitation_key(oid)
>>> print('key: ', key)
```

refresh_org_invitation_key(*oid: str*)

刷新指定组织邀请用的最新的密钥 (PUT /siteapi/v1/org/<oid>/invitation/)

Parameters:

oid (*str*) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> key = oc.refresh_org_invitation_key(oid)
>>> print('key: ', key)
```

view_org_by_invitation_key(*oid: str*, *invite_link_key: str*)

使用邀请密钥查看指定组织的信息 (GET /siteapi/v1/org/<oid>/invitation/<invite_link_key>/)

Parameters:

oid (*str*) 组织的唯一标识

invite_link_key (*str*) 组织邀请密钥

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.view_org_by_invitation_key(oid, invite_link_key)
>>> print('org: ', org)
```

join_org_by_invitation_key(*oid: str*, *invite_link_key: str*)

使用邀请密钥加入指定组织 (POST /siteapi/v1/org/<oid>/invitation/<invite_link_key>/)

Parameters:

oid (*str*) 组织的唯一标识

invite_link_key (*str*) 组织邀请密钥

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.join_org_by_invitation_key(oid, invite_link_key)
>>> print('org: ', org)
```

1.3.4 节点服务客户端

class arkid_client.node.NodeClient(*base_url*, *authorizer=None*, ***kwargs*)

基类：*arkid_client.base.BaseClient*

节点管理客户端，用于与 ArkID 服务端节点管理相关接口的访问操作。

Methods

- *query_node()*
- *view_node()*
- *update_node()*
- *delete_node()*
- *get_node_tree_list()*
- *get_node_tree()*
- *view_node_tree()*
- *get_subnode()*
- *create_subnode()*
- *add_subnode()*
- *sort_subnode()*
- *query_user_under_node()*
- *add_user_under_node()*
- *delete_user_under_node()*
- *override_user_under_node()*
- *sort_user_under_node()*
- *move_out_user_under_node()*

query_node(*node_uid: str*)

查询指定节点的信息 (GET /siteapi/v1/node/<node_uid>/)

Parameters:

node_uid (***str***) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node = nc.query_node(node_uid)
>>> print('node is', node)
```

`view_node`(*node_uid: str*)

用户查询指定节点的信息 (`GET /siteapi/v1/ucenter/node/<node_uid>/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node = nc.view_node(node_uid)
>>> print('node is', node)
```

`update_node`(*node_uid: str*, *json_body: dict*)

修改指定节点的信息 (`PATCH /siteapi/v1/node/<node_uid>/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

`json_body` (***dict***) 节点的元信息, 参数详情请参考接口文档

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_data = {
>>>     'name': 'example'
>>> }
>>> node = nc.update_node(node_uid, node_data)
>>> print('node is', node)
```

External Documentation

关于 节点的元数据 详情请参阅 API 文档。

`delete_node`(*node_uid: str*, ***params*)

删除指定节点的信息 (`DELETE /siteapi/v1/node/<node_uid>/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

`ignore_user` (***bool***) 用于删除节点，当 “true“时，若节点下有人员存在时，会先将人员从节点内删除，再删除此节点。

Examples self.logger.info(“正在调用 NodeClient.delete_node() 接口与 ArkID 服务端进行交互”) >>> nc = arkid_client.NodeClient(···) >>> nc.delete_node(node_uid, ignore_user=True)

`get_node_tree_list`(*node_uid: str*)

需要管理员权限，获取节点及其子孙节点列表，将某节点下的子树以列表形式返回，包括该节点自身，前序遍历。(`GET /siteapi/v1/node/<node_uid>/list/`)

Parameters:

node_uid (*str*) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> tree_list = nc.get_node_tree_list(node_uid)
>>> print('tree_list is', tree_list)
```

get_node_tree(*node_uid: str*, ***params*)

需要管理员权限，管理员访问到的数据将由管理范围决定，数据包括从该节点起始的完整树。(GET /siteapi/v1/node/<node_uid>/tree/)

Parameters:

node_uid (*str*) 节点的唯一标识

user_required (*bool*) 是否需要成员信息

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> tree = nc.get_node_tree(node_uid, user_required=True)
>>> print('tree is', tree)
```

view_node_tree(*node_uid: str*, ***params*)

普通用户访问节点下结构，访问到的数据将由节点可见范围决定，数据包括从该节点起始的完整树。(GET /siteapi/v1/ucenter/node/<node_uid>/tree/)

Parameters:

node_uid (*str*) 节点的唯一标识

user_required (*bool*) 是否需要成员信息

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> tree = nc.view_node_tree(node_uid, user_required=True)
>>> print('tree is', tree)
```

get_subnode(*node_uid: str*)

获取指定节点的子节点信息 (GET /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (*str*) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> subnode = nc.get_subnode(node_uid)
>>> print('subnode is', subnode)
```

create_subnode(*node_uid: str*, *json_body: dict*)

创建指定节点的子节点 (POST /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (***str***) 节点的唯一标识

json_body (***dict***) 节点的元信息, 参数详情请参考接口文档

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_data = {
>>>     'name': 'example'
>>> }
>>> subnode = nc.create_subnode(node_uid, node_data)
>>> print('subnode is', subnode)
```

External Documentation

关于 节点的元数据 详情请参阅 API 文档。

add_subnode(*node_uid: str*, *node_uids: list*)

向指定节点添加子节点 (PATCH /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (***str***) 节点的唯一标识

node_uids (***list***) 节点唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_uids = [
>>>     'node1',
>>>     'node2',
>>>         ...
>>>     'noden'
>>> ]
>>> node = nc.add_subnode(node_uid, node_uids)
>>> print('node is', node)
```

sort_subnode(*node_uid: str, node_uids: list*)

对指定子节点按指定位置进行排序 (PATCH /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (***str***) 节点的唯一标识

node_uids (***list***) 节点唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_uids = [
>>>     'node1',
>>>     'node2',
>>>         ...
>>>     'noden'
>>> ]
>>> node = nc.sort_subnode(node_uid, node_uids)
>>> print('node is', node)
```

query_user_under_node(*node_uid: str, **params*)

查询指定节点下的直属人员的信息 (GET /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (***str***) 节点的唯一标识

****params** (***dict***) 用户的元信息，参数详情请参考接口文档

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> users = nc.query_user_under_node(node_uid)
>>> for user in users:
>>>     print('user is', user)
```

External Documentation

关于 节点的元数据 详情请参阅 API 文档。

add_user_under_node(*node_uid: str, user_uids: list*)

向指定节点添加指定成员 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (***str***) 节点的唯一标识

user_uids (***list***) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.add_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

delete_user_under_node(*node_uid: str, user_uids: list*)

从指定节点移除指定成员 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (***str***) 节点的唯一标识

user_uids (***list***) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.delete_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

override_user_under_node(*node_uid: str, user_uids: list*)

重置指定节点的指定用户 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (***str***) 节点的唯一标识

user_uids (***list***) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
```

(下页继续)

(续上页)

```
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.override_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

sort_user_under_node(*node_uid: str, user_uids: list*)

对指定人按指定位置进行排序 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (***str***) 节点的唯一标识

user_uids (***list***) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.sort_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

move_out_user_under_node(*node_uid: str, user_uids: list*)

将这些人从该节点移除，并加到指定节点 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (***str***) 节点的唯一标识

user_uids (***list***) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
```

(下页继续)

(续上页)

```
>>> node = nc.move_out_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

1.3.5 用户中心服务客户端

class arkid_client.ucenter.**UcenterClient**(*base_url*, *authorizer=None*, ***kwargs*)

基类：*arkid_client.base.BaseClient*

用户中心管理客户端，用于与 ArkID 服务端用户中心管理相关接口的访问操作。

Methods

- *view_perm()*
- *view_profile()*
- *view_current_org()*
- *switch_current_org()*

view_perm()

获取用户权限，只返回用户拥有的权限（只读）(GET /siteapi/v1/ucenter/perm/)

Examples

```
>>> uc = arkid_client.UsercenterClient(...)
>>> perm = uc.query_perm()
>>> print('perm is', perm)
```

view_profile()

获取用户自身信息 (GET /siteapi/v1/ucenter/profile/)

Examples

```
>>> uc = arkid_client.UsercenterClient(...)
>>> perm = uc.query_profile()
>>> print('perm is', perm)
```

view_current_org()

获取用户当前所在组织的信息 (GET /siteapi/v1/ucenter/org/)

Examples

```
>>> oc = arkid_client.UsercenterClient(...)
>>> org = oc.get_current_org()
>>> print('org: ', org)
```

switch_current_org(*json_body: dict*)

切换用户当前所在的组织 (PUT /siteapi/v1/ucenter/org/)

Parameters:

json_body (*dict*)

oid (***str***) 组织的唯一标识

Examples

```
>>> oc = arkid_client.UsercenterClient(...)
>>> org = oc.switch_current_org({'oid': 'example'})
>>> print('org: ', org)
```

query_apps(***params*)

普通用户获取可见应用列表 (GET /siteapi/v1/ucenter/apps/)

Parameters:

name (***str***) 应用名称

Examples

```
>>> oc = arkid_client.UsercenterClient(...)
>>> apps = oc.query_apps()
>>> print('apps: ', apps)
```

1.3.6 权限服务客户端

class arkid_client.perm.**PermClient**(*base_url*, *authorizer=None*, ***kwargs*)

基类：*arkid_client.base.BaseClient*

权限管理客户端，用于与 ArkID 服务端权限管理相关接口的访问操作。

Methods

- *query_all_perm()*
- *create_perm()*
- *query_perm()*
- *update_perm()*
- *query_perm_owner()*
- *update_perm_owner()*
- *query_specified_user_perm()*
- *update_specified_user_perm()*

- *query_dept_perm()*
- *update_dept_perm()*
- *query_group_perm()*
- *update_group_perm()*
- *query_node_perm()*
- *update_node_perm()*

query_all_perm()

获取所有权限 (GET /siteapi/v1/perm/)

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.query_perm()
>>> print('perm is', perm)
```

create_perm(*json_body: dict*)

创建权限 (POST /siteapi/v1/perm/)

Parameters:

json_body (*dict*)

scope (***str***) 应用 uid

name (***str***) 应用名称

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = uc.create_perm(...)
>>> print('perm is', perm)
```

query_perm(*uid: str*)

获取指定权限 (GET /siteapi/v1/perm/<uid>/)

Parameters:

uid (***str***) 权限唯一标识

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.query_perm(...)
>>> print('perm is', perm)
```

update_perm(*uid: str, json_body: dict*)

更新指定权限 (PATCH /siteapi/v1/perm/<uid>/)

Parameters:

uid (***str***) 权限唯一标识

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.update_perm(...)
>>> print('perm is', perm)
```

query_perm_owner(*uid: str, **params*)

获取某权限指定类型的所有者该接口内的 uid，对于 user 为 username，对于 node 为 node_uid (GET /siteapi/v1/perm/<uid>/)

Parameters:

uid (***str***) 权限唯一标识

owner_subject (***str***) 权限所有者类型

value (***bool***) 最终判定结果

status (***int***) 授权状态

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.query_perm_owner(...)
>>> print('perm is', perm)
```

External Documentation

关于 params 参数 详情请参阅 API 文档。

update_perm_owner(*uid: str, **params*)

更新某权限指定类型的所有者，仅按提供的数据做局部修改该接口内的 uid ，对于 user 为 username，对于 node 为 node_uid (PATCH /siteapi/v1/perm/<uid>/)

Parameters:

uid (***str***) 权限唯一标识

owner_subject (***str***) 权限所有者类型

value (***bool***) 最终判定结果

status (***int***) 授权状态

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.update_perm_owner(...)
>>> print('perm is', perm)
```

External Documentation

关于 params 参数 详情请参阅 API 文档。

query_specified_user_perm(*username: str*, ***params*)

获取用户所有权限, 包括所有授权、未授权的权限 (`GET /siteapi/v1/perm/user/<username>/`)

Parameters:

username (***str***) 用户唯一标识

action (***str***) 特定操作

action_except (***bool***) 排除某操作，惯用 ‘‘action_except=access‘‘获取应用内权限

scope (***int***) 与应用 ‘‘uid‘‘对应，惯用该字段获取某应用下权限

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.query_specified_user_perm(...)
>>> print('perm is', perm)
```

update_specified_user_perm(*username: str*, *json_body: dict*)

更新用户权限 (`PATCH /siteapi/v1/perm/user/<username>/`)

Parameters:

username (***str***) 用户唯一标识

action (***str***) 特定操作

action_except (***bool***) 排除某操作，惯用 ‘‘action_except=access‘‘获取应用内权限

scope (***int***) 与应用 ‘‘uid‘‘对应，惯用该字段获取某应用下权限

json_body (***int***) 权限的部分元信息，详见接口文档

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.update_specified_user_perm(...)
>>> print('perm is', perm)
```

External Documentation

关于 权限的元数据 详情请参阅 API 文档。

query_dept_perm(*uid: str, **params*)

获取部门所有权限, 包括所有授权、未授权的权限 (GET /siteapi/v1/perm/dept/<uid>/)

Parameters:

uid (***str***) 部门唯一标识

action (***str***) 特定操作

action_except (***bool***) 排除某操作，惯用 "action_except=access"获取应用内权限

scope (***int***) 与应用 "uid"对应，惯用该字段获取某应用下权限

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.query_dept_perm(...)
>>> print('perm is', perm)
```

update_dept_perm(*uid: str, json_body: dict*)

获取部门所有权限, 包括所有授权、未授权的权限 (PATCH /siteapi/v1/perm/dept/<uid>/)

Parameters:

uid (***str***) 部门唯一标识

action (***str***) 特定操作

action_except (***bool***) 排除某操作，惯用 "action_except=access"获取应用内权限

scope (***int***) 与应用 "uid"对应，惯用该字段获取某应用下权限

json_body (***dict***) 权限的元信息, 详情参见接口文档

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.update_dept_perm(...)
>>> print('perm is', perm)
```

query_group_perm(*uid: str, **params*)

获取组所有权限, 包括所有授权、未授权的权限 (GET /siteapi/v1/perm/group/<uid>/)

Parameters:

uid (***str***) 组的唯一标识

action (***str***) 特定操作

action_except (***bool***) 排除某操作，惯用 "action_except=access"获取应用内权限

scope (***int***) 与应用 "uid"对应，惯用该字段获取某应用下权限

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.query_group_perm(...)
>>> print('perm is', perm)
```

update_group_perm(*uid: str*, *json_body: dict*)

获取组所有权限, 包括所有授权、未授权的权限 (PATCH /siteapi/v1/perm/group/<uid>/)

Parameters:

uid (***str***) 组唯一标识

action (***str***) 特定操作

action_except (***bool***) 排除某操作，惯用 ‘‘action_except=access‘‘获取应用内权限

scope (***int***) 与应用 ‘‘uid‘‘对应，惯用该字段获取某应用下权限

json_body (***dict***) 权限的元信息, 详情参见接口文档

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.update_group_perm(...)
>>> print('perm is', perm)
```

query_node_perm(*node_uid: str*, ***params*)

获取节点所有权限, 包括所有授权、未授权的权限 (GET /siteapi/v1/perm/node/<uid>/)

Parameters:

node_uid (***str***) 节点的唯一标识

action (***str***) 特定操作

action_except (***bool***) 排除某操作，惯用 ‘‘action_except=access‘‘获取应用内权限

scope (***int***) 与应用 ‘‘uid‘‘对应，惯用该字段获取某应用下权限

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.query_node_perm(...)
>>> print('perm is', perm)
```

update_node_perm(*node_uid: str*, *json_body: dict*)

获取组所有权限, 包括所有授权、未授权的权限 (PATCH /siteapi/v1/perm/node/<uid>/)

Parameters:

node_uid (***str***) 节点的唯一标识

action (***str***) 特定操作

action_except (*bool*) 排除某操作，惯用 ‘‘action_except=access‘‘获取应用内权限

scope (*int*) 与应用 ‘‘uid‘‘对应，惯用该字段获取某应用下权限

json_body (*dict*) 权限的元信息, 详情参见接口文档

Examples

```
>>> pc = arkid_client.PermClient(...)
>>> perm = pc.update_node_perm(...)
>>> print('perm is', perm)
```

1.3.7 基础设施服务客户端

class arkid_client.infrastructure.**InfrastructureClient**(*base_url*, *authorizer=None*, ***kwargs*)

基类：*arkid_client.base.BaseClient*

基础设施管理客户端，用于与 ArkID 服务端基础设施管理相关接口的访问操作。

Methods

- *get_sms_captcha()*
- *verify_sms_captcha()*

get_sms_captcha(*action: str*, *mobile: str*, ***kwargs*)

获取短信验证码操作包括：注册、登录、重置密码、激活账号、重置手机、通用 (POST /siteapi/v1/service/sms/*/)

Parameters:

action (*str*) 1）register (注册) 2）login (登录) 3）reset_password (重置密码) 4）activate_user (激活账号) 5）update_mobile (重置手机) 6) general (通用)

mobile (*str*) 手机号

Examples

```
>>> ic = arkid_client.InfrastructureClient(...)
>>> node = ic.get_sms_captcha('register', 'example')
```

verify_sms_captcha(*action: str*, *mobile: str*, *code: str*)

验证短信验证码操作包括：注册、登录、重置密码、激活账号、重置手机、通用 (GET /siteapi/v1/service/sms/*/)

Parameters:

action (*str*) 1）register (注册) 2）login (登录) 3）reset_password (重置密码) 4）activate_user (激活账号) 5）update_mobile (重置手机) 6) general (通用)

mobile (*str*) 手机号

Examples

```
>>> ic = arkid_client.InfrastructureClient(...)
>>> node = ic.verify_sms_captcha('register', 'example')
```

1.3.8 应用服务客户端

class arkid_client.app.**AppClient**(*base_url*, *authorizer=None*, ***kwargs*)

基类: *arkid_client.base.BaseClient*

应用管理客户端，用于与 ArkID 服务端应用管理相关接口的访问操作。

Methods

- *query_app_list()*
- *create_app()*
- *query_app()*
- *update_app()*
- *delete_app()*
- *register_app()*

query_app_list(*oid: str*, ***params*)

获取应用信息列表 (GET /siteapi/v1/org/<oid>/app/)

Parameters:

name (***str***) 查询关键字，进行用户名、姓名、邮箱、手机号模糊搜索

node_uid (***str***) 查询该节点的权限

user_uid (***int***) 查询该用户权限

owner_access (***Boolean***) 限定访问权限结果

Examples

```
>>> ac = arkid_client.AppClient(...)
>>> apps = ac.query_app_list(...)
>>> for app in apps:
>>>     print(app['name'], 'uid: '
>>>           ,app['uid'])
```

create_app(*oid: str*, *json_body: dict*)

创建应用 (POST /siteapi/v1/org/<oid>/app/)

Parameters:

json_body (*dict*) 应用的元信息, 参数详情请参考接口文档

Examples

```
>>> ac = arkid_client.AppClient(...)
>>> app = ac.create_app(...)
>>> print('app is', app)
```

External Documentation

关于 应用的元数据 详情请参阅 API 文档。

`query_app`(*oid: str*, *uid: str*)

获取特定应用 (`GET /siteapi/v1/org/<oid>/app/<uid>/`)

Parameters:

`oid` (***str***) 组织的唯一标识

`uid` (***str***) 应用的唯一标识

Examples

```
>>> ac = arkid_client.AppClient(...)
>>> app = ac.query_app()
>>> print('app: ', app)
```

`update_app`(*oid: str*, *uid: str*, *json_body: dict*)

修改特定应用 (`PATCH /siteapi/v1/org/<oid>/app/<uid>/`)

Parameters:

`oid` (***str***) 组织的唯一标识

`uid` (***str***) 应用的唯一标识

Examples

```
>>> ac = arkid_client.AppClient(...)
>>> app = ac.update_app(...)
>>> print('app: ', app)
```

`delete_app`(*oid: str*, *uid: str*)

修改特定应用 (`DELETE /siteapi/v1/org/<oid>/app/<uid>/`)

Parameters:

`oid` (***str***) 组织的唯一标识

`uid` (***str***) 应用的唯一标识

Examples

```
>>> ac = arkid_client.AppClient(...)
>>> app = ac.delete_app(...)
```

`register_app`(*oid: str*, *uid: str*, *protocol: str*, *json_body: dict*)

注册应用 (`PATCH /siteapi/v1/org/<oid>/app/<uid>/*/`)

Parameters:

`oid` (***str***) 组织的唯一标识

`uid` (***str***) 应用的唯一标识

`protocol` (***str***) 应用所采用的协议

`json_body` (***dict***) 应用的元信息

Examples

```
>>> ac = arkid_client.AppClient(...)
>>> app = ac.register_app(...)
>>> print('app: ', app)
```

1.3.9 ArkID 客户端

ArkID Client 集成之前所述的所有客户端 (除认证客户端以外) 的功能, 为用户提供统一的接口来访问 ArkID 的服务

`class arkid_client.client.ArkIDClient`(*base_url*, *authorizer=None*, **args*, ***kwargs*)

由各种 ArkID 客户端集成而生，他提供的所有功能都只是各个客户端的简单封装，用来一致对外界用户使用; 默认加载的为 user 相关服务接口当然，如果您足够熟悉本项目，您也可以直接实例化所需要的指定客户端。

Methods

- *query_user()*
- *query_isolated_user()*
- *query_user()*
- *create_user()*
- *update_user()*
- *delete_user()*
- *query_specified_perm()*
- *query_own_org()*

- *query_org()*
- *create_org()*
- *delete_org()*
- *update_org()*
- *query_orguser()*
- *add_orguser()*
- *delete_orguser()*
- *query_orguser()*
- *update_orguser()*
- *get_org_invitation_key()*
- *refresh_org_invitation_key()*
- *view_org_by_invitation_key()*
- *join_org_by_invitation_key()*
- *query_node()*
- *view_node()*
- *update_node()*
- *delete_node()*
- *get_node_tree_list()*
- *get_node_tree()*
- *view_node_tree()*
- *get_subnode()*
- *create_subnode()*
- *add_subnode()*
- *sort_subnode()*
- *query_user_under_node()*
- *add_user_under_node()*
- *delete_user_under_node()*
- *override_user_under_node()*
- *sort_user_under_node()*
- *move_out_user_under_node()*

- *view_perm()*
- *view_profile()*
- *view_current_org()*
- *switch_current_org()*
- *query_apps()*
- *get_sms_captcha()*
- *verify_sms_captcha()*
- *query_all_perm()*
- *create_perm()*
- *query_perm()*
- *update_perm()*
- *query_perm_owner()*
- *update_perm_owner()*
- *query_specified_user_perm()*
- *update_specified_user_perm()*
- *query_dept_perm()*
- *update_dept_perm()*
- *query_group_perm()*
- *update_group_perm()*
- *query_node_perm()*
- *update_node_perm()*
- *query_app_list()*
- *create_app()*
- *query_app()*
- *update_app()*
- *delete_app()*
- *register_app()*

query_user_list(***kwargs*)

获取用户信息列表 (`GET /siteapi/v1/user/`)

Parameters:

keyword (*str*) 查询关键字，进行用户名、姓名、邮箱、手机号模糊搜索

wechat_unionid (*str*) 微信客户端 openid

page (*int*) 用于分页，*Default: 1*

page_size (*int*) 指定分页大小，*Default: 30*

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> users = uc.query_user(...)
>>> for user in users:
>>>     print(user['username'], 'id: '
>>>           ,user['id'])
```

External Documentation

关于 用户的元数据 详情请参阅 API 文档。

query_isolated_user(***kwargs*)

获取所有独立用户 (GET /siteapi/v1/user/isolated/)

Parameters:

page (*int*) 用于分页，*Default: 1*

page_size (*int*) 指定分页大小，*Default: 30*

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> users = uc.query_isolated_user(...)
>>> for user in users:
>>>     print(user['username'], 'id: '
>>>           ,user['id'])
```

query_user(***kwargs*)

获取指定用户的信息 (GET /siteapi/v1/user/<username>/)

Parameters:

username (*str*) 用户唯一标识

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> user = uc.query_user(...)
>>> print(user['username'], 'id: '
>>>       ,user['id'])
```

create_user(**kwargs)

创建用户 (需要管理员权限) (POST /siteapi/v1/user/)

Parameters:

json_body (*dict*)

group_uids (*list[str]*) 用户组 uuid 的集合

dept_uids (*list[str]*) 部门 uuid 的集合

user (*dict*) 用户的元信息, 参数详情请参考接口文档

node_uids (*list[str]*) (可选的) 此字段提供时会忽略 group_uids, dept_uids

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> user_data = {
>>>   "user": {
>>>     "password": "example",
>>>     "username": "example"
>>>     }
>>> }
>>> user = uc.create_user(user_data)
>>> print(user['username'], 'id: '
>>>           ,user['id'])
```

External Documentation

关于 用户的元数据 详情请参阅 API 文档。

update_user(**kwargs)

修改指定用户的信息 (PATCH /siteapi/v1/user/<username>)

Parameters:

username (*str*) 用户唯一标识

json_body (*dict*) 用户的元信息, 参数详情请参考接口文档

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> query_data = {
>>>   "password": "example",
>>>   "private_email": "example@org.com"
>>> }
>>> user = uc.update_user(username, query_data)
```

(下页继续)

(续上页)

```
>>> print(user['username'], 'id: '
>>>           ,user['id'])
```

External Documentation

关于 用户的元数据 详情请参阅 API 文档。

`delete_user`(***kwargs*)

删除指定用户的信息 (`DELETE /siteapi/v1/user/<username>/`)

Parameters:

username (***str***) 用户唯一标识

Examples

```
>>> uc = arkid_client.UserClient(...)
>>> uc.delete_user(username)
```

`query_specified_perm`(***kwargs*)

获取用户权限详情，包括权限来源 (`GET /siteapi/v1/user/<username>/perm/<uid>/`)

Parameters:

username (***str***) 用户唯一标识

uid (***str***) 权限唯一标识

Examples

```
>>> pc = arkid_client.UserClient(...)
>>> perm = pc.query_specified_perm(...)
>>> print('perm is', perm)
```

`query_own_org`(***kwargs*)

查询用户所在的组织 (`GET /siteapi/v1/org/`)

Parameters:

role (***str***) 在组织内的角色

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> orgs = oc.query_own_org(role='admin')
>>> for org in orgs:
>>>     print(org['oid'], 'name: '
>>>           ,org['name'])
```

query_org(***kwargs*)

查看指定组织的信息 (GET /siteapi/v1/org/<oid>/)

Parameters:

oid (***str***) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.query_org(oid)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

create_org(***kwargs*)

创建组织 (POST /siteapi/v1/org/)

Parameters:

json_body (***dict***) 组织的元信息，参数详情请参考接口文档

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org_data = {
>>>   "name": "example",
>>> }
>>> org = oc.create_org(org_data)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

External Documentation

关于 组织的元数据 详情请参阅 API 文档。

delete_org(***kwargs*)

删除指定组织的信息 (DELETE /siteapi/v1/org/<oid>/)

Parameters:

oid (***str***) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> oc.delete_org(oid)
```

update_org(***kwargs*)

修改指定组织的信息 (PATCH /siteapi/v1/org/<oid>/)

Parameters:

`oid` (*str*) 组织的唯一标识

`json_body` (*dict*) 组织的元信息，参数详情请参考接口文档

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org_data = {
>>>   "name": "example",
>>> }
>>> org = oc.update_org(oid, org_data)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

External Documentation

关于 组织的元数据 详情请参阅 API 文档。

`query_orguser_list`(***kwargs*)

查看特定组织的成员信息 (`GET /siteapi/v1/org/<oid>/user/`)

Parameters:

`oid` (*str*) 组织的唯一标识

`page` (*int*) 用于分页，*Default: 1*

`page_size` (*int*) 指定分页大小，*Default: 30*

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.query_orguser_list(oid)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

`add_orguser`(***kwargs*)

向指定组织中添加成员 (`PATCH /siteapi/v1/org/<oid>/user/`)

Parameters:

`oid` (*str*) 组织的唯一标识

`usernames` (*list*) 用户的唯一标识组成的列表

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> usernames = [
```

(下页继续)

(续上页)

```
>>>     'username1',
>>>     'username2',
>>>         ...
>>>     'usernamen'
>>> ]
>>> org = oc.add_orguser(oid, usernames)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

delete_orguser(***kwargs*)

从指定组织中移除成员 (PATCH /siteapi/v1/org/<oid>/user/)

Parameters:

oid (*str*) 组织的唯一标识

usernames (*list*) 用户的唯一标识组成的列表

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> usernames = [
>>>     'username1',
>>>     'username2',
>>>         ...
>>>     'usernamen'
>>> ]
>>> org = oc.delete_orguser(oid, usernames)
>>> print(org['oid'], 'name: '
>>>       ,org['name'])
```

query_orguser(***kwargs*)

查看指定组织的指定成员的信息 (GET /siteapi/v1/org/<oid>/user/<username>/)

Parameters:

oid (*str*) 组织的唯一标识

username (*str*) 用户唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> user = oc.query_orguser(oid, username)
>>> print('user is', user)
```

update_orguser(***kwargs*)

编辑指定组织的指定成员的信息 (PATCH /siteapi/v1/org/<oid>/user/<username>/)

Parameters:

oid (*str*) 组织的唯一标识

username (*str*) 用户唯一标识

json_body (*dict*)

email (*str*) 成员邮箱

employee_number (*str*) 成员工号

position (*str*) 成员职位

hiredate (*str*) 成员雇佣日期

remark (*str*) 成员备注

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> user_data = {
>>>   "email": "example@org.com",
>>> }
>>> user = oc.update_orguser(oid, username, user_data)
>>> print(user['id'], 'name: '
>>>       ,user['name'])
```

External Documentation

关于 成员的元数据 详情请参阅 API 文档。

get_org_invitation_key(***kwargs*)

获取指定组织邀请用的最新的密钥 (GET /siteapi/v1/org/<oid>/invitation/)

Parameters:

oid (*str*) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> key = oc.get_org_invitation_key(oid)
>>> print('key: ', key)
```

refresh_org_invitation_key(***kwargs*)

刷新指定组织邀请用的最新的密钥 (PUT /siteapi/v1/org/<oid>/invitation/)

Parameters:

oid (*str*) 组织的唯一标识

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> key = oc.refresh_org_invitation_key(oid)
>>> print('key: ', key)
```

view_org_by_invitation_key(***kwargs*)

使用邀请密钥查看指定组织的信息 (GET /siteapi/v1/org/<oid>/invitation/<invite_link_key>/)

Parameters:

oid (*str*) 组织的唯一标识

invite_link_key (*str*) 组织邀请密钥

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.view_org_by_invitation_key(oid, invite_link_key)
>>> print('org: ', org)
```

join_org_by_invitation_key(***kwargs*)

使用邀请密钥加入指定组织 (POST /siteapi/v1/org/<oid>/invitation/<invite_link_key>/)

Parameters:

oid (*str*) 组织的唯一标识

invite_link_key (*str*) 组织邀请密钥

Examples

```
>>> oc = arkid_client.OrgClient(...)
>>> org = oc.join_org_by_invitation_key(oid, invite_link_key)
>>> print('org: ', org)
```

query_node(***kwargs*)

查询指定节点的信息 (GET /siteapi/v1/node/<node_uid>/)

Parameters:

node_uid (*str*) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node = nc.query_node(node_uid)
>>> print('node is', node)
```

`view_node`(***kwargs*)

用户查询指定节点的信息 (`GET /siteapi/v1/ucenter/node/<node_uid>/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node = nc.view_node(node_uid)
>>> print('node is', node)
```

`update_node`(***kwargs*)

修改指定节点的信息 (`PATCH /siteapi/v1/node/<node_uid>/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

`json_body` (***dict***) 节点的元信息, 参数详情请参考接口文档

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_data = {
>>>     'name': 'example'
>>> }
>>> node = nc.update_node(node_uid, node_data)
>>> print('node is', node)
```

External Documentation

关于 节点的元数据 详情请参阅 API 文档。

`delete_node`(***kwargs*)

删除指定节点的信息 (`DELETE /siteapi/v1/node/<node_uid>/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

`ignore_user` (***bool***) 用于删除节点，当 “true“时，若节点下有人员存在时，会先将人员从节点内删除，再删除此节点。

Examples self.logger.info(“正在调用 NodeClient.delete_node() 接口与 ArkID 服务端进行交互”) >>> nc = arkid_client.NodeClient(···) >>> nc.delete_node(node_uid, ignore_user=True)

`get_node_tree_list`(***kwargs*)

需要管理员权限，获取节点及其子孙节点列表，将某节点下的子树以列表形式返回，包括该节点自身，前序遍历。(`GET /siteapi/v1/node/<node_uid>/list/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> tree_list = nc.get_node_tree_list(node_uid)
>>> print('tree_list is', tree_list)
```

`get_node_tree`(***kwargs*)

需要管理员权限，管理员访问到的数据将由管理范围决定，数据包括从该节点起始的完整树。(`GET /siteapi/v1/node/<node_uid>/tree/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

`user_required` (***bool***) 是否需要成员信息

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> tree = nc.get_node_tree(node_uid, user_required=True)
>>> print('tree is', tree)
```

`view_node_tree`(***kwargs*)

普通用户访问节点下结构，访问到的数据将由节点可见范围决定，数据包括从该节点起始的完整树。(`GET /siteapi/v1/ucenter/node/<node_uid>/tree/`)

Parameters:

`node_uid` (***str***) 节点的唯一标识

`user_required` (***bool***) 是否需要成员信息

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> tree = nc.view_node_tree(node_uid, user_required=True)
>>> print('tree is', tree)
```

get_subnode(**kwargs)

获取指定节点的子节点信息 (GET /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (*str*) 节点的唯一标识

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> subnode = nc.get_subnode(node_uid)
>>> print('subnode is', subnode)
```

create_subnode(**kwargs)

创建指定节点的子节点 (POST /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (*str*) 节点的唯一标识

json_body (*dict*) 节点的元信息, 参数详情请参考接口文档

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_data = {
>>>     'name': 'example'
>>> }
>>> subnode = nc.create_subnode(node_uid, node_data)
>>> print('subnode is', subnode)
```

External Documentation

关于 节点的元数据 详情请参阅 API 文档。

add_subnode(**kwargs)

向指定节点添加子节点 (PATCH /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (*str*) 节点的唯一标识

node_uids (*list*) 节点唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_uids = [
>>>     'node1',
>>>     'node2',
```

(下页继续)

(续上页)

```
>>>         ...
>>>     'noden'
>>> ]
>>> node = nc.add_subnode(node_uid, node_uids)
>>> print('node is', node)
```

sort_subnode(***kwargs*)

对指定子节点按指定位置进行排序 (PATCH /siteapi/v1/node/<node_uid>/node/)

Parameters:

node_uid (***str***) 节点的唯一标识

node_uids (***list***) 节点唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> node_uids = [
>>>     'node1',
>>>     'node2',
>>>         ...
>>>     'noden'
>>> ]
>>> node = nc.sort_subnode(node_uid, node_uids)
>>> print('node is', node)
```

query_user_under_node(***kwargs*)

查询指定节点下的直属人员的信息 (GET /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (***str***) 节点的唯一标识

****params** (***dict***) 用户的元信息，参数详情请参考接口文档

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> users = nc.query_user_under_node(node_uid)
>>> for user in users:
>>>     print('user is', user)
```

External Documentation

关于 节点的元数据 详情请参阅 API 文档。

add_user_under_node(***kwargs*)

向指定节点添加指定成员 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (*str*) 节点的唯一标识

user_uids (*list*) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.add_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

delete_user_under_node(***kwargs*)

从指定节点移除指定成员 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (*str*) 节点的唯一标识

user_uids (*list*) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.delete_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

sort_user_under_node(***kwargs*)

对指定人按指定位置进行排序 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (*str*) 节点的唯一标识

user_uids (*list*) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.sort_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

override_user_under_node(***kwargs*)

重置指定节点的指定用户 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (*str*) 节点的唯一标识

user_uids (*list*) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.override_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

move_out_user_under_node(***kwargs*)

将这些人从该节点移除，并加到指定节点 (PATCH /siteapi/v1/node/<node_uid>/user/)

Parameters:

node_uid (*str*) 节点的唯一标识

user_uids (*list*) 用户唯一标识组成的列表

Examples

```
>>> nc = arkid_client.NodeClient(...)
>>> user_uids = [
>>>     'user1',
>>>     'user2',
>>>         ...
>>>     'usern'
>>> ]
>>> node = nc.move_out_user_under_node(node_uid, user_uids)
>>> print('node is', node)
```

1.3.10 基类客户端

所有类型的服务客户端均支持由 BaseClient 提供的底层 API 。

class arkid_client.base.BaseClient(*base_url: str*, *service: str*, *environment: str = None*, *base_path: str = None*, *authorizer: object = None*, *app_name: str = None*, *http_timeout: float = None*, **args*, ***kwargs*)

简单的基类客户端，可处理 ArkID REST APIs 返回的错误信息。封装 requests.Session 对象为一个简化的接口，该接口不公开来自请求的任何内容。注意：强烈建议您不要尝试直接实例化 BaseClient

Parameters

base_url (***str***) ArkID 服务端的根地址，用户无需进行任何有关 ArkID 服务端的地址配置操作，只需在初始化 BaseClient 实例时传入 base_url 参数即可

authorizer (ArkIDAuthorizer)

authorizer 认证授权器用于生成 HTTP 请求的头部认证信息

app_name (***str***) (可选) 用于标识调用方，往往指代正在使用 ArkID SDK 进行开发的项目。此参数与客户端的任何操作无关。仅仅作为请求头部 User-Agent 的一部分发送给 ArkID 团队，以方便调试出现的问题。

http_timeout (***float***) HTTP 连接响应的等待时间（单位：s)。默认 60 。如果传入的值为 -1 ，代表请求发送后将无限期挂起。

所有其它的初始化参数用于子类内部使用

set_app_name(*app_name: str*)

设置一个应用名称（用户代理）并发送给 ArkID 服务端。

注意：建议应用开发者提供一个应用名称给 ArkID 团队，以便在可能的情况下促进与 ArkID 的交互问题的解决。

`get`(*path: str*, *params: dict = None*, *headers: dict = None*, *response_class: object = None*, *retry_401: bool = True*)
以 GET 方式向指定地址发送 HTTP 请求。

Parameters

`path` (***string***) 请求的路径，有无正斜杠均可

`params` (***dict***) 编码为 Query String 的参数

`headers` (***dict***) 添加到请求中的 HTTP 标头

`response_class` (***class***) 响应对象的类型，由客户端的 `default_response_class` 重写

`retry_401` (***bool***) 如果响应码为 401 并且 `self.authorizer` 支持重试，那么会自动进行新的认证。

返回 *ArkIDHTTPResponse* object

`post`(*path: str*, *json_body: dict = None*, *params: dict = None*, *headers: dict = None*, *text_body: dict = None*, *response_class: object = None*, *retry_401: bool = True*)
以 POST 方式向指定地址发送 HTTP 请求。

Parameters

`path` (***string***) 请求的路径，有无正斜杠均可

`params` (***dict***) 编码为 Query String 的参数

`headers` (***dict***) 添加到请求中的 HTTP 标头

`json_body` (***dict***) 请求体中通过 JSON 编码的数据

`text_body` (***string or dict***) 用作请求主体的原始字符串，或是以 HTTP 形式编码的字典数据

`response_class` (***class***) 响应对象的类型，由客户端的 `default_response_class` 重写

`retry_401` (***bool***) 如果响应码为 401 并且 `self.authorizer` 支持重试，那么会自动进行新的认证。

返回 *ArkIDHTTPResponse* object

`delete`(*path: str*, *params: dict = None*, *headers: dict = None*, *response_class: object = None*, *retry_401: bool = True*)
以 DELETE 方式向指定地址发送 HTTP 请求。

Parameters

`path` (***string***) 请求的路径，有无正斜杠均可

`params` (***dict***) 编码为 Query String 的参数

`headers` (***dict***) 添加到请求中的 HTTP 标头

response_class (*class*) 响应对象的类型，由客户端的 default_response_class 重写

retry_401 (*bool*) 如果响应码为 401 并且 self.authorizer 支持重试，那么会自动进行新的认证。

返回 *ArkIDHTTPResponse* object

put(*path: str*, *json_body: dict = None*, *params: dict = None*, *headers: dict = None*, *text_body: dict = None*, *response_class: object = None*, *retry_401: bool = True*)
以 PUT 方式向指定地址发送 HTTP 请求。

Parameters

path (*string*) 请求的路径，有无正斜杠均可

params (*dict*) 编码为 Query String 的参数

headers (*dict*) 添加到请求中的 HTTP 标头

json_body (*dict*) 请求体中通过 JSON 编码的数据

text_body (*string or dict*) 用作请求主体的原始字符串，或是以 HTTP 形式编码的字典数据

response_class (*class*) 响应对象的类型，由客户端的 default_response_class 重写

retry_401 (*bool*) 如果响应码为 401 并且 self.authorizer 支持重试，那么会自动进行新的认证。

返回 *ArkIDHTTPResponse* object

patch(*path: str*, *json_body: dict = None*, *params: dict = None*, *headers: dict = None*, *text_body: dict = None*, *response_class: object = None*, *retry_401: bool = True*)
以 PATCH 方式向指定地址发送 HTTP 请求。

Parameters

path (*string*) 请求的路径，有无正斜杠均可

params (*dict*) 编码为 Query String 的参数

headers (*dict*) 添加到请求中的 HTTP 标头

json_body (*dict*) 请求体中通过 JSON 编码的数据

text_body (*string or dict*) 用作请求主体的原始字符串，或是以 HTTP 形式编码的字典数据

response_class (*class*) 响应对象的类型，由客户端的 default_response_class 重写

retry_401 (*bool*) 如果响应码为 401 并且 self.authorizer 支持重试，那么会自动进行新的认证。

返回 *ArkIDHTTPResponse* object

1.4 接口响应

如果没有另作说明，ArkID SDk 客户端的所有方法的返回值都是 ArkIDResponse 对象。一些 ArkIDResponse 对象是可迭代的，在这些情况下，它们的内容也是 ArkIDResponse 对象。

在调用 ArkID 服务的接口时，使用的返回对象为 ArkIDResponse 的子类型 ArkIDHTTPResponse，它丰富了一些自定义数据信息的细节。

1.4.1 泛型响应类

class arkid_client.response.**ArkIDResponse**(*data*, *client=None*)

通用响应基类，只有一个简单的 data 成员。最常见的响应数据是 JSON 字典。为了尽可能不去处理这种类型的反应，ArkIDResponse 对象支持通过字典的形式来直接访问响应内容，如果 data 不是字典结构，将会抛出 TypeError 异常。

```
>>> print('"Response ID": response["id"]') # alias for response.data["id"]
```

ArkIDResponse 对象封装 HTTP 响应的数据给调用者，大多数的操作都是基于与这些数据进行交互。

data

以 Python 形式的数据结构返回响应数据，通常是一个 dict 或者 list。

get(**args*, ***kwargs*)

BaseResponse.get(...) 方法是 BaseResponse.data.get(...) 方法的别名

class arkid_client.response.**ArkIDHTTPResponse**(*http_response*, *client=None*)

基类：*arkid_client.response.ArkIDResponse*

封装底层 HTTP 响应对象。如果响应数据类型是 JSON，则解析后的数据将在 data 中，否则 data 将为 None ，并且 text 将被使用。

变量

- **http_status** – ArkID 服务端返回的 HTTP 响应的状态码 (int)
- **content_type** – ArkID 服务端返回的 HTTP 响应的内容类型 (str)

data

以 Python 形式的数据结构返回响应数据，通常是一个 dict 或者 list。

text

HTTP 响应数据的字符串形式

1.5 异常处理

所有的 ARKID Client 错误都继承自 ArkIDError，所有 SDK 错误类型均可从 arkid_client 导入。因此，您可以通过 ArkIDError 来捕获 ARKID Client 抛出的 * 所有 * 错误, 例如

```
import logging
from arkid_client import UserClient, ArkIDError

try:
    uc = UserClient(...)
    # create with no parameters will throw an exception
    user = uc.create_user()
except ArkIDError:
    logging.exception("ArkID Error!")
    raise
```

在大多数情况下，最好查找 ArkIDError 的特定子类。比如，想要区分网络故障和意外的 API 条件，您需要通过 NetworkError 和 ArkIDAPIError 来定位问题所在:

```
import logging
from arkid_client import (UserClient,
                          ArkIDError, ArkIDAPIError, NetworkError)

try:
    uc = UserClient(...)
    users = uc.query_user()

    for user in users:
        print(user['name'])

    ...
except ArkIDAPIError as e:
    # Error response from the REST service, check the code and message for details.
    logging.error(("Got a ArkID API Error\n"
                   "Error Code: {}\n"
                   "Error Message: {}").format(e.code, e.message))
    raise e
except NetworkError:
    logging.error(("Network Failure. "
                   "Possibly a firewall or connectivity issue"))
    raise
except ArkIDError:
    logging.exception("Totally unexpected ArkIDError!")
    raise
else:
    ...
```

当然，如果您想要了解更多关于响应的信息，您要做的不仅仅只有这些。

由 ArkID SDK 引起的所有错误都应该是 `ArkIDError` 的实例。对 ArkID Client 方法的错误调用通常会引发 `ArkIDSDKUsageError` ，但是，在极少数情况下，可能会引发标准的 python 异常（ `ValueError` , `OSError` 等)

1.5.1 错误类型

class arkid_client.**ArkIDError**

基类：Exception

ArkID Client 错误的基类

class arkid_client.**ArkIDSDKUsageError**

基类：arkid_client.exceptions.ArkIDError, ValueError

当 ArkID Client 检测到它被不正确的使用时会抛出 `ArkIDSDKUsageError` 异常，这些错误通常表示某些关于 ArkID Client 使用的规范 (例如所需的操作顺序) 已被违反。

class arkid_client.**ArkIDAPIError**(*response*, **args*, ***kwargs*)

基类：arkid_client.exceptions.ArkIDError

封装 REST API 的响应信息

变量

- **http_status** – HTTP 响应的状态码 (int)
- **message** – 来自 API 的错误信息。一般来说，其对开发人员是有用的，但在某些情况下它也许适合显示给终端用户。

raw_json

获取从 ArkID API 接收到的响应信息，尝试以 JSON 格式解析数据，并转化为 dict 。

如果响应信息无法通过 JSON 格式加载，则返回 None 。

raw_text

以 string 形式获取响应信息。

class arkid_client.**AuthAPIError**(*response*, **args*, ***kwargs*)

基类：arkid_client.exceptions.ArkIDAPIError

认证服务客户端的错误类型，继承了 message 变量

class arkid_client.exceptions.**UserAPIError**(*response*)

基类：arkid_client.exceptions.ArkIDAPIError

用户服务客户端的错误类型，继承了 message 变量

class arkid_client.exceptions.**OrgAPIError**(*response*)

基类：arkid_client.exceptions.ArkIDAPIError

组织服务客户端的错误类型，继承了 message 变量

class arkid_client.exceptions.NodeAPIError(*response*)

基类：arkid_client.exceptions.ArkIDAPIError

节点服务客户端的错误类型，继承了 message 变量

class arkid_client.NetworkError(*message*, *exception*, **args*, ***kwargs*)

基类：arkid_client.exceptions.ArkIDError

当与 ArkID 服务通信发生错误时会抛出 NetworkError 错误，其为通信方面出现的错误的基类。其在保留原始异常数据的基础上，也可以接受其他一些有用的消息，方便用户明确错误所在。

class arkid_client.ArkIDConnectionError(*message*, *exception*, **args*, ***kwargs*)

基类：arkid_client.exceptions.NetworkError

在发出 REST 请求时发生连接错误。

class arkid_client.ArkIDTimeoutError(*message*, *exception*, **args*, ***kwargs*)

基类：arkid_client.exceptions.NetworkError

REST API 请求超时

class arkid_client.ArkIDConnectionTimeoutError(*message*, *exception*, **args*, ***kwargs*)

基类：arkid_client.exceptions.ArkIDTimeoutError

请求在连接建立期间超时，这些错误可以进行安全地重试。

1.6 API 认证

针对 ArkID Client 的授权操作，目前仅支持 用户名 + 密码的形式，但是这样的授权流程对于用户的隐私信息是不安全的，所以在后续的 ArkID Client 开发中，我们也许会引入 oauth2.0 或者 OIDC 相关的授权协议来加固 ArkID Client。

1.6.1 授权器基类

ArkIDAuthorizer 为一个授权器的抽象基类

class arkid_client.authorizers.base.ArkIDAuthorizer

授权器基类，用于生成有效的授权头部。支持处理无效的的授权头部。

set_authorization_header(*header_dict: dict*)

获取 HTTP 请求头部的 dict 数据，并将 {"Authorization": "..."} 形式的授权信息加入其中。注意：若 Authorization 授权信息已经设置，则此方法将会覆盖原来的授权信息。

handle_missing_authorization(**args*, ***kwargs*)

若 HTTP 请求使用此授权器进行访问时出现 401 （HTTP 请求未经授权）响应，若授权器可以采取某些措施补救这种情况，其将会更新状态并返回 True；若授权器针对这种情况无能为力，其也许会更新一些操作，但是更重要的是，会返回 False。

默认情况下，总是返回 False ，不采取任何操作。

1.6.2 授权器类型

以下所有类型的授权器均可从 `arkid_client.authorizers` 导入

class arkid_client.**NullAuthorizer**

基类：*arkid_client.authorizers.base.ArkIDAuthorizer*

该授权器不实现任何身份验证功能，并尝试去掉请求头部的认证信息。

set_authorization_header(*header_dict*)

如果存在授权头部信息，则从给定头部信息的 `dict` 中尝试删除授权标头。

class arkid_client.**BasicAuthorizer**(*oneid_token: str*)

基类：*arkid_client.authorizers.base.ArkIDAuthorizer*

使用 ArkID 官方默认的 oneid_token 进行基本认证。将在请求头中设置 Token 来向 ArkID 服务端发起请求。**Parameters**

oneid_token (***str***) An basic token for ArkID Auth

set_authorization_header(*header_dict: dict*)

Sets the *Authorization* header to "token <oneid_token>"

Python 模块索引

a

索引

D

G

H

I

J

M

N

O

P

Q

R

S

T

U

V