Misskey.py

YuzuRyo61

2022 年 03 月 03 日

目次

第 1 章

はじめに

Misskey.py は Misskey の API の利用を容易にするためのラッパー API ライブラリです。

1.1 前提

Misskey.py は Python 3.7 以降からサポートしています。

Python 2.x や Python 3.6 以前はサポートされませんのでご注意ください。

1.2 インストール

PyPI から直接インストールすることができます。

```
pip install Misskey.py
```

1.3 基本概念

Misskey.py はメソッドで API リクエストを送信する仕組みになっています。

インスタンス化するときに少なくとも Misskey のインスタンスアドレスを指定するだけで使用できます。

```
from misskey import Misskey

mk = Misskey("mk.example.com")

info = mk.meta()
```

(次のページに続く)

(前のページからの続き)

```
print(info["name"]) # インスタンス名
```

第 2 章

クイックスタート

2.1 基本

Misskey.py のメインのクラスは以下のクラスです。

class misskey.**Misskey**(*address: str = 'https://misskey.io'*, *i: Optional[str] = None*, *session: Optional[requests.sessions.Session] = None*)

Misskey API client class.

Args: address (str): Instance address. You can also include the URL protocol. If not specified, it will be automatically recognized as `https`.

i (`str`, optional): Misskey API token. If you have an API token, you can assign it at instantiation.

session (`requests.Session`, optional): If you have prepared the `requests.Session` class yourself, you can assign it here. Normally you do not need to specify it.

Raises: MisskeyAuthorizeFailedException: Raises if token validation fails during instantiation.

クラスをインスタンス化するために、少なくとも、Misskey インスタンスアドレスを指定してください。

必要な場合は URL プロトコルを含めることもできます。

```
from misskey import Misskey

mk = Misskey("mk.example.com")

mk_local = Misskey("http://localhost:3000") # URL プロトコルを含めた場合
```

各メソッドにエンドポイント名を記載しています。

使用するインスタンスの Misskey API ドキュメントと併用してご確認下さい。

例: `i` → Misskey API エンドポイント `/api/i` へ HTTP リクエスト

2.2 トークンを加える

Misskey インスタンスで使用できるトークンを持っている場合は、Misskey.py はインスタンス化時、もしくはプロパティに代入することで使用することができます。

2.2.1 インスタンス化時に代入

コンストラクタに i という引数にトークンを代入します。

```
from misskey import Misskey

mk = Misskey("mk.example.com", i="xxxxxxxxxx")

print(mk.i()) # あなたのプロフィールが表示されます
```

トークンが有効でない場合は以下の例外が出されます。

exception misskey.exceptions.**MisskeyAuthorizeFailedException**

2.2.2 プロパティに代入

インスタンスの token プロパティに代入します。

```
from misskey import Misskey

mk = Misskey("mk.example.com")

mk.token = "xxxxxxxxxx"

print(mk.i()) # あなたのプロフィールが表示されます
```

トークンが有効でない場合は以下の例外が出されます。

exception misskey.exceptions.**MisskeyAuthorizeFailedException**

2.3 投稿をしてみる

トークンを含めたインスタンスで、Misskey に Note を投稿してみましょう。

メソッド `notes_create` を使用すると、Note を作成することができます。

`Misskey.notes_create`(*text: Optional[str] = None*, *cw: Optional[str] = None*, *visibility: Union[misskey.enum.NoteVisibility, str] = NoteVisibility.PUBLIC*, *visible_user_ids: Optional[List[str]] = None*, *via_mobile: bool = False*, *local_only: bool = False*, *no_extract_mentions: bool = False*, *no_extract_hashtags: bool = False*, *no_extract_emojis: bool = False*, *file_ids: Optional[List[str]] = None*, *reply_id: Optional[str] = None*, *renote_id: Optional[str] = None*, *poll_choices: Optional[Union[List[str], Tuple[str]]] = None*, *poll_multiple: bool = False*, *poll_expires_at: Optional[Union[int, datetime.datetime]] = None*, *poll_expired_after: Optional[Union[int, datetime.timedelta]] = None*) → dict

Create a note.

Args: text (`str`, optional): Specify the text.

cw (`str`, optional): Specify the CW(Content Warning).

visibility (`str`, default: `public`): Post range. Specifies the enumeration in `NoteVisibility`.

visible_user_ids (`list` of `str`, optional): If `visibility` is `specified`, specify the user ID in the list.

via_mobile (`bool`, optional): Specify whether to post from mobile. It doesn't work with recent Misskey versions.

local_only (`bool`, optional): Specifies whether to post only the instance you are using.

no_extract_mentions (`bool`, optional): Specifies whether to detect mentions from the text.

no_extract_hashtags (`bool`, optional): Specifies whether to detect hashtags from the text.

no_extract_emojis (`bool`, optional): Specifies whether to detect emojis from the text.

file_ids (`list` of `str`, optional): Specify the file ID to attach in the list.

reply_id (`str`, optional): Specify the Note ID of the reply destination.

renote_id (`str`, optional): Specify the Note ID to renote.

poll_choices (`list` of `str`, optional): Specify the voting item. You can specify 2 or more and 10 or less.

poll_multiple (`bool`, optional): Specifies whether to allow multiple votes. This is valid only when `poll_choices` is specified.

poll_expires_at (`datetime.datetime`, optional): Specify the expiration date of the vote. If not specified, it will be indefinite. Cannot be used with `poll_expired_after`.

poll_expired_after (`datetime.timedelta`, optional): Specifies the validity period of the vote. If not specified, it will be indefinite. Cannot be used with `poll_expired_at`.

Endpoint: `notes/create`

Note: `token` must be set in the instance.

You must specify at least either `text` or `files_id`.

Returns: dict: The dict of the posted result is returned.

Raises: MisskeyAPIException: Raise if the API request fails.

以下がサンプルコードです。

```
from misskey import Misskey

mk = Misskey("mk.example.com", i="xxxxxxxxxx")

new_note = mk.notes_create(text="Hello Misskey.py!")

print(new_note["createdNote"]["id"]) # 投稿された Note の ID が表示されます
```

コードを実行すると Note が投稿されます。ブラウザなどで確認してみてください。

第 3 章

ファイルのアップロード

Misskey.py は Misskey のドライブにファイルをアップロードするためのメソッドを含んでいます。

`Misskey.`**`drive_files_create`**(*file: IO*, *folder_id: Optional[str] = None*, *name: Optional[str] = None*, *is_sensitive: bool = False*, *force: bool = False*) → dict

Upload a file to the drive.

Args: file (`IO`): Assign a file stream. As an example, the one opened by the `open` function is included.

folder_id (str, optional): Specify the folder ID.

name (str, optional): Specify the file name.

is_sensitive (bool, optional): Specify whether the file is sensitive.

force (bool, optional): Specify whether to overwrite the file if it already exists.

Endpoint: `drive/files/create`

Returns: dict: Returns the file information.

Raises: MisskeyAPIException: Raise if the API request fails.

ファイルをアップロードするには、ファイルストリーミングを開き、それをメソッドに渡します。

```python
from misskey import Misskey

mk = Misskey("mk.example.com", i="xxxxxxxxxx")

with open("test.png", "rb") as f:
    data = mk.drive_files_create(f)

print(data["id"]) # アップロードされたファイルの ID を表示
```

ファイルストリーミング以外指定してない場合は、名前はファイル名、フォルダはルートフォルダが指定されます。

第 4 章

MiAuth でトークンを取得する

Misskey は認証方法の一つとして MiAuth があります。

この認証方法は Misskey 12.39.1 以降のインスタンスでサポートされます。

MiAuth での認証を容易にするため、Misskey.py には MiAuth クラスがあります。

class misskey.**MiAuth**(*address: str = 'https://misskey.io'*, *session_id: Optional[Union[uuid.UUID, str]] = None*, *name: str = 'Misskey.py'*, *icon: Optional[str] = None*, *callback: Optional[str] = None*, *permission: Optional[Union[List[Optional[Union[*misskey.enum.Permissions*, str]]], Tuple[Optional[*misskey.enum.Permissions*]], Set[Optional[*misskey.enum.Permissions*]]]] = None*, *session: Optional[requests.sessions.Session] = None*)

Misskey Authentication Class

Args: address (str): Instance address. You can also include the URL protocol. If not specified, it will be automatically recognized as `https`.

session_id (`uuid.UUID` or `str`, optional): Session ID for performing MiAuth. If not specified, it will be set automatically.

name (`str`, optional): App name.

icon (`str`, optional): The URL of the icon.

callback (`str`, optional): App callback URL.

permission (`list` of `str`, optional): The permissions that the app can use. You can specify an enumeration of Permissions.

session (`requests.Session`, optional): If you have prepared the `requests.Session` class yourself, you can assign it here. Normally you do not need to specify it.

4.1 認証 URL を作成する

まず、MiAuth クラスをインスタンス化します。

少なくとも、インスタンスアドレス (address) と名前 (name) を入力します。

また、必要に応じて権限 (permission) やコールバック URL(callback) を追加で指定します。

```
from misskey import MiAuth

# MiAuth インスタンスを作成
mia = MiAuth("mk.example.com", name="Misskey.py App")

# permission を指定する場合
mia_p = MiAuth(
    "mk.example.com",
    name="Misskey.py App",
    permission=[
        "read:account",
        "write:notes",
    ]
)
```

インスタンス化したら、メソッド generate_url で URL を作成します。

作成した URL をエンドユーザー (認証してもらうユーザー) のブラウザで開きます。

```
from misskey import MiAuth
import webbrowser

mia = MiAuth("mk.example.com", name="Misskey.py App")

url = mia.generate_url()

webbrowser.open(url) # 例としてブラウザを開きます
```

4.2 トークンを取得する

エンドユーザーの認証が完了したら、メソッド check でトークンの取得を行います。

```
from misskey import MiAuth
import webbrowser

mia = MiAuth("mk.example.com", name="Misskey.py App")

token = mia.check() # 認証が行えたかをチェックします
```

正しく認証された場合は、トークンが返されます。問題があった場合は例外 MisskeyMiAuthFailedException が送出されます。

exception misskey.exceptions.**MisskeyMiAuthFailedException**

また、認証されたトークンはプロパティ token にも保存されます。

取得したトークンは、そのまま Misskey クラスなどで使用します。

```
from misskey import MiAuth, Misskey

mia = MiAuth("mk.example.com", name="Misskey.py App")
token = mia.check()

mk = Misskey("mk.example.com", i=token)
```

4.3 セッション ID を他で保存する場合

スクリプト上、MiAuth インスタンスを保持できない場合は、プロパティ session_id を使用してセッション ID を保存することもできます。

```
from misskey import MiAuth

mia = MiAuth("mk.example.com", name="Misskey.py App")

session_id = str(mia.session_id) # セッション ID を取得
```

MiAuth インスタンス化時に引数 session_id を代入することで同じセッション ID で再開できます。

```
from misskey import MiAuth

session_id = "00000000-0000-0000-0000-000000000000"

mia_n = MiAuth("mk.example.com", session_id=session_id)
```

第 5 章

Misskey class

Misskey の API プレフィックスは /api/ です。

Endpoint	Method
Meta	
announcements	*Misskey.announcements()*
meta	*Misskey.meta()*
stats	*Misskey.stats()*
Account(i)	
i	*Misskey.i()*
i/favorites	*Misskey.i_favorites()*
i/pin	*Misskey.i_pin()*
i/unpin	*Misskey.i_unpin()*
i/notifications	*Misskey.i_notifications()*
i/update	*Misskey.i_update()*
Note	
notes/create	*Misskey.notes_create()*
notes/delete	*Misskey.notes_delete()*
notes/show	*Misskey.notes_show()*
notes/conversation	*Misskey.notes_conversation()*
notes/children	*Misskey.notes_children()*
notes/replies	*Misskey.notes_replies()*
notes/renotes	*Misskey.notes_renotes()*
notes/unrenote	*Misskey.notes_unrenote()*
notes/reactions	*Misskey.notes_reactions()*
notes/reactions/create	*Misskey.notes_reactions_create()*
notes/reactions/delete	*Misskey.notes_reactions_delete()*

次のページに続く

表 1 – 前のページからの続き

Endpoint	Method
notes/polls/vote	*Misskey.notes_polls_vote()*
notes/state	*Misskey.notes_state()*
notes/favorites/create	*Misskey.notes_favorites_create()*
notes/favorites/delete	*Misskey.notes_favorites_delete()*
notes/watching/create	*Misskey.notes_watching_create()*
notes/watching/delete	*Misskey.notes_watching_delete()*
notes/timeline	*Misskey.notes_timeline()*
notes/local-timeline	*Misskey.notes_local_timeline()*
notes/hybrid-timeline	*Misskey.notes_hybrid_timeline()*
notes/global-timeline	*Misskey.notes_global_timeline()*
Users	
users/show	*Misskey.users_show()*
users/following	*Misskey.users_following()*
users/followers	*Misskey.users_followers()*
users/notes	*Misskey.users_notes()*
users/stats	*Misskey.users_stats()*
users/relation	*Misskey.users_relation()*
users/lists/create	*Misskey.users_lists_create()*
users/lists/show	*Misskey.users_lists_show()*
users/lists/push	*Misskey.users_lists_push()*
users/lists/pull	*Misskey.users_lists_pull()*
users/lists/update	*Misskey.users_lists_update()*
users/lists/delete	*Misskey.users_lists_delete()*
users/report-abuse	*Misskey.users_report_abuse()*
Relationships	
following/create	*Misskey.following_create()*
following/delete	*Misskey.following_delete()*
following/requests/accept	*Misskey.following_requests_accept()*
following/requests/reject	*Misskey.following_requests_reject()*
following/requests/cancel	*Misskey.following_requests_cancel()*
following/requests/list	*Misskey.following_requests_list()*
mute/create	*Misskey.mute_create()*
mute/list	*Misskey.mute_list()*
mute/delete	*Misskey.mute_delete()*
blocking/create	*Misskey.blocking_create()*
blocking/list	*Misskey.blocking_list()*
blocking/delete	*Misskey.blocking_delete()*

次のページに続く

表 1 – 前のページからの続き

Endpoint	Method
Drive	
drive	*Misskey.drive()*
drive/stream	*Misskey.drive_stream()*
drive/files	*Misskey.drive_files()*
drive/files/create	*Misskey.drive_files_create()*
drive/files/delete	*Misskey.drive_files_delete()*
drive/files/show	*Misskey.drive_files_show()*
drive/files/update	*Misskey.drive_files_update()*
drive/files/check-existence	*Misskey.drive_files_check_existence()*
drive/files/attached-notes	*Misskey.drive_files_attached_notes()*
drive/files/find-by-hash	*Misskey.drive_files_find_by_hash()*
drive/folders	*Misskey.drive_folders()*
drive/folders/create	*Misskey.drive_folders_create()*
drive/folders/show	*Misskey.drive_folders_show()*
drive/folders/update	*Misskey.drive_folders_update()*
drive/folders/delete	*Misskey.drive_folders_delete()*
Notifications	
notifications/mark-all-as-read	*Misskey.notifications_mark_all_as_read()*

class misskey.**Misskey**(*address: str = 'https://misskey.io'*, *i: Optional[str] = None*, *session: Optional[requests.sessions.Session] = None*)

Misskey API client class.

Args: address (str): Instance address. You can also include the URL protocol. If not specified, it will be automatically recognized as `https`.

i (`str`, optional): Misskey API token. If you have an API token, you can assign it at instantiation.

session (`requests.Session`, optional): If you have prepared the `requests.Session` class yourself, you can assign it here. Normally you do not need to specify it.

Raises: MisskeyAuthorizeFailedException: Raises if token validation fails during instantiation.

property address

Misskey instance address. Cannot be edited.

announcements(*limit: int = 10*, *with_unreads: bool = True*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*) → List[dict]

Get announcements.

Endpoint: `announcements`

Returns: *list* of *dict*: List of announcements.

Raises: MisskeyAPIException: Raise if the API request fails.

blocking_create(*user_id: str*) → dict

Block the specified user.

Args: user_id (str): Specify the user ID.

Endpoint: `blocking/create`

Returns: dict: Returns the blocking information.

Raises: MisskeyAPIException: Raise if the API request fails.

blocking_delete(*user_id: str*) → dict

Unblock the specified user.

Args: user_id (str): Specify the user ID.

Endpoint: `blocking/delete`

Returns: dict: Returns the blocking information.

Raises: MisskeyAPIException: Raise if the API request fails.

blocking_list(*limit: int = 30*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*) → List[dict]

Get list of blocked users.

Args: limit (int): Specify the number of users to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

Endpoint: `blocking/list`

Returns: `list` of `dict`: Returns the list of blocked users.

drive() → dict

Get drive usage information.

Endpoint: `drive`

Returns: dict: Returns the drive usage information.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *folder_id: Optional[str] = None*, *file_type: Optional[str] = None*) → List[dict]

Get drive files in specified folder(optional).

Args: limit (int): Specify the number of files to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

folder_id (str, optional): Specify the folder ID.

file_type (str, optional): Specify the file type.

Endpoint: `drive/files`

Returns: `list` of `dict`: Returns the list of files.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files_attached_notes(*file_id: str*) → List[dict]

Get notes that have the specified file.

Args: file_id (str): Specify the file ID.

Endpoint: `drive/files/attached-notes`

Returns: `list` of `dict`: Returns the list of notes.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files_check_existence(*md5: str*) → bool

Validate if the specified md5 hash exists in the drive.

Args: md5 (str): Specify the md5 hash.

Endpoint: `drive/files/check-existence`

Returns: bool: Returns `True` if the file exists.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files_create(*file: IO*, *folder_id: Optional[str] = None*, *name: Optional[str] = None*, *is_sensitive: bool = False*, *force: bool = False*) → dict

Upload a file to the drive.

Args: file (`IO`): Assign a file stream. As an example, the one opened by the `open` function is included.

folder_id (str, optional): Specify the folder ID.

name (str, optional): Specify the file name.

is_sensitive (bool, optional): Specify whether the file is sensitive.

force (bool, optional): Specify whether to overwrite the file if it already exists.

Endpoint: `drive/files/create`

Returns: dict: Returns the file information.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files_delete(*file_id: str*) → bool

Delete a file.

Args: file_id (str): Specify the file ID.

Endpoint: `drive/files/delete`

Returns: bool: Returns `True` if the file is deleted.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files_find_by_hash(*md5: str*) → List[dict]

Get files that have the specified md5 hash.

Args: md5 (str): Specify the md5 hash.

Endpoint: `drive/files/find-by-hash`

Returns: `list` of `dict`: Returns the list of files.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files_show(*file_id: Optional[str] = None*, *url: Optional[str] = None*) → dict

Get file information.

Args: file_id (str, optional): Specify the file ID.

url (str, optional): Specify the file URL.

Endpoint: `drive/files/show`

Note: You need to specify either `file_id` or `url`.

Returns: dict: Returns the file information.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_files_update(*file_id: str*, *folder_id: Optional[str] = ''*, *name: Optional[str] = None*, *is_sensitive: Optional[bool] = None*, *comment: Optional[str] = ''*) → dict

Update file information.

Args: file_id (str): Specify the file ID.

folder_id (str, optional): Specify the folder ID.

name (str, optional): Specify the file name.

is_sensitive (bool, optional): Specify whether the file is sensitive.

comment (str, optional): Specify a comment.

Endpoint: `drive/files/update`

Returns: dict: Returns the file information.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_folders(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *folder_id: Optional[str] = None*) → List[dict]

Get the folder list.

Args: limit (int, optional): Specify the number of folders to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

folder_id (str, optional): Specify the folder ID.

Endpoint: `drive/folders`

Returns: `list` of `dict`: Returns the list of folders.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_folders_create(*name: str = 'Untitled'*, *parent_id: Optional[str] = None*) → dict

Create a folder.

Args: name (str, optional): Specify the folder name.

parent_id (str, optional): Specify the parent folder ID.

Endpoint: `drive/folders/create`

Returns: dict: Returns the folder information.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_folders_delete(*folder_id: str*) → bool

Delete a folder.

Args: folder_id (str): Specify the folder ID.

Endpoint: `drive/folders/delete`

Returns: bool: Returns `True` if the folder is deleted.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_folders_show(*folder_id: str*) → dict

Get folder information.

Args: folder_id (str): Specify the folder ID.

Endpoint: `drive/folders/show`

Returns: dict: Returns the folder information.

Raises: MisskeyAPIException: Raise if the API request fails.

drive_folders_update(*folder_id: str*, *name: Optional[str] = None*, *parent_id: Optional[str] = ''*) → dict

Update folder information.

Args: folder_id (str): Specify the folder ID.

name (str, optional): Specify the folder name.

parent_id (str, optional): Specify the parent folder ID.

Endpoint: `drive/folders/update`

Returns: dict: Returns the folder information.

drive_stream(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *file_type: Optional[str] = None*) → List[dict]

Get drive files.

Args: limit (int): Specify the number of files to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

file_type (str, optional): Specify the file type.

Endpoint: `drive/stream`

Returns: `list` of `dict`: Returns the list of files.

Raises: MisskeyAPIException: Raise if the API request fails.

following_create(*user_id: str*) → dict

Follow the specified user.

Args: user_id (str): Specify the user ID.

Endpoint: `following/create`

Returns: dict: Returns the following information.

Raises: MisskeyAPIException: Raise if the API request fails.

following_delete(*user_id: str*) → dict

Unfollow the specified user.

Args: user_id (str): Specify the user ID.

Endpoint: `following/delete`

Returns: dict: Returns the following information.

Raises: MisskeyAPIException: Raise if the API request fails.

following_requests_accept(*user_id: str*) → bool

Accept the following request.

Args: user_id (str): Specify the user ID.

Endpoint: `following/requests/accept`

Returns: bool: Returns `True` if the request is successful.

Raises: MisskeyAPIException: Raise if the API request fails.

following_requests_cancel(*user_id: str*) → dict

Cancel the following request.

Args: user_id (str): Specify the user ID.

Endpoint: `following/requests/cancel`

Returns: dict: Returns the following information.

Raises: MisskeyAPIException: Raise if the API request fails.

following_requests_list() → List[dict]

Get list of following requests.

Endpoint: `following/requests/list`

Returns: `list` of `dict`: Returns the list of following requests.

Raises: MisskeyAPIException: Raise if the API request fails.

following_requests_reject(*user_id: str*) → bool

Reject the following request.

Args: user_id (str): Specify the user ID.

Endpoint: `following/requests/reject`

Returns: bool: Returns `True` if the request is successful.

Raises: MisskeyAPIException: Raise if the API request fails.

i() → dict

Get your credentials.

Endpoint: `i`

Returns: dict: A dict containing your profile information will be returned.

Note: `token` must be set in the instance.

Raises: MisskeyAPIException: Raise if the API request fails.

i_favorites(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*) → List[dict]

Get your favorites.

Args: limit (int): Specify the amount to get. You can specify from 1 to 100.

since_id (`str`, optional): Specify the first ID to get.

until_id (`str`, optional): Specify the last ID to get.

Endpoint: `i/favorites`

Note: `token` must be set in the instance.

Returns: *list* of *dict*: List of notes.

Raises: MisskeyAPIException: Raise if the API request fails.

i_notifications(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *following: bool = False*, *mark_as_read: bool = True*, *include_types: Optional[Union[List[Union[*misskey.enum.NotificationsType*, str]], Tuple[*misskey.enum.NotificationsType*], Set[*misskey.enum.NotificationsType*]]] = None*, *exclude_types: Optional[Union[List[Union[*misskey.enum.NotificationsType*, str]], Tuple[*misskey.enum.NotificationsType*], Set[*misskey.enum.NotificationsType*]]] = None*) → List[dict]

Get your notifications.

Args: limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

since_id (`str`, optional): Specify the first ID to get.

until_id (`str`, optional): Specify the last ID to get.

following (bool): Only following.

mark_as_read (bool): Specify whether to mark it as read when it is acquired.

include_types (`list`, `tuple` or `set`): Specifies the type of notification to include.

exclude_types (`list`, `tuple` or `set`): Specifies the type of notification to exclude.

Endpoint: `i/notifications`

Note: `token` must be set in the instance.

Returns: *list* of *dict*: List of notifications.

Raises: MisskeyAPIException: Raise if the API request fails.

i_pin(*note_id: str*) → dict

Pin a note.

Args: note_id (str): Note id.

Endpoint: `i/pin`

Note: `token` must be set in the instance.

Raises: MisskeyAPIException: Raise if the API request fails.

i_unpin(*note_id: str*) → dict

Unpin a note.

Args: note_id (str): Note id.

Endpoint: `i/unpin`

Note: `token` must be set in the instance.

Raises: MisskeyAPIException: Raise if the API request fails.

i_update(*name: Optional[str] = None*, *description: Optional[str] = None*, *lang: Optional[Union[*misskey.enum.LangType*, str]] = None*, *location: Optional[str] = None*, *birthday: Optional[Union[datetime.date, datetime.datetime, str]] = None*, *avatar_id: Optional[str] = None*, *banner_id: Optional[str] = None*, *fields: Optional[List[dict]] = None*, *is_locked: Optional[bool] = None*, *is_explorable: Optional[bool] = None*, *hide_online_status: Optional[bool] = None*, *careful_bot: Optional[bool] = None*, *auto_accept_followed: Optional[bool] = None*, *no_crawle: Optional[bool] = None*, *is_bot: Optional[bool] = None*, *is_cat: Optional[bool] = None*, *inject_featured_note: Optional[bool] = None*, *receive_announcement_email: Optional[bool] = None*, *always_mark_nsfw: Optional[bool] = None*, *pinned_page_id: Optional[str] = None*, *muted_words: Optional[List[List[str]]] = None*, *muting_notification_types: Optional[Union[List[Union[*misskey.enum.NotificationsType*, str]], Tuple[*misskey.enum.NotificationsType*], Set[*misskey.enum.NotificationsType*]]] = None*, *email_notification_types: Optional[List[str]] = None*) → dict

Update your profiles.

Args: name (`str`, optional): Your name to display.

description (`str`, optional): Write an introductory text.

lang (`str`, optional): Specify your language.

location (`str`, optional): Specify your location.

birthday (`datetime.date`, `datetime.datetime` or `str`, optional): Specify your birthday date.

avatar_id (`str`, optional): Avatar's drive id.

banner_id (`str`, optional): Banner's drive id.

fields (`list` of `dict`, optional): Profile supplementary information.

is_locked (`bool`, optional): Whether to make follow-up approval system.

is_explorable (`bool`, optional): Whether to set as a discoverable user.

hide_online_status (`bool`, optional): Whether to hide online status.

careful_bot (`bool`, optional): Whether to approve follow-ups from bots.

auto_accept_followed (`bool`, optional): Whether to automatically follow from the users you are following

no_crawle (`bool`, optional): Specifies whether to prevent it from being tracked by search engines.

is_bot (`bool`, optional): Whether to operate as a bot.

is_cat (`bool`, optional): Specifies whether to use nyaise.

inject_featured_note (`bool`, optional):

receive_announcement_email (`bool`, optional):

always_mark_nsfw (`bool`, optional): Whether to give NSFW to the posted file by default.

pinned_page_id (`str`, optional): ID of the page to be fixed.

muted_words (`list` of `list` of `str`, optional): Word to mute.

muting_notification_types (`list`, optional): Notification type to hide.

email_notification_types (`list` of `str`, optional): Specify the notification type for email notification.

Endpoint: `i/update`

Note: `token` must be set in the instance.

Returns: bool: Returns `True` if successful.

Raises: MisskeyAPIException: Raise if the API request fails.

meta(*detail: bool = True*) → dict

Get instance meta.

Args: detail (bool): Add the details of the instance information.

Endpoint: `meta`

Returns: dict: A dict containing instance information.

Raises: MisskeyAPIException: Raise if the API request fails.

mute_create(*user_id: str*) → bool

Mute the specified user.

Args: user_id (str): Specify the user ID.

Endpoint: `mute/create`

Returns: bool: Returns `True` if the user is muted.

Raises: MisskeyAPIException: Raise if the API request fails.

mute_delete(*user_id: str*) → bool

Unmute the specified user.

Args: user_id (str): Specify the user ID.

Endpoint: `mute/delete`

Returns: bool: Returns `True` if the user is unmuted.

Raises: MisskeyAPIException: Raise if the API request fails.

mute_list(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*) → List[dict]

Get list of muted users.

Args: limit (int): Specify the number of users to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

Endpoint: `mute/list`

Returns: `list` of `dict`: Returns the list of muted users.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_children(*note_id: str*, *limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*) → List[dict]

Show note children.

Args: note_id (str): Specify the Note ID to get.

limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

Endpoint: `notes/children`

Returns: `list` of `dict`: Gets the Note associated with that Note.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_conversation(*note_id: str*, *limit: int = 10*, *offset: Optional[int] = None*) → List[dict]

Show note conversations.

Args: note_id (str): Specify the Note ID to get.

limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

offset (int, optional): Specify the offset to get.

Endpoint: `notes/conversation`

Returns: `list` of `dict`: Gets the Note associated with that Note.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_create(*text: Optional[str] = None*, *cw: Optional[str] = None*, *visibility: Union[*misskey.enum.NoteVisibility*, str] = NoteVisibility.PUBLIC*, *visible_user_ids: Optional[List[str]] = None*, *via_mobile: bool = False*, *local_only: bool = False*, *no_extract_mentions: bool = False*, *no_extract_hashtags: bool = False*, *no_extract_emojis: bool = False*, *file_ids: Optional[List[str]] = None*, *reply_id: Optional[str] = None*, *renote_id: Optional[str] = None*, *poll_choices: Optional[Union[List[str], Tuple[str]]] = None*, *poll_multiple: bool = False*, *poll_expires_at: Optional[Union[int, datetime.datetime]] = None*, *poll_expired_after: Optional[Union[int, datetime.timedelta]] = None*) → dict

Create a note.

Args: text (`str`, optional): Specify the text.

cw (`str`, optional): Specify the CW(Content Warning).

visibility (`str`, default: `public`): Post range. Specifies the enumeration in `NoteVisibility`.

visible_user_ids (`list` of `str`, optional): If `visibility` is `specified`, specify the user ID in the list.

via_mobile (`bool`, optional): Specify whether to post from mobile. It doesn't work with recent Misskey versions.

local_only (`bool`, optional): Specifies whether to post only the instance you are using.

no_extract_mentions (`bool`, optional): Specifies whether to detect mentions from the text.

no_extract_hashtags (`bool`, optional): Specifies whether to detect hashtags from the text.

no_extract_emojis (`bool`, optional): Specifies whether to detect emojis from the text.

file_ids (`list` of `str`, optional): Specify the file ID to attach in the list.

reply_id (`str`, optional): Specify the Note ID of the reply destination.

renote_id (`str`, optional): Specify the Note ID to renote.

poll_choices (`list` of `str`, optional): Specify the voting item. You can specify 2 or more and 10 or less.

poll_multiple (`bool`, optional): Specifies whether to allow multiple votes. This is valid only when `poll_choices` is specified.

poll_expires_at (`datetime.datetime`, optional): Specify the expiration date of the vote. If not specified, it will be indefinite. Cannot be used with `poll_expired_after`.

poll_expired_after (`datetime.timedelta`, optional): Specifies the validity period of the vote. If not specified, it will be indefinite. Cannot be used with `poll_expired_at`.

Endpoint: `notes/create`

Note: `token` must be set in the instance.

You must specify at least either `text` or `files_id`.

Returns: dict: The dict of the posted result is returned.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_delete(*note_id: str*) → bool

Delete a note.

Args: note_id (str): Specify the Note ID to delete.

Endpoint: `notes/delete`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_favorites_create(*note_id: str*) → bool

Mark as favorite a note.

Args: note_id (str): Specify the Note ID to make a favorite.

Endpoint: `notes/favorites/create`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_favorites_delete(*note_id: str*) → bool

Delete favorite a note.

Args: note_id (str): Specify the Note ID to unmark a favorite.

Endpoint: `notes/favorites/delete`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_global_timeline(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *since_date: Optional[Union[int, datetime.datetime]] = None*, *until_date: Optional[Union[int, datetime.datetime]] = None*, *with_files: bool = True*) → List[dict]

Show global timeline.

Args: limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

since_date (int, datetime.datetime, optional): Specify the first date to get.

until_date (int, datetime.datetime, optional): Specify the last date to get.

with_files (bool, optional): Specify whether to include files.

Endpoint: `notes/global-timeline`

Returns: `list` of `dict`: Returns a list of notes.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_hybrid_timeline(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *since_date: Optional[Union[int, datetime.datetime]] = None*, *until_date: Optional[Union[int, datetime.datetime]] = None*, *include_my_renotes: bool = True*, *include_renoted_my_notes: bool = True*, *include_local_renotes: bool = True*, *with_files: bool = True*) → List[dict]

Show hybrid(home + local) timeline.

Args: limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

since_date (int, datetime.datetime, optional): Specify the first date to get.

until_date (int, datetime.datetime, optional): Specify the last date to get.

include_my_renotes (bool, optional): Specify whether to include your notes.

include_renoted_my_notes (bool, optional): Specify whether to include renotes of your notes.

include_local_renotes (bool, optional): Specify whether to include local renotes.

with_files (bool, optional): Specify whether to include files.

Endpoint: `notes/hybrid-timeline`

Returns: `list` of `dict`: Returns a list of notes.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_local_timeline(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *since_date: Optional[Union[int, datetime.datetime]] = None*, *until_date: Optional[Union[int, datetime.datetime]] = None*, *with_files: bool = True*, *file_type: Optional[List[str]] = None*, *exclude_nsfw: bool = False*) → List[dict]

Show local timeline.

Args: limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

since_date (int, datetime.datetime, optional): Specify the first date to get.

until_date (int, datetime.datetime, optional): Specify the last date to get.

with_files (bool, optional): Specify whether to include files.

file_type (`list` of `str`, optional): Specify the file type to get.

exclude_nsfw (bool, optional): Specify whether to exclude NSFW (Not safe for work) notes.

Endpoint: `notes/local-timeline`

Returns: `list` of `dict`: Returns a list of notes.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_polls_vote(*note_id: str*, *choice: int*) → bool

Vote in a note poll.

Args: note_id (str): Specify the Note ID to vote.

choice (int): Specify the choice to vote. Specify from 0.

Endpoint: `notes/polls/vote`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_reactions(*note_id: str*, *reaction_type: Optional[str] = None*, *limit: int = 10*, *offset: Optional[int] = None*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*) → List[dict]

Show note reactions.

Args: note_id (str): Specify the Note ID to get.

reaction_type (str, optional): Specify the reaction type to get.

limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

offset (int, optional): Specify the offset to get.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

Endpoint: `notes/reactions`

Returns: `list` of `dict`: Get reactions in associated note.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_reactions_create(*note_id: str*, *reaction: str*) → bool

Create a reaction in a note.

Args: note_id (str): Specify the Note ID to create your reaction.

reaction (str): Specify the reaction type.

Endpoint: `notes/reactions/create`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_reactions_delete(*note_id: str*) → bool

Delete a reaction in a note.

Args: note_id (str): Specify the Note ID to delete your reaction.

Endpoint: `notes/reactions/delete`

Returns: bool: Returns `True` if the reaction was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_renotes(*note_id: str*, *limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*) → List[dict]

Show renotes.

Args: note_id (str): Specify the Note ID to get.

limit(int, optional): Specify the amount to get.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

Endpoint: `notes/renotes`

Returns: `list` of `dict`: Gets the Note associated with that Note.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_replies(*note_id: str*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *limit: int = 10*) → List[dict]

Show note replies.

Args: note_id (str): Specify the Note ID to get.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

Endpoint: `notes/replies`

Returns: `list` of `dict`: Gets the Note associated with that Note.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_show(*note_id: str*) → dict

Show a note.

Args: note_id (str): Specify the Note ID to get.

Endpoint: `notes/show`

Returns: dict: A dict with the specified Note ID is returned.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_state(*note_id: str*) → dict

Show a state of a note.

Args: note_id (str): Specify the Note ID to get.

Endpoint: `notes/state`

Returns: `dict`: Get state of a note.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_timeline(*limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *since_date: Optional[Union[int, datetime.datetime]] = None*, *until_date: Optional[Union[int, datetime.datetime]] = None*, *include_my_renotes: bool = True*, *include_renoted_my_notes: bool = True*, *include_local_renotes: bool = True*, *with_files: bool = True*) → List[dict]

Show your home timeline.

Args: limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

since_date (int, datetime.datetime, optional): Specify the first date to get.

until_date (int, datetime.datetime, optional): Specify the last date to get.

include_my_renotes (bool, optional): Specify whether to include your notes.

include_renoted_my_notes (bool, optional): Specify whether to include renotes of your notes.

include_local_renotes (bool, optional): Specify whether to include local renotes.

with_files (bool, optional): Specify whether to include files.

Endpoint: `notes/timeline`

Returns: `list` of `dict`: Returns a list of notes.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_unrenote(*note_id: str*) → bool

Unrenote a note.

Args: note_id (str): Specify the Note ID to unrenote.

Endpoint: `notes/unrenote`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_watching_create(*note_id: str*) → bool

Watch a note.

Args: note_id (str): Specify the Note ID to watch.

Endpoint: `notes/watching/create`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notes_watching_delete(*note_id: str*) → bool

Unwatch a note.

Args: note_id (str): Specify the Note ID to unwatch.

Endpoint: `notes/watching/delete`

Returns: bool: Returns `True` if the request was successful.

Raises: MisskeyAPIException: Raise if the API request fails.

notifications_mark_all_as_read() → bool

Mark all as read to your notifications.

Endpoint: `notifications/mark-all-as-read`

Note: `token` must be set in the instance.

Returns: bool: Returns `True` if successful.

Raises: MisskeyAPIException: Raise if the API request fails.

stats() → dict

Get instance statuses.

Endpoint: `stats`

Returns: dict: A dict containing the number of users, the number of notes, etc. is returned.

Raises: MisskeyAPIException: Raise if the API request fails.

timeout: Optional[Any] = 15.0

Specifies the number of seconds for HTTP communication timeout. Comply with "requests".

property token: Optional[str]

Get a token.

When you assign a new token, it automatically verifies whether the token can be used. If validation fails, the exception `MisskeyAuthorizeFailedException` is raised.

If using `del`, `token` will be `None`.

users_followers(*user_id: Optional[str] = None*, *username: Optional[str] = None*, *host: Optional[str] = None*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *limit: int = 10*) → List[dict]

Get the follower list of the specified user.

Args: user_id (str, optional): Specify the user ID.

username (str, optional): Specify the username.

host (str, optional): Specify the host.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

Endpoint: `users/following`

Note: You must specify one of user_id, username (and host).

Returns: `list` of `dict`: Returns a list of users.

Raises: MisskeyAPIException: Raise if the API request fails.

users_following(*user_id: Optional[str] = None*, *username: Optional[str] = None*, *host: Optional[str] = None*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *limit: int = 10*) → List[dict]

Get the follow list of the specified user.

Args: user_id (str, optional): Specify the user ID.

username (str, optional): Specify the username.

host (str, optional): Specify the host.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

Endpoint: `users/following`

Note: You must specify one of user_id, username (and host).

Returns: `list` of `dict`: Returns a list of users.

Raises: MisskeyAPIException: Raise if the API request fails.

users_lists_create(*name: str*) → dict

Create user list.

Args: name (str): Specify the list name.

Endpoint: `users/lists/create`

Returns: dict: Returns the new list information.

Raises: MisskeyAPIException: Raise if the API request fails.

users_lists_delete(*list_id: str*) → bool

Delete user list.

Args: list_id (str): Specify the list ID.

Endpoint: `users/lists/delete`

Returns: bool: Returns `True` if the list is deleted.

Raises: MisskeyAPIException: Raise if the API request fails.

users_lists_list() → List[dict]

Get user list.

Endpoint: `users/lists/list`

Returns: list of dict: Returns a list of user lists.

Raises: MisskeyAPIException: Raise if the API request fails.

users_lists_pull(*list_id: str*, *user_id: str*) → bool

Remove user from user list.

Args: list_id (str): Specify the list ID.

user_id (str): Specify the user ID.

Endpoint: `users/lists/pull`

Returns: bool: Returns `True` if the user is removed from the list.

Raises: MisskeyAPIException: Raise if the API request fails.

users_lists_push(*list_id: str*, *user_id: str*) → bool

Add user to user list.

Args: list_id (str): Specify the list ID.

user_id (str): Specify the user ID.

Endpoint: `users/lists/push`

Returns: bool: Returns `True` if the user is added to the list.

Raises: MisskeyAPIException: Raise if the API request fails.

users_lists_show(*list_id: str*) → dict

Get user list detail.

Args: list_id (str): Specify the list ID.

Endpoint: `users/lists/show`

Returns: dict: Returns the list information.

Raises: MisskeyAPIException: Raise if the API request fails.

users_lists_update(*list_id: str*, *name: str*) → dict

Update user list.

Args: list_id (str): Specify the list ID.

name (str): Specify the new list name.

Endpoint: `users/lists/update`

Returns: dict: Returns the updated list information.

Raises: MisskeyAPIException: Raise if the API request fails.

users_notes(*user_id: str*, *include_replies: bool = True*, *limit: int = 10*, *since_id: Optional[str] = None*, *until_id: Optional[str] = None*, *since_date: Optional[Union[int, datetime.datetime]] = None*, *until_date: Optional[Union[int, datetime.datetime]] = None*, *include_my_renotes: bool = True*, *with_files: bool = False*, *file_type: Optional[List[str]] = None*, *exclude_nsfw: bool = False*) → List[dict]

Get the note list of the specified user.

Args: user_id (str): Specify the user ID.

include_replies (bool, optional): Specify whether to include replies.

limit (int, optional): Specify the amount to get. You can specify from 1 to 100.

since_id (str, optional): Specify the first ID to get.

until_id (str, optional): Specify the last ID to get.

since_date (`datetime.datetime`, optional): Specify the first date to get.

until_date (`datetime.datetime`, optional): Specify the last date to get.

include_my_renotes (bool, optional): Specify whether to include my renotes.

with_files (bool, optional): Specify whether to include files.

file_type (`list` of `str`, optional): Specify the file type to get.

exclude_nsfw (bool, optional): Specify whether to exclude NSFW notes.

Endpoint: `users/notes`

Returns: `list` of `dict`: Returns a list of notes.

Raises: MisskeyAPIException: Raise if the API request fails.

users_relation(*user_id: Union[str, List[str]]*) → Union[dict, List[dict]]

Get the relation of the specified user(s).

Args: user_id (`str` or `list` of `str`): Specify the user ID(s).

Endpoint: `users/relation`

Note: If `user_id` is specified by str, it will be returned by dict.

If `user_id` is specified by list, it will be returned by dict of list.

Returns: `dict` or `list` of `dict`: Returns the relation.

Raises: MisskeyAPIException: Raise if the API request fails.

users_report_abuse(*user_id: str*, *comment: str*) → bool

Report abuse to user.

Args: user_id (str): Specify the user ID.

comment (str): Specify the comment.

Endpoint: `users/report/abuse`

Returns: bool: Returns `True` if the report is sent.

Raises: MisskeyAPIException: Raise if the API request fails.

users_show(*user_id: Optional[str] = None*, *user_ids: Optional[List[str]] = None*, *username: Optional[str] = None*, *host: Optional[str] = None*) → Union[dict, List[dict]]

Show user.

Args: user_id (str, optional): Specify the user ID.

user_ids (list of str, optional): Specify the user IDs.

username (str, optional): Specify the username.

host (str, optional): Specify the host.

Endpoint: `users/show`

Note: You must specify one of user_id, user_ids, username (and host).

If you specify `user_ids`, it returns a list of users.

Returns: `dict` or `list` of `dict`: Returns a user or a list of users.

Raises: MisskeyAPIException: Raise if the API request fails.

users_stats(*user_id: str*) → dict

Gets the count for the specified user

Args: user_id (str): Specify the user ID.

Endpoint: `users/stats`

Returns: `dict`: Returns a count for the specified user.

Raises: MisskeyAPIException: Raise if the API request fails.

第 6 章

MiAuth class

class misskey.**MiAuth**(*address: str = 'https://misskey.io'*, *session_id: Optional[Union[uuid.UUID, str]] = None*, *name: str = 'Misskey.py'*, *icon: Optional[str] = None*, *callback: Optional[str] = None*, *permission: Optional[Union[List[Optional[Union[*misskey.enum.Permissions*, str]]], Tuple[Optional[*misskey.enum.Permissions*]], Set[Optional[*misskey.enum.Permissions*]]]] = None*, *session: Optional[requests.sessions.Session] = None*)

Misskey Authentication Class

Args: address (str): Instance address. You can also include the URL protocol. If not specified, it will be automatically recognized as `https`.

session_id (`uuid.UUID` or `str`, optional): Session ID for performing MiAuth. If not specified, it will be set automatically.

name (`str`, optional): App name.

icon (`str`, optional): The URL of the icon.

callback (`str`, optional): App callback URL.

permission (`list` of `str`, optional): The permissions that the app can use. You can specify an enumeration of Permissions.

session (`requests.Session`, optional): If you have prepared the `requests.Session` class yourself, you can assign it here. Normally you do not need to specify it.

property address: str

Misskey instance address for MiAuth. Cannot be edited.

property callback: Optional[str]

App callback URL.

It can be changed by assignment.

If using `del`, `icon` will be `None`.

check() → str

Validate MiAuth authentication.

Returns: str: If the authentication is successful, a MiAuth token will be returned.

Note: The token obtained by this method is also assigned to the property `token`.

Raises: MisskeyMiAuthFailedException: Raise if MiAuth authentication fails.

generate_url() → str

Create a URL for your end user.

Returns: str: The URL for the end user is returned. Instruct the end user to open it in a web browser or the like.

property icon: Optional[str]

The URL of the icon.

It can be changed by assignment.

If using `del`, `icon` will be `None`.

property name: str

App name.

It can be changed by assignment.

property permission: Optional[Union[List[Optional[Union[*misskey.enum.Permissions*, str]]], Tuple[Optional[*misskey.enum.Permissions*]], Set[*misskey.enum.Permissions*]]]

The permissions that the app can use.

property session_id: Union[uuid.UUID, str]

Session ID for performing MiAuth. Cannot be edited.

property token: Optional[str]

Contains the token authenticated by MiAuth. If not authenticated, `None` will be returned.

第 7 章

Enum classes

class misskey.enum.**LangType**(*value*)

ベースクラス: enum.Enum

Language type enumuration.

class misskey.enum.**NoteVisibility**(*value*)

ベースクラス: enum.Enum

Note visibility enumeration.

class misskey.enum.**NotificationsType**(*value*)

ベースクラス: enum.Enum

Notifications type enumeration.

class misskey.enum.**Permissions**(*value*)

ベースクラス: enum.Enum

Permissions enumuration.

第 8 章

Exceptions

exception misskey.exceptions.**MisskeyAPIException**(*response_dict: dict*)

exception misskey.exceptions.**MisskeyAuthorizeFailedException**

exception misskey.exceptions.**MisskeyMiAuthFailedException**

第 9 章

索引

- genindex

Python モジュール索引

m

索引