## CONTENTS

1 Application Programming Interface .......................................................... 3
  1.1 OAuth2 ........................................................................................................ 3
  1.2 Requests .................................................................................................... 4
  1.3 Responses ................................................................................................ 4
  1.4 Endpoints .................................................................................................. 5
  1.5 Objects ..................................................................................................... 11
Contents:
1.1 OAuth2

Pixelmerge uses OAuth2 to authorize API Clients for specific Endpoints. We implement the latest OAuth2 Draft (at the moment 31).

1.1.1 Authorization Code Grant

Pixelmerge supports the Authorization Code Grant as preferred grant:

1. Redirect the user to https://pixelmerge.me/oauth2/connect with the following parameters as query components encoded as application/x-www-urlencoded:
   - **response_type**: Expected response, must be “code” for Authorization Code Grant
   - **client_id**: Your applications Client ID
   - **redirect_url**: A URL to redirect after permission grant
   - **scope**: List of requested permissions. Must be a space-separated list of permission tokens obtained from /api/info. If omitted, the request will implicitly contain the standard scopes of your Client.
   - **state**: An optional state to determine the request in your callback. We (and the draft) recommend, that you are using a state

2. The user will see a dialog, which asks him to permit or deny the request. If the user permits the authorization grant, then the user will be redirected to redirect_url with the following additional parameters in the Query component, which is encoded as application/x-www-urlencoded:
   - **code**: The authorization code to obtain an access token
   - **state**: If you supplied a state in step 1, then you will get it here

3. Call https://pixelmerge.me/oauth2/access_token on server side with the following parameters as query component encoded as application/x-www-urlencoded:
   - **grant_type**: The OAuth2 grant type, in this case “authorization_code”
   - **code**: The obtained authorization code from step 2
   - **redirect_url**: The redirect_url parameter from step 1

---

client_id  Your Client ID
client_secret  Your Client secret

4. Step 3 will respond with an body encoded as application/json.

### 1.2 Requests

The Pixelmerge API receives requests only by HTTPS. A request is sent to an endpoint, which replies with data to the request. A request takes a method, a path, an optional query with parameters, and an optional application/json-encoded body. Authorization is performed by OAuth 2, access tokens must be supplied as a query parameter called “access_token”.

At the moment, the requests are not rate-limited, so you can request until we block your IP address or access token manually. This could change in near time, so you should always accept errors.

#### 1.2.1 Generic Collection Parameters

Some API endpoints, which return a collection, accept generic collection parameters, to subset the responses. In general, any collection, which returns objects with a time, is supported.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>number</td>
<td>Maximum amount of the objects returned</td>
</tr>
<tr>
<td>since_time</td>
<td>number</td>
<td>Instruct, that the API should not return objects with a time, which is before the time specified in this parameter as UTC Unix timestamp.</td>
</tr>
</tbody>
</table>

All of these parameters must be supplied as query parameters.

### 1.3 Responses

An API endpoint returns responses, which consist of a HTTP status code and an optional application/json-encoded body.

#### 1.3.1 Errors

If an error while performing the request occurs, the API will return a HTTP status code equal or greater than 400. Then, the API will return also a JSON object in the body with the parameter error, which describes the error. This is compatible to OAuth2, especially because OAuth2 errors are handled by the same functionality. In addition to the error field, the object can contain other error-specific fields, which describe the errors more specific.

```
{
    "error": {string},
    ...
}
```

See Errors for a full list of API errors.

---

1.4 Permissions

Pixelmerge maintains several permissions for its API endpoints. These permissions are also OAuth2 scopes, that means, you have to obtain them on Authorization Grant.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_private_scope</td>
<td>Access to objects which hold a scope which is private</td>
</tr>
<tr>
<td>create_post</td>
<td>Create new objects which are related to posts</td>
</tr>
<tr>
<td>get_services</td>
<td>Read service objects</td>
</tr>
<tr>
<td>add_service</td>
<td>Create service objects</td>
</tr>
<tr>
<td>delete_service</td>
<td>Delete service objects</td>
</tr>
<tr>
<td>get_views</td>
<td>Read view objects</td>
</tr>
<tr>
<td>create_view</td>
<td>Create view objects</td>
</tr>
<tr>
<td>delete_view</td>
<td>Delete view objects</td>
</tr>
<tr>
<td>get_settings</td>
<td>Read settings</td>
</tr>
<tr>
<td>set_settings</td>
<td>Set settings</td>
</tr>
</tbody>
</table>

1.5 Endpoints

Pixelmerge provides a RESTful API interface.

1.5.1 User

GET /users/<username>

Method GET
Path /users/<username>

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>string</td>
<td>Name of the user</td>
</tr>
</tbody>
</table>

Authorization None
Permissions None

Returns User object

1.5.2 Post

GET /posts/private

This endpoint returns all private posts

Method GET
Path /posts/private

Parameters This endpoint accepts all generic collection parameters

Authorization OAuth2
Permissions get_private_scope

Returns Array of post objects
GET /posts/public

Method GET
Path /posts/public
Parameters This endpoint accepts all generic collection parameters
Authorization OAuth2
Permissions None
Returns Array of post objects

GET /posts/public/<username>

Method GET
Path /posts/public/<username>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>string</td>
<td>Name of a registered user</td>
</tr>
</tbody>
</table>

In addition to this, this endpoint accepts all generic collection parameters
Authorization None
Permissions None
Returns Array of post objects

GET /posts/id/<id>

Method GET
Path /posts/id/<id>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>ID of a post object</td>
</tr>
</tbody>
</table>

Authorization None if the scope of the object is “public”, otherwise OAuth2
Permissions If the scope of the object is “private”:
- get_private_scope

Returns Post object

POST /posts

Method POST
Path /posts
Parameters This endpoint accepts a JSON encoded object with the following parameters in the request body:

```json
{
    "text": {string},
    "groups": [
        {string}
    ]
}
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>string</td>
<td>Text for the post</td>
</tr>
<tr>
<td>groups</td>
<td>array</td>
<td>List of service groups IDs, on which members the text will be posted.</td>
</tr>
</tbody>
</table>

Authorization  

**OAuth2**

Permissions

- create_post

Returns

- 202 if the post is acceptable
- 400 if the request is invalid
- 401 if the user hasn’t the permissions

### 1.5.3 Medium

**GET /media/private**

Method  

**GET**

Path  

/media/private

Parameters  

This endpoint accepts all generic collection parameters

Authorization  

**OAuth2**

Permissions  

get_private_scope

Returns  

Array of medium objects

**GET /media/public**

Method  

**GET**

Path  

/media/public

Parameters  

This endpoints accepts all generic collection parameters

Authorization  

**OAuth2**

Permissions  

None

Returns  

Array of medium objects

**GET /media/public/<username>**

Method  

**GET**

Path  

/media/public/<username>

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>string</td>
<td>Name of a registered user</td>
</tr>
</tbody>
</table>

In addition to this, this endpoint accepts all generic collection parameters

Authorization  

None

Permissions  

None
Returns *Array of medium objects*

**GET /media/id/<id>**

**Method** GET  
**Path** /media/id/<id>  

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td></td>
<td>ID of a medium object</td>
</tr>
</tbody>
</table>

**Authorization** None if the scope of the object is “public”, otherwise OAuth2  

**Permissions** If the scope of the object is not “private”:
- get_private_scope  

**Returns** *Medium object*

### 1.5.4 Location

**GET /locations/private**

**Method** GET  
**Path** /locations/private  

**Parameters** This endpoint accepts all generic collection parameters  

**Authorization** OAuth2  

**Permissions** get_private_scope  

**Returns** *Array of location objects*

**GET /locations/public**

**Method** GET  
**Path** /locations/public  

**Parameters** This endpoint accepts all generic collection parameters  

**Authorization** OAuth2  

**Permissions** None  

**Returns** *Array of location objects*

**GET /locations/public/<username>**

**Method** GET  
**Path** /locations/public/<username>  

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>string</td>
<td></td>
<td>Name of a registered user</td>
</tr>
<tr>
<td>count</td>
<td>number</td>
<td></td>
<td>Number of records to receive as query parameter</td>
</tr>
<tr>
<td>since_time</td>
<td>number</td>
<td></td>
<td>Time of the earliest object to return (query)</td>
</tr>
</tbody>
</table>

In addition to this, this endpoint accepts all generic collection parameters.
Authorization  None
Permissions  None
Returns  Array of location objects

**GET /locations/id/<id>**

**Method**  GET

**Path**  /locations/id/<id>

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>ID of a location object</td>
</tr>
</tbody>
</table>

Authorization  None if the scope of the object is “public”, otherwise OAuth2
Permissions  If the scope of the object is “private”:
  • get_private_scope

Returns  Location object

### 1.5.5 Permissions

**GET /permissions**

**Method**  GET

**Path**  /permissions

**Parameters**  None

Authorization  OAuth2
Permissions  None
Returns  Array of Permissions

[ ]

**DELETE /permissions/<permission>**

**Method**  DELETE

**Path**  /permissions/<permission>

**Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>permission</td>
<td>string</td>
<td>Permission, which should removed from the current access token</td>
</tr>
</tbody>
</table>

Authorization  OAuth2
Permissions  None
Returns  Nothing
1.5.6 Service

GET /services

Method GET
Path /services
Parameters None
Authorization OAuth2
Permissions get_services
Returns Array of service objects

GET /services/<id>

Method GET
Path /services/<id>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>ID of a service object</td>
</tr>
</tbody>
</table>

Authorization OAuth2
Permissions get_services
Returns Array of service objects

DELETE /services/<id>

Method DELETE
Path /services/<id>
Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>ID of a service object</td>
</tr>
</tbody>
</table>

Authorization OAuth2
Permissions delete_service
Returns Nothing
POST /services

1.5.7 Service groups

GET /services/groups
POST /services/groups
GET /services/groups/<id>
DELETE /services/groups/<id>
POST /services/groups/<id>/<service_id>
DELETE /services/groups/<id>/<service_id>

1.5.8 Settings

GET /settings
POST /settings

1.5.9 View

GET /views
POST /views
GET /views/<id>
POST /views/<id>
DELETE /views/<id>

1.5.10 Info

GET /info

1.6 Objects

The Pixelmerge API responds with API objects, which are specified in this section:

1.6.1 Post Objects

Post objects represent raw posts, which are not merged and imported direct from services. They contain medium and location objects.
Chapter 1. Application Programming Interface
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>An unique ID which specifies a post object</td>
</tr>
<tr>
<td>external_url</td>
<td>string</td>
<td>A URL which references the specific object on service side.</td>
</tr>
<tr>
<td>url</td>
<td>string</td>
<td>A URL which references the specific object on Pixelmerges side</td>
</tr>
<tr>
<td>scope</td>
<td>string</td>
<td>Scope of the post object</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Name which references the user object of the service owner</td>
</tr>
<tr>
<td>text</td>
<td>string</td>
<td>The text part of the post as unformatted plain text</td>
</tr>
<tr>
<td>location</td>
<td>object</td>
<td>Specifies the location of the object; null, if the post hasn’t any location data.</td>
</tr>
<tr>
<td>location[longitude]</td>
<td>number</td>
<td>Longitude of the location</td>
</tr>
<tr>
<td>location[latitude]</td>
<td>number</td>
<td>Latitude of the location</td>
</tr>
<tr>
<td>media</td>
<td>array</td>
<td>List of media supplied by the post</td>
</tr>
<tr>
<td>media[]</td>
<td>object</td>
<td>Object which holds the media data</td>
</tr>
<tr>
<td>media[]тип</td>
<td>string</td>
<td>Type of the media, at the moment the following types are supported:</td>
</tr>
<tr>
<td>media[]url</td>
<td>string</td>
<td>URL which references the source of the medium</td>
</tr>
<tr>
<td>time</td>
<td>number</td>
<td>Unix Timestamp (UTC) which specifies the creation datetime of the post</td>
</tr>
<tr>
<td>creator</td>
<td>object</td>
<td>Object which holds information of the creator on service side</td>
</tr>
<tr>
<td>creator[username]</td>
<td>string</td>
<td>Username of the creator of the post on service side</td>
</tr>
<tr>
<td>creator[display_name]</td>
<td>string</td>
<td>Display name of the creator of the post on service side</td>
</tr>
<tr>
<td>creator[url]</td>
<td>string</td>
<td>Link to the resource of the creator on service side</td>
</tr>
<tr>
<td>creator[avatar]</td>
<td>string</td>
<td>Link to the user image on service side (Avatar)</td>
</tr>
</tbody>
</table>

### 1.6.2 User Objects

User objects specify Pixelmerge users.

```json
{
  "username": {string},
  "created_at": {number},
  "views": [
   {string}
  ]
}
```
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>string</td>
<td>Name of the user</td>
</tr>
<tr>
<td>created_at</td>
<td>number</td>
<td>Time of the creation of the user</td>
</tr>
<tr>
<td>views</td>
<td>array</td>
<td>List of public user views</td>
</tr>
<tr>
<td>views[]</td>
<td>string</td>
<td>Name of a user view</td>
</tr>
</tbody>
</table>

### 1.6.3 Location Objects

Location objects represent locations from posts, but extracted to query them directly.

```json
{
    "id": {string},
    "username": {string},
    "service": {string},
    "time": {number},
    "position": {
        "longitude": {number},
        "latitude": {number}
    },
    "scope": {string},
    "post": {string}
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>Unique identifier of the location</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Name of the location object owner</td>
</tr>
<tr>
<td>service</td>
<td>string</td>
<td>ID which references the service object associated with the location object</td>
</tr>
<tr>
<td>time</td>
<td>number</td>
<td>Time of the creation of the post related to this location object</td>
</tr>
<tr>
<td>position</td>
<td>object</td>
<td>Object which holds the location information</td>
</tr>
<tr>
<td>position[longitude]</td>
<td>number</td>
<td>Longitude of the position</td>
</tr>
<tr>
<td>position[latitude]</td>
<td>number</td>
<td>Latitude of the position</td>
</tr>
<tr>
<td>scope</td>
<td>string</td>
<td>Scope of the location object</td>
</tr>
<tr>
<td>post</td>
<td>string</td>
<td>ID which references the post object, from which the location data was obtained.</td>
</tr>
</tbody>
</table>

### 1.6.4 Medium Objects

Medium objects represent any type of mediums, e.g. images, videos, documents.

```json
{
    "id": {string},
    "username": {string},
    "service": {string},
    "time": {number},
    "url": {string},
    "scope": {string},
    "post": {string},
    "type": {string}
}
```
### Field | Type | Description
--- | --- | ---
id | string | An unique ID which specifies a media object
username | string | Name which references a user which owns the post
service | string | An ID which references the service object
time | number | Time of the creation of the referenced post object
url | string | URL which references the source of the medium
scope | string | Scope of the medium
post | string | ID which references the post object
type | string | Type of the medium

#### 1.6.5 Service Objects

Service objects represent external services from which Pixelmerge imports posts.

```
{
    "id": {string},
    "username": {string},
    "service": {string},
    "state": {string},
    "realname": {string}
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>An unique ID which specifies a service object</td>
</tr>
<tr>
<td>username</td>
<td>string</td>
<td>Name which references the user object of the owner of the service</td>
</tr>
<tr>
<td>service</td>
<td>string</td>
<td>Name of the external service (e.g. twitter)</td>
</tr>
<tr>
<td>state</td>
<td>string</td>
<td>State of the service. Valid states are “pending”, “inactive”, “active”</td>
</tr>
<tr>
<td>realname</td>
<td>string</td>
<td>Display name of the service account</td>
</tr>
</tbody>
</table>

#### 1.6.6 View Objects

View objects represent the meta data used to process posts.

```
{
    "id": {string},
    "settings": {object},
    "order": {number},
    "type": {string},
    "scope": {string}
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>string</td>
<td>An unique ID which specifies a view object</td>
</tr>
<tr>
<td>settings</td>
<td>object</td>
<td>Type-specific settings for the view</td>
</tr>
<tr>
<td>order</td>
<td>number</td>
<td>Number by which the service is sorted in the user interface</td>
</tr>
<tr>
<td>type</td>
<td>string</td>
<td>Type of the view</td>
</tr>
<tr>
<td>scope</td>
<td>string</td>
<td>Scope of the view</td>
</tr>
</tbody>
</table>