

---

# AniList Api Docs Documentation

*Release 0.0.1*

**Josh Star**

April 17, 2015



<b>1</b>	<b>Contents:</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.2	Authentication . . . . .	4
1.3	User . . . . .	7
1.4	User Lists . . . . .	11
1.5	Anime . . . . .	13
1.6	Manga . . . . .	16
1.7	Characters . . . . .	18
1.8	Staff . . . . .	19
1.9	Studio . . . . .	21
1.10	Forum . . . . .	22
<b>2</b>	<b>Indices and tables</b>	<b>29</b>



Welcome to the [AniList.co](#) JSON API documentation.

For any questions, feature requests or issues with our API, or API documentation please use our [GitHub Issue tracker](#), not the AniList forums.



---

## Contents:

---

## 1.1 Introduction

All AniList Api url's featured in this documentation require the following url prefix:

`https://anilist.co/api/`

**HTTPS is required** If a client requests a HTTP url they will **not** be redirected to the HTTPS variant.

### API Terms of use

- Free for non-commercial use.
- Using the AniList API as a backup or data storage service is strictly prohibited.
- 'Hoarding' or mass collection of data from the AniList API is strictly prohibited.

If you would like to use the AniList API in a commercial client (this includes ads and in-app purchases) just drop me an email `joshstar [a-t] mail [do-t] com` and I'll get back to you eventually. With the current usage of the API it's rather unlikely any kind of profit cut etc will be necessary.

**Naming Guidelines** Applications or services that utilizes the AniList API must adhere to our naming guidelines (case insensitive):

- If "AniList" or "AniChart" are used in the title/name of the application you must clearly state they are an unofficial app by appending either "UNOFFICIAL" or "for AniList"/"for AniChart" to the title/name of the application.
- Just the title/name "AniList" and "AniChart" are strictly NOT permitted.

**Adult content and application stores** We feel we should mention many app-stores like the Apple app store prohibit 18+ content. If you're thinking about providing your client via a third-party service you should check their official state on the matter. Where we can, we provide an adult boolean in anime/manga API data, however we can not 100% ensure that this will always be accurate or that our definition of 'adult' content meets the same standards of other services. Specifically 'Ecchi' shows are not included in the 'adult' boolean, which is known to have caused issues with Google Adsense and the Apple app-store in the past.

We can also not ensure the data provided by our user's always be non-adult, (primarily activity and forum data) however we do try to remove this data as soon as possible.

We recommend you do additional checks and limit the data shown on your client if necessary, to comply with the standards of any service you may be using.

### 1.1.1 Creating a client

Log into your AniList account (or create one if you haven't already) and go to [/developer](#). Click the plus in the top right and enter your client's information, once saved you will receive your client id and secret.

### 1.1.2 Terminology

- **Access token** - A token used to access protected resources
- **Authorization code** - An intermediary token generated when a user authorizes a client to access protected resources on their behalf. The client receives this token and exchanges it for an access token.
- **Client** - An application which accesses protected resources on behalf of the resource owner. The client could be hosted on a server, desktop, mobile app or other device.
- **Grant** - A grant is a method of acquiring an access token.
- **Resource owner** - The owner of a protected resource, usually the user.
- **Resource server** - The AniList API server, which sits in front of protected resources and is capable of accepting and responding to protected resource requests using access tokens.

Terminology courtesy of Alex Bilbie

## 1.2 Authentication

### 1.2.1 Which Grant type to use?

If you only require reading of AniList data, then use *client credentials* grant type.

If you want to save, edit or remove (POST, PUT, DELETE) any AniList user (resource owner) data then you'll need to use the *authorization code* or *authorization pin* grant type.

The *authorization code* and *authorization pin* grant types are very similar. The only main difference being the *authorization code* grant will redirect the user to a redirect uri with the authorization code, while the *authorization pin* grant will display a pin that the user must copy and paste into the client.

Both the current grant types have a client secret key, you are required to keep this **private**.

### 1.2.2 Grant: Authorization Code

The authorization code grant type allows the client access to view, add, edit and remove a resource owner's data on their behalf. To do this we require the permission from the resource owner themselves, this will provide us with an *authorization code*, which we can later exchange for the required *access token*.

Request authorization code:

```
GET: auth/authorize
```

```
Url Params:
grant_type    : "authorization_code"
client_id     : Client id
redirect_uri  : Client redirect uri
response_type : "code"
```



This will direct the resource owner to a web page where they may choose to accept or deny the client. If the resource owner is not currently logged in, they will be redirected to the standard AniList login page, then redirect back to the client approval page once logged in.

If the resource owner accepts the client, they will be redirected to the client's redirected uri. A *code* parameter will be included in the redirect uri, this is **not** the access token, but instead the *authorization code* which will be exchanged for the access token in the next step.

Request access token:

```
POST: auth/access_token
```

Url Params:

```
grant_type      : "authorization_code"
client_id       : Client id
client_secret   : Client secret
redirect_uri    : Client redirect uri
code            : Authorization code
```

Return example:

```
{
  access_token: "ACXD3snrImEP5R6IHs6gGgqpGgoZp54TDaWZkgc"
  token_type: "bearer"
  expires: 1414232110
  expires_in: 3600
  refresh_token: "X2BxjlKzjsoaD4FCj6A0MGFWdYlGgoc31L70eSAQ"
}
```

For security this access token will expire in 1 hour. We don't want the resource owner to re-accept the client every time the access token becomes invalid, so we use the *refresh token* to request a new one.

Request access token via refresh token:

```
POST: auth/access_token
```

Url Params:

```
grant_type      : "refresh_token"
client_id       : Client id
client_secret   : Client secret
refresh_token   : Refresh Token
```

Return example:

```
{
  access_token: "n6c4Rk1lnTD3CY1lKfJVlRXvIGOH4yLhAVyf5Iz"
  token_type: "bearer"
  expires: 1414233512
  expires_in: 3600
}
```

Once again this access token will expire in 1 hour. Use the refresh token from before to repeat this step whenever necessary.

Now to access the resource server on the resource owner's behalf, simply include the following header with all your requests

```
Authorization: Bearer access_token
```

Ensure your Content type header is set to URL encoded.

Content-Type: application/x-www-form-urlencoded

### 1.2.3 Grant: Authorization Pin

The authorization pin grant type allows the client access to view, add, edit and remove a resource owner's data on their behalf. To do this we require the permission from the resource owner themselves, this will provide us with an *authorization pin*, which we can later exchange for the required *access token*.

Request authorization pin:

GET: auth/authorize

Url Params:  
grant\_type : "authorization\_pin"  
client\_id : Client id  
response\_type : "pin"

This will direct the resource owner to a web page where they may choose to accept or deny the client. If the resource owner is not currently logged in, they will be redirected to the standard AniList login page, then redirect back to the client approval page once logged in.

If the resource owner accepts the client, the authorization pin will be displayed for the user to copy and paste into the client. The client can then use this pin to request an access token.

Request access token:

POST: auth/access\_token

Url Params:  
grant\_type : "authorization\_pin"  
client\_id : Client id  
client\_secret : Client secret  
code : Authorization pin

Return example:

```
{
  access_token: "ACXD3snrImEP5R6IHs6gGgqpnGgoZp54TDaWZkgc"
  token_type: "bearer"
  expires: 1414232110
  expires_in: 3600
  refresh_token: "X2Bxj1KzjsoaD4FCj6A0MGFWdYlGgoc31L70eSAQ"
```

For security this access token will expire in 1 hour. We don't want the resource owner to re-accept the client every time the access token becomes invalid, so we use the *refresh token* to request a new one.

Request access token via refresh token:

POST: auth/access\_token

Url Params:  
grant\_type : "refresh\_token"  
client\_id : Client id  
client\_secret : Client secret  
refresh\_token : Refresh Token

Return example:

```
{
  access_token: "n6c4Rk1lnTD3CY1lKfJVlRXvIGOH4yLhAVyf5Iz"
  token_type: "bearer"
  expires: 1414233512
  expires_in: 3600
}
```

Once again this access token will expire in 1 hour. Use the refresh token from before to repeat this step whenever necessary.

Now to access the resource server on the resource owner's behalf, simply include the following header with all your requests

```
Authorization: Bearer access_token
```

Ensure your Content type header is set to URL encoded.

```
Content-Type: application/x-www-form-urlencoded
```

### 1.2.4 Grant: Client Credentials

The client credentials grant type allows the client itself permission to read (GET) data from the AniList API. Reading certain current-user specific data, general editing, adding, deleting of data is not accessible from this grant type. However this grant type doesn't require any resource owner's permission, thus is much quicker and easier to set up and use.

Request access token:

```
POST: auth/access_token
```

```
Url Params:
grant_type      : "client_credentials"
client_id       : Client id
client_secret   : Client secret
```

Return example:

```
{
  access_token: "NR3M3vXgHK0kmluOcJVlRXvbGOg4yLhAVyf5If"
  token_type: "bearer"
  expires: 1414234981
  expires_in: 3600
}
```

You can now access the majority of the resource server's GET end points by including this access token as a "access\_token" header or url parameter. For security this access token will expire in 1 hour, to receive a new one simply repeat this step.

## 1.3 User

### User Model

```
{
  "id": 1,
  "display_name": "Josh",
  "anime_time": 54067,
```

```
"manga_chap": 587,
"about": "Admin of this site and AniChart.net. Basically a Comp Sci student with interest in",
"list_order": 0,
"adult_content": true,
"following": false,
"image_url_lge": "http://img.anilist.co/user/reg/1.png",
"image_url_med": "http://img.anilist.co/user/sml/1.png",
"image_url_banner": "http://i.imgur.com/ZHAUS4K.jpg",
"title_language": "romaji",
"score_type": 4,
"custom_list_anime": [
    "",
    "Fall Anime",
    "Summer Anime",
    "",
    ""
],
"custom_list_manga": [
    "",
    "",
    "",
    "",
    ""
],
"advanced_rating": true,
"advanced_rating_names": [
    "Story",
    "Characters",
    "Visuals",
    "Audio",
    "Enjoyment"
],
"notifications": 0
}
```

### Small User Model

```
{
  "id": 1,
  "display_name": "Josh",
  "image_url_lge": "http://img.anilist.co/user/reg/1.png",
  "image_url_med": "http://img.anilist.co/user/sml/1.png"
}
```

## 1.3.1 Basic

Url

GET: user/{id || displayname}

Returns a user model.

Current authenticated user

GET: user

Returns user model. Only available for *authorization code* or *authorization pin* grant types.

### 1.3.2 Activity

#### User Activity:

GET: `user/{id || displayname}/activity`

Url Params:  
page : page number

Returns the activity of the user and activity messages from of other users.

#### Current user's activity feed:

GET: `user/activity`

Url Params:  
page : page number

Returns the activity of the current user and the users they are following.

### 1.3.3 Create activity [POST]

#### Activity status

POST: `user/activity`

#### Payload

text: (string) activity text

#### Activity message

POST: `user/activity`

#### Payload

text: (string) activity text  
messenger\_id: (int) recipient user id

#### Activity reply

POST: `user/activity`

#### Payload

text: (string) activity text  
reply\_id: (int) activity id

### 1.3.4 Remove activity [DELETE]

#### Remove activity

DELETE: `user/activity`

#### Payload

id: (int) activity id

#### Remove activity reply

DELETE: user/activity/reply

Payload

id: (int) activity reply id

### 1.3.5 Notifications

Url

GET: user/notifications

Returns up to 10 notifications of the current user.

**Count**

GET: user/notifications/count

Returns int of current outstanding notifications of current user.

*Note: Only available via authorization code grant.*

### 1.3.6 Followers & Following

**Following:**

GET: user/{id || displayname}/following

**Followers:**

GET: user/{id || displayname}/followers

### 1.3.7 Follow/Unfollow [POST]

**Toggle follow**

POST: user/follow

Payload

id: (int) user id

### 1.3.8 Favourites

Url

GET: user/{id || displayname}/favourites

Returns a user's favourites.

### 1.3.9 Airing

Url:

GET: `user/airing`

Url Params:

`limit` : int number of entries returned

Returns anime list entry with small model anime, where the anime is currently airing and being currently watched by the user.

*Note: Only available via authorization code/pin grant.*

### 1.3.10 Search

Url

GET: `user/search/{query}`

Returns small user models.

## 1.4 User Lists

### List score type

0. 10 Point (0-10 int)
1. 100 Point (0-100 int)
2. 5 Star (0-5 int)
3. 3 Smiles ("`:`", "`:/`", "`:/`") String)
4. 10 Point decimal (0.0 - 10.0 Float)

- The AniList API will automatically convert and output the correct score format for the user's type.
- String score types with the score value 0 will output "-".

### List score order

0. Score
1. Alphabetical

### 1.4.1 Animelist

Url

GET: `user/{id || displayname}/animelist`

Raw

GET: `user/{id || displayname}/animelist/raw`

Raw outputs the same as the standard animelist output but the without anime relation data.

## 1.4.2 Anime list - Create entry [POST] / Edit entry [PUT]

### Create

POST: animelist

### Edit

PUT: animelist

### Payload

```
id: (int) anime_id of list item
list_status: (String) "watching" || "completed" || "on-hold" || "dropped" || "plan to watch"
score: (See top of page - List score types)
episodes_watched: (int)
rewatched: (int)
notes: (String)
advanced_rating_scores: comma separated scores, same order as advanced_rating_names
custom_lists: comma separated 1 or 0, same order as custom_list_anime
hidden_default: (int) 0 || 1
```

## 1.4.3 Mangalist

### Url

GET: user/{id || displayname}/mangalist

### Raw

GET: user/{id || displayname}/mangalist/raw

Raw outputs the same as the standard mangalist output but without the manga relation data.

## 1.4.4 Manga list - Create entry [POST] / Edit entry [PUT]

### Create

POST: mangalist

### Edit

PUT: mangalist

### Payload

```
id: (int) manga_id of list item
list_status: (String) "reading" || "completed" || "on-hold" || "dropped" || "plan to read"
score: (See top of page - List score types)
volumes_read: (int)
chapters_read: (int)
reread: (int)
notes: (String)
advanced_rating_scores: comma separated scores, same order as advanced_rating_names
custom_lists: comma separated 1 or 0, same order as custom_list_manga
hidden_default: (int) 0 || 1
```



### 1.4.5 Remove entry [DELETE]

Anime list

DELETE: animelist/{anime\_id}

Manga list

DELETE: mangalist/{manga\_id}

## 1.5 Anime

**Title languages** To make supporting the user *title\_language* option easier, if the english or japanese titles are not available their values will be replaced the the standard romaji title.

**Anime Model**

```
{
  "id": 1,
  "title_romaji": "Cowboy Bebop",
  "type": "TV",
  "image_url_med": "http://anilist.co/img/dir/anime/med/1.jpg",
  "image_url_sml": "http://anilist.co/img/dir/anime/sml/1.jpg",
  "start_date": "1998-04-03T21:00:00+09:00",
  "end_date": "1999-04-24T21:00:00+09:00",
  "classification": "R - 17+ (violence & profanity)",
  "hashtag": null,
  "source": null,
  "title_japanese": "",
  "title_english": "Cowboy Bebop",
  "synonyms": [],
  "description": "Enter a world in the distant future, where Bounty Hunters roam the solar system",
  "genres": [
    "Action",
    "Adventure",
    "Comedy",
    "Drama",
    "Sci-Fi",
    "Space"
  ],
  "image_url_lge": "http://anilist.co/img/dir/anime/reg/1.jpg",
  "image_url_banner": "http://anilist.co/img/dir/anime/banner/1.jpg",
  "duration": 24,
  "airing_status": "finished airing",
  "average_score": "86.8",
  "total_episodes": 26,
  "youtube_id": null,
  "adult": false,
  "popularity": 7574,
  "relation_type": null,
  "role": null,
  "list_stats": {
    "plan_to_watch": 1673,
    "watching": 421,
    "completed": 4855,
    "on_hold": 511,
    "dropped": 114
  }
}
```

```
    },
    "airing": {
      "time": "2015-04-12T00:00:00+09:00",
      "countdown": 497524,
      "next_episode": 2
    }
  }
}
```

### Small Anime Model

```
{
  "id": 1,
  "title_romaji": "Cowboy Bebop",
  "type": "TV",
  "image_url_med": "http://anilist.co/img/dir/anime/med/1.jpg",
  "image_url_sml": "http://anilist.co/img/dir/anime/sml/1.jpg",
  "title_japanese": "",
  "title_english": "Cowboy Bebop",
  "synonyms": [],
  "image_url_lge": "http://anilist.co/img/dir/anime/reg/1.jpg",
  "airing_status": "finished airing",
  "average_score": "86.8",
  "total_episodes": 26,
  "adult": false,
  "popularity": 7574,
  "relation_type": null,
  "role": null
}
```

## 1.5.1 Basic

Url

GET: anime/{id}

Returns anime model.

## 1.5.2 Page

Url

GET: anime/{id}/page

Returns anime model with the following:

Up to 9 small model characters (ordered by main role) with Japanese small model actors  
Up to 9 small model staff  
Up to 2 small model reviews with their users  
Relations (small model anime)  
Manga relations (small model manga)  
Studios  
External links

## 1.5.3 Characters / Staff

Url

```
GET: anime/{id}/characters
alt: anime/{id}/staff
alt: anime/{id}/actors
```

Returns anime model with the following:

Small model characters (ordered by main role) with small model actors  
Small model staff

## 1.5.4 Airing

Url

```
GET: anime/{id}/airing
```

- Key: Episode number
- Value: Airing Time

## 1.5.5 Browse

Returns up to 40 small anime models if paginating.

Browse

```
Get: browse/anime
```

Url Params:

```
year          : 4 digit year e.g. "2014"
season        : "winter" || "spring" || "summer" || "fall"
type          : "Tv" || "Movie" || "Special" || "OVA" || "ONA" || "Tv Short"
status        : "Not Yet Aired" || "Currently Airing" || "Finished Airing" || "Cancelled"
genres        : Comma separated genre strings. e.g. "Action,Comedy" Returns anime that have ALL the
genres_exclude: Comma separated genre strings. e.g. "Drama" Excludes returning anime that have ANY of
sort          : "id" || "score" || "popularity" || "start date" || "end date" Sorts results, default
airing_data   : "airing_data=true" Includes anime airing data in small models
full_page     : "full_page=true" Returns all available results. Ignores pages. Only available when s
page          : int
```

Genre List

```
GET: genre_list
```

List of genres for use with browse queries

*The old browse API endpoints will continue to be supported until the next major API version update*

## 1.5.6 Favourite [POST]

Toggle favourite

```
POST: anime/favourite
```

Payload

```
id: (int) anime id
```

## 1.5.7 Search

Url

GET: `anime/search/{query}`

Returns small anime models.

## 1.6 Manga

**Title languages** To make supporting the user *title\_language* option easier, if the english or japanese titles are not available their values will be replaced the the standard romaji title.

### Manga Model

```
{
  "id": 1,
  "title_romaji": "Monster",
  "type": "Manga",
  "start_date": "1994-12-05T00:00:00+09:00",
  "end_date": "2001-12-20T00:00:00+09:00",
  "title_japanese": "",
  "title_english": "Monster",
  "synonyms": [],
  "description": "Dr. Kenzo Tenma is a renowned young brain surgeon of Japanese descent working",
  "genres": [
    "Mystery",
    "Drama",
    "Psychological",
    "Seinen"
  ],
  "image_url_lge": "http://anilist.co/img/dir/manga/reg/1.jpg",
  "image_url_med": "http://anilist.co/img/dir/manga/med/1.jpg",
  "image_url_banner": null,
  "publishing_status": "finished",
  "average_score": "90.2",
  "total_chapters": 162,
  "total_volumes": 18,
  "adult": false,
  "popularity": 7574,
  "relation_type": null,
  "role": null,
  "list_stats": {
    "plan_to_read": 1673,
    "reading": 421,
    "completed": 4855,
    "on_hold": 511,
    "dropped": 114
  }
}
```

### Small Manga Model

```
{
  "id": 698,
  "title_romaji": "Neon Genesis Evangelion",
  "type": "Manga",
```

```
"title_japanese": "",
"title_english": "Neon Genesis Evangelion",
"synonyms": [],
"image_url_lge": "http://anilist.co/img/dir/manga/reg/698.jpg",
"image_url_med": "http://anilist.co/img/dir/manga/med/698.jpg",
"publishing_status": "finished",
"average_score": "85.3",
"total_chapters": 95,
"total_volumes": 14,
"adult": false,
"popularity": 7574,
"relation_type": null,
"role": null
}
```

### 1.6.1 Basic

Url

GET: manga/{id}

Returns manga model.

### 1.6.2 Page

Url

GET: manga/{id}/page

Returns manga model with the following:

Up to 9 small model characters (ordered by main role)  
Up to 9 small model staff  
Up to 2 small model reviews with their users  
Relations (small model manga)  
Anime relations (small model anime)

### 1.6.3 Characters / Staff

Url

GET: manga/{id}/characters  
alt: manga/{id}/staff

Returns anime model with the following:

Small model characters (ordered by main role)  
Small model staff

### 1.6.4 Browse

Returns up to 40 small manga models.

Browse

Get: browse/manga

Url Params:

```
year          : 4 digit year e.g. "2014"
type          : "Manga" || "Novel" || "Manhua" || "Manhwa" || "One" || "Doujin"
status        : "Not Yet Published" || "Currently Publishing" || "Finished" || "Cancelled"
genres        : Comma separated genre strings. e.g. "Action,Comedy" Returns manga that have ALL the
genres_exclude : Comma separated genre strings. e.g. "Drama" Excludes returning manga that have ANY of
sort          : "id" || "score" || "popularity" || "start date" || "end date" Sorts results, default
page          : int
```

### Genre List

GET: genre\_list

List of genres for use with browse queries

*The old browse API endpoints will continue to be supported until the next major API version update*

## 1.6.5 Favourite [POST]

Toggle favourite

POST: manga/favourite

Payload

id: (int) manga id

## 1.6.6 Search

Url

GET: manga/search/{query}

Returns small manga models.

## 1.7 Characters

### Character model

```
{
  "name_alt": "",
  "info": "Height: 6'10"; Weight: 155    Spike Spiegel is a tall and lean 27-year-old
  "id": 1,
  "name_first": "Spike",
  "name_last": "Spiegel",
  "name_japanese": "",
  "image_url_lge": "http://anilist.co/img/dir/character/reg/1.jpg",
  "image_url_med": "http://anilist.co/img/dir/character/med/1.jpg",
  "id_actor": null,
  "role": null
}
```

### Small Character Model

```
{
  "id": 1,
  "name_first": "Spike",
  "name_last": "Spiegel",
  "image_url_lge": "http://anilist.co/img/dir/character/reg/1.jpg",
  "image_url_med": "http://anilist.co/img/dir/character/med/1.jpg",
  "id_actor": null,
  "role": "Main",
}
```

### 1.7.1 Basic

Url:

GET: `character/{id}`

Returns character model.

### 1.7.2 Page

Url:

GET: `character/{id}/page`

Returns characters model with the following:

Small model anime with small model character  
Small model anime staff  
Small model manga staff

### 1.7.3 Favourite [POST]

Toggle favourite

POST: `character/favourite`

Payload

`id: (int) character id`

### 1.7.4 Search

Url

GET: `character/search/{query}`

Returns small character models.

## 1.8 Staff

**Staff/Actor Model**

```
{
  "dob": 9081972,
  "website": "http://www.atomicmonkey.jp/jp/amprofile/seki.html",
  "info": "Hometown: Tokyo, Japan Blood type: AB Alias: Monto Hiraku Also Known as: Seki Mon",
  "id": 1,
  "name_first": "Tomokazu",
  "name_last": "Seki",
  "name_first_japanese": "",
  "name_last_japanese": "",
  "image_url_lge": "http://anilist.co/img/dir/person/reg/1.jpg",
  "image_url_med": "http://anilist.co/img/dir/person/med/1.jpg",
  "language": "Japanese",
  "role": null
}
```

### Small Staff/Character Model

```
{
  "id": 14,
  "name_first": "Megumi",
  "name_last": "Hayashibara",
  "image_url_lge": "http://anilist.co/img/dir/person/reg/14.jpg",
  "image_url_med": "http://anilist.co/img/dir/person/med/14.jpg",
  "language": "Japanese",
  "role": null
}
```

## 1.8.1 Basic

Url:

```
GET: staff/{id}
alt: actor/{id}
```

Returns staff model.

## 1.8.2 Page

Url:

```
GET: staff/{id}/page
alt: actor/{id}
```

Returns staff model with the following:

```
Small model anime with small model actors
Small model manga
```

## 1.8.3 Favourite [POST]

Toggle favourite

```
POST: actor/favourite
alt: staff/favourite
```



### Payload

```
id: (int) staff id
```

## 1.8.4 Search

### Url

```
GET: staff/search/{query}
alt: actor/search/{query}
```

Returns small staff models.

## 1.9 Studio

### Studio Model

```
{
  "studio_name": "Studio Pierrot",
  "studio_wiki": "http://en.wikipedia.org/wiki/Pierrot_(company)",
  "id": 1,
  "main_studio": null
}
```

### 1.9.1 Basic

#### Url

```
GET: studio/{id}
```

Returns a studio model.

### 1.9.2 Page

#### Url

```
GET: studio/{id}/page
```

Returns a studio model with small anime models.

### 1.9.3 Search

#### Url

```
GET: studio/search/{query}
```

Returns studio models.

## 1.10 Forum

### Tag Ids

1. Anime
2. Manga
3. Light Novels
4. Visual Novels
5. Release Discussion
6. (Unused)
7. General
8. News
9. Music
10. Gaming
11. Site Feedback
12. Bug Reports
13. Site Announcements
14. List Customisation
15. Recommendations
16. Forum Games
17. Misc
18. AniList Apps

### 1.10.1 Feeds

All of the following feeds are returned in the same data format: First they include pagination data

```
"total": 56,  
"per_page": 22,  
"current_page": 1,  
"last_page": 3,  
"from": 1,  
"to": 22,
```

Next, the threads array is within “data”:

```
"data": [  
  {  
    "id": 61,  
    "title": "Example Thread",  
    "sticky": 1,  
    "last_reply": "2014-10-20 09:31:57",  
    "reply_count": 1,  
    "view_count": 19,  
    "tags": [  
      {  
        "id": 7,  
        "name": "General"  
      }  
    ],  
    "tags_anime": [  
      {  
        "id": 18897,  
        "thread_id": 60,  
        "tag_id": 18897,  
        "anime": [  
          {
```

```

        "id": 18897,
        "title_romaji": "Nisekoi",
        "type": "TV",
        "image_url_med": "http://anilist.co/img/dir/anime/med/18897.jpg",
        "image_url_sml": "http://anilist.co/img/dir/anime/sml/18897.jpg",
        "title_japanese": "",
        "title_english": "Nisekoi",
        "image_url_lge": "http://anilist.co/img/dir/anime/reg/18897.jpg",
        "airing_status": "currently airing",
        "average_score": "79.3",
        "total_episodes": 20,
        "adult": false,
        "relation_type": null,
        "role": null
      }
    ]
  },
  "tags_manga": [],
  "user": {
    "id": 1,
    "display_name": "Josh",
    "image_url_lge": "http://img.anilist.co/user/reg/1.png",
    "image_url_med": "http://img.anilist.co/user/sml/1.png"
  },
  "reply_user": {
    "id": 1,
    "display_name": "Josh",
    "image_url_lge": "http://img.anilist.co/user/reg/1.png",
    "image_url_med": "http://img.anilist.co/user/sml/1.png"
  }
}, ...etc

```

- When there is no reply to a thread *last\_reply* will be set to the creation date of the thread.
- *tags\_anime* and *tags\_manga* include an array of small anime/manga models
- *user* includes the small user model of the thread's creator.
- *reply\_user* includes the small user model for the user who last commented in the thread.
- *sticky* is a boolean.

### Recent

GET: `forum/recent`

Url Params:  
 page : page number

Returns threads ordered by most recent activity or creation.

### New

GET: `forum/new`

Url Params:  
 page : page number

Returns threads ordered by most recent creation.

### Subscribed

GET: forum/subscribed

Url Params:  
page : page number

Returns threads the user has subscribed to, ordered by most recent activity or creation.

*Note: Only available via authorization code/pin grant.*

## Tags

GET: forum/tag

Url Params:  
tag : Comma separated tag ids  
anime : Comma separated anime ids  
manga : Comma separated manga ids  
page : page number

Returns threads which belong to all of the included tags, ordered by most recent activity or creation.

## 1.10.2 Thread

Url

GET: forum/thread/{id}

Thread data (No comments):

```
{
  "id": 1,
  "user_id": 2,
  "title": "[Spoilers] Anime! (Episode 1 Discussion)",
  "body": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed quis posuere urna.",
  "sticky": null,
  "locked": null,
  "last_reply": "2014-10-07 10:29:22",
  "last_reply_user": 1,
  "deleted_at": null,
  "created_at": "2014-10-07 10:23:21",
  "reply_count": 2,
  "view_count": 61,
  "subscribed": false,
  "page_data": {
    "total_root": 11,
    "per_page": 10,
    "current_page": 1,
    "last_page": 2,
    "from": 1,
    "to": 10
  },
  "tags": [
    {
      "id": 3,
      "name": "Light Novels"
    }
  ],
  "tags_anime": [
    {
```

```

    "id": 30,
    "thread_id": 1,
    "tag_id": 30,
    "anime": [
      {
        "id": 30,
        "title_romaji": "Neon Genesis Evangelion",
        "type": "TV",
        "image_url_med": "http://anilist.co/img/dir/anime/med/30.jpg",
        "image_url_sml": "http://anilist.co/img/dir/anime/sml/30.jpg",
        "title_japanese": "",
        "title_english": "Neon Genesis Evangelion",
        "image_url_lge": "http://anilist.co/img/dir/anime/reg/30.jpg",
        "airing_status": "finished airing",
        "average_score": "82",
        "total_episodes": 26,
        "adult": false,
        "relation_type": null,
        "role": null
      }
    ]
  },
  "tags_manga": [],
  "user": {
    "id": 1,
    "display_name": "Josh",
    "image_url_lge": "http://img.anilist.co/user/reg/1.png",
    "image_url_med": "http://img.anilist.co/user/sml/1.png"
  },
  "reply_user": {
    "id": 1,
    "display_name": "Josh",
    "image_url_lge": "http://img.anilist.co/user/reg/1.png",
    "image_url_med": "http://img.anilist.co/user/sml/1.png"
  }
}

```

- *page\_data* includes pagination for only the root level comments
- *tags\_anime* and *tags\_manga* include an array of small anime/manga models
- *user* includes the small user model of the thread's creator.
- *reply\_user* includes the small user model for the user who last commented in the thread.
- *sticky*, *locked* and *subscribed* are booleans.

## Thread Comments

Included within "comments":

```

"comments": [
  {
    "id": 139,
    "parent_id": null,
    "user_id": 1,
    "thread_id": 61,
    "comment": "root comment 1",
    "created_at": "2014-10-20 09:31:57",
    "updated_at": "2014-10-26 23:52:58",

```

```
"user": {
  "id": 1,
  "display_name": "Josh",
  "image_url_lge": "http://img.anilist.co/user/reg/1.png",
  "image_url_med": "http://img.anilist.co/user/sml/1.png"
},
"children": [
  {
    "id": 142,
    "parent_id": 139,
    "user_id": 1,
    "thread_id": 61,
    "comment": "child comment 1",
    "created_at": "2014-10-26 23:52:39",
    "updated_at": "2014-10-26 23:53:06",
    "user": {
      "id": 1,
      "display_name": "Josh",
      "image_url_lge": "http://img.anilist.co/user/reg/1.png",
      "image_url_med": "http://img.anilist.co/user/sml/1.png"
    },
    "children": []
  }
]
},
{
  "id": 143,
  "parent_id": null,
  "user_id": 1,
  "thread_id": 61,
  "comment": "root comment 2",
  "created_at": "2014-10-26 23:52:53",
  "updated_at": "2014-10-26 23:53:16",
  "user": {
    "id": 1,
    "display_name": "Josh",
    "image_url_lge": "http://img.anilist.co/user/reg/1.png",
    "image_url_med": "http://img.anilist.co/user/sml/1.png"
  },
  "children": []
}
]
```

- Comments have children comments themselves, that can have children comments, and so on and so forth.

### 1.10.3 Create thread [POST]

#### Create thread

POST: forum/thread

#### Payload

```
title      : (string) thread title
body       : (string) thread body
tags       : Comma separated tag ids
tags_anime : Comma separated anime ids
tags_manga : Comma separated manga ids
```

### 1.10.4 Edit thread [PUT]

#### Edit thread

PUT: forum/thread

#### Payload

```
id          : (int) thread id
title       : (string) thread title
body        : (string) thread body
tags        : Comma separated tag ids
tags_anime  : Comma separated anime ids
tags_manga  : Comma separated manga ids
```

### 1.10.5 Remove thread [DELETE]

#### Remove thread

DELETE: forum/thread/{thread\_id}

### 1.10.6 Thread subscribe [POST]

#### Toggle thread subscribe

POST: forum/comment/subscribe

#### Payload

```
thread_id: (int) thread id
```

### 1.10.7 Create comment [POST]

#### Edit thread

POST: forum/comment

#### Payload

```
thread_id : (int) thread id
comment   : (string) comment text
reply_id  : (int) comment id (only when replying)
```

### 1.10.8 Edit comment [PUT]

#### Edit thread

PUT: forum/comment

#### Payload

```
id       : (int) comment id
comment  : (string) comment text
```

### 1.10.9 Remove comment [DELETE]

Remove thread

DELETE: `forum/comment/{comment_id}`

### 1.10.10 Search

Url

GET: `forum/search/{query}`

Returns search feed threads.



---

## Indices and tables

---

- *genindex*
- *modindex*
- *search*