General advice

RFC notices

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The HTTP specification used throughout this document is RFC7231, which replaces the older RFC2616.

Choosing an endpoint

Use the most specific endpoint possible, while minimizing the number of requests sent. If your application only knows the snowflake of a specific message, then use the form `/api/v1/messages/<message-snowflake>`; however, if you already know what channel the message belongs to, then use `/api/v1/text-channels/<text-channel-snowflake>/messages/<message-snowflake>`. Do not request information just for the purpose of using a different endpoint. Using more specific endpoints allows us to optimize the request, but this is not worth the additional cost of fulfilling two requests.

Acceptable Content-Types

Currently, the API only accepts payloads of the type `application/json`. Support for XML may be added at a later time.

General error responses

All endpoints may respond with the following error codes.

These error codes SHOULD be accompanied by a body detailing the error as precisely as possible:

```
HTTP/1.1 500 Internal Server Error
```

These error codes MAY be accompanied by a body detailing the error:

```
HTTP/1.1 503 Service Unavailable
```

How to read this document

You know it's going to be dry when there's a heading that says "How to read this document".

Placeholders

When a URI, request body, or other piece of code requires variables, the syntax `<variable-description>`

shall be used. Common examples used in this document include:

- `<snowflake>` represents a snowflake
- `<guild-snowflake>` represents a snowflake that identifies a guild
- `<string>` represents any string
- `<uri-string>` represents a string that has been encoded as a URI component (e.g., ' ' becomes '%20')

Query string parameters

When a request accepts query string parameters, those parameters will be explained in bulleted lists like so:

- [0..n] `title=<string>` : returns only resources whose title contains the string
- [0..1] `id=<snowflake>` : returns only resources whose id matches the snowflake exactly
- [1..n] `text-channel=<snowflake>` : returns only resources from the text channel whose id is the snowflake

The bracketed intervals along the left indicate how many times the parameter may appear:

- "[0..n]" indicates that the parameter MAY appear any number of times
- "[0..1]" indicates that the parameter MUST appear either once or not at all
- "[1..n]" indicates that the parameter MUST appear one or more times

0 and 1 may be changed to arbitrary constants.

Endpoints

Guilds

All guilds

```
/api/v1/guilds
```

Create a guild

Request:

```
POST /api/v1/guilds HTTP/1.1
Content-Type: application/json

{
    "name": "Sam Ramaham's Auto Zone and Stripper Dome"
}
```

Response on success:

```
HTTP/1.1 201 Created
Location: /api/v1/guilds/<guild-snowflake>
```

Possible modes of failure:

- If the request body is invalid, the response MUST have a status code of 400 and SHOULD be accompanied by a body detailing the error(s) as precisely as possible.

  ```
  HTTP/1.1 400 Bad Request

  +(The request did not contain a body.
  | The request body did not contain valid JSON.
  | The request body did not contain an object.
  | The object in the request body did not contain a "name" key.
  | The name provided must have a length between 1 and 64, but has length 0.
  | The name provided must have a length between 1 and 64, but has length 69.
  | The name provided must consist only of printable characters, but an unprintable
    character was found at index 42.)
  ```

Retrieve a list of guilds

A request for a collection of guilds may be modified with the following query string parameters:

- [0..1] `before=<snowflake>` : return only guilds created before the timestamp in the snowflake
- [0..1] `after=<snowflake>` : return only guilds created after the timestamp in the snowflake
- [0..n] `name=<string>` : return only guilds whose names include the string

Request:

```
GET /api/v1/guilds HTTP/1.1
GET /api/v1/guilds?before=<snowflake>&after=<snowflake>&name=<string>
```

Response on success

A specific guild

```
/api/v1/guilds/<guild-snowflake>
```

Text Channels

All text channels

```
/api/v1/text-channels
```

All text channels in a specific guild

```
/api/v1/guilds/<guild-snowflake>/text-channels
```

A specific text channel

```
/api/v1/text-channels/<text-channel-snowflake>
/api/v1/guilds/<guild-snowflake>/text-channels/<text-channel-snowflake>
```

Messages

TODO: Messages may be filtered by passing a filter object into a GET request.

All messages

```
/api/v1/messages
```

All messages in a specific channel

```
/api/v1/text-channels/<text-channel-snowflake>/messages
/api/v1/guilds/<guild-snowflake>/text-channels/<text-channel-snowflake>/messages
```

A specific message

```
/api/v1/messages/<message-snowflake>
/api/v1/text-channels/<text-channel-snowflake>/messages/<message-snowflake>
/api/v1/guilds/<guild-snowflake>/text-channels/<text-channel-snowflake>/messages/<message-
```

Users

All users

```
/api/v1/users
```

A specific user

```
/api/v1/users/<user-snowflake>
```

Icons

These endpoints are used to manipulate images stored by the server, including user profile pictures and guild cover images. Every icon is assigned a snowflake.

All icons

```
/api/v1/icons
```

A specific icon

```
/api/v1/icons/<icon-snowflake>
```

The current icon of a guild

```
/api/v1/guilds/<guild-snowflake>/icon
```

The current icon of a user

```
/api/v1/users/<user-snowflake>/icon
```