Discord API: History, Features, and Technical Evaluation

Brennan Miller
Dr. Flores
CPSC 501
Christopher Newport University
11/18/2024

Introduction

Goal: Examine the Discord API's development, applications, as well as its strengths and weaknesses

Structure:

1) History and overview of features
2) Uses cases and applications
3) Technical evaluation
4) Pros and cons
5) Demo

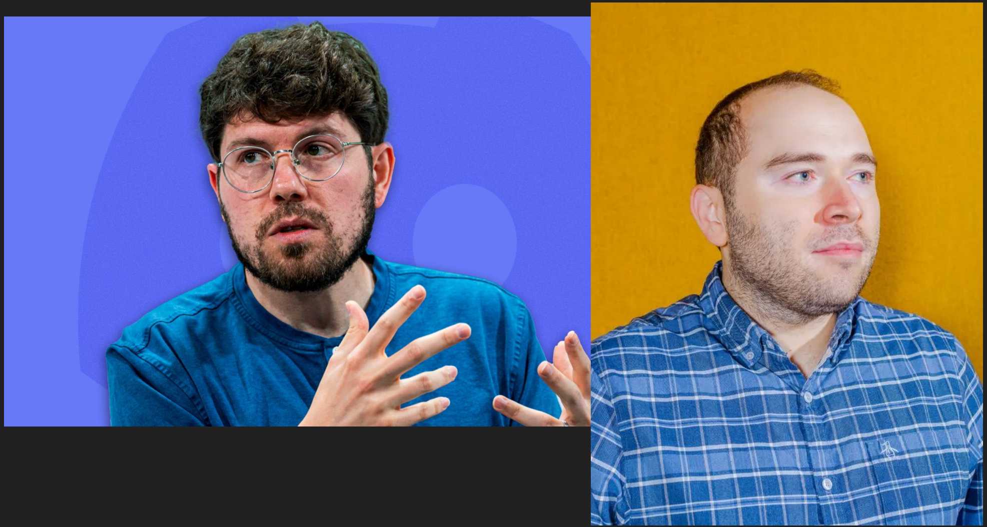
Background

Launched in 2015 as service for gamers

Evolved quickly to support other communities (education, hobbies, etc.)

Major milestones include:

- Bot automation
- Real-time event handling
- Third-party integrations expansion


Pictured: Jason Citron (left) and Stan Vishnevskiy, founders

Key Features

Bot Development:

- Simple creation of bots for community management

Real-Time Event Handling:

- WebSockets for notifications and announcements

User/Role Management:

- Permissions, channels, roles, sanction/kick/ban, etc.

Third Party Integrations:

- Links with tools like Trello, Google Apps, YouTube, Twitch, Patreon, and many more

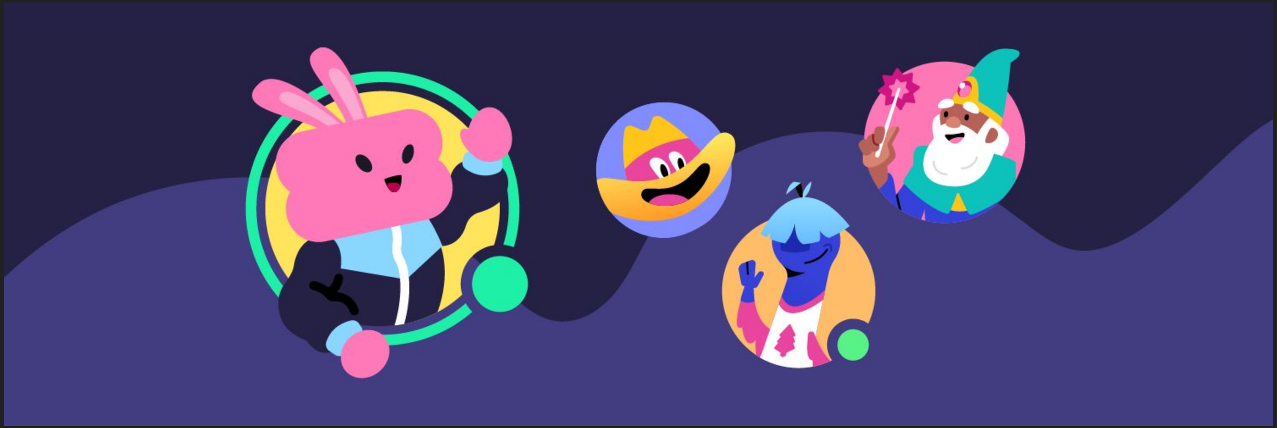
Uses Cases/Applications

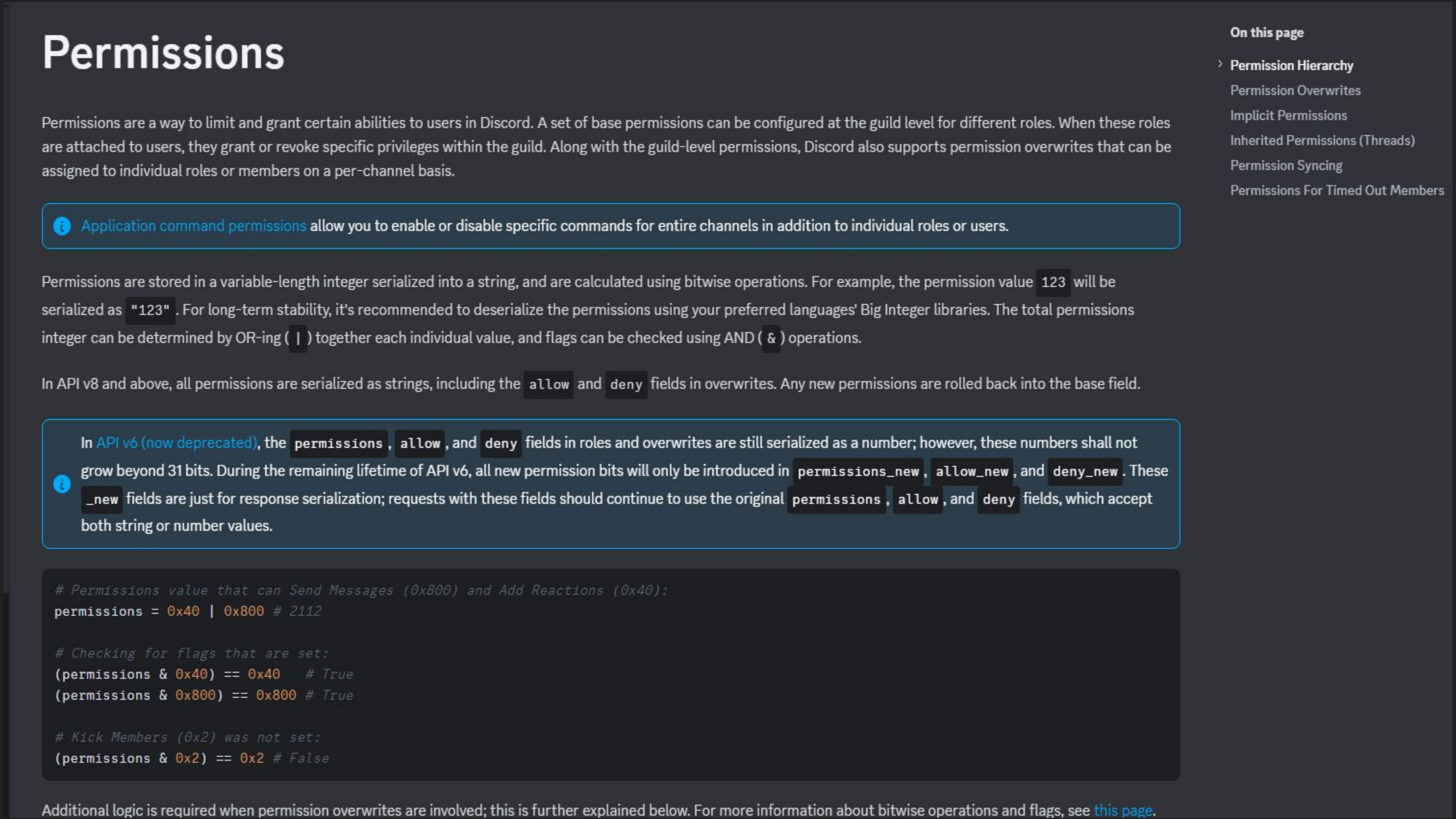
Gaming: Manage communities, announce events, chat moderation

Education: Study sessions, reminders, collaborative projects

Productivity: Easy links with project management tools

Hobby Communities: Collaborative music/drawing tools, image and video sharing





Permissions

Permissions are a way to limit and grant certain abilities to users in Discord. A set of base permissions can be configured at the guild level for different roles. When these roles are attached to users, they grant or revoke specific privileges within the guild. Along with the guild-level permissions, Discord also supports permission overwrites that can be assigned to individual roles or members on a per-channel basis.

Application command permissions allow you to enable or disable specific commands for entire channels in addition to individual roles or users.

Permissions are stored in a variable-length integer serialized into a string, and are calculated using bitwise operations. For example, the permission value `123` will be serialized as `"123"`. For long-term stability, it's recommended to deserialize the permissions using your preferred languages' Big Integer libraries. The total permissions integer can be determined by OR-ing (`|`) together each individual value, and flags can be checked using AND (`&`) operations.

In API v8 and above, all permissions are serialized as strings, including the `allow` and `deny` fields in overwrites. Any new permissions are rolled back into the base field.

In API v6 (now deprecated), the `permissions`, `allow`, and `deny` fields in roles and overwrites are still serialized as a number; however, these numbers shall not grow beyond 31 bits. During the remaining lifetime of API v6, all new permission bits will only be introduced in `permissions_new`, `allow_new`, and `deny_new`. These `_new` fields are just for response serialization; requests with these fields should continue to use the original `permissions`, `allow`, and `deny` fields, which accept both string or number values.

```
# Permissions value that can Send Messages (0x800) and Add Reactions (0x40):
permissions = 0x40 | 0x800 # 2112

# Checking for flags that are set:
(permissions & 0x40) == 0x40   # True
(permissions & 0x800) == 0x800 # True

# Kick Members (0x2) was not set:
(permissions & 0x2) == 0x2 # False
```

Additional logic is required when permission overwrites are involved; this is further explained below. For more information about bitwise operations and flags, see this page.

On this page

Documentation Evaluation

Documentation

- Discord API is not particularly vast
- Well organized and easy to follow documentation
- Sufficient guides for both new and experienced users

Version Compatibility:

- Frequent updates
- Backwards compatible changes
- Very clear deprecation warnings

Design Patterns, Extensibility

API Architecture: RESTful and WebSocket support

Design Pattern: Modular endpoints, adhering to Single Responsibility Principle

- “A module should be responsible to one, and only one, actor”

Extensibility: supports custom commands, third party integrations, and customization to user experience

Strengths and Weaknesses

Strengths

- User friendly documentation
- Real-time interactivity
- Highly customizable
- Sizable and active developer community

Weaknesses

- Strict rate limits
- Limited voice and video controls
- Performance possibly affected by the limitations of Discord’s servers

Demonstration

https://github.com/BJRenz113/DiscordAPIDemoCPSC501

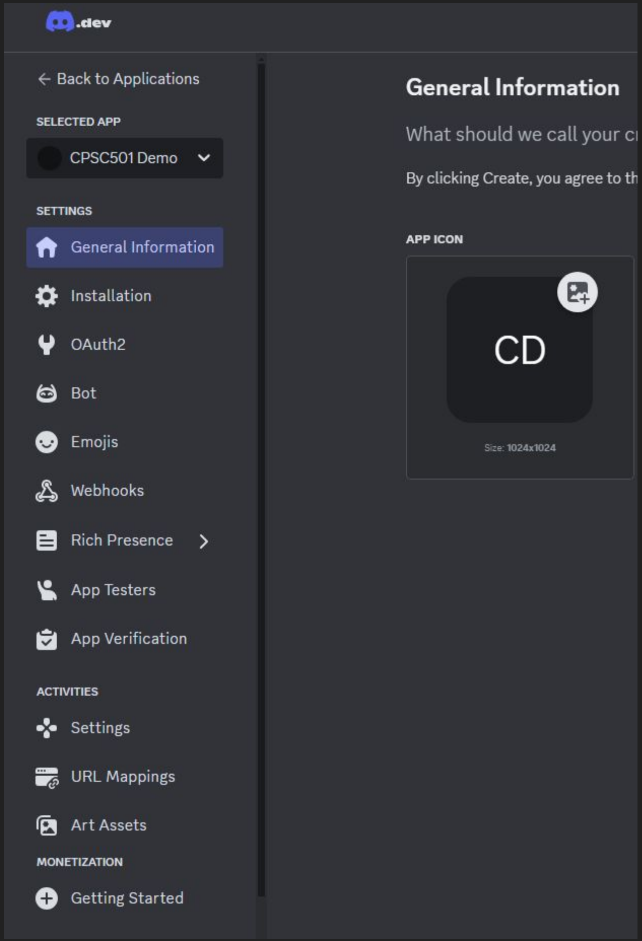
I have constructed a Discord bot using Node.js to demonstrate basics

This includes:

- Bot Authentication/Setup
- Slash Commands
- Event Handling
- User Interactions
- Creating Channels
- Sending Embeds

Bot Creation

1) Access Discord Developer Portal, sign in
2) Select New Application
3) Use the OAuth2 tab
   a) OAuth2 URL Generator -> select “bot” and “applications.commands”
   b) Bot Permissions -> select “Administrator”
4) This will create a link that you can copy/paste into the desired Discord Server
5) This is also where you can access your bot’s token (“password”)



General Information

What should we call your c

By clicking Create, you agree to th

Set Up

When first beginning to code:

- npm init -y -> creates initial package, with -y indicating a “yes” to all questions
- npm install discord.js -> installs the Discord package
- npm install -g nodemon -> nodemon will be used to automatically restart bot after changes, make sure to install globally (in packages.json)
- npm install dotenv -> sets environment variables, used to hide bot’s token, again, install globally

Useful commands:

- nodemon -> starts the bot
- Ctrl+C -> ends the bot’s process
- (for this particular demonstration) src/commandregistration.js -> ensures that commands are free of syntax errors/are usable

Additionally, in order to see Server ID, ensure your Discord Account is set to Developer Mode

Conclusion

The Discord API’s strong points make it ideal for community centered use cases, especially in more informal contexts, however there are certainly some performance and control limitations.

Outlook: potential for expanded functionality and improvement in documentation

Sources

https://discord.com/company

https://discord.com/developers/docs/reference

Images:

https://logos-world.net/wp-content/uploads/2020/12/Discord-Logo.png

https://www.nytimes.com/2021/12/29/business/discord-server-social-media.html

https://www.fastcompany.com/90983953/discords-founder-we-have-video-games-to-thank-for-the-ai-revolution-now-its-time-to-push-it-to-better-places

https://user-images.githubusercontent.com/13700/35731649-652807e8-080e-11e8-88fd-1b2f6d553b2d.png

https://media.licdn.com/dms/image/D5612AQHt-Y26_Im7IA/article-cover_image-shrink_720_1280/0/1714463825421?e=2147483647&v=beta&t=tziCy9eNtdHvurhoCp00iuHTdG0gUpa8zI2CHZ2ToLg

https://discord.com/community

Questions?

Questions?

- In terms of professional applications, how do you think the limitations of the Discord API like voice/video controls and rate limits might affect its adoption?
    - Rate limits hurt professions that need a high number of API calls per minute (stock trading notifs) and limited voice/video controls hurt remote work/conference call software applications

- What features might make the Discord API more enticing to nongaming online communities?
    - Expanded integration, better privacy/security features, increased rate limits, more advanced media controls (like custom audio/video processing)