

NOTES [things to do]

- Look into pooled connections for fast querying
- At each step, consider what data you can leverage for analytics and AI purpose. Perhaps a recommendation system for joining a certain 'group' in a server?
- <https://www.pinecone.io/learn/vector-database/>
- A group_type is a string or phase an admin can set for the allowed groups in the channel. Each group_type is essentially a pointer to a "Category" in a discord server.
- /preset add (preset for project, team, study)
- /preset remove (all, project, team, study)

Index of [commands]:

Global Key:

[] = required parameter

() = optional parameter

Command Prefixes:

/create

/disband

/form

Server Owner Commands:

/wipe

/recover

Admin Commands:

/create group

/create poll

/disband group

User Commands:

/form

/disband

/create [command]

/create Command List:

- /create group
- /create poll

/create Admin Command(s):

1. Create a Group Type

Command:

```
/create group [group_type] [post_channel_id]  
(emoji, default=👍) (std_admin_msg, default="")
```

Example Admin Usage:

```
/create [team] [#find-teams]  
(👍) (**This is a team group type.**)
```

Function:

Only server admin can call this method. When invoked, a server [category_id] is created with the name [group_type]. If set, an (emoji) and (std_admin_msg) is stored for this specific [group_type]. Anytime a user invokes /form group command, it will refer to the [group_type] created by the admin, and gather the necessary [category_id], [post_channel_id], (emoji), and (std_admin_msg) to initialize a user group.

2. Create a Poll



Command:

```
/create poll [question]  
(choices) (existing channel_ids)
```

Example Admin Usage:

```
/create poll [What group do you want to join?]  
(a, b, c) (#a, #b, #c)
```

Function:

A server admin can initiate a poll in any channel. When invoked, a message containing the poll will be posted. If the admin specifies choices, the question will populate with reaction emotes of letters , , ... and so on, otherwise it will populate with  and . Moreover, the admin can optionally map these choices to a specific channel_id. By doing so, users can directly join the corresponding channel by reacting with the appropriate choice.

/disband [command]

Admin Command(s):

1. Disband a Group

Command:

/disband [channel_id] (retype confirmation message)

Example Admin Usage:

/disband [#jacks-team] (I understand this will delete #jacks-team permanently)

Function:

Only a server admin can execute this command. It can be called anywhere inside the server. The admin must retype the confirmation message as shown to confirm deletion. When invoked, the entire channel data is archived, and sent to the [group_owner] and admin in a private DM. Then, the associated [text_channel_id] and [voice_channel_id] are removed from the respective [category_id]. The post with [post_id] is deleted from the [post_channel_id] location. The admin can choose to be notified that a group has been disbanded in the form of a log.

User Command(s):

1. Disband a Group

Command:

/disband (confirmation message)

Example User Usage:

/disband (I understand this will delete #jacks-team permanently)

Function:

Only [group_owner] can call this method inside the

[channel_id] where the appropriate group exists. The user must retype the confirmation message as shown to confirm deletion. When invoked, entire channel data is archived, and sent to the [group_owner] in a private DM. Then, the associated [text_channel_id] and [voice_channel_id] are removed from the respective [category_id]. The post with [post_id] is deleted from the [post_channel_id] location. The admin can choose to be notified that a group has been disbanded in the form of a log.

/form [command]

/form Command List:

- /form group

/form Admin Command(s):

/form User Command(s):

1. Form a Group

Command:

/form group [group_type] [group_name] (user message)

Example User Usage:

/form group [team] [Jack's Team]
(Read this resource: xyz.com)

Function:

FILL THIS OUT

Standard Message Format:

Welcome to our [group_type] "[group_name]"!
Please react with [emoji, default=👍] to board this
group.
(std_admin_msg)
(user message)
[group_type] by @user

Example Output: font size and spacing may vary

Welcome to my team "Jack's Team"!
Please react with 👍 to board this group.
This is a team group type.

Read this resource: xyz.com
Team by @user

/create [database dev-guide]

Relevant Data: [server_id], [group_type], [group_type_color, default=], [emoji, default=👍],
[std_admin_msg, default=]

/create Command List:

- /create group
- /create poll

/create Admin Command(s):

1. Create a Group Type

Command:

```
/create group [group_type] [post_channel_id]  
(emoji, default=👍) (std_admin_msg, default="")
```

Method:

When an admin invokes the /create group command, the bot stores predefined data such as server_id, group_type, category_id, post_channel_id, emoji (defaulting to "👍" if not specified), and std_admin_msg (empty by default) in a dedicated table. Later, when a user uses the /form [group_type] command, the bot refers to this table to fetch details about the group type in the server, using them to generate user posts and server channels accordingly.

Considerations:

- Admin creates duplicate group_type
- deletes Category without bot.

Data Variables:

[server_id] - The server where command was invoked.

[group_type] - The type/category of group set by admin.

[category_id] - The category_id generated from creation of a group_type.

[emoji, default=👍] - Reaction role emoji associated with posts with group_type.

[std_admin_msg, default=""] - Standard message by admin

for posts with group_type.

Table Name: 'group_types'

Table Relationships:

[server_id] (Foreign Key)

Ensures every 'group_type' is associated with an existing server.

[server_id, group_type] (Composite Primary Key)

Ensures within a specific server, duplicate group_types cannot exist.

Table Columns:

server_id	group_type	category_id	post_channel_id	emoji	std_admin_msg

User Command(s): _____

/disband [database dev-guide]

[group_owner], [group_type], [group_type_color, default=], [group_name],
[emoji, default=👍], [std_admin_msg, default=], [post_id], [role_id], [channel_id]

Admin Command(s):

User Command(s):

1. Disband a Group

Command:

/disband

Method:

In the database table 'form_group_type', any rows with [channel_id] (foreign key) are removed.

Table Name: 'form_group_type'

Data Variables:

[server_id] - Reference to server where command was invoked.

[group_owner] - Reference to user who created the group.

[group_name] - Name of the group, role, and channel.

[role_id] - Role ID create for the group_name.

[post_id] - ID of the post with the reaction role.

[channel_id] - ID of the private channel created for the group.

Table Relationships:

[server_id] (Foreign Key)

references primary key in the Servers Index.

[group_type] (Foreign Key)

references 'group_type' in the 'form_type' table, indicating the type of group, [std_admin_msg] and [emoji] associated with type.

/form [database dev-guide]

Admin Command(s):

User Command(s):

1. Form a Group

Command:

`/form group [group_type] [group_name] (user message)`

Method:

When a user invokes the `/form group` command, the system logs the relevant details into a table. This table captures details like the server's ID, group type, group name, group owner, post ID, emoji, and the IDs of the created text and voice channels, along with the date of creation. When a user executes the command, it leads to the creation of new text and voice channels under the specified category in the server, both named after the group name provided. The invoking user is tagged as the group owner. Concurrently, a message gets posted with an emoji reaction, setting up a reaction-based access control. Users who react to this post with the correct emoji gain access to the channels. To manage inactive groups, a `created_date` timestamp is noted. Groups with 30 days of inactivity alert the admin and the group owner, giving a 7-day grace period to respond to the bot before automated removal.

Data Variables:

[server_id] - The server where command was invoked.

[group_type] - Links [category_id] from 'group_types' table.

[group_owner] - Records the user who created the group.

[group_name] - Name of the text and voice channels.

[post_id] - ID of the post with the reaction role.

[text_channel_id] - ID of the private text channel created for the group.

[voice_channel_id] - ID of the private voice channel created for the group.
[created_date] - Date the group was created.

Table Name: 'groups'

Table Relationships:

[server_id] (Foreign Key)
Ensures every group created is for an existing server.

[server_id, group_type] (Composite Primary Key)
Ensures that a group_type is defined for that server.
Also links to 'group_types' Table giving access to details like [category_id],[post_channel_id], [emoji], and [std_admin_msg].

Table Columns:

server_id	group_type	group_owner	post_id	text_channel_id	Voice_channel_id	Created_date