Botisimo

Jul 23, 2020

Contents

CHAPTER 1

What is Botisimo?

Botisimo is a cross-platform chat bot & viewer engagement tools. Botisimo supports leading stream and chat platforms such as Twitch, Youtube, Facebook and Discord. Botisimo provides analytics for your chats as well as user tracking, custom commands, timers, polls, chat logs, stream overlays, song requests, and more.

Learn More at Botisimo.com →

CHAPTER 2

Membership

Using Botisimo requires a monthly or annual membership.

- **Pro** ($5/mo)
- **Master** ($10/mo)
- **Guild** (starting at $50/mo)

View Membership Details →

CHAPTER 3

Moderation

Many features of Botisimo require moderator privileges in your channel. It is highly recommended that you make Botisimo a moderator on all streaming channels you use.

Learn How to Make Botisimo a Moderator →

CHAPTER 4

Commands

Commands are used to allow you, your moderators, and your viewers to interact with the bot via chat.

Learn About Commands →

4.1 Commands

Commands are used to allow you, your moderators, and your viewers to interact with the bot via chat. Default commands are the commands that come built-in to Botisimo. Custom commands are commands created by you.

4.1.1 Default

Default commands are the commands that come built-in to Botisimo. See below for the complete list. Click on any command to see how it works.

You can change the minimum permissions or disable any of the default commands by visiting the commands page and clicking on "Edit Default Command Permissions".

Note: If a command argument is required it will be wrapped in < > and if the argument is optional it will be wrapped in `[ ]`.

Note: If viewers are not receiving whispers from Botisimo in Twitch, they may need to send it a whisper first to get it working for their account.

!amazon

Default Permission: Everyone

Search Amazon products. Returns top 10 results in Discord and top result in Twitch, YouTube, & Facebook.

Usage: !amazon `<search>`

Arguments:

- `search` **<required>** - The search term(s) to use

Example:

```
user:     !amazon dr. seuss
botisimo: ...
```

Note: We are a participant in the Amazon Services LLC Associates Program, an affiliate advertising program designed to provide a means for us to earn fees by linking to Amazon.com and affiliated sites.

!bridge (Discord only)

Default Permission: Moderator

A Discord bridge is a way to send messages between multiple Discord servers running Botisimo. The *!bridge* command is used to create/join/leave/delete Discord bridges via chat.

- `delete` - Delete or leave a bridge *view docs*
- `join` - Join a bridge *view docs*
- `new` - Create a new bridge to host *view docs*

delete

Delete the entire bridge if you are the host. Disconnect from the bridge if you are not the host

Usage: !bridge delete

Example:

```
user:     !bridge delete
botisimo: Bridge deleted
```

join

Join a bridge using a token

Usage: !bridge `<token>`

Arguments:

- `token` **<required>** - The token of the bridge to join

Example:

```
user:     !bridge ggh4jd7f
botisimo: You have been connected to the bridge
```

new

Create a new bridge and get a token for other channels to join the bridge

Usage: !bridge

Example:

```
user:     !bridge
botisimo: This channel is now hosting a bridge. Add other channels to this bridge␣
↪by typing '!bridge ggh4jd7f' in those channels.
```

!buy

Default Permission: Everyone

Purchase an item from the shop using currency.

Usage: !buy `<key>`

Arguments:

- `key` **<required>** - The key for the item you wish to purchase

Example:

```
bob:      !buy 1
botisimo: @user    @bob purchased Super Cool Shoutout for 250 Points
```

!clear (Discord only)

Default Permission: Moderator

Clear a number of messages from the chat (max: 99).

Usage: !clear `<number>`

Arguments:

- `number` **<required>** - The number of messages to clear from chat

Example:

```
user: !clear 50
```

!clip

Default Permission: Everyone

Responds with the top Twitch clip for the current user or the `username` (must be a Twitch username).

Usage: !clip `[username]`

Arguments:

- `username` **[optional]** - The username to get the clip for (if no user then it will use the username of the user issuing the command)

Example:

```
user:     !clip drdisrespect
botisimo: https://clips.twitch.tv/VivaciousEncouragingSparrowCeilingCat
```

!command

Default Permission: Moderator

The `!command` command is used to create/edit/delete/enable/disable custom commands via chat.

- `delete` - Delete command *view docs*
- `disable` - Disable command *view docs*
- `edit` - Edit command response *view docs*
- `enable` - Enable command *view docs*
- `info` - View command info *view docs*
- `list` - List commands *view docs*
- `new` - Create a new command *view docs*
- `permission` - Edit command minimum permission *view docs*
- `reset` - Reset the command use counter *view docs*

Note: Commands are enabled and disabled based on the platform (Twitch, YouTube, Facebook, etc.) the command was issued from. Commands will respond to the channel they are called from.

---

delete

Delete command

Usage: !command delete `<name>`

Arguments:

- `name` **<required>** - The name of the command to delete

Example:

```
user:     !command delete !social
botisimo: Command deleted: !social
```

disable

Disable command

Usage: !command disable `<name>`

Arguments:

- `name` **<required>** - The name of the command to disable

Example:

```
user:     !command disable !social
botisimo: Command disabled: !social
```

edit

Edit command response

Usage: !command edit `<name> <response>`

Arguments:

- `name` **<required>** - The name of the command to delete
- `response` **<required>** - The command response

Example:

```
user:     !command edit !social You can follow me on twitter and instagram␣
↪@botisimo
botisimo: Command updated: !social
```

enable

Enable command

Usage: !command enable `<name>`

Arguments:

- `name` **<required>** - The name of the command to enable

Example:

```
user:     !command enable !social
botisimo: Command enabled: !social
```

info

Get info for a command

Usage: !command info `<name>`

Arguments:

- `name` **<required>** - The name of the command to get info for

Example:

```
user:     !command info !social
botisimo: Command info: !social | enabled | You can follow me on twitter @botisimo
```

list

List commands

Usage: !command list

Example:

```
user:     !command list
botisimo: Commands: !social, !wizard, !roll, !super
```

new

Create a new command

Usage: !command new `<name>` `<response>`

Arguments:

- `name` **<required>** - The name of the command
- `response` **<required>** - The command response

Example:

```
user:     !command new !social You can follow me on twitter @botisimo
botisimo: New command: !social
```

permission

Edit command minimum permission

Usage: !command permission `<name>` `<permission=everyone|regs|subs|mods|admin>`

Arguments:

- `name` **<required>** - The name of the command to update
- `permission` **<required>** - The minimum permission required to use the command (valid values: `everyone`, `regs`, `subs`, `mods`, `admin`)

Example:

```
user:     !command permission !social everyone
botisimo: Command updated: !social
```

reset

Reset the command use counter

Usage: !command reset `<name>`

Arguments:

- `name` **<required>** - The name of the command

Example:

```
user:     !command reset !social
botisimo: Command reset: !social
```

!commands

Default Permission: Everyone

Receive list of custom commands available to you via whisper.

Usage: !commands

Example:

```
user:     !commands
botisimo: !social, !wizard, !roll, !super
```

!currency

Default Permission: Everyone

Receive a DM telling your how much currency you have

Usage: !currency

Example:

```
user:     !currency
botisimo: Currency: 850
```

!enter

Default Permission: Everyone

The `!enter` command is used to enter giveaways via chat.

Usage: !enter

Example:

```
user:     !enter
botisimo: You have been entered into the giveaway
```

!game

Default Permission: Moderator

Get or set the game for the stream.

Usage: !game `[game]`

Arguments:

- `game` **[optional]** - The name of the game to set (if no game then it will respond with the current game)

Example:

```
user:     !game Pickle Wars
botisimo: Game updated: Pickle Wars
```

Example:

```
user:     !game
botisimo: Current game: Pickle Wars
```

!gameall

Default Permission: Moderator

Set the game on all connections on the account.

Usage: !gameall `<game>`

Arguments:

- `game` **<required>** - The name of the game to set

Example:

```
user:     !gameall Pickle Wars
botisimo: Game updated: Pickle Wars
```

!give

Default Permission: Moderator

Give to users by username

Usage: !give `<amount>` `<username>` `[username]` `[username]` ...

Example:

```
user:     !give 500 @username1 @username2
botisimo: @username1 @username2 received 500 currency! !currency
```

Give to all users

Usage: !give `<amount>` all

Example:

```
user:     !give 500 all
botisimo: All users received 500 currency! !currency
```

Give to users currently active in chat

Usage: !give `<amount>` chat

Example:

```
user:     !give 500 chat
botisimo: @username1 received 500 currency! !currency
```

!giveaway

Default Permission: Moderator

The `!giveaway` command is used to create/view/close/end giveaways via chat.

- `close` - Close entry for the current giveaway *view docs*

- `end` - End a giveaway *view docs*
- `new` - Create a new giveaway *view docs*
- `open` - Open entry for the current giveaway *view docs*
- `rules` - View the rules for the current giveaway *view docs*
- `view` - View current giveaway *view docs*
- `winner` - Select a winner *view docs*

Note: You can display the active giveaway in your stream using the stream overlay for giveaways.

---

close

Closes the giveaway to stop further entry, but keeps the results live

Usage: !giveaway close

Example:

```
user:     !giveaway close
botisimo: Giveaway closed: Xbox Controller Giveaway
```

end

Ends the giveaway

Usage: !giveaway end

Example:

```
user:     !giveaway end
botisimo: Giveaway ended: Xbox Controller Giveaway
```

new

Create a new giveaway. This will end the current giveaway if it exists.

Usage: !giveaway new `<title>` | `[cost=0]` | `[multiple_entries]` | `[max_entries=0]` | `[multiple_wins]`

Arguments:

- `title` **<required>** - The title of the giveaway
- `cost` **[optional]** - The cost to enter the giveaway (default: 0)
- `multiple_entries` **[optional]** - Set to any value to allow multiple entries (default: false)
- `max_entries` **[optional]** - The maximum number of entries that a single user can enter the giveaway (0 means unlimited) (default: unlimited)
- `multiple_wins` **[optional]** - Set to any value to allow a single user to win multiple times (default: false)

Example:

```
user:     !giveaway new Xbox Controller Giveaway | 1500 | 1 | 0 | 1
botisimo: New giveaway: Xbox Controller Giveaway **Type !enter to join**
```

open

Opens the giveaway to allow entry

Usage: !giveaway open

Example:

```
user:     !giveaway open
botisimo: Giveaway opened: Xbox Controller Giveaway **Type !enter to join**
```

rules

Get the rules for the current giveaway

Usage: !giveaway rules

Example:

```
user:     !giveaway rules
botisimo: Giveaway rules: Must be following the channel. Must be present when␣
 ↪doing the giveaway raffle.
```

view

Get the current giveaway info

Usage: !giveaway

Example:

```
user:     !giveaway
botisimo: Current giveaway: Xbox Controller Giveaway **Type !enter to join**
```

winner

Select a random winner for the giveaway

Usage: !giveaway winner

Example:

```
user:     !giveaway winner
botisimo: @username has won the Xbox Controller Giveaway Giveaway!
```

!help

Default Permission: Everyone

Link to Botisimo docs.

Usage: !help

Example:

```
user:     !help
botisimo: https://docs.botisimo.com/
```

!irregular

Default Permission: Moderator

Remove user from regulars.

Usage: !irregular `<username>`

Arguments:

- `username` **<required>** - The name of the user to remove from regular list

Example:

```
user:     !irregular username
botisimo: username is no longer a regular
```

!leaderboard

Default Permission: Everyone

Display link to the leaderboard.

Usage: !leaderboard

Example:

```
user:     !leaderboard
botisimo: https://botisimo.com/u/user/leaderboard
```

!leave

Default Permission: Admin

Disable Botisimo for your channel (can be reenabled on the connections page of your account).

Usage: !leave

Example:

```
user:     !leave
botisimo: Bye!
```

!link

Default Permission: Everyone

The `!link` command is used to link users from different platforms together to share currency and XP. Each group of linked users can have 1 user from each platform linked (Twitch, YouTube, Facebook, and Discord). You must have login access to the user account on each platform to be able to link them.

- *Link Users*
- *Check Linked Users*
- *Unlink Users*

Link Users

Log in to a chat platform (in this example we'll use Twitch) and join the channel with Botisimo and initiate the linking process by targeting another platform and username:

```
(twitch) Player1: !link youtube Player1
```

If the targeted user is not in the system, you may need to log in and send a message in chat to allow Botisimo to register that user to the system. If the user is in the system, Botisimo will respond with instructions to complete the link:

```
(twitch) Botisimo: Link initiated. Please log in to youtube as Player1 and type "!
↪link twitch Player1" to finish linking with Player1 [twitch]
```

Log in to the your user on the other platform (in this example we are using YouTube) and complete the link:

```
(youtube) Player1: !link twitch Player1
```

If everything is correct, Botisimo will let you know your users have been linked:

```
(youtube) Botisimo: Player1 [youtube] has been linked with Player1 [twitch]
```

That's it! To link another user you would just repeat the steps:

```
(twitch) Player1: !link discord Player1
(twitch) Botisimo: Link initiated. Please log in to discord as Player1 and type "!
↪link twitch Player1" to finish linking with Player1 [twitch], Player1 [youtube]
```

```
(discord) Player1: !link twitch Player1
(discord) Botisimo: Player1 [discord] has been linked with Player1 [twitch], Player1␣
↪[youtube]
```

Check Linked Users

You can use `!link` to see what users are linked together:

```
(twitch) Player1: !link
(twitch) Botisimo: Player1 [twitch], Player1 [youtube], Player1 [discord]
```

Unlink Users

To unlink a user you can target a platform with the `!unlink` command:

```
(twitch) Player1: !unlink discord
(twitch) Botisimo: Player1 [discord] has been unlinked
(twitch) Player1: !link
(twitch) Botisimo: Player1 [twitch], Player1 [youtube]
```

!live

Default Permission: Moderator

Force a stream up announcement in Discord (useful in case the automatic announcement fails).

Usage: !live

Example:

```
user: !live
```

!music

Default Permission: Moderator

The `!music` command is used to manage the music player via chat.

- `next` - Skip to the next song *view docs*
- `play` - Enable music player *view docs*
- `stop` - Disable music player *view docs*

next

Skip to the next song

Usage: !music next

Example:

```
user:     !music next
botisimo: Song skipped
```

play

Start the music player

Usage: !music play

Example:

```
user:     !music play
botisimo: Music player started
```

stop

Stop the music player

Usage: !music stop

Example:

```
user:     !music stop
botisimo: Music player stopped
```

!nextsong

Default Permission: Moderator

Skip the current song (works with the music player).

Usage: !nextsong

Example:

```
user:     !nextsong
botisimo: Song skipped
```

!optin

Default Permission: Everyone

Opt in to Level Up and DyneOps announcements for your user.

Usage: !optin

Example:

```
user:     !optin
botisimo: You have been opted back in to Botisimo levels and DyneOps notifications
```

!optout

Default Permission: Everyone

Opt out of Level Up and DyneOps announcements for your user.

Usage: !optout

Example:

```
user:     !optout
botisimo: You have been opted out of Botisimo levels and DyneOps notifications
```

!permit

Default Permission: Moderator

Permit a user to bypass the whitelist domain spam filter for 1 minute.

Usage: !permit `<username>`

Arguments:

- `username` **<required>** - Name of the user to permit

Example:

```
user:     !permit @bob
botisimo: @bob has been permitted to send links for 1 minute
```

!ping

Default Permission: Moderator

Responds with `pong` in chat (quick way to check if Botisimo is responding in your channel).

Usage: !ping

Example:

```
user:     !ping
botisimo: pong
```

!poll

Default Permission: Moderator

The `!poll` command is used to create/view/close/end polls via chat.

- `close` - Close voting for a poll *view docs*
- `end` - End a poll *view docs*
- `new` - Create a new poll *view docs*
- `open` - Open voting for a poll *view docs*
- `view` - View current poll *view docs*
- `winner` - Select a winner *view docs*

Note: You can display the active poll in your stream using the stream overlay for polls.

---

close

Closes the poll to stop further voting, but keeps the results live

Usage: !poll close

Example:

```
user:     !poll close
botisimo: Poll voting closed
```

end

Ends the poll

Usage: !poll end

Example:

```
user:     !poll end
botisimo: poll ended
```

new

Create a new poll. This will end the current poll if it exists.

Usage: !poll new `<title>|<option>|[option]|...`

Arguments:

- `title` **<required>** - The title of the poll
- `option` **<required>** - The name of the option for the poll (to add multiple options separate with a `|` character)

Example:

```
user:     !poll new Is this a fun poll? | yes | no | hell no | what a stupid poll
botisimo: New poll: Is this a fun poll? 1. yes (0 votes) 2. no (0 votes) 3. hell␣
↪no (0 votes) 4. what a stupid poll (0 votes)
```

open

Opens the poll to allow voting

Usage: !poll open

Example:

```
user:     !poll open
botisimo: Poll voting opened: Is this a fun poll? 1. yes (0 votes) 2. no (0␣
↪votes) 3. hell no (0 votes) 4. what a stupid poll (0 votes)
```

view

Get the current poll info

Usage: !poll

Example:

```
user:     !poll
botisimo: Current poll: Is this a fun poll? 1. yes (0 votes) 2. no (0 votes) 3.␣
↪hell no (0 votes) 4. what a stupid poll (0 votes)
```

winner

Select a random winner from the given option

Usage: !poll winner `<option>`

Arguments:

- `option` **<required>** - The poll option to select as the winner

Example:

```
user:     !poll winner 1
botisimo: #1 yes wins!
```

!prefix

Default Permission: Admin

Get or set the Botisimo command prefix (only affects Botisimo's built-in commands).

Usage: !prefix `[prefix]`

Arguments:

- `prefix` **[optional]** - The prefix to set on the account (if no prefix then it will respond with the current prefix)

Example:

```
user:     !prefix #
botisimo: Prefix has been updated: #
user:     #prefix !
botisimo: Prefix has been updated: !
```

!queue

Default Permission: Moderator

The `!queue` command is used to join/leave/list users in the queue via chat

- `clear` - Clear the queue *view docs*
- `close` - Close the queue *view docs*
- `join` - Join the queue *view docs*
- `leave` - Leave the queue *view docs*
- `list` - List users in queue *view docs*
- `next` - Get the next user in the queue *view docs*
- `open` - Open the queue *view docs*
- `permission` - Edit queue minimum permission *view docs*
- `random` - Gets a random user in the queue as the next user *view docs*

clear

Clear the queue

Usage: !queue clear

Example:

```
user:     !queue clear
botisimo: Queue cleared
```

close

Close the queue

Usage: !queue close

Example:

```
user:     !queue close
botisimo: Queue closed
```

join

Join the queue

Usage: !queue join

Example:

```
user:     !queue join
botisimo: Added to queue: @user
```

leave

Leave the queue

Usage: !queue leave

Example:

```
user:     !queue leave
botisimo: Removed from queue: @user
```

list

List queue

Usage: !queue list

Example:

```
user:     !queue list
botisimo: Queue: @user
```

next

Retrieve the next user(s) from the queue

Usage: !queue next `[users=1]`

Arguments:

- `users` **[optional]** - The number of users to get from the queue (default: 1)

Example:

```
user:     !queue next
botisimo: Next in queue: @user
```

open

Open the queue

Usage: !queue open

Example:

```
user:     !queue open
botisimo: Queue opened
```

permission

Edit queue minimum permission

Usage: !queue permission `<permission=everyone|regs|subs|mods|admin>`

Arguments:

- `permission` **<required>** - The minimum permission required to use the queue (valid values: `everyone`, `regs`, `subs`, `mods`, `admin`)

Example:

```
user:     !queue permission everyone
botisimo: Queue permission updated: everyone
```

random

Retrieve a random user from the queue as the next user

Usage: !queue random

Example:

```
user:     !queue random
botisimo: Next in queue: @user
```

!rank

Default Permission: Everyone

Display rank information for the current user or the `username`.

Usage: !rank `[username]`

Arguments:

- `username` **[optional]** - The name of the user to get rank for (if no user then it will use the username of the user issuing the command)

Example:

```
user:     !rank
botisimo: @user | Rank 1/3 | Lvl 7 | XP 1,755/6,865 (tot. 6,840)
```

Example:

```
user:     !rank bob
botisimo: @bob | Rank 2/3 | Lvl 6 | XP 1,345/3,765 (tot. 4,540)
```

!reddit

Default Permission: Everyone

Get the latest post(s) for a subreddit. Returns top 10 results in Discord and top result in Twitch, YouTube, & Facebook.

Usage: !reddit `<subreddit>`

Arguments:

- `subreddit` **<required>** - The subreddit to search

Example:

```
user:     !reddit FoodPorn
botisimo: Instead of frying I tried Double baked Garlic Parmesan chilli wings.␣
↪Tossing the wings in baking powder is a game changer. - https://i.redd.it/
↪i45refx73ld41.jpg
```

!regular

Default Permission: Moderator

Add this user to regulars.

Usage: !regular `<username>`

Arguments:

- `username` **<required>** - The name of the user to add to the regular list

Example:

```
user:     !regular username
botisimo: username is now a regular
```

!shop

Default Permission: Everyone

Display available shop items for channel in chat.

Usage: !shop

Example:

```
user:     !shop
botisimo: User's Shop - 1. Pizza (100,000 Points) "!buy 1" 2. Featured Account␣
↪(250 Points) "!buy 2" - more at https://botisimo.com/u/user/shop
```

!songrequest

Default Permission: Everyone

Request a song (works with the music player).

Usage: !songrequest `<search>`

Arguments:

- `search` **<required>** - The search term(s) or video url to use to search YouTube

Example:

```
user:     !songrequest dont stop believing
botisimo: Song added to queue: Journey - Don't Stop Believin'
```

!stackoverflow

Default Permission: Everyone

Search Stackoverflow questions. Returns top 10 results in Discord and top result in Twitch, YouTube, & Facebook.

Usage: !stackoverflow `<search>`

Arguments:

- `search` **<required>** - The search term(s) to use

Example:

```
user:     !stackoverflow how to get words in string javascript
botisimo: Javascript regex how to get all occurences between two words in a␣
↪string - https://stackoverflow.com/questions/23594062/javascript-regex-how-to-
↪get-all-occurences-between-two-words-in-a-string
```

!subscribe (Discord only)

Default Permission: Moderator

Subscribe to other Botisimo users' stream up announcements (platform should be `twitch` or `mixer`).

Usage: !subscribe `<platform>` `<username>`

Arguments:

- `platform` **<required>** - The platform of the user to subscribe to (valid values: `twitch`, `mixer`)
- `username` **<required>** - The username of the user to subscribe to

Example:

```
user:     !subscribe twitch bob
botisimo: You have subscribed to stream up announcement for #bob on Twitch
```

!timer

Default Permission: Moderator

The `!timer` command is used to create/edit/delete/start/stop custom timers via chat.

- `delete` - Delete a timer *view docs*
- `edit` - Edit a timer *view docs*
- `info` - View timer info *view docs*
- `list` - List timers *view docs*
- `new` - Create a new timer *view docs*
- `start` - Start a timer *view docs*
- `stop` - Stop a timer *view docs*

Note: Timers are started and stopped based on the platform (Twitch, YouTube, Facebook, etc.) the command was issued from. Timers will only post to a single channel per platform.

Note: Timers are only activated if 10 or more messages have been sent since the last time it was activated.

---

delete

Delete a timer

Usage: !timer delete `<name>`

Arguments:

- `name` **<required>** - The name of the timer to delete

Example:

```
user:     !timer delete social
botisimo: Timer deleted: social
```

edit

Edit a timer

Usage: !timer edit `<name>` `<interval>` `<message>`

Arguments:

- `name` **<required>** - The name of the timer
- `interval` **<required>** - The interval of the timer in minutes (valid values: `5` to `60`)
- `message` **<required>** - The message for the timer

Example:

```
user:     !timer edit social 10 Follow me on twitter and instagram @botisimo
botisimo: Timer updated: social
```

info

Get info for a timer

Usage: !timer info `<name>`

Arguments:

- `name` **<required>** - The name of the timer to get the info for

Example:

```
user:     !timer info social
botisimo: Timer info: social | 5m | started | Follow me on twitter @botisimo
```

list

List timers

Usage: !timer list

Example:

```
user:     !timer list
botisimo: Timers: social, follow, botisimo
```

new

Create a new timer

Usage: !timer new `<name>` `<interval>` `<message>`

Arguments:

- `name` **<required>** - The name of the timer
- `interval` **<required>** - The interval of the timer in minutes (valid values: `5` to `60`)
- `message` **<required>** - The message for the timer

Example:

```
user:     !timer new social 5 Follow me on twitter @botisimo
botisimo: New timer: social
```

start

Starts a timer in the current channel

Usage: !timer start `<name>`

Arguments:

- `name` **<required>** - The name of the timer to start

Example:

```
user:     !timer start social
botisimo: Timer started: social
```

stop

Stops a timer

Usage: !timer stop `<name>`

Arguments:

- `name` **<required>** - The name of the timer to stop

Example:

```
user:     !timer stop social
botisimo: Timer stopped: social
```

!title

Default Permission: Moderator

Get or set the title for the stream.

Usage: !title `[title]`

Arguments:

- `title` **[optional]** - The title of the stream to set (if no title then it will respond with the current title)

Example:

```
user:     !title Streamin like a demon! Let's GOOO!!!!
botisimo: Title updated: Streamin like a demon! Let's GOOO!!!!
```

Example:

```
user:     !title
botisimo: Current title: Streamin like a demon! Let's GOOO!!!!
```

!titleall

Default Permission: Moderator

Set the title on all connections on the account.

Usage: !titleall `<title>`

Arguments:

- `title` **<required>** - The title of the stream to set

Example:

```
user:     !titleall Streamin like a demon! Let's GOOO!!!!
botisimo: Title updated: Streamin like a demon! Let's GOOO!!!!
```

!twitter

Default Permission: Everyone

Responds with the latest tweet from the given `username` (must be a Twitter username).

Usage: !twitter `<username>`

Arguments:

- `username` **<required>** - The twitter username to search for

Example:

```
user:     !twitter Dadsaysjokes
botisimo: What do you call an angsty teenage robot? A sigh borg. - @Dadsaysjokes␣
↪(22 minutes ago)
```

!unsubscribe (Discord only)

Default Permission: Moderator

Unsubscribe from other Botisimo users' stream up announcements (platform should be `twitch` or `mixer`).

Usage: !unsubscribe `<platform>` `<username>`

Arguments:

- `platform` **<required>** - The platform of the user to subscribe to (valid values: `twitch`, `mixer`)
- `username` **<required>** - The username of the user to subscribe to

Example:

```
user:     !unsubscribe twitch bob
botisimo: You have unsubscribed from stream up announcements for #bob on Twitch
```

!uptime

Default Permission: Everyone

Display time the stream started in chat.

Usage !uptime

Example:

```
user:     !uptime
botisimo: The stream started 2 hours 16 minutes 30 seconds ago
```

!vote

Default Permission: Everyone

The `!vote` command is used to vote in polls via chat.

> **Warning:** Casting multiple votes for a poll may disqualify you from the poll if enabled by the streamer.

Usage: !vote `<option> [bet]`

Arguments:

- `option` **<required>** - The poll option to vote for
- `bet` **[optional]** - The amount of currency to bet (default: 0)

Example:

```
user:     !vote 1
botisimo: Your vote has been tallied!
```

!wikipedia

Default Permission: Everyone

Search Wikipedia topics. Returns top 10 results in Discord and top result in Twitch, YouTube, & Facebook.

Usage: !wikipedia `<search>`

Arguments:

- `search` **<required>** - The search term(s) to use

Example:

```
user:     !wikipedia quadratic equation
botisimo: Quadratic equation - No description found - https://en.wikipedia.org/
↪wiki/Quadratic_equation
```

!winner

Default Permission: Moderator

Pick a random winner from chat who sent a message in last 10 minutes. Optionally, use the `message` to filter users who typed a specific message (default: any message). You can also alter the minutes by using the `|` character followed by a number (example: `!winner enter|30`).

Usage: !winner `[message] [|minutes]`

Arguments:

- `message` **[optional]** - The message that the user must have typed in chat to be qualified to enter the giveaway
- `minutes` **[optional]** - The number of minutes to look back to find qualifying messages

Example:

```
user:     type "go for the win" to qualify
bob:      go for the win
user:     !winner go for the win|5
botisimo: @bob wins!
```

4.1.2 Custom

Custom commands are commands created by you.

Custom command responses are often made up of text that combines *response variables* and *response directives* to achieve dynamic output.

You can create custom commands by visiting the commands page and clicking on **"+ Create New"**.

Related Videos

CHAPTER 5

Timers

Timers are used to automate messages based on time interval and chat activity. This is a great way to promote your brand and your message. Timers are often used to do things like promote social media channels or remind people to like, follow, and subscribe to your streaming channel so they know when you are going to be live next.

Learn About Timers →

5.1 Timers

Timers are used to automate messages based on time interval and chat activity. This is a great way to promote your brand and your message. Timers are often used to do things like promote social media channels or remind people to like, follow, and subscribe so they know when you are going to be live next.

Timer responses are often made up of text that combines *response variables* and *response directives* to achieve dynamic output.

You can create timers by visiting the timers page and clicking on **"+ Create New"**.

5.1.1 Related Videos

CHAPTER 6

Polls

Polls are used to get feedback from your viewers. Polls are often used to do things like ask viewers what game you should play or how many times they think you will win a battle today.

Learn About Polls →

6.1 Polls

Polls are used to get feedback from your viewers. Polls are often used to ask viewers what game we should play or how many times they think I will win a battle today.

You can create polls by visiting the polls page and clicking on **"+ Create New"**.

CHAPTER 7

Giveaways

Giveaways are used to reward your viewers. This is a great way to give back to your viewers for supporting you. Giveaways are often used to give a random viewer gift card codes or in-game items.

Learn About Giveaways →

7.1 Giveaways

Giveaways are used to reward your viewers. This is a great way to give back to your viewers for supporting you. Giveaways are often used to give a random viewer gift card codes or in-game items.

You can create giveaways by visiting the giveaways page and clicking on **"+ Create New"**.

CHAPTER 8

Shop

The shop is used to incentivize your viewers to watch and participate in your stream regularly. When viewers watch your stream they gets currency that they can spend in your shop. The shop often includes items like a personal shout out or maybe you have to do something silly like the ice bucket challenge, be creative.

Learn About Shop →

8.1 Shop

The shop is used to incentivize your viewers to watch and participate in your stream regularly. When viewers watch your stream they gets currency that they can spend in your shop. The shop often includes items like a personal shout out or maybe you have to do something silly like the ice bucket challenge, be creative.

You can create shop items by visiting the shop page and clicking on **"+ Create New"**.

CHAPTER 9

Response Variables

Response variables are used to make dynamic custom command and timer responses.

View Response Variables →

9.1 Variables

Response variables are used to make dynamic custom command and timer responses. See also *response directives*.

Note: If a variable argument is required it will be wrapped in < > and if the argument is optional it will be wrapped in `[ ]`.

9.1.1 Arguments

Resolves the argument from the command input.

Usage: $(`<argument> [fallback]`)

Arguments: `argument` **<required>** - The number of the argument to get from the command starting with 1
`fallback` **[optional]** - The text to display if the argument does not exist

Example Command: **name**: !hello

response: Hello $(1 World)!

output:

```
user:     !hello @username
botisimo: Hello @username!
user:     !hello
botisimo: Hello World!
```

9.1.2 $(bot)

Resolve bot username.

Usage: $(bot)

Example Command: **name**: !checkin

response: $(bot) is here!

output:

```
user:     !checkin
botisimo: @botisimo is here!
```

9.1.3 $(botplain)

Resolve bot username without @ symbol.

Usage: $(botplain)

Example Command: **name**: !checkin

response: $(botplain) is here!

output:

```
user:     !checkin
botisimo: botisimo is here!
```

9.1.4 $(cache)

Resolve a value from the cache. Values can be save to the cache using the *cache directive*.

Usage: $(cache <key>)

Arguments: `key` **<required>** - The key of the value to fetch from the cache

Example Command: **name**: !cache

response: You have a cached value of "$(cache $(username))"

output:

```
user:     !cache
botisimo: You have a cached value of "some value to save for later"
```

9.1.5 $(channel)

Resolves the current channel name.

Usage: $(channel)

Example Command: **name**: !channel

response: Welcome to $(channel)!

output:

```
user:     !channel
botisimo: Welcome to #botisimo!
```

9.1.6 $(count)

Resolve the number of times the command has been used.

Usage: $(count)

Example Command: **name**: !count

response: This command has been used $(count) times

output:

```
user:     !count
botisimo: This command has been used 25 times
```

9.1.7 $(countdown)

Resolves the time between now and the given date.

Usage: $(countdown `<date>` `[time]`)

Arguments:

- `date` **<required>** - The date to compare (ex: 12/25/2020)
- `time` **[optional]** - The time of day (ex: 12:30pm)

Example Command: **name**: !countdown

response: The new year begins in $(countdown 1/1/2021 00:00:00)

output:

```
user:     !countdown
botisimo: The new year begins in 360 days 12 hours 37 minutes
```

9.1.8 $(currency)

Resolve the currency amount for the command issuer

Usage: $(currency)

Example Command: **name**: !currency

response: Currency: $(currency)

output:

```
user:     !currency
botisimo: Currency: 375
```

9.1.9 $(discord)

Show text in Discord chat only.

Usage: $(discord `<message>`)

Arguments:

- `message` **<required>** - The message to display in chat

Example Command: **name**: !example

response: You are a $(discord super awesome and) cool dude$(discord !)

discord output:

```
user:     !example
botisimo: You are a super awesome and cool dude!
```

twitch/mixer/youtube output:

```
user:     !example
botisimo: You are a cool dude
```

9.1.10 $(fetch)

Resolves the response from the url using a GET request.

Note: To avoid hitting any rate limits against remote APIs, all responses are cached for 5 seconds.

Usage: $(fetch `<url>`)

Arguments:

- `url` **<required>** - The url to send the request to

Example Command: **name**: !kanye

response: $(fetch https://api.kanye.rest/?format=text) - Kanye

output:

```
user:     !kanye
botisimo: I'm a creative genius - Kanye
```

9.1.11 $(fetchp)

Resolves the response from the url using a POST request. Similar to *$(fetch)*.

Note: To avoid hitting any rate limits against remote APIs, all responses are cached for 5 seconds.

Usage: $(fetchp `<url> [data]`)

Arguments:

- `url` **<required>** - The url to send the request to

- `data` **[optional]** - The data string to send to the url

Example Command: **name**: !kanye

response: $(fetchp https://api.kanye.rest/?format=text) - Kanye

output:

```
user:     !kanye
botisimo: I'm a creative genius - Kanye
```

9.1.12 $(hours)

Resolves the number of hours the command issuers has watched the live stream

Usage: $(hours)

Example Command: **name**: !hours

response: Hours: $(hours)

output:

```
user:     !hours
botisimo: Hours: 9.28
```

9.1.13 $(js)

Resolves the value of a javascript expression

Usage: $(js `<javascript>`)

Arguments:

- `javascript` **<required>** - The javascript expression to evaluate

Example Command: **name**: !js

response: $(js ['need', 'a', 'better', 'example'].join(' '))

output:

```
user:     !js
botisimo: need a better example
```

9.1.14 $(lastfm)

Resolves the last scrobbled song from lastfm for the given username (must be lastfm username).

Usage: $(lastfm `<username>`)

Arguments:

- `username` **<required>** - The name of the lastfm user

Example Command: **name**: !lastfm

response: $(lastfm botisimo)

output:

```
user:     !lastfm
botisimo: Egzod - Universe
```

9.1.15 $(level)

Resolve the level of the command issuer.

Usage: $(level)

Example Command: **name**: !level

response: Get on my level: $(level)

output:

```
user:     !level
botisimo: Get on my level: 5
```

9.1.16 $(minutes)

Resolves the number of minutes the command issuers has watched the live stream.

Usage: $(minutes)

Example Command: **name**: !minutes

response: Minutes: $(minutes)

output:

```
user:     !minutes
botisimo: Minutes: 600
```

9.1.17 $(mixer)

Show text in Mixer chat only.

Usage: $(mixer `<message>`)

Arguments:

- `message` **<required>** - The message to display in chat

Example Command: **name**: !example

response: You are a $(mixer super awesome and) cool dude$(mixer !)

mixer output:

```
user:     !example
botisimo: You are a super awesome and cool dude!
```

twitch/youtube/discord output:

```
user:     !example
botisimo: You are a cool dude
```

9.1.18 $(pick)

Resolves a random response from the given options.

Usage: $(pick `<option>` | `[option]` | `[option]` | ...)

Arguments:

- `option` **<required>** - The message to display in chat (to add multiple options separate with a `|` character)

Example Command: **name**: !pick

response: I'm thinking... $(pick yes | no | maybe so | I don't know)

output:

```
user:     !pick
botisimo: I'm thinking...maybe so
user:     !pick
botisimo: I'm thinking...yes
```

9.1.19 $(query)

Resolves the query string from the command.

Usage: $(query `[start]`)

Arguments:

- `start` **[optional]** - The word to start on from the beginning of the query

Example Command: **name**: !query

response: $(query) -> $(query 3)

output:

```
user:     !query this is the query
botisimo: this is the query -> the query
```

9.1.20 $(rank)

Resolve the rank of the command issuer.

Usage: $(rank)

Example Command: **name**: !rank

response: You are rank: $(rank)

output:

```
user:     !rank
botisimo: You are rank: 5
```

9.1.21 $(repeat)

Resolves some text repeatedly.

Usage: $(repeat `<count>` `<text>`)

Arguments:

- `count` **<required>** - The number of times to repeat the text
- `text` **<required>** - The text to repeat

Example Command: **name**: !repeat

response: $(repeat 5 PogChamp Kappa)

output:

```
user:     !repeat
botisimo: PogChamp Kappa PogChamp Kappa PogChamp Kappa PogChamp Kappa PogChamp␣
↪Kappa
```

9.1.22 $(rng)

Resolve a random number between given numbers.

Usage: $(rng `<min>` `<max>`)

Arguments:

- `min` **<required>** - The minimum number that can be chosen
- `max` **<required>** - The maximum number that can be chosen

Example Command: **name**: !roll

response: You rolled a $(rng 1 100)

output:

```
user:     !roll
botisimo: You rolled a 76
user:     !roll
botisimo: You rolled a 33
```

9.1.23 $(rotate)

Resolves a response from the given options in sequence.

Usage: $(rotate `<option>` | `[option]` | `[option]` | ...)

Arguments:

- `option` **<required>** - The message to display in chat (to add multiple options separate with a `|` character)

Example Command: **name**: !rotate

response: $(rotate First response | Second response | Third response)

output:

```
user:     !rotate
botisimo: First response
user:     !rotate
botisimo: Second response
user:     !rotate
botisimo: Third response
user:     !rotate
botisimo: First response
```

9.1.24 $(songrequester)

Resolve the requester text of the current song playing on the music player.

Usage: $(songrequester)

Example Command: **name**: !song

response: Now Playing: $(songtitle)$(songrequester)

output:

```
user:     !song
botisimo: Now Playing: Culture Code - Make Me Move (feat. Karra) [NCS Release]␣
↪(requested by @user)
```

9.1.25 $(songthumbnail)

Resolve the url to the thumbnail of the current song playing on the music player.

Usage: $(songthumbnail)

Example Command: **name**: !songthumbnail

response: $(songthumbnail)

output:

```
user:     !songthumbnail
botisimo: https://i1.ytimg.com/vi/vBGiFtb8Rpw/mqdefault.jpg
```

9.1.26 $(songtitle)

Resolves the title of the song currently set to play on the music player.

Usage: $(songtitle)

Example Command: **name**: !song

response: Now Playing: $(songtitle)

output:

```
user:     !song
botisimo: Now Playing: Culture Code - Make Me Move (feat. Karra) [NCS Release]
```

9.1.27 $(songurl)

Resolves the url of the song currently set to play on the music player.

Usage: $(songurl)

Example Command: **name**: !song

response: Now Playing: $(songurl)

output:

```
user:     !song
botisimo: Now Playing: https://youtube.com/watch?v=vBGiFtb8Rpw
```

9.1.28 $(songusername)

Resolve the requester text of the current song playing on the music player.

Usage: $(songusername)

Example Command: **name**: !song

response: Now Playing: $(songtitle) ($(songusername))

output:

```
user:     !song
botisimo: Now Playing: Culture Code - Make Me Move (feat. Karra) [NCS Release]␣
↪(@username)
```

9.1.29 $(streamer)

Resolve streamer username.

Usage: $(streamer)

Example Command: **name**: !schedule

response: $(streamer) is live Mon-Fri 12-3pm!

output:

```
user:     !schedule
botisimo: @streamername is live Mon-Fri 12-3pm!
```

9.1.30 $(streamerplain)

Resolve streamer username without @ symbol.

Usage: $(streamerplain)

Example Command: **name**: !schedule

response: $(streamerplain) is live Mon-Fri 12-3pm!

output:

```
user:     !schedule
botisimo: streamername is live Mon-Fri 12-3pm!
```

9.1.31 $(stripchar)

Strip a character from text.

Usage: $(stripchar `<character>` `<text>`)

Arguments:

- `character` **<required>** - The character to remove from the text
- `text` **<required>** - The text to remove the character from

Example Command: **name**: !shoutout

response: $(1) is one cool streamer, check them out at https://mixer.com/\protect\T1\textdollar(stripchar @ $(1))

output:

```
user:     !shoutout @username
botisimo: @username is one cool streamer, check them out at https://mixer.com/
↪username
```

9.1.32 $(target)

Resolve targeted username from command.

Usage: $(target)

Example Command: **name**: !shoutout

response: Shout out to $(target)!

output:

```
user:     !shoutout @bob
botisimo: Shout out to @bob!
```

9.1.33 $(targetplain)

Resolve targeted username from command without @ symbol.

Usage: $(targetplain)

Example Command: **name**: !follow

response: Go to mixer.com/$(targetplain) and drop a follow!

output:

```
user:     !follow @bob
botisimo: Go to mixer.com/bob and drop a follow!
```

9.1.34 $(total)

Resolves the total numbers of ranked users.

Usage: $(total)

Example Command: **name**: !total

response: Total users: $(total)

output:

```
user:     !total
botisimo: Total users: 300
```

9.1.35 $(twitch)

Show text in Twitch chat only.

Usage: $(twitch <message>)

Arguments:

- `message` **<required>** - The message to display in chat

Example Command: **name**: !example

response: You are a $(twitch super awesome and) cool dude$(twitch !)

twitch output:

```
user:     !example
botisimo: You are a super awesome and cool dude!
```

mixer/youtube/discord output:

```
user:     !example
botisimo: You are a cool dude
```

9.1.36 $(urlencode)

Resolves a url encoded string.

Usage: $(urlencode <text>)

Arguments:

- `text` **<required>** - The text to url encode

Example Command: **name**: !encode

response: $(urlencode please encode me)

output:

```
user:     !encode
botisimo: please%20encode%20me
```

9.1.37 $(userid)

Resolves the command issuer's user id for the platform the command was issued.

Usage: $(userid)

Example Command: **name**: !id

response: Your user id is $(userid)

output:

```
user:     !id
botisimo: Your user id is 837455
```

9.1.38 $(username)

Resolves the command issuer's username as a mention.

Usage: $(username)

Example Command: **name**: !hi

response: Hi, $(username)!

twitch/mixer/youtube output:

```
user:     !hi
botisimo: Hi, @bob!
```

discord output:

```
user:     !hi
botisimo: Hi, <@112338494039>!
```

9.1.39 $(usernameplain)

Resolves the command issuer's username as plain text.

Usage: $(usernameplain)

Example Command: **name**: !hi

response: Hi, $(usernameplain)!

twitch/mixer/youtube output:

```
user:     !hi
botisimo: Hi, bob!
```

discord output:

```
user:     !hi
botisimo: Hi, bob!
```

9.1.40 $(usertype)

Resolves the command issuer's user type (everyone, regular, subscriber, moderator, admin).

Usage: $(usertype)

Example Command: **name**: !type

response: Your user type is $(usertype)

output:

```
user:     !type
botisimo: Your user type is moderator
```

9.1.41 $(winner)

Resolve the username as a mention of a random user currently online in the channel.

Usage: $(winner `[minutes=10] [keyword=]`)

Arguments:

- `minutes` **[optional]** - The number of minutes to look back in chat for the keyword
- `keyword` **[optional]** - The keyword to qualify to be selected as a winner (if no keyword then any text will qualify)

Example Command: **name**: !winner

response: The winners is $(winner 5 enter)!

output:

```
user:     !winner
botisimo: The winner is @bob!
```

9.1.42 $(xp)

Resolves the command issuers xp.

Usage: $(xp)

Example Command: **name**: !xp

response: You have $(xp) XP

output:

```
user:     !xp
botisimo: You have 5,000 XP
```

9.1.43 $(youtube)

Show text in YouTube chat only.

Usage: $(youtube `<message>`)

Arguments:

- `message` **<required>** - The message to display in chat

Example Command: **name**: !example

response: You are a $(youtube super awesome and) cool dude$(youtube !)

youtube output:

```
user:     !example
botisimo: You are a super awesome and cool dude!
```

twitch/mixer/discord output:

```
user:     !example
botisimo: You are a cool dude
```

9.1.44 Advanced Example

This example shows you how to combine the `$(fetch)`, `$(urlencode)`, and `$(query)` response variables to make a !ascii command that translates the user's input into formatted ASCII text for Discord.

Example Command: **command**: !ascii

response: ‘‘‘$(fetch http://artii.herokuapp.com/make?text=\protect\T1\textdollar(urlencode $(query))&font=colossal)‘‘‘

output:

```
user:     !ascii Hello World
botisimo: 888    888          888 888             888       888                 ␣
↪888      888
          888    888          888 888             888   o   888                 ␣
↪888      888
          888    888          888 888             888  d8b  888                 ␣
↪888      888
          8888888888  .d88b.  888 888  .d88b.      888 d888b 888  .d88b.  888d888␣
↪888  .d88888
          888    888 d8P  Y8b 888 888 d88""88b     888d88888b888 d88""88b 888P"  ␣
↪888 d88" 888
          888    888 88888888 888 888 888  888     88888P Y88888 888  888 888    ␣
↪888 888  888
          888    888 Y8b.     888 888 Y88..88P     8888P   Y8888 Y88..88P 888    ␣
↪888 Y88b 888
          888    888  "Y8888  888 888  "Y88P"      888P     Y888  "Y88P"  888    ␣
↪888  "Y88888
```

CHAPTER 10

Response Directives

Response directives are used to make dynamic actions for your custom commands and timers. Directives will be processed after *response variables* and in the order they appear in the response text.

View Response Directives →

10.1 Directives

Response directives are used to make dynamic actions for your custom commands and timers. Directives will be processed after *response variables* and in the order they appear in the response text.

Note: If a directive argument is required it will be wrapped in < > and if the argument is optional it will be wrapped in `[ ]`.

10.1.1 $[cache]

Saves a value temporarily to be used in other commands. Fetch values from the cache using the *cache variable*. The cache is not meant to be used as a long term or permanent place to store any sensitive or important information.

Usage: $[cache `<key>` `<seconds>` `<value>`]

Arguments:

- `key` **<required>** - The key to store the value in the cache
- `seconds` **<required>** - Number of seconds to keep the value in the cache
- `value` **<required>** - The value to store in the cache

Example Command: **name**: !cache

response: $[cache $(username) 60 some value to save for later] value saved

output:

```
user:     !cache
botisimo: value saved
```

10.1.2 $[chain]

Break response into multiple, chained responses.

- Pro members are limited to chaining 3 responses
- Master members are limited to chaining 5 responses
- Guild members are limited to chaining 10 responses

Usage: $[chain]

Example Command: **name**: !chain

response: This is the first response $[chain] This is the second response $[chain] This is a third response

output:

```
user:     !chain
botisimo: This is the first response
botisimo: This is the second response
botisimo: This is a third response
```

10.1.3 $[cooldown]

Directs the bot to enforce a cooldown in seconds for the command.

Usage: $[cooldown `<seconds>`]

Arguments:

- `seconds` **<required>** - Number of seconds it should wait to cooldown

Options:

- `-p, --public` **[optional]** - If present then error message will be sent to public chat (instead of direct message)
- `-s, --silent` **[optional]** - If present then no error message will be sent
- `-g, --global` **[optional]** - If present then cooldown will apply to any user who tries to use the command
- `--error-message <message>` **[optional]** - URL encoded error message

Example Command: **name**: !cool

response: $[cooldown 30 –public] This response will be sent only if 30 seconds has passed

output:

```
user:     !cool
botisimo: This response will be sent only if 30 seconds has passed
user:     !cool
botisimo: You must wait 30 seconds between !cool commands
```

Example Command: **name**: !cool

response: $[cooldown 30 –silent] This response will be sent only if 30 seconds has passed

output:

```
user:     !cool
botisimo: This response will be sent only if 30 seconds has passed
user:     !cool
```

Example Command: **name**: !cool

response: $[cooldown 30 –error-message $(urlencode Cool it dude, you gotta wait 30 seconds to use this command again)] This response will be sent only if 30 seconds has passed

output:

```
user:     !cool
botisimo: This response will be sent only if 30 seconds has passed
user:     !cool
botisimo: Cool it dude, you gotta wait 30 seconds to use this command again
```

10.1.4 $[cost]

Directs the bot to add a currency cost to the command.

Usage: $[cost `<amount>`]

Arguments:

- `amount` **<required>** - Amount of currency it should cost

Example Command: **name**: !cost

response: $[cost 50] You just paid 50 Goldfrags to use this command, ha!

output:

```
user:     !cost
botisimo: You just paid 50 Goldfrags to use this command, ha!
user:     !cost
botisimo: Not enough Goldfrags to use that command 37/50
```

10.1.5 $[delay]

Directs the bot to wait some seconds before responding.

Usage: $[delay `<seconds>`]

Arguments:

- `seconds` **<required>** - The number of seconds it should delay

Example Command: **name**: !delay

response: $[delay 5] This response will be sent after 5 seconds

output:

```
user:     !delay

... wait 5 seconds ...

botisimo: This response will be sent after 5 seconds
```

10.1.6 $[delete]

Directs the bot to delete the command message before responding.

Usage: $[delete]

Example Command: **name**: !delete

response: $[delete] Your command message was deleted

output:

```
user:     !delete
botisimo: Your command message was deleted
```

10.1.7 $[discord role add]

Adds a role to the user issuing the command.

Usage: $[discord role add `<role>` `[user]`]

Arguments:

- `role` **<required>** - URL encoded role name
- `user` **[optional]** - Discord user to add the role to

Example Command: **name**: !awesome

response: $[discord role add $(urlencode My Awesome Role)] You have been added to the My Awesome Role role

output:

```
user:     !awesome
botisimo: You have been added to the My Awesome Role role
```

10.1.8 $[discord role check]

Requires a user to have a discord role.

Usage: $[discord role check `<role>`]

Arguments:

- `role` **<required>** - URL encoded role name

Options:

- `-p, --public` **[optional]** - If present then error message will be sent to public chat (instead of direct message)
- `-s, --silent` **[optional]** - If present then no error message will be sent

- `--error-message <message>` **[optional]** - URL encoded error message

Example Command: name: !expelliarmus

response: $[discord role check $(urlencode My Awesome Role)] (-´).*

output:

```
user:     !expelliarmus
botisimo: (-´).*
```

10.1.9 $[discord role remove]

Removes a role from the user issuing the command.

Usage: $[discord role remove `<role>` `[user]`]

Arguments:

- `role` **<required>** - URL encoded role name
- `user` **[optional]** - Discord user to remove the role from

Example Command: name: !notawesome

response: $[discord role remove $(urlencode My Awesome Role)] You have been removed from the My Awesome Role role

output:

```
user:     !notawesome
botisimo: You have been removed from the My Awesome Role role
```

10.1.10 $[event]

Sends a custom event to the events overlay.

Usage: $[event]

Options:

- `-t, --text <text>` **[optional]** - URL encoded primary text for the event
- `-s, --secondary-text <text>` **[optional]** - URL encoded secondary text for the event
- `-d, --duration <milliseconds>` **[optional]** - Duration in milliseconds to display event
- `-i, --image <url>` **[optional]** - URL for image to display for event
- `-a, --audio <url>` **[optional]** - URL to audio to play for event

Example Command: name: !event

response: $[event -t $(urlencode This is the primary text) -s $(urlencode This is the secondary text) -d 5000 -i https://media2.giphy.com/media/KXtq8oYQrYMIF9Esi7/giphy.gif -a http://soundbible.com/grab.php?id=1817&type=mp3] event sent

output:

```
user:     !event
botisimo: event sent
```

10.1.11 $[give]

Directs the bot to give currency to the user who uses the command or optionally the targeted user.

Usage: $[give <amount> [user]]

Arguments:

- amount **<required>** - The amount of currency to give
- user **[optional]** - The user to give the currency to (if no user then it will give to the user issuing the command)

Example Command: **name**: !give

response: $[give 50] You just received 50 Goldfrags for using this command, woo!

output:

```
user:     !give
botisimo: You just received 50 Goldfrags for using this command, woo!
```

Example Command: **name**: !give

response: $[give 50 $(1)] $(1) just received 50 Goldfrags, woo!

output:

```
user:     !give @username
botisimo: @username just received 50 Goldfrags, woo!
```

10.1.12 $[hours]

Directs the bot to require minimum hours for the command.

Usage: $[hours <hours>]

Arguments:

- hours **<required>** - The number of hours to require the user to have

Options:

- -p, --public **[optional]** - If present then error message will be sent to public chat (instead of direct message)
- -s, --silent **[optional]** - If present then no error message will be sent
- --error-message <message> **[optional]** - URL encoded error message

Example Command: **name**: !special

response: $[hours 10] This response will be sent only if the user has watched the stream for 10 hours

output:

```
user:     !special
botisimo: This response will be sent only if the user has watched the stream for␣
↪10 hours
newbie:   !special
botisimo: You must watch the stream for 10 hours to unlock the !special command
```

Example Command: **name**: !special

response: $[hours 1 –error-message $(urlencode Please enjoy the show a little longer before gaining access to this command)] This response will be sent only if the user has watched the stream for 10 hours

output:

```
user:     !special
botisimo: This response will be sent only if the user has watched the stream for␣
↪1 hours
newbie:   !special
botisimo: Please enjoy the show a little longer before gaining access to this␣
↪command
```

10.1.13 $[norelay]

Directs the bot to not relay the response to this command.

Usage: $[norelay]

Example Command: **name**: !norelay

response: $[norelay] This response will not be sent to other platforms via relay

output:

```
user:     !norelay
botisimo: This response will not be sent to other platforms via relay
```

10.1.14 $[precooldown]

Same as *$[cooldown]* but applied before parsing *command variables*.

10.1.15 $[precost]

Same as *$[cost]* but applied before parsing *command variables*.

10.1.16 $[predelay]

Same as *$[delay]* but applied before parsing *command variables*.

10.1.17 $[prehours]

Same as *$[hours]* but applied before parsing *command variables*.

10.1.18 $[transfer]

Directs the bot to transfer currency from the user who uses the command to the targeted user.

Usage: $[transfer `<amount>` `<user>`]

Arguments:

- `amount` **<required>** - The amount of currency to transfer
- `user` **<required>** - The user to transfer the currency to

Example Command: **name**: !transfer

response: $[transfer 100 $(1)] You just transferred 100 Goldfrags to $(1) for using this command, woo!

output:

```
user:     !transfer @username
botisimo: You just transferred 100 Goldfrags to @username for using this command,␣
↪woo!
user:     !transfer @username
botisimo: Not enough Goldfrags to use that command 37/100
```

10.1.19 $[whisper]

Directs the bot to respond to the command via a direct message.

Usage: $[whisper]

Example Command: **name**: !whisper

response: $[whisper] This response will be sent in a direct message

output:

```
user:             !whisper
botisimo -> user: This response will be sent in a direct message
```

10.1.20 Advanced Example

This example shows you how to combine the `$(pick)` and `$(repeat)` *response variables* with the `$[cost]` and `$[give]` *response directives* to make a !gamble command.

Example Command: **name**: !gamble

response: $[cost 100] $(pick $(repeat 5 $[give 100] You broke even +0 |)$(repeat 5 You lost it all -100 |)$(repeat 5 $[give 150] You got a return on your investment +50 |)$[give 300] You hit the jackpot! +200)

output:

```
user:     !gamble
botisimo: You got a return on your investment +50
user:     !gamble
botisimo: You lost it all -100
```

CHAPTER 11

Extras

11.1 Extras

11.1.1 Alexa Skill

Welcome to the Botisimo skill for Amazon Alexa. If you are new to Amazon Alexa you can learn more about it on Amazon's website.

Install

To install the Botisimo skill for Amazon Alexa you should open the Alexa app on your device. Open the main menu and select the "Skills & Games" option. Search for "Botisimo" and install the Botisimo skill.

Account Linking

The Botisimo skill for Amazon Alexa requires a linked Botisimo account to use many of the features. You can link your account by asking Alexa:

```
Alexa, ask Botisimo to link my account.
```

Alexa will respond with instructions to link your Botisimo account.

Skills

You can use the Botisimo skill for Amazon Alexa to control some features within your Botisimo account.

Note: It may take up to a few seconds for any overlays to sync after using a voice command.

> Music Player

You can control the music player using the following voice commands:

```
Alexa, ask Botisimo to start the music player.
Alexa, ask Botisimo to stop the music player.
Alexa, ask Botisimo to mute the music player.
Alexa, ask Botisimo to unmute the music player.
Alexa, ask Botisimo to skip the current song.
```

> Giveaways

You can control the active giveaway using the following voice commands:

```
Alexa, ask Botisimo to read the active giveaway.
Alexa, ask Botisimo to open the active giveaway.
Alexa, ask Botisimo to close the active giveaway.
Alexa, ask Botisimo to end the active giveaway.
```

> Polls

You can control the active poll using the following voice commands:

```
Alexa, ask Botisimo to read the active poll.
Alexa, ask Botisimo to open the active poll.
Alexa, ask Botisimo to close the active poll.
Alexa, ask Botisimo to end the active poll.
```

> Currency

You can give currency to users in chat:

```
Alexa, ask Botisimo to give five hundred currency to chat
Alexa, ask Botisimo to give five hundred currency to followers
Alexa, ask Botisimo to give one thousand currency to subscribers
```

11.1.2 DyneOps

DyneOps is a fun chat game that will test your memory and put your 10-key numpad typing skills to the test. This page defines some important terms and concepts to understand to be able to play DyneOps.

If you are a streamer and you have Botisimo in your channel, you can enable DyneOps here.

Note: If viewers are not receiving whispers from Botisimo in Twitch, they may need to send it a whisper first to get it working for their account.

Dyne

Dynes are the main aspect of DyneOps. They are found randomly when chatting with other viewers in chat and each viewer can generate a Dyne using the command `!dyne` in chat every 60 minutes. Most Dynes contain *Goldfrags* and some Dynes contain *decryption chips* that are used to more easily unlock more Dynes. To retrieve the loot from a Dyne you must successfully perform the *TranceDecryption* process.

Note: Viewers that are subscribed to the streamer will be able to generate a Dyne every 30 minutes.

Dyne TranceDecryption

TranceDecryption is the process of unlocking a Dyne to get the loot out of it. Once you have found or generated a Dyne Botisimo will notify you:

```
Botisimo: @viewer1 You intercepted a Dyne! Start TranceDecryption now! BudStar !start␣
↪!readme [5 min remain]
```

Once you have been notified of the Dyne you will have 5 minutes to start the decryption before the Dyne corrupts. To start TranceDecryption type `!start` in the chat:

```
viewer1: !start
```

Shortly after you start the TranceDecryption, Botisimo will relay you the Synapse for decryption:

```
Botisimo: @viewer1 Your Synapse is '##(@)*4&0&^7*9*((421('. Remove symbols and give␣
↪me only numbers! [10 sec remain]
```

After starting TranceDecryption you will have 10 seconds to return the numbers from the Synapse in the exact order they are in before the Dyne is corrupted:

```
viewer1: 4079421
```

Botisimo will alert you whether your TranceDecryption was a success or not and fetch your loot if successful:

```
Botisimo: @viewer1 TranceDecryption success! PogChamp PogChamp PogChamp You found 6␣
↪Goldfrags - completed in 7 seconds
```

That's it, now you're ready to TranceDecrypt Dyne!

HYPERDyne

Sometimes a Dyne will appear as a HYPERDyne. HYPERDynes are very similar to regular Dynes but they have more loot and they are longer and you have to sort the numbers from lowest to highest when doing *HYPERDyne TranceDecryption*.

HYPERDyne TranceDecryption

If you have found or generated a HYPERDyne Botisimo will notify you:

```
Botisimo: @viewer1 You intercepted a HYPERDyne! Start TranceDecryption now! BudStar !
↪start !readme [5 min remain]
```

Once you have been notified of the HYPERDyne you will have 5 minutes to start the decryption before the HYPERDyne corrupts. To start TranceDecryption type `!start` in the chat:

```
viewer1: !start
```

Shortly after you start the TranceDecryption, Botisimo will relay you the Synapse for decryption:

```
Botisimo: @viewer1 Your Synapse is '##(@)*4&0&^7*9*((421(&3^8))'. Remove symbols and␣
↪give me the numbers IN ASCENDING ORDER (lowest to highest)! [10 sec remain]
```

After starting TranceDecryption you will have 10 seconds to return the numbers from the Synapse sorted from lowest to highest before the Dyne is corrupted:

```
viewer1: 012344789
```

Botisimo will alert you whether your TranceDecryption was a success or not and fetch your loot if successful:

```
Botisimo: @viewer1 TranceDecryption success! PogChamp PogChamp PogChamp You found 12␣
↪Goldfrags - completed in 9 seconds
```

That's it, now you're ready to TranceDecrypt HYPERDynes!

Goldfrags

Goldfrags are the currency of DyneOps and can be gained randomly as you chat or by unlocking Dynes. Viewers can exchange their currency for items in the *Shop*. If you are a streamer using Botisimo you can change the currency name in your account settings.

Shop

The Shop is where viewers can exchange their currency for items created by the streamer. To view the shop, type `!shop` in the chat:

```
viewer1: !shop
```

Botisimo will display the shop items in chat:

```
Botisimo: Streamer's Shop - 1. Dance (500 Goldfrags) 2. Song Request (250 Goldfrags) -
↪ More info https://botisimo.com/u/streamer/shop
```

Each item will have a corresponding number that you can use to buy it using the `!buy <key>` command:

```
viewer1: !buy 2
```

If you have enough currency to buy the item, Botisimo will announce the purchase in chat to let the streamer know:

```
Botisimo: @streamer TwitchRPG TwitchRPG TwitchRPG @viewer1 has purchased Song Request!
```

If you are a streamer, you can configure your shop here.

Decryption Chips

Decryption Chips assist you during TranceDecryption and can be found by opening Dynes or merging. You can hold a maximum of 3 of each item.

> Noobuster

A noobuster increases the amount of time you have to complete the TranceDecryption by 1. So if you have 1 noobuster then you will have 11 seconds instead of 10 to complete the TranceDecryption.

> Syncswitch

A syncswitch decreases the numbers you have to return for a successful TranceDecryption by 1. So if you have 1 syncswitch and the synapse is `##(@)*4&0&^7*9*((421(` then you only have to return `407942` instead of `4079421` for the TranceDecryption to be successful. if you have 3 syncswitches and the synapse is `##(@)*4&0&^7*9*((421(` then you only have to return `4079` instead of `4079421` for the TranceDecryption to be successful.

> Skullshunt

A set of 3 skullshunts will auto-sort your HyperDyne Synapse before you have to decrypt it. So if you have 3 skullshunts and you find a HyperDyne then you do not need to sort the numbers when doing the TranceDecryption because they will already be sorted for you. example: `##(@)*0&1&^2*3*((447(&8^9))`.

How to acquire Decryption Chips:

- `Noobuster` - Increases decryption time by 1 second
  - Successful TranceDecryption
  - Purchase in shop for 1,000 currency "!buy noobuster"
- `Syncswitch` - Decrease required numbers by 1 during TranceDecryption
  - Successful TranceDecryption
  - Merge 3 Noobusters "!merge noobusters"
  - Purchase in shop for 4,500 currency "!buy syncswitch"
- `Skullshunt` - Set of 3 Skullshunts will auto-sort your Synapse decryption if needed
  - Merge 3 Syncswitches "!merge syncswitches"
  - Purchase in shop for 16,000 currency "!buy skullshunt"

Commands

- `!buy <key>` - Purchase an item from the shop using currency
- `!dyne` - Generate a Dyne (can be used 1 time per 60 minutes, 30 minutes for subscribers)
- `!give <amount> <username> [username=] ...` - Give some currency to 1 or more users (amount should be a number or the word "dyne") **[mods only]**
- `!inventory` - Receive inventory (currency, decryption chips, etc.) info via whisper
- `!leaders` - Display link to leaderboard in chat
- `!merge <item>` - Merge 3 items into a better item (valid items: noobuster, syncswitch, skullshunt)
- `!optin` - Opt in to DyneOps announcements for your user
- `!optout` - Opt out of DyneOps announcements for your user

- `!readme` - Display link to these docs
- `!shop` - Display available shop items in chat
- `!start` - Start a TranceDecryption for a Dyne
- `!stats` - Receive stats via whisper

11.1.3 GraphQL API

Welcome to the Botisimo GraphQL API. If you are new to GraphQL you can learn more about it here https://graphql.org/learn.

GraphQL Endpoint

The Botisimo GraphQL API endpoint is located at:

```
https://botisimo.com/graphql
```

Authentication

The Botisimo GraphQL API requires a Client ID to make authenticated requests. You can obtain your Client ID by visiting your account settings and copying the Client ID found in the "Developer" section. Keep this Client ID a secret because anyone who has it has access to your account.

This Client ID will need to be included as a `Client-ID` header in all GraphQL requests.

Query the API

You can access the Botisimo GraphQL API endpoint using cURL or any other HTTP client. For the following example, make sure to replace **<CLIENT_ID>** with the Client ID you copied from your account settings.

> cURL

To make a query to the Botisimo GraphQL API using cURL, send a POST request with your query as the payload:

```
curl -X POST "https://botisimo.com/graphql" \
-H "Content-Type: application/graphql" \
-H "Client-ID: <CLIENT_ID>" \
-d '
query {
  account {
    id
    email
    createdAt
  }
}'
```

Examples

Here's how you might fetch the Twitch connections on your account and then add currency to a user on one of the Twitch connections

> Fetch Twitch Connection IDs

request:

```
query {
  twitchConnections(pagination: {limit: 1}) {
    edges {
      node {
        id
        twitchName
      }
    }
  }
}
```

response:

```
{
  "data": {
    "twitchConnections": {
      "edges": [
        {
          "node": {
            "id": 1,
            "twitchName": "botisimo"
          }
        }
      ]
    }
  },
}
```

> Fetch Twitch Connection Users

request:

```
query {
  twitchConnectionUsers(twitchConnectionId: 1, pagination: {limit: 1}) {
    edges {
      node {
        id
        currency
        twitchUser {
          name
        }
      }
    }
  }
}
```

response:

```
{
  "data": {
    "twitchConnectionUsers": {
      "edges": [
```

(continues on next page)

(continued from previous page)

```
        {
          "node": {
            "id": 1,
            "currency": 1400,
            "twitchUser": {
              "name": "otothea"
            }
          }
        }
      ]
    }
  }
}
```

> Add Currency to a Twitch Connection User

request:

```
mutation {
  addCurrency(
    platform: TWITCH,
    id: "otothea",
    amount: 100,
    connectionId: 1
  ) {
    ... on TwitchConnectionUser {
      id
      currency
      twitchUser {
        name
      }
    }
  }
}
```

response:

```
{
  "data": {
    "addCurrency": {
      "id": 1,
      "currency": 1500,
      "twitchUser": {
        "name": "otothea"
      }
    }
  }
}
```

11.1.4 Twitch.Center Quote System

Botisimo does not have a quote system, however you can use the `$(fetch)`, `$(query)`, and `$(urlencode)` response variables combined with a third-party api to make your own.

What can it do?

- Allow users to add quotes
- Allow users to delete quotes
- Respond with random quote
- Respond with specific quote

How do I make it?

> Step 1.

Generate three command responses to use in the commands by visiting this URL: https://twitch.center/customapi/quote/generate:

```
$(urlfetch http://twitch.center/customapi/quote?token=********&data=$(querystring))
$(urlfetch http://twitch.center/customapi/addquote?token=********&data=$(querystring))
$(urlfetch http://twitch.center/customapi/delquote?token=********&data=$(querystring))
```

In the generated command responses, there is a token (represented by asterisks * in the example above). **Keep this token a secret!**

> Step 2.

In the command responses that are generated.

- Find `urlfetch` and replace it with `fetch`
- Find `$(querystring)`
  - **Only** in the `/addquote?` command response, replace it with: `$(urlencode $(query))`
  - In the other two, replace it with: `$(query)`

It should look like this now:

```
$(fetch http://twitch.center/customapi/quote?token=********&data=$(query))
$(fetch http://twitch.center/customapi/addquote?token=********&data=$(urlencode
↪$(query)))
$(fetch http://twitch.center/customapi/delquote?token=********&data=$(query))
```

> Step 3.

Create a custom command for each one:

```
name: !quote
response: $(fetch http://twitch.center/customapi/quote?token=********&data=$(query))
```

```
name: !addquote
response: $(fetch http://twitch.center/customapi/addquote?token=********&data=
↪$(urlencode $(query)))
```

```
name: !delquote
response: $(fetch http://twitch.center/customapi/delquote?token=********&data=
↪$(query))
```

Remember to set the appropriate permissions! For example, you might not want everyone to be able to delete quotes.

Note: Special thanks to @Tylegn & @Superlisaa