discord-ext-music

Release 0.3.0

Rahman Yusuf

Nov 16, 2021

CONTENTS

An easy to use music extension for discord.py

CHAPTER

ONE

FEATURES:

- It's easy to use and can be used for complex process.
- Complete playback controls and thread-safe.
- The audio source can be used in discord.py audio library.

CHAPTER

TWO

QUICK USAGE

```python
from discord.ext.commands import Bot
from discord.ext.music import MusicClient, WAVAudio, Track

bot = Bot()

@bot.command()
async def play(ctx):
    voice_user = ctx.message.author.voice
    music_client = await voice_user.channel.connect(cls=MusicClient)
    track = Track(
        WAVAudio('audio.wav'), # AudioSource
        'This is audio' # name
    )
    await music_client.play(track)

bot.run('token')
```

CHAPTER

THREE

LINKS

3.1 Installation

discord-ext-music require Python 3.8 or higher, Python 2.7 or lower are not supported and Python 3.7 or lower are not supported too.

3.1.1 Via PyPI

Note: On linux, before you installing discord-ext-music you need to install these required packages:

- libffi
- libnacl
- python3-dev

You can install discord-ext-music via PyPI:

```
pip install -U discord-ext-music
```

3.1.2 Optional Dependencies

- av for embedded FFmpeg libraries music sources
- miniaudio for Miniaudio-based music sources
- scipy for equalizer
- pydub for equalizer

3.1.3 Installing Optional Dependencies

PyAV

You can do the following command:

```
pip install -U discord-ext-music[av]
```

For more infomation about installing PyAV, please go to here

pyminiaudio

You can do the following command:

Note: On windows, you may need to install Visual Studio with C++ extension (or Visual studio build tools) before installing miniaudio.

```
pip install -U discord-ext-music[miniaudio]
```

scipy and pydub

You can do the following command:

```
pip install -U discord-ext-music[equalizer]
```

3.2 API Reference

3.2.1 Music Clients

MusicClient

class discord.ext.music.**MusicClient**(*client*, *channel*)

Same like `discord.VoiceClient` but with playback controls for music.

Each coroutine functions are thread-safe.

You usually don't create these, you can get it from `discord.VoiceChannel.connect()`

Warning: It is important to add parameter `cls` with value *`MusicClient`* to `discord.VoiceChannel.connect()`, otherwise you wont get these features. For example:

```
# Use this method
music_client = await voice_channel.connect(cls=MusicClient)

# But not this method
music_client = await voice_channel.connect()
```

on_disconnect(*func*)

A decorator that register a callable function as hook when disconnected

Note: The function must be coroutine / async, otherwise it will raise error. The function must be not take any parameters.

Parameters **func** – a callable function (must be a coroutine function)

Raises `TypeError` – The function is not coroutine or async

on_player_error(*func*)
A decorator that register a callable function as hook when player encountered error.

The function can be normal function or coroutine (async) function.

Parameters **func** (Callable[[Exception], Any]) – a callable function (can be a coroutine function)

Raises **TypeError** – The function is not coroutine or async

async disconnect(*, *force=False*)
Disconnects this voice client from voice.

async move_to(*channel*)
Moves you to a different voice channel.

Parameters **channel** (discord.VoiceChannel) – The channel to move to. Must be a voice channel.

async reconnect(*reconnect=True*, *timeout=10*)
Disconnect forcefully from voice channel and connect it again

Parameters

- **reconnect** (bool) – Reconnect when connection is failing.
- **timeout** (float) – The timeout for the connection.

property playlist
Return current playlist

Type *Playlist*

async set_playlist(*playlist*, *stop_player=False*)
Replace current playlist with given playlist

Parameters

- **playlist** (*Playlist*) – A playlist that want to be setted
- **stop_player** (bool) – Stop the player when changing playlist (if playing) and play current track from new playlist.

Raises **TypeError** – "playlist" parameter is not *Playlist*

before_play_next(*func*)
A decorator that register callable function (can be coroutine function) as a pre-play next track

This is useful for checking a streamable url or any type of set up required.

The func callable must accept 1 parameter *Track* that is the next track that want to be played.

Parameters **func** (Callable[[Union[*Track*, None]], Any]) – The callable function to register as pre-play next track.

Raises **TypeError** – Not a callable function

after_play_next(*func*)
A decorator that register callable function (can be coroutine function) as a post-play next track

This is useful for sending announcement or any type of clean up required.

The func callable must accept 1 parameter *Track* that is the next track that want to be played.

Parameters **func** (Callable[[Union[*Track*, None]], Any]) – The callable function to register as post-play next track.

Raises TypeError – Not a callable function

add_track(*track*)
Add a track to playlist

Parameters **track** (*Track*) – Audio Track that we're gonna add to playlist.

async play(*track*)
Play a Track

This function is automatically add track to playlist, even it still playing songs.

Parameters **track** (*Track*) – Audio Track that we're gonna play.

Raises

- *NotConnected* – Not connected to voice
- TypeError – "track" paramater is not *Track*

async play_track_from_pos(*pos*)
Play track from given pos

Parameters **pos** (int) – Track position that we want to play.

Raises

- *NotConnected* – Not connected to voice
- *TrackNotExist* – Given track position is not exist

async stop()
Stop playing audio

Raises *MusicNotPlaying* – Not playing any audio

async pause(*play_silence=True*)
Pauses the audio playing.

Parameters **play_silence** (bool) – if *True* play silence audio.

Raises *MusicNotPlaying* – Not playing any audio

async resume()
Resumes the audio playing.

Raises

- *MusicAlreadyPlaying* – Already playing audio
- *MusicNotPlaying* – Not playing any audio

async seek(*seconds*)
Jump forward to specified durations

Parameters **seconds** (Union[int, float]) – Time to seek in seconds

Raises *MusicNotPlaying* – Not playing any audio

async rewind(*seconds*)
Jump back to specified durations

Parameters **seconds** (Union[int, float]) – Time to rewind in seconds

Raises *MusicNotPlaying* – Not playing any audio

get_stream_durations()
Optional[float]: Get current stream durations in seconds, if playing.

async next_track()
Play next track

Raises

- *NotConnected* – Not connected to voice.
- *NoMoreSongs* – No more songs in playlist.

async previous_track()
Play previous track

Raises

- *NotConnected* – Not connected a voice.
- *NoMoreSongs* – No more songs in playlist.

async remove_track(*track*)
Remove a track and stop the player (if given track same as playing track)

Parameters **track** (*Track*) – A *Track* you want to remove

Raises *TrackNotExist* – Given track is not exist

async remove_track_from_pos(*pos*)
Remove a track from given position and stop the player (if given pos same as playing track pos)

Parameters **pos** (int) – Track position that we want to remove.

Raises *TrackNotExist* – Given track is not exist

async remove_all_tracks()
Remove all tracks and stop the player (if playing)

property equalizer
Return current equalizer music source being played (if playing).

Type Optional[*Equalizer*]

set_equalizer(*equalizer*)
Set equalizer for music source

Note: This will apply equalizer to all music sources to this playlist. music sources that don't support equalizer or volume adjust will be ignored.

Parameters **equalier** (*Equalizer*) – Equalizer that want to be setted in music source

Raises

- *MusicNotPlaying* – Not playing any audio
- *MusicClientException* – current music source does not support equalizer

property volume
Return current volume music source being played (if playing).

Type Optional[float]

set_volume(*volume*)
Set volume for music source for this channel.

Note: This will apply volume to all music sources to this playlist. music sources that don't support equalizer or volume adjust will be ignored.

Parameters **volume** (float) – Volume that want to be setted in music source

Raises

- *MusicNotPlaying* – Not playing any audio
- *MusicClientException* – current music source does not support volume adjust

property source
The audio source being played, if playing.

Type Optional[*MusicSource*]

property track
The audio track being played, if playing.

Type Optional[*Track*]

3.2.2 Tracks

Track

class discord.ext.music.**Track**(*source*, *name*, *url=None*, *thumbnail=None*, ***kwargs*)
A audio track containing audio source, name, url, thumbnail

Parameters

- **source** (*MusicSource*) – The audio source of this track
- **name** (str) – Name of this track
- **url** (str) – Webpage url of this track
- **thumbnail** (str) – Valid thumbnail url of this track

source
The audio source of this track

Type *MusicSource*

name
Name of this track

Type str

url
Webpage url of this track

Type str

thumbnail
Valid thumbnail url of this track

Type str

3.2.3 Playlists

Playlist

class discord.ext.music.**Playlist**
a class representing playlist for tracks

This class is thread-safe.

property pos
Return current position of the playlist.

add_track(*track*)
Add a track

Parameters **track** (*Track*) – The audio track that we want to put in playlist.

jump_to_pos(*pos*)
Change playlist pos and return *Track* from given position

Parameters **pos** (int) – Track position that we want jump to

Raises ***TrackNotExist*** – Given track position is not exist

Returns The audio track from given position

Return type *Track*

remove_track(*track*)
Remove a track

Parameters **track** (*Track*) – The audio track that we want to remove from playlist.

Raises ***TrackNotExist*** – Given track is not exist

remove_track_from_pos(*pos*)
Remove a track from given position

Parameters **pos** (int) – Track position that we want remove from playlist

Raises ***TrackNotExist*** – Given track position is not exist

remove_all_tracks()
Remove all tracks from playlist

reset_pos_tracks()
Reset current position playlist

is_track_exist(*track*)
Check if given track is exist in this playlist

Parameters **track** (*Track*) – The audio track that we want to check

Returns *True* if exist, or *False* if not exist

Return type bool

get_all_tracks()
Get all tracks in this playlist

Returns All tracks in playlist

Return type List[*Track*]

get_current_track()
Get current track in current position

Returns The current track in current position

Return type *Track*

get_pos_from_track(*track*)
Get a position track from given track

Parameters **track** (*Track*) – Track that we want to retrieve from playlist

Raises ***TrackNotExist*** – Given track position is not exist

Returns The track position from given track

Return type *Track*

get_track_from_pos(*pos*)
Get a track from given position

Parameters **pos** (int) – Track position that we want retrieve from playlist

Raises ***TrackNotExist*** – Given track position is not exist

Returns The track from given position

Return type *Track*

get_next_track()
Get next track

Returns The next track of this playlist

Return type Union[*Track*, None]

get_previous_track()
Get previous track

Returns The previous track of this playlist

Return type Union[*Track*, None]

3.2.4 Equalizers

class discord.ext.music.**Equalizer**
Represent a equalizer for converting audio.

Sub-classes must implement this.

property stream
Return the audio stream

Type io.BufferedIOBase

setup(*stream*)
Initialize audio stream to Equalizer

> **Warning:** This is important to call it first, before equalizing audio.

Parameters **stream** (io.BufferedIOBase) – The audio stream that we want to equalize.

read()
Read audio stream and return equalized audio data.

close()
Close audio stream and do some cleanup to the equalizer.

pydub Equalizer

class discord.ext.music.pydubEqualizer(*freqs=None*)
A pydub equalizer for Signed-PCM codec

The audio specifications must be 16-bit 48KHz

Warning: You must have scipy and pydub installed, otherwise you will get error.

Parameters **freqs** (Optional[List[dict]]) – a list containing dict, each dict has frequency (in Hz) and gain (in dB) inside it. For example, [{"freq": 20, "gain": 20}, ...] You cannot add same frequencys, if you try to add it, it will raise pydubError.

Raises **pydubError** – pydub and scipy is not installed

add_frequency(*freq*, *gain*)
Add a frequency

Parameters

- **freq** (int) – The frequency that want to add
- **gain** (Union[int, float]) – The gain frequency

Raises **ValueError** – given frequency is already exist

remove_frequency(*freq*)
Remove a frequency

Parameters **freq** (int) – The frequency that want to add

Raises **ValueError** – given frequency is not exist

set_gain(*freq*, *gain*)
Set frequency gain in dB,

Parameters

- **freq** (int) – The frequency want to increase the gain
- **gain** (Union[int, float]) – The value want to increase or lower

Raises **ValueError** – given frequency is not exist

read()
Read audio stream and return equalized audio data.

class discord.ext.music.pydubSubwooferEqualizer(*volume=0.5*)
An easy to use *pydubEqualizer* for subwoofer

The frequency is 60Hz.

Parameters **volume** (float) – Set initial volume as float percent. For example, 0.5 for 50% and 1.75 for 175%.

property volume
The subwoofer volume in float numbers

This property can also be used to change the subwoofer volume.

Type Optional[float]

set_gain(*dB*)
Set frequency gain in dB.

setup(*stream*)
Initialize audio stream to Equalizer

> **Warning:** This is important to call it first, before equalizing audio.

Parameters **stream** (io.BufferedIOBase) – The audio stream that we want to equalize.

read()
Read audio stream and return equalized audio data.

3.2.5 Music sources

Legacy music sources

class discord.ext.music.**MusicSource**
same like discord.AudioSource, but its have seek, rewind, equalizer and volume built-in to AudioSource

read()
Reads 20ms worth of audio.

Subclasses must implement this.

If the audio is complete, then returning an empty bytes to signal this is the way to do so.

If is_opus() method returns True, then it must return 20ms worth of Opus encoded audio. Otherwise, it must be 20ms worth of 16-bit 48KHz stereo PCM, which is about 3,840 bytes per frame (20ms worth of audio).

Returns A bytes like object that represents the PCM or Opus data.

Return type bytes

recreate()
Recreate audio source, useful for next and previous playback

seekable()
Check if this source support seek() and rewind() operations or not

return bool

seek(*seconds*)
Jump forward to specified durations

Parameters **seconds** (float) – The duration in seconds that we want to jump forward

Raises ***IllegalSeek*** – current stream doesn't support seek() operations

rewind(*seconds*)
Jump back to specified durations

Parameters **seconds** (float) – The duration in seconds that we want to jump backward

Raises ***IllegalSeek*** – current stream doesn't support seek() operations

get_stream_durations()
Get current stream durations in seconds

Returns The current stream duration in seconds

Return type float

property volume
Return current volume

Type Optional[float]

set_volume(*volume*)
Set volume in float percentage. Set to None to disable volume adjust.

For example, 0.5 = 50%, 1.5 = 150%

Parameters **volume** (float) – Set volume to music source

property equalizer
Return current equalizer

Type Optional[*Equalizer*]

set_equalizer(*equalizer=None*)
Set a *Equalizer* to MusicSource.

Parameters **equalizer** (*Equalizer*) – Set equalizer to music source

class discord.ext.music.**RawPCMAudio**(*stream*, *volume=None*)
Represents raw 16-bit 48KHz stereo PCM audio source.

Parameters

- **stream** (io.BufferedIOBase) – file-like object
- **volume** (float or NoneType) – Set initial volume for AudioSource

stream
A file-like object that reads byte data representing raw PCM.

Type io.BufferedIOBase

read()
Reads 20ms worth of audio.

Subclasses must implement this.

If the audio is complete, then returning an empty bytes to signal this is the way to do so.

If is_opus() method returns True, then it must return 20ms worth of Opus encoded audio. Otherwise, it must be 20ms worth of 16-bit 48KHz stereo PCM, which is about 3,840 bytes per frame (20ms worth of audio).

Returns A bytes like object that represents the PCM or Opus data.

Return type bytes

cleanup()
Called when clean-up is needed to be done.

Useful for clearing buffer data or processes after it is done playing audio.

recreate()
Recreate audio source, useful for next and previous playback

seekable()
Check if this source support seek() and rewind() operations or not

return bool

get_stream_durations()
Get current stream durations in seconds

Returns The current stream duration in seconds

Return type float

set_volume(*volume*)
Set volume in float percentage. Set to None to disable volume adjust.

For example, 0.5 = 50%, 1.5 = 150%

Parameters **volume** (float) – Set volume to music source

set_equalizer(*eq=None*)
Set a *Equalizer* to MusicSource.

Parameters **equalizer** (*Equalizer*) – Set equalizer to music source

seek(*seconds*)
Jump forward to specified durations

Parameters **seconds** (float) – The duration in seconds that we want to jump forward

Raises ***IllegalSeek*** – current stream doesn't support seek() operations

rewind(*seconds*)
Jump back to specified durations

Parameters **seconds** (float) – The duration in seconds that we want to jump backward

Raises ***IllegalSeek*** – current stream doesn't support seek() operations

class discord.ext.music.**WAVAudio**(*file*, *volume=None*, ***kwargs*)
Represents WAV audio stream

file: Union[str, io.BufferedIOBase] valid file location or file-like object.

volume: float or NoneType Set initial volume for AudioSource

kwargs: These parameters will be passed in *RawPCMAudio*

Miniaudio music sources

class discord.ext.music.**Miniaudio**(*stream*, *volume*)
Representing miniaudio-based audio source

Audio formats that miniaudio can play:

- MP3
- FLAC
- Vorbis
- WAV

> **Warning:** You must have miniaudio installed, otherwise it didn't work.

Raises *MiniaudioError* – miniaudio not installed

class discord.ext.music.**MP3toPCMAudio**(*data*, *volume=0.5*, ***kwargs*)
Represents miniaudio-based mp3 to PCM audio source.

This audio source will convert mp3 to pcm format (16-bit 48KHz).

Note: When you initiate this class, the audio data will automatically coverted to pcm. This may cause all asynchronous process is blocked by this process. If you want to avoid this, use *MP3toPCMAudio.from_data* or *MP3toPCMAudio.from_file*.

Parameters

- **data** (bytes) – MP3 bytes data
- **volume** (float) – Set initial volume
- **kwargs** – These parameters will be passed in *RawPCMAudio*

stream
A file-like object that reads byte data representing raw PCM.

Type io.BufferedIOBase

Raises *InvalidMP3* – The audio data is not mp3 format

async classmethod from_data(*data*, *volume=0.5*)
Asynchronously convert mp3 data to pcm.

Parameters

- **data** (bytes) – MP3 bytes data
- **volume** (float) – Set initial volume, default to *0.5*

async classmethod from_file(*filename*, *volume=0.5*)
Asynchronously convert mp3 data to pcm.

Parameters

- **filename** (str) – MP3 File
- **volume** (float) – Set initial volume, default to *0.5*

seekable()
Check if this source support seek() and rewind() operations or not

return bool

class discord.ext.music.**FLACtoPCMAudio**(*data*, *volume=0.5*, ***kwargs*)
Represents miniaudio-based flac to PCM audio source.

This audio source will convert flac to pcm format (16-bit 48KHz).

Note: When you initiate this class, the audio data will automatically coverted to pcm. This may cause all asynchronous process is blocked by this process. If you want to avoid this, use *FLACtoPCMAudio.from_data* or *FLACtoPCMAudio.from_file*

Parameters

- **data** (bytes) – FLAC bytes data
- **volume** (float) – Set initial volume
- **kwargs** – These parameters will be passed in *RawPCMAudio*

stream
A file-like object that reads byte data representing raw PCM.

Type io.BufferedIOBase

Raises ***InvalidFLAC*** – The audio data is not flac format

async classmethod from_data(*data*, *volume=0.5*)
Asynchronously convert flac data to pcm.

Parameters

- **data** (bytes) – FLAC bytes data
- **volume** (float) – Set initial volume, default to *0.5*

async classmethod from_file(*filename*, *volume=0.5*)
Asynchronously convert flac data to pcm.

Parameters

- **filename** (str) – FLAC File
- **volume** (float) – Set initial volume, default to *0.5*

seekable()
Check if this source support seek() and rewind() operations or not

return bool

class discord.ext.music.**VorbistoPCMAudio**(*data*, *volume=0.5*, ***kwargs*)
Represents miniaudio-based vorbis to PCM audio source.

This audio source will convert vorbis to pcm format (16-bit 48KHz).

Note: When you initiate this class, the audio data will automatically coverted to pcm. This may cause all asynchronous process is blocked by this process. If you want to avoid this, use *VorbistoPCMAudio.from_data* or *VorbistoPCMAudio.from_file*

Parameters

- **data** (bytes) – Vorbis bytes data
- **volume** (float) – Set initial volume
- **kwargs** – These parameters will be passed in *RawPCMAudio*

stream
A file-like object that reads byte data representing raw PCM.

Type io.BufferedIOBase

Raises ***InvalidVorbis*** – The audio data is not vorbis codec

async classmethod from_data(*data*, *volume=0.5*)
Asynchronously convert vorbis data to pcm.

Parameters

- **data** (bytes) – Vorbis bytes data
- **volume** (float) – Set initial volume, default to *0.5*

async classmethod from_file(*filename*, *volume=0.5*)
Asynchronously convert vorbis data to pcm.

Parameters

- **filename** (str) – Vorbis File
- **volume** (float) – Set initial volume, default to *0.5*

seekable()
Check if this source support seek() and rewind() operations or not

return bool

class discord.ext.music.**WAVtoPCMAudio**(*data*, *volume=0.5*, ***kwargs*)
Represents miniaudio-based WAV to PCM audio source.

This audio source will convert wav to pcm format (16-bit 48KHz).

Note: When you initiate this class, the audio data will automatically coverted to pcm. This may cause all asynchronous process is blocked by this process. If you want to avoid this, use *WAVtoPCMAudio.from_data* or *WAVtoPCMAudio.from_file*

Parameters

- **data** (bytes) – WAV bytes data
- **volume** (float) – Set initial volume
- **kwargs** – These parameters will be passed in *RawPCMAudio*

stream
A file-like object that reads byte data representing raw PCM.

Type io.BufferedIOBase

Raises ***InvalidWAV*** – The audio data is not WAV format

async classmethod from_data(*data*, *volume=0.5*)
Asynchronously convert WAV data to pcm.

Parameters

- **data** (bytes) – WAV bytes data
- **volume** (float) – Set initial volume, default to *0.5*

async classmethod from_file(*filename*, *volume=0.5*)
Asynchronously convert WAV data to pcm.

Parameters

- **filename** (str) – WAV File

- **volume** (float) – Set initial volume, default to *0.5*

seekable()
Check if this source support seek() and rewind() operations or not

return bool

PyAV / Embedded FFmpeg libraries music sources

class discord.ext.music.**LibAVOpusAudio**(*url_or_file*)
Represents embedded FFmpeg-based Opus audio source.

> **Warning:** You must have av installed, otherwise it didn't work.

Parameters **url_or_file** (str) – Valid URL or file location

url
Valid URL or file location

Type str

stream
a file-like object that returning ogg opus encoded data

Type io.RawIOBase

Raises **LibAVError** – Something happened when opening connection stream url.

recreate()
Recreate audio source, useful for next and previous playback

is_opus()
Checks if the audio source is already encoded in Opus.

set_volume(*volume*)
Set volume in float percentage. Set to None to disable volume adjust.

For example, 0.5 = 50%, 1.5 = 150%

Parameters **volume** (float) – Set volume to music source

set_equalizer(*equalizer*)
Set a *Equalizer* to MusicSource.

Parameters **equalizer** (*Equalizer*) – Set equalizer to music source

read()
Reads 20ms worth of audio.

Subclasses must implement this.

If the audio is complete, then returning an empty bytes to signal this is the way to do so.

If *is_opus()* method returns True, then it must return 20ms worth of Opus encoded audio. Otherwise, it must be 20ms worth of 16-bit 48KHz stereo PCM, which is about 3,840 bytes per frame (20ms worth of audio).

Returns A bytes like object that represents the PCM or Opus data.

Return type bytes

get_stream_durations()
Get current stream durations in seconds

Returns The current stream duration in seconds

Return type float

seekable()
Check if this source support seek() and rewind() operations or not

return bool

seek(*seconds*)
Jump forward to specified durations

Parameters **seconds** (float) – The duration in seconds that we want to jump forward

Raises *IllegalSeek* – current stream doesn't support seek() operations

rewind(*seconds*)
Jump back to specified durations

Parameters **seconds** (float) – The duration in seconds that we want to jump backward

Raises *IllegalSeek* – current stream doesn't support seek() operations

cleanup()
Called when clean-up is needed to be done.

Useful for clearing buffer data or processes after it is done playing audio.

class discord.ext.music.**LibAVPCMAudio**(*url_or_file*)
Represents embedded FFmpeg-based audio source producing pcm packets.

> **Warning:** You must have av installed, otherwise it didn't work.

Parameters **url_or_file** (str) – Valid URL or file location

url
Valid URL or file location

Type str

stream
a file-like object that returning pcm packets.

Type io.RawIOBase

Raises **LibAVError** – Something happened when opening connection stream url.

recreate()
Recreate audio source, useful for next and previous playback

get_stream_durations()
Get current stream durations in seconds

Returns The current stream duration in seconds

Return type float

seek(*seconds*)
Jump forward to specified durations

Parameters **seconds** (*float*) – The duration in seconds that we want to jump forward

Raises ***IllegalSeek*** – current stream doesn't support seek() operations

rewind(*seconds*)
Jump back to specified durations

Parameters **seconds** (*float*) – The duration in seconds that we want to jump backward

Raises ***IllegalSeek*** – current stream doesn't support seek() operations

cleanup()
Called when clean-up is needed to be done.

Useful for clearing buffer data or processes after it is done playing audio.

3.2.6 Exceptions

exception discord.ext.music.**EqualizerError**
Raised when something happened in Equalizer class

exception discord.ext.music.**IllegalSeek**
Raised when MusicSource trying to seek when stream doesn't support seek() operations

exception discord.ext.music.**InvalidMP3**
Raised when audio data is not mp3 format

exception discord.ext.music.**InvalidFLAC**
Raised when audio data is not flac format

exception discord.ext.music.**InvalidVorbis**
Raised when audio data is not vorbis codec

exception discord.ext.music.**InvalidWAV**
Raised when audio data is not WAV format

exception discord.ext.music.**MiniaudioError**
Raised when something happened in miniaudio module

exception discord.ext.music.**StreamHTTPError**
Raised when something happened in audio HTTP stream

exception discord.ext.music.**TrackNotExist**
Raised when track is trying to be removed while it not exist

exception discord.ext.music.**MusicClientException**
Base exception for MusicClient class

exception discord.ext.music.**MusicNotPlaying**
Music is not playing

exception discord.ext.music.**MusicAlreadyPlaying**
Music is already playing

exception discord.ext.music.**NoMoreSongs**
No more songs in playlist

exception discord.ext.music.**NotConnected**
Not connected to voice

3.3 Changelog

3.3.1 v0.3.0

Improvements

- Optimized PyAV music sources stream.
- Optimized pydub equalizer.

Fix bugs

- **Fixed All of miniaudio music sources malfunctioning when decoding, for more information about what sources are affecte**
  - *MP3toPCMAudio.from_file()*
  - *MP3toPCMAudio.from_data()*
  - *FLACtoPCMAudio.from_file()*
  - *FLACtoPCMAudio.from_data()*
  - *VorbistoPCMAudio.from_file()*
  - *VorbistoPCMAudio.from_data()*
  - *WAVtoPCMAudio.from_file()*
  - *WAVtoPCMAudio.from_data()*
- Fixed *LibAVOpusAudio.seek()* error caused by unconverted floating numbers.
- Fixed a deadlock when stopping audio caused by *MusicClient.stop()* is waiting audio player thread to exit.
- Fixed after function is called in audio player when *MusicClient.stop()* is called.
- Fixed WAVAudio malfunctioning if given stream is valid wav.
- Fixed after function called when voice is disconnected

New features

- Added new opus encoder using PyAV library.
- Added equalizer support for PyAV-based music source for *LibAVPCMAudio*
- Added *Playlist.get_pos_from_track()* to retrieve track position from given track
- Added *MusicSource.volume* property to return current volume.
- Added *MusicSource.equalizer* property to return current equalizer.
- Added *MusicClient.playlist* property to retrieve current playlist in *MusicClient*
- Added *MusicClient.set_playlist()* to set new playlist.
- Added *MusicClient.volume* property to return current volume from music client.
- Added *MusicClient.set_volume()* to set volume music source in music client.
- Added *MusicClient.equalizer* property to return current equalizer from music client.

- Added *MusicClient.set_equalizer()* to set equalizer in music client.
- Added hook *MusicClient.on_disconnect()* on MusicClient.
- Added hook *MusicClient.on_player_error()* on MusicClient.

Breaking changes

- Replaced `Equalizer` and `SubwooferEqualizer` with *pydubEqualizer* and *pydubSubwooferEqualizer*.
- Removed module `discord.ext.music.voice_source.av.encoder` as it unused because new opus encoder.
- Removed `LibAVAudio` as it unused.
- Replaced `LibAVError` with *StreamHTTPError*
- **Removed `MusicClient.register_after_callback()`, replaced with:**
  - *MusicClient.before_play_next()*
  - *MusicClient.after_play_next()*
- Removed `Track.stream_url` attribute and `stream_url` parameter
- Player error handling now are called from *MusicClient.on_player_error()*

3.3.2 v0.2.0

New features

- Added API Documentation
- Added *WAVtoPCMAudio* miniaudio-based music sources.
- Added Keyword-arguments only in *Track*, all Keyword-arguments will be setted in *Track* class attributes.

Fixed bugs

- Fix unhandled error "cannot allocate memory in static TLS block" when importing `discord.ext.music.voice_source.av` on ARM-based CPU. **NOTE:** The error is not fixed automatically inside python, because you need to fix it manually outside python. The solution is given inside the error.

Removals

- Removed `WorkerError` exception class as it unused.
- Removed `ConverterError` exception class as it unused.

Improvements

- Improved how next song playback system work in *MusicClient*

Breaking changes

- Changed module name from `discord.ext.music.voice_source.pyav` to `discord.ext.music.voice_source.av`
- module `discord.ext.music.equalizer` no longer raising error when you try to import it.
- module `discord.ext.music.voice_source.miniaudio` no longer raising error when you try to import it.
- module `discord.ext.music.voice_source.av` no longer raising error when you try to import it.
- Now `PCMEqualizer` and `SubwooferPCMEqualizer` will raise error when you dont have the required modules and try to create it.
- Changed name `WavAudio` to *WAVAudio*
- Calling *LibAVOpusAudio.recreate()* now will close the stream before re-creating it.
- Now `LibAVAudioStream.seek()` method will directly seek to the stream instead of re-creating it.

3.3.3 v0.1.0

This is First release of discord-ext-music

INDEX

T

U

V

W