xdcc-dl

Release 3.1.0

Feb 22, 2019

CONTENTS

Contents:

CHAPTER

ONE

XDCC_DL PACKAGE

1.1 Subpackages

1.1.1 xdcc_dl.entities package

Submodules

xdcc_dl.entities.IrcServer module

class xdcc_dl.entities.IrcServer.**IrcServer**(*server_address: str*, *server_port: int = 6667*)
Bases: object

Class that models an IRC server

__init__(*server_address: str*, *server_port: int = 6667*)
Initializes the Server's information

Parameters

- **server_address** – the address of the IRC Server
- **server_port** – the port of the server, which defaults to 6667 and can usually safely stay that way

get_address() → str

Returns the server address

get_port() → int

Returns the server port

xdcc_dl.entities.User module

class xdcc_dl.entities.User.**User**(*username: str = 'random'*)
Bases: object

Models an IRC user

__init__(*username: str = 'random'*)
Initializes the User

Parameters **username** – the user's username. If left empty, or the string 'random' is passed, a random username consisting only of ASCII characters will be generated as the username. An empty string will also result in a random username

static generate_random_username(*length: int = 10*) → str
Generates a random username of given length

Parameters **length** – The length of the username

Returns The random username

get_name() → str

Returns The user's username

xdcc_dl.entities.XDCCPack module

class xdcc_dl.entities.XDCCPack.**XDCCPack**(*server: xdcc_dl.entities.IrcServer.IrcServer*, *bot: str*, *packnumber: int*)
Bases: object

Class that models an XDCC Pack

__init__(*server: xdcc_dl.entities.IrcServer.IrcServer*, *bot: str*, *packnumber: int*)
Initializes an XDCC object. It contains all the necessary information for joining the correct IRC server and channel and sending the download request to the correct bot, then storing the received file in the predetermined location. If the destination is a directory, the file will be stored in the directory with the default file name, if not the file will be saved at the destination exactly. The file extension will stay as in the original filename

Parameters

- **server** – The Sever to be used by the XDCC Bot
- **bot** – The bot serving the file
- **packnumber** – The packnumber of the desired file

classmethod from_xdcc_message(*xdcc_message: str*, *destination_directory: str = '/home/gitlab-runner/builds/18226848/0/namibsun/python/xdcc-dl/doc/sphinx'*, *server: str = 'irc.rizon.net'*) → list
Generates XDCC Packs from an xdcc message of the form "/msg <bot> xdcc send #<packnumber>[-<packnumber>]"

Parameters

- **xdcc_message** – the XDCC message to parse
- **destination_directory** – the destination directory of the file
- **server** – the server to use, defaults to irc.rizon.net for simplicity's sake

Returns The generated XDCC Packs in a list

get_bot() → str

Returns The bot

get_filename() → str

Returns The currently set filename

get_filepath() → str

Returns The full destination file path

get_packnumber() → int

Returns the pack number

get_request_message(*full: bool = False*) → str
Generates an xdcc send message to be sent to the bot to initiate the XDCC connection

Parameters **full** – Returns the entire message string, including the bot's name, as seen on packlist sites

Returns The generated message string

get_server() → xdcc_dl.entities.IrcServer.IrcServer

Returns The server

get_size() → int

Returns The currently set file size

is_filename_valid(*filename: str*) → bool
Checks if a filename is the same as the original filename, if one was set previously. This is used internally by the IRC Bot to check if a file that was offered to the bot actually matches the file we want to download.

Parameters **filename** – The file name to check

Returns True, if the names match, or no original filename was set, otherwise False

set_directory(*directory: str*)
Sets the target directory of the XDCC PAck

Parameters **directory** – the target directory

Returns None

set_filename(*filename: str*, *override: bool = False*)
Sets the filename (or only the file extension) of the target file

Parameters

- **filename** – the filename as provided by the XDCC bot
- **override** – Overrides the current filename

Returns None

set_original_filename(*filename: str*)
Sets the 'original' filename, a.k.a the name of the actual file to download. This is a method that should only be used by the pack searchers to add filename checks during the download.

Parameters **filename** – The original filename as found by the PackSearcher

Returns None

set_size(*size: int*)
Sets the file size of the XDCC pack

Parameters **size** – the size of the pack

Returns None

Module contents

1.1.2 xdcc_dl.logging package

Submodules

xdcc_dl.logging.Logger module

class xdcc_dl.logging.Logger.**Logger**
Bases: object

Class that handles log and print calls in the bot

debug(*message: str*, *back: <colorama.ansi.AnsiBack object at 0x7f65e9265710> = '\x1b[47m'*, *fore: <colorama.ansi.AnsiFore object at 0x7f65e9265668> = '\x1b[30m'*, *end: str = '\n'*)
Logs a message at the DEBUG level :param message: The message to log :param back: The background color to print :param fore: The foreground color to print :param end: Characters to append to the string (Default newline) :return: None

error(*message: str*, *back: <colorama.ansi.AnsiBack object at 0x7f65e9265710> = '\x1b[41m'*, *fore: <colorama.ansi.AnsiFore object at 0x7f65e9265668> = '\x1b[34m'*, *end: str = '\n'*)
Logs a message at the ERROR level :param message: The message to log :param back: The background color to print :param fore: The foreground color to print :param end: Characters to append to the string (Default newline) :return: None

info(*message: str*, *back: <colorama.ansi.AnsiBack object at 0x7f65e9265710> = '\x1b[40m'*, *fore: <colorama.ansi.AnsiFore object at 0x7f65e9265668> = '\x1b[32m'*, *end: str = '\n'*)
Logs a message at the INFO level :param message: The message to log :param back: The background color to print :param fore: The foreground color to print :param end: Characters to append to the string (Default newline) :return: None

last_end = '\n'
Keeps track of the last character to be printed

log(*message: str, level: Optional[int], back: <colorama.ansi.AnsiBack object at 0x7f65e9265710> = '\x1b[40m', fore: <colorama.ansi.AnsiFore object at 0x7f65e9265668> = '\x1b[32m', end: str = '\n'*)
Logs a message at the specified logging level :param message: The message to log :param level: The level at which to log the message.

If set as None, will always be printed.

Parameters

- **back** – The background color to print
- **fore** – The foreground color to print
- **end** – Characters to append to the string (Default newline)

Returns None

logging_level = 30
The logging level to display

print(*message: str*, *back: <colorama.ansi.AnsiBack object at 0x7f65e9265710> = '\x1b[40m'*, *fore: <colorama.ansi.AnsiFore object at 0x7f65e9265668> = '\x1b[32m'*, *end: str = '\n'*)
Prints a message :param message: The message to print :param back: The background color to print :param fore: The foreground color to print :param end: Characters to append to the string (Default newline) :return: None

warning(*message: str*, *back: <colorama.ansi.AnsiBack object at 0x7f65e9265710> = '\x1b[43m'*, *fore: <colorama.ansi.AnsiFore object at 0x7f65e9265668> = '\x1b[34m'*, *end: str = '\n'*)
Logs a message at the WARNING level :param message: The message to log :param back: The background color to print :param fore: The foreground color to print :param end: Characters to append to the string (Default newline) :return: None

Module contents

1.1.3 xdcc_dl.pack_search package

Subpackages

xdcc_dl.pack_search.procedures package

Submodules

xdcc_dl.pack_search.procedures.horriblesubs module

xdcc_dl.pack_search.procedures.horriblesubs.**find_horriblesubs_packs**(*search_phrase: str*) → List[xdcc_dl.entities.XDCCPack.
Method that conducts the xdcc pack search for xdcc.horriblesubs.info

Returns the search results as a list of XDCCPack objects

xdcc_dl.pack_search.procedures.horriblesubs.**parse_result**(*result: str*) → Dict[str, str]
Turns the weird horriblesubs response syntax into a useable dictionary :param result: The result to parse :return: The result as a dictionary

xdcc_dl.pack_search.procedures.ixirc module

xdcc_dl.pack_search.procedures.ixirc.**find_ixirc_packs**(*search_phrase: str*) → List[xdcc_dl.entities.XDCCPack.XDCCPack]
Searches for XDCC Packs matching the specified search string on ixirc.com

Parameters **search_phrase** – The search phrase to search for

Returns The list of found XDCC Packs

xdcc_dl.pack_search.procedures.ixirc.**get_page_results**(*page_content: bs4.BeautifulSoup*) → List[xdcc_dl.entities.XDCCPack.XDCCPack]
This parses a single ixIRC page to find all search results from that page

Parameters **page_content** – The Beautifulsoup-parsed content of the page

Returns A list of XDCC Packs on that page

xdcc_dl.pack_search.procedures.nibl module

xdcc_dl.pack_search.procedures.nibl.**find_nibl_packs**(*search_phrase: str*) → List[xdcc_dl.entities.XDCCPack.XDCCPack]
Searches for XDCC Packs matching the specified search string on nibl.co.uk

Parameters **search_phrase** – The search phrase to search for

Returns The list of found XDCC Packs

Module contents

Submodules

xdcc_dl.pack_search.SearchEngine module

class xdcc_dl.pack_search.SearchEngine.**SearchEngine**(*name: str*, *procedure: callable*)

Bases: object

An XDCC Pack Search Engine

__init__(*name: str*, *procedure: callable*)

Initializes the Search Engine :param name: The name of the search engine :param procedure: A function that performs the XDCC pack search

search(*term: str*) → List[xdcc_dl.entities.XDCCPack.XDCCPack]

Searches for packs that match the provided term :param term: The term to search for :return: A list of XDCC Packs

class xdcc_dl.pack_search.SearchEngine.**SearchEngineType**

Bases: enum.Enum

The different implemented search engines

HORRIBLESUBS = <xdcc_dl.pack_search.SearchEngine.SearchEngine object>

IXIRC = <xdcc_dl.pack_search.SearchEngine.SearchEngine object>

NIBL = <xdcc_dl.pack_search.SearchEngine.SearchEngine object>

choices = <bound method SearchEngineType.choices of <enum 'SearchEngineType'>>

resolve = <bound method SearchEngineType.resolve of <enum 'SearchEngineType'>>

Module contents

1.1.4 xdcc_dl.xdcc package

Submodules

xdcc_dl.xdcc.XDCCClient module

class xdcc_dl.xdcc.XDCCClient.**XDCCClient**(*pack: xdcc_dl.entities.XDCCPack.XDCCPack*, *retry: bool = False*)

Bases: irc.client.SimpleIRCClient

IRC Client that can download an XDCC pack

__init__(*pack: xdcc_dl.entities.XDCCPack.XDCCPack*, *retry: bool = False*)

Initializes the XDCC IRC client :param pack: The pack to downloadX :param retry: Set to true for retried downloads.

download() → str

Downloads the pack :return: The path to the downloaded file

download_limit = -1
Maximum speed of the download in bytes/s If set to -1, will be unlimited

event = 'nickcollision'

on_action(*c*, *e*)

on_adminemail(*c*, *e*)

on_adminloc1(*c*, *e*)

on_adminloc2(*c*, *e*)

on_adminme(*c*, *e*)

on_alreadyregistered(*c*, *e*)

on_away(*c*, *e*)

on_badchanmask(*c*, *e*)

on_badchannelkey(*c*, *e*)

on_banlist(*c*, *e*)

on_banlistfull(*c*, *e*)

on_bannedfromchan(*c*, *e*)

on_cannotknock(*c*, *e*)

on_cannotsendtochan(*c*, *e*)

on_cantkillserver(*c*, *e*)

on_channelcreate(*c*, *e*)

on_channelisfull(*c*, *e*)

on_channelmodeis(*c*, *e*)

on_chanoprivsneeded(*c*, *e*)

on_closeend(*c*, *e*)

on_closing(*c*, *e*)

on_created(*c*, *e*)

on_ctcp(*conn: irc.client.ServerConnection*, *event: irc.client.Event*)
The 'ctcp' event indicates that a CTCP message was received. The downloader receives a CTCP from the bot to initialize the XDCC file transfer. Handles DCC ACCEPT and SEND messages. Other DCC messages will result in a raised InvalidCTCP exception. DCC ACCEPT will only occur when a resume request was sent successfully. DCC SEND will occur when the bot offers a file. :param conn: The connection :param event: The 'ctcp' event :return: None :raise InvalidCTCPException: In case no valid DCC message was received

on_ctcpreply(*c*, *e*)

on_currenttopic(*c*, *e*)

on_dcc_connect(*c*, *e*)

on_dcc_disconnect(*_: irc.client.ServerConnection*, *__: irc.client.Event*)
The 'dccmsg' event contains the file data. :param _: The connection :param __: The 'dccmsg' event :return: None

on_dccmsg(*_: irc.client.ServerConnection*, *event: irc.client.Event*)
The 'dccmsg' event contains the file data. :param _: The connection :param event: The 'dccmsg' event :return: None

on_disconnect(*c*, *e*)

on_endofbanlist(*c*, *e*)

on_endofexceptlist(*c*, *e*)

on_endofinfo(*c*, *e*)

on_endofinvitelist(*c*, *e*)

on_endoflinks(*c*, *e*)

on_endofmotd(*c*, *e*)

on_endofnames(*c*, *e*)

on_endofservices(*c*, *e*)

on_endofstats(*c*, *e*)

on_endoftrace(*c*, *e*)

on_endofusers(*c*, *e*)

on_endofwho(*c*, *e*)

on_endofwhois(*conn: irc.client.ServerConnection*, *_: irc.client.Event*)
The 'endofwhois' event indicates the end of a whois request. This manually calls on_join in case the bot has not joined any channels. :param conn: The connection :param _: The 'endofwhois' event :return: None

on_endofwhowas(*c*, *e*)

on_erroneusnickname(*c*, *e*)

on_error(*_: irc.client.ServerConnection*, *event: irc.client.Event*)
Sometimes, the connection gives an error which may prove fatal for the download process. A possible cause of error events is a banned IP address. :param _: The connection :param event: The error event :return: None

on_exceptlist(*c*, *e*)

on_featurelist(*c*, *e*)

on_fileerror(*c*, *e*)

on_info(*c*, *e*)

on_infostart(*c*, *e*)

on_invalidcapcmd(*c*, *e*)

on_invite(*c*, *e*)

on_invitelist(*c*, *e*)

on_inviteonlychan(*c*, *e*)

on_inviting(*c*, *e*)

on_ison(*c*, *e*)

on_join(*conn: irc.client.ServerConnection*, *event: irc.client.Event*)
The 'join' event indicates that a channel was successfully joined. The first on_join call will send a message to the bot that requests the initialization of the XDCC file transfer. :param conn: The connection :param event: The 'join' event :return: None

on_keyset(*c*, *e*)

on_kick(*c*, *e*)

on_killdone(*c*, *e*)

on_links(*c*, *e*)

on_list(*c*, *e*)

on_listend(*c*, *e*)

on_liststart(*c*, *e*)

on_luserchannels(*c*, *e*)

on_luserclient(*c*, *e*)

on_luserconns(*c*, *e*)

on_luserme(*c*, *e*)

on_luserop(*c*, *e*)

on_luserunknown(*c*, *e*)

on_mode(*c*, *e*)

on_motd(*c*, *e*)

on_motd2(*c*, *e*)

on_motdstart(*c*, *e*)

on_myinfo(*c*, *e*)

on_myportis(*c*, *e*)

on_n_global(*c*, *e*)

on_n_local(*c*, *e*)

on_namreply(*c*, *e*)

on_needmoreparams(*c*, *e*)

on_nick(*c*, *e*)

on_nickcollision(*c*, *e*)

on_nicknameinuse(*c*, *e*)

on_noadmininfo(*c*, *e*)

on_nochanmodes(*c*, *e*)

on_nologin(*c*, *e*)

on_nomotd(*c*, *e*)

on_none(*c*, *e*)

on_nonicknamegiven(*c*, *e*)

on_nooperhost(*c*, *e*)

on_noorigin(*c*, *e*)

on_nopermforhost(*c*, *e*)

on_noprivileges(*c*, *e*)

on_norecipient(*c*, *e*)

on_noservicehost(*c*, *e*)

on_nosuchchannel(*c*, *e*)

on_nosuchnick(*c*, *e*)

on_nosuchserver(*c*, *e*)

on_notexttosend(*c*, *e*)

on_notonchannel(*c*, *e*)

on_notopic(*c*, *e*)

on_notoplevel(*c*, *e*)

on_notregistered(*c*, *e*)

on_nousers(*c*, *e*)

on_nowaway(*c*, *e*)

on_part(*c*, *e*)

on_passwdmismatch(*c*, *e*)

on_ping(*c*, *e*)

on_pong(*c*, *e*)

on_privmsg(*c*, *e*)

on_privnotice(*_: irc.client.ServerConnection*, *event: irc.client.Event*)

Handles privnotices. Bots sometimes send privnotices when a pack was already requested or the user is put into a queue.

If the privnotice indicates that a pack was already requested, the downloader will pause for 60 seconds

Parameters

- **_** – The connection
- **event** – The privnotice event

Returns None

on_pubmsg(*c*, *e*)

on_pubnotice(*c*, *e*)

on_quit(*c*, *e*)

on_rehashing(*c*, *e*)

on_restricted(*c*, *e*)

on_service(*c*, *e*)

on_serviceinfo(*c*, *e*)

on_servlist(*c*, *e*)

on_servlistend(*c*, *e*)

on_statscline(*c*, *e*)

on_statscommands(*c*, *e*)

on_statshline(*c*, *e*)

on_statsiline(*c*, *e*)

on_statskline(*c*, *e*)

on_statslinkinfo(*c*, *e*)

on_statslline(*c*, *e*)

on_statsnline(*c*, *e*)

on_statsoline(*c*, *e*)

on_statsqline(*c*, *e*)

on_statsuptime(*c*, *e*)

on_statsyline(*c*, *e*)

on_summondisabled(*c*, *e*)

on_summoning(*c*, *e*)

on_time(*c*, *e*)

on_toomanychannels(*c*, *e*)

on_toomanytargets(*c*, *e*)

on_topic(*c*, *e*)

on_topicinfo(*c*, *e*)

on_traceclass(*c*, *e*)

on_traceconnecting(*c*, *e*)

on_tracehandshake(*c*, *e*)

on_tracelink(*c*, *e*)

on_tracelog(*c*, *e*)

on_tracenewtype(*c*, *e*)

on_traceoperator(*c*, *e*)

on_tracereconnect(*c*, *e*)

on_traceserver(*c*, *e*)

on_traceservice(*c*, *e*)

on_traceunknown(*c*, *e*)

on_traceuser(*c*, *e*)

on_tryagain(*c*, *e*)

on_umodeis(*c*, *e*)

on_umodeunknownflag(*c*, *e*)

on_unavailresource(*c*, *e*)

on_unaway(*c*, *e*)

on_uniqopprivsneeded(*c*, *e*)

on_unknowncommand(*c*, *e*)

on_unknownmode(*c*, *e*)

on_userhost(*c*, *e*)

on_usernotinchannel(*c*, *e*)

on_useronchannel(*c*, *e*)

on_users(*c*, *e*)

on_usersdisabled(*c*, *e*)

on_usersdontmatch(*c*, *e*)

on_usersstart(*c*, *e*)

on_version(*c*, *e*)

on_wasnosuchnick(*c*, *e*)

on_welcome(*conn: irc.client.ServerConnection*, *_: irc.client.Event*)
The 'welcome' event indicates a successful connection to the server Sends a whois command to find the bot on the server :param conn: The connection :param _: The 'welcome' event :return: None

on_whoisaccount(*c*, *e*)

on_whoischannels(*conn: irc.client.ServerConnection*, *event: irc.client.Event*)
The 'whoischannels' event indicates that a whois request has found channels that the bot is a part of. Channels that the bot has joined will be joined as well. :param conn: The connection :param event: The 'whoischannels' event :return: None

on_whoischanop(*c*, *e*)

on_whoisidle(*c*, *e*)

on_whoisoperator(*c*, *e*)

on_whoisserver(*c*, *e*)

on_whoisuser(*c*, *e*)

on_whoreply(*c*, *e*)

on_whospcrpl(*c*, *e*)

on_whowasuser(*c*, *e*)

on_wildtoplevel(*c*, *e*)

on_yourebannedcreep(*c*, *e*)

on_youreoper(*c*, *e*)

on_yourhost(*c*, *e*)

on_youwillbebanned(*c*, *e*)

xdcc_dl.xdcc.exceptions module

exception xdcc_dl.xdcc.exceptions.**AlreadyDownloadedException**
Bases: Exception

Exception thrown when a file was already downloaded

exception xdcc_dl.xdcc.exceptions.**DownloadCompleted**
Bases: Exception

Exception thrown once the download has been completed

exception xdcc_dl.xdcc.exceptions.**DownloadIncomplete**
Bases: Exception

Exception thrown if a download did not complete

exception xdcc_dl.xdcc.exceptions.**InvalidCTCPException**
Bases: Exception

Exception thrown when an invalid CTCP DCC message type is received

exception xdcc_dl.xdcc.exceptions.**PackAlreadyRequested**
Bases: Exception

Exception raised if a pack was already requested

exception xdcc_dl.xdcc.exceptions.**UnrecoverableError**
Bases: Exception

Exception raised when an unrecoverable error occurs

Module contents

xdcc_dl.xdcc.**download_packs**(*packs: List[xdcc_dl.entities.XDCCPack.XDCCPack]*)
Downloads a list of XDCC Packs :param packs: The packs to download :return: None

1.2 Submodules

1.3 xdcc_dl.helper module

xdcc_dl.helper.**prepare_packs**(*packs: List[xdcc_dl.entities.XDCCPack.XDCCPack], location: Optional[str]*)
Prepares the output path of a list of packs based on a location string :param location: The location at which to save the packs. :param packs: The packs to prepare :return: None

xdcc_dl.helper.**set_logging_level**(*quiet: bool*, *verbose: bool*, *debug: bool*)
Sets the logging level based on a combination of flags If all flags are False, the logging level will be set to WARNING :param quiet: If set to True, will set logging to ERROR :param verbose: If set to True, will set logging to INFO :param debug: If set to True, will set logging to DEBUG :return: None

xdcc_dl.helper.**set_throttle_value**(*throttle_string: str*)
Sets the throttle value of the XDCC Client globally based on a string in the form <Bytes><|k|m|g> (kilo, mega, giga) :param throttle_string: The string to parse :return: None

1.4 Module contents

CHAPTER

TWO

XDCC_DL

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

X

INDEX

Symbols

A

C

D

E

F

G

H

I

P

R

S

U

W

X