
Whalesong Documentation

Release 0.9.1

alfred82santa

May 29, 2019

Contents

1	Binary Requirements	3
1.1	Using Firefox	3
1.2	Using Chromium	3
1.3	Development requirements	3
2	Installation	5
2.1	Using Firefox	5
2.2	Using Chromium	5
3	Table of contents	7
3.1	Features	7
3.1.1	Whatsapp features	7
3.2	Browser backends	8
3.2.1	Firefox backend	9
3.2.1.1	Pros	9
3.2.1.2	Contras	9
3.2.1.3	How to use	9
3.2.2	Chromium backend	9
3.2.2.1	Pros	9
3.2.2.2	Contras	10
3.2.2.3	How to use	10
3.2.3	Other backends	10
3.3	Examples	10
3.3.1	Raw driver examples	10
3.3.1.1	State monitor	10
3.3.1.2	Get contacts	11
3.3.1.3	Get chats	11
3.3.1.4	Get messages	11
3.3.2	Whatsapp driver examples	11
3.3.2.1	State monitor	11
3.3.2.2	Get contacts	12
3.3.2.3	Get chats	12
3.3.2.4	Get messages	12
3.3.2.5	Minibot	12
3.3.2.6	Get stickers	15
3.3.2.7	Monitor presences	15
3.3.2.8	Get statuses	15

3.3.2.9	Get live locations	15
3.4	API reference	16
3.4.1	Whalesong Service	16
3.4.2	Whalesong Driver	18
3.4.2.1	Base driver	18
3.4.2.2	Firefox driver	18
3.4.2.3	Chromium driver	19
3.4.3	Result types	19
3.4.4	Base Models	21
3.4.5	Base Managers	22
3.4.6	Contact references	26
3.4.6.1	Managers	26
3.4.6.2	Models	30
3.4.7	Chat references	33
3.4.7.1	Managers	33
3.4.7.2	Models	44
3.4.8	Message references	47
3.4.8.1	Managers	47
3.4.8.2	Models	57
3.4.9	Group metadata references	132
3.4.9.1	Managers	132
3.4.9.2	Models	140
3.4.10	Wap references	141
3.4.10.1	Managers	141
3.4.10.2	Models	143
3.4.11	Sticker pack references	143
3.4.11.1	Managers	143
3.4.11.2	Models	151
3.4.12	Connection references	152
3.4.12.1	Managers	152
3.4.12.2	Models	154
3.4.13	Stream references	156
3.4.13.1	Managers	156
3.4.13.2	Models	158
3.4.14	Presence references	160
3.4.14.1	Managers	160
3.4.14.2	Models	167
3.4.15	Profile pictures references	169
3.4.15.1	Managers	169
3.4.15.2	Models	173
3.4.16	Status references	173
3.4.16.1	Managers	173
3.4.16.2	Models	177
3.4.17	Live location references	177
3.4.17.1	Managers	177
3.4.17.2	Models	185
3.4.18	Mute references	186
3.4.18.1	Managers	186
3.4.18.2	Models	190
3.4.19	Display information references	191
3.4.19.1	Managers	191
3.4.19.2	Models	192
3.4.20	Status V3 references	195
3.4.20.1	Managers	195

3.4.20.2	Models	199
3.4.21	Storage references	200
3.4.21.1	Managers	200
3.4.22	Whalesong Errors	201
3.4.23	Firefox Profile	201
3.5	How to develop	201
3.5.1	Development requirements	201
3.5.2	Install library requirements	202
3.5.3	Build Javascript scriptlet	202
3.5.4	Beautify code	202
3.6	Changelog	202
3.6.1	Version 0.9.1 (Work in progress)	202
3.6.2	Version 0.9.0	202
3.6.3	Version 0.8.4	202
3.6.4	Version 0.8.2	203
3.6.5	Version 0.8.1	203
3.6.6	Version 0.8.0	203
3.6.7	Version 0.7.2	203
3.6.8	Version 0.7.1	203
3.6.9	Version 0.7.0	203
3.6.10	Version 0.6.0	204
3.6.11	Version 0.5.3	204
3.6.12	Version 0.5.2	205
3.6.13	Version 0.5.1	205
3.6.14	Version 0.5.0	205
3.6.15	Version 0.4.4	205
3.6.16	Version 0.4.0	205
3.6.17	Version 0.3.0	205
3.6.18	Version 0.2.0	206
4	Related projects	207
5	Indices and tables	209
6	Legal	211
	Python Module Index	213

Whalesong is an asyncio python library to manage WebApps remotely. Currently WhatsappWeb is implemented

Warning: NEW VERSION 0.7.0

With new version some new requirements have been defined and some API change have been committed.

Version 0.7.0 introduces new driver for Chromium browser. It has required some changes on base driver's API. It should not affect anyone. But be aware.

Warning: NEW VERSION 0.7.

Stream model now uses enumerations. It means that if you check stream states you must be aware that they are not a string anymore, now they are enumerations. Look at [documentation](#).

Binary Requirements

- Python 3.6+

1.1 Using Firefox

- Geckodriver
- Firefox 50+

1.2 Using Chromium

- Execute this after installation:

```
$ pypeteer-install
```

1.3 Development requirements

- node (only for development)
- npm (only for development)
- make (only for development)

CHAPTER 2

Installation

Starting with version *0.7.0* Whalesong introduces new **browsers backends**. It means you must decide which browser backend you want to use. So, depending on it:

Tip: You could install both with no problems.

2.1 Using Firefox

```
$ pip3 install whalesong[firefox]
```

2.2 Using Chromium

```
$ pip3 install whalesong[chromium]
```

If you choose Chromium backend, you should execute this after installation:

```
$ pyppeteer-install
```

It will download a patched Chromium distribution needed for that backend.

3.1 Features

- Non blocking driver. It can do more than one thing at same time.
- Monitor feature. Python code could react to Javascript event.
- Iterator feature. Large item list are passed to Python as async iterators.
- Persistent browser profile.
- Easy way to build new features.
- AmpersandJS/BackboneJS models and collection monitor.
- AmpersandJS/BackboneJS field monitor.
- Monitor localStorage.
- Take screenshots (page and css elements).
- Powered by AsyncIO.
- Firefox backend.
- Chromium backend.

3.1.1 Whatsapp features

- Monitor connection state.
- Monitor QR changes.
- Monitor stream state.
- It's able to refresh QR.
- It's able to take over session.

- List chats.
- List contacts.
- List messages
- Monitor new messages.
- Monitor unread messages.
- Monitor message acks.
- Monitor new contacts.
- Monitor new chats.
- Send text messages.
- Reply messages.
- Send VCard messages.
- Send Media (image/video/pdf) messages.
- Send seen to chats.
- Create groups.
- Manage groups (add/kick/promote/demote people).
- Auto load link previews.
- Allow to check whether a contact is registered on Whatsapp.
- Load and send stickers (even with a quoted message).
- Presence monitor.
- Profile and group's picture management.
- (Un)Pin and (un)archive chats.
- Report spam.
- (Un)Star messages.
- Status management.
- Pushname management.
- Display information management.
- Live location management.
- Mutes management.
- It's able to revoke messages (delete messages for others).
- List and manage WhatsApp Status version 3 (Stories).

3.2 Browser backends

Whalesong uses a browser backend in order to execute a WebApp (currently only WhatsAppWeb). All backends have an interface to manage webviews and that is what Whalesong uses to manage applications. That interface changes depending on browser, but there is a standard interface called [WebDriver](#). Firsts Whalesong versions used to use a [Selenium](#) library in order to communicate with Firefox browser. This backend is the default one for now, **but it will be deprecated in next versions and removed in version 1.0.**

3.2.1 Firefox backend

It was the first backend developed. It uses [Selenium](#) library and [Geckodriver](#) to communicate with Firefox process. It is the most tested (the most, but not well).

3.2.1.1 Pros

- Tested (more or less).
- Use Firefox (I prefer it in front Chromium).

3.2.1.2 Contras

- Selenium is a huge library. It is wonderful for what it was created, but not for Whalesong.
- Selenium is a synchronous library. It is a problem, because Whalesong is an asynchronous library. It means, Whalesong creates a thread pool to communicate with Selenium.
- We need Geckodriver. Firefox does not implement Webdriver protocol by itself. Firefox has its own protocol called [Marionette](#). So Geckodriver is used as proxy between Webdriver protocol and Marionette protocol.
- As Webdriver is a synchronous protocol. Whalesong must poll continuously to Firefox in order to get new events. There is no way to make Firefox notify Whalesong proactively. It means, Whalesong is polling for new results continuously, with an interval (by default 0.5 seconds).

Note: There is only one way for the Firefox backend to survive:

Drop Selenium, drop Geckodriver, implement Marionette protocol directly and implement a notification system (I'm not sure it is possible in Marionette, currently).

3.2.1.3 How to use

Currently Firefox backend is the default one. But it will change on next versions. So, in order to ensure you use Firefox backend you must instantiate Whalesong with proper driver.

```
from whalesong import Whalesong
from whalesong.driver_firefox import WhalesongDriver

driver = WhalesongDriver(profile='/path/to/your/firefox/profile')
whaleapp = Whalesong(driver=driver)
```

3.2.2 Chromium backend

It is the new one. It is implemented using [Pyppetter](#) which is inspired on [Puppeteer](#) (a *node* library to control Chromium headless, mainly, for testing). It uses [Devtools protocol](#) in order to communicate with the browser. It is an asynchronous protocol over websocket.

3.2.2.1 Pros

- No more polling! When a result is produced it will send proactively to Whalesong. No more *sleeps*, no more waitings.

- Small footprint (at least, it looks like, even being Chromium).
- No extra processes (No more Geckodriver).
- Application mode. No tabs, no URL field.
- No huge libraries (No more Selenium).

3.2.2.2 Contrasts

- It is Chromium. It uses Blink: over-vitaminized Webkit render. A memory eater.
- Currently Pyppeter has a bug. It makes to loose connection after 20 seconds. It is resolved in [miyakogi/pyppeteer/#160](#) but is not approved yet (some test errors).
- It uses a patched Chromium version from Puppeteer. Whalesong needs this patch because it uses *Runtime.addBinding* command. It is not available in regular stable version. So, you must [download it](#) before to use the backend.
- Poorly tested.
- There is a bug in Chromium under Wayland. It makes impossible to get WhastappWeb QR when Chromium is executed with window (no headless).

3.2.2.3 How to use

In order to use Chromium backend you must inject Chromium driver to Whalesong service constructor.

```
from whalesong import Whalesong
from whalesong.driver_chromium import WhalesongDriver

driver = WhalesongDriver(profile='/path/to/your/chromium/profile')
whaleapp = Whalesong(driver=driver)
```

3.2.3 Other backends

No, there are no other backends. But I'm thinking about other possibilities:

- Create a small footprint webview using Webkit directly (GTK or QT ways are not an option).
- Create a small footprint webview using Servo directly (I want to learn rust language).

Of course, any contribution will be welcome (so much).

3.3 Examples

3.3.1 Raw driver examples

3.3.1.1 State monitor

It monitors Stream state, Connection state and localStorage. It prints any change on them. It takes a page screenshot on each stream state change.

It tries to own WhatsappWeb session, it means that it will restore session if you open a new session in other browser.

On the other hand, if session is not started, it will renew QR automatically when it expires. It will save QR image each time it changes.

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/driver/statemonitor.py
```

3.3.1.2 Get contacts

It prints contact list.

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/driver/getcontacts.py
```

3.3.1.3 Get chats

It prints chat list.

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/driver/getchats.py
```

3.3.1.4 Get messages

It prints message list and monitors it. So, if new messages are received it will print them. It monitors message acknowledgments and prints them, as well.

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/driver/getmessages.py
```

3.3.2 Whatsapp driver examples

3.3.2.1 State monitor

It monitors Stream state, Connection state and localStorage. It prints any change on them. It takes a page screenshot on each stream state change.

It tries to own WhatsappWeb session, it means that it will restore session if you open a new session in other browser.

On the other hand, if session is not started, it will renew QR automatically when it expires. It will save QR image each time it changes.

Firefox backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/statemonitor.py
```

Chromium backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/statemonitor-chromium.py
```

3.3.2.2 Get contacts

It prints contact list.

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getcontacts.py
```

3.3.2.3 Get chats

It prints chat list.

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getchats.py
```

3.3.2.4 Get messages

It prints message list and monitors it. So, if new messages are received it will print them. It monitors message acknowledgments and prints them, as well.

It stores files and thumbnails from media messages.

Firefox backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getmessages.py
```

Chromium backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getmessages-chromium.py
```

3.3.2.5 Minibot

Mini bot to test features.

Firefox backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/minibot.py
```

Chromium backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/minibot-chromium.py
```

Minibot implements some test features:

Echo

When a contact sends `/echo [text]` it replies with `[text]`.

Example

```
/echo Hello!
```

Contact

When a contact sends `/contact [contactID]` it replies with the contact in VCard format.

Example

```
/contact 495555555555
```

Download

When a contact sends `/download [url]` it replies with content pointed by URL (image, pdf, video).

Example

```
/download http://example.com/image.jpg
```

Send

When a contact sends `/send [contactId] [text]` it will send `[text]` to `[contactId]`. `[contactId]` must be a phone number with country prefix: 495555555555 where 49 is Germany prefix.

Example

```
/send 495555555555 Hello!
```

Link

When a contact sends `/link [text]` it replies with `[text]`. It's very similar to `/echo`, but never quote original message and if there was a link it will try to get link preview and attach it.

Example

```
/link https://www.google.com
```

Exist

When a contact sends `/exist [contactId]`, it will return whether a phone number is registered on Whatsapp.

Example

```
/exist 495555555555
```

List sticker packs

When a contact sends */sticker list*, it will send all sticker pack names with main image attached.

Example

```
/sticker list
```

Send random sticker

When a contact sends */sticker [stickerPackName]*, it will send a random sticker from sticker pack with name *[sticker-PackName]*.

Example

```
/sticker Cuppy
```

Set bot status

When a contact sends */status [newStatus]*, it will change its own status to *[newStatus]*.

Example

```
/status I'm a bot
```

Set bot pushname

When a contact sends */pushname [name]*, it will change its own pushname to *[name]*.

Example

```
/pushname I'm a bot
```

3.3.2.6 Get stickers

It fetch all sticker packs installed. It fetch all sticker for each sticker pack. And, finally, it downloads all sticker images.

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getstickers.py
```

3.3.2.7 Monitor presences

It monitors user presences. It prints any change on them.

Firefox backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/presencemonitor.py
```

Chromium backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/presencemonitor-chromium.py
```

3.3.2.8 Get statuses

It will get all unread statuses, download the media content to the *output* folder and send a read command.

Firefox backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getstatusv3.py
```

Chromium backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getstatusv3-chromium.py
```

3.3.2.9 Get live locations

It will get all active live locations and will monitor them.

Firefox backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getlivelocations.py
```

Chromium backend

```
$ PYTHONPATH=.:$PYTHONPATH python3 examples/getlivelocations-chromium.py
```

3.4 API reference

3.4.1 Whalesong Service

class whalesong.**Whalesong** (*profile=None, *, autostart=True, headless=False, extra_options=None, driver=None, loop=None, **kwargs*)

Bases: *whalesong.managers.BaseManager*

Whalesong service.

The main Whalesong manager.

Parameters

- **profile** (*Optional[str]*) – Path to firefox profile.
- **autostart** (*bool*) – Whether driver must start immediately.
- **headless** (*bool*) – Whether browser must be started with headless flag. In production environments it should be set to *True*.
- **extra_options** (*Optional[dict]*) – Extra parametres for browser commandline.
- **loop** (*Optional[AbstractEventLoop]*) – Event loop.
- **loadstyles** (*bool*) – Whether CSS styles must be loaded. It is need in order to get QR image. (Only for Firefox)
- **interval** (*float*) – Polling responses interval in seconds. Default 0.5 seconds. (Only for Firefox)

storage

StorageManager

Manager for local storage.

stream

StreamManager

Manager for stream object.

conn

ConnManager

Manager for connection object

contacts

ContactCollectionManager

Manager for contact collection.

chats

ChatCollectionManager

Manager for chat collection.

messages

MessageCollectionManager

Manager for messages collection.

wap

WapManager

Manager for wap object.

sticker_packs*StickerPackCollectionManager*

Manager for sticker pack collection.

status*StatusCollectionManager*

Manager for status collection.

display_info*DisplayInfoManager*

Manager for display information.

live_locations*LiveLocationCollectionManager*

Manager for live locations collection.

mutes*MuteCollectionManager*

Manager for mutes collection.

status_v3*StatusV3CollectionManager*

Manager for statuses version 3 (Stories) collection.

loop

Event loop.

Returns Event loop.**await start ()**

Start Whalesong service.

await stop ()

Stop Whalesong service.

await wait_until_stop ()

Wait until Whalesong service is stopped.

await screenshot ()

Take a screenshot of whole page.

Return type `BytesIO`**Returns** It returns a stream of a PNG image.**await qr ()**

Take a screenshot of QR.

Return type `BytesIO`**Returns** It returns a stream of a PNG image.**stop_monitor (monitor)**

Stop a given monitor.

Parameters **monitor** (*MonitorResult*[~T]) – Monitor object to stop.**Return type** *Result*[None]**await cancel_iterators ()**

Cancel all iterators.

await `download_file(url)`
Download a file by URL

Parameters `url` (`str`) – URL to the file

Return type `BytesIO`

Returns It returns a stream.

3.4.2 Whalesong Driver

3.4.2.1 Base driver

```
class whalesong.driver.BaseWhalesongDriver(*,      autostart=True,      headless=False,
                                           extra_options=None,      logger=None,
                                           loop=None)

Bases: abc.ABC

await start_driver()

abstractmethod await connect()

abstractmethod await refresh()

await run_scriptlet()

await screenshot()
    Return type BytesIO

await screenshot_element(css_selector)
    Return type BytesIO

execute_command(command, params=None, *, result_class=None)

process_result_sync(result)

await process_result(result)

await close()

await cancel_iterators()

await cancel_monitors()

await download_file(url)
    Return type BytesIO

await wait_until_stop()
```

3.4.2.2 Firefox driver

```
class whalesong.driver_firefox.WholesongDriver(profile=None, *, autostart=True,
                                              headless=False, interval=0.5, load-
                                              styles=False, extra_options=None,
                                              logger=None, loop=None)

Bases: whalesong.driver.BaseWhalesongDriver

free_port()
    Determines a free port using sockets.

await connect()
```



```
await refresh()
await poll()
```

3.4.2.3 Chromium driver

```
class whalesong.driver_chromium.WholesongDriver (profile=None, *, autostart=True, head-
less=False, extra_options=None, log-
ger=None, loop=None)

Bases: whalesong.driver.BaseWholesongDriver

await connect()
await refresh()
```

3.4.3 Result types

```
class whalesong.results.BaseResultMixin (result_id, *, fn_map=None)
Bases: abc.ABC, typingAwaitable

Base result mixin.

result_id: str
    Result unique identifier.

fn_map: Callable[[dict], Union[BaseModel, dict]]
    Mapping function used to map result.
```

Parameters

- **result_id** (`str`) – Result unique identifier.
- **fn_map** (`Optional[Callable[[dict], ~T]]`) – Mapping function used to map result.

map (`data`)

Maps data from browser to an object if `fn_map` function is defined.

Parameters `data` (`dict`) – Data from browser.

Return type `~T`

Returns Mapped object.

```
await set_final_result (data)
```

```
await set_error_result (data)
```

```
class whalesong.results.Result (result_id, *, fn_map=None)
Bases: whalesong.results.BaseResultMixin, asyncio.Future
```

Result of command. It is a subtype of `asyncio.Future`, so in order to get result value you must *await* it.

```
value = await result
```

Parameters

- **result_id** (`str`) – Result unique identifier.
- **fn_map** (`Optional[Callable[[dict], ~T]]`) – Mapping function used to map result.

```
class whalesong.results.BasePartialResult (result_id, *, fn_map=None)
    Bases: whalesong.results.BaseResultMixin, typing.AsyncIterable

    await set_partial_result (data)

    cancel ()
```

```
class whalesong.results.BaseIteratorResult (result_id, *, fn_map=None)
    Bases: whalesong.results.BasePartialResult

    Base iterable result.
```

```
class whalesong.results.IteratorResult (result_id, *, fn_map=None)
    Bases: whalesong.results.BaseIteratorResult

    Iterator result. It is used as result of command which returns a list of object.
```

Warning: It is an async iterator.

How to use

```
async for item in result_iterator:
    print (item)
```

map (data)

Maps data from browser to an object if *fn_map* function is defined.

Parameters *data* – Data from browser.

Return type ~T

Returns Mapped object.

```
class whalesong.results.MonitorResult (result_id, *, fn_map=None)
    Bases: whalesong.results.BaseIteratorResult
```

Monitor result. It is used as result of monitor command. It is a infinite iterator. Each change on object or field it is monitoring will be a new item on iterator.

Warning: It is an async iterator.

How to use

```
async for item_changed in monitor:
    print (item_changed)
```

add_callback (fn)

Add a callback to be called each time object or field change.

Parameters *fn* (*Callable*[[~T], *Awaitable*[Any]]) – Callback function

start_monitor ()

Starts automatic monitor iteration. Useful when callback functions are defined.

```
class whalesong.results.ResultManager
    Bases: object
```

get_next_id ()

Return type *str*

```

remove_result (result_id)
    Return type Union[Result[~T], IteratorResult[~T], MonitorResult[~T], None]
request_result (result_class)
    Return type ~TypeResult
request_final_result ()
    Return type Result[~T]
request_iterator_result ()
    Return type IteratorResult[~T]
request_monitor_result ()
    Return type MonitorResult[~T]
cancel_result (result_id)
cancel_all ()
await set_final_result (result_id, data)
await set_error_result (result_id, data)
await set_partial_result (result_id, data)
get_iterators ()
    Return type List[IteratorResult[~T]]
get_monitors ()
    Return type List[MonitorResult[~T]]

```

3.4.4 Base Models

```

class whalesong.models.Base64Field (name=None, alias=None, getter=None, setter=None,
                                     read_only=False, default=None, doc=None)
    Bases: dirty_models.fields.BytesField

```

Byte field which allows to set base64 string data.

```

convert_value (value)
    Converts value to field type

```

```

class whalesong.models.DateTimeField (parse_format=None, default_timezone=None,
                                       force_timezone=False, **kwargs)
    Bases: dirty_models.fields.DateTimeField

```

Date time field that allow timestamps in microseconds.

Parameters

- **parse_format** (*str or dict*) – String format to cast string to datetime. It could be an string format or a *dict* with two keys:
 - *parser* key to set how string must be parsed. It could be a callable.
 - *formatter* key to set how datetime must be formatted. It could be a callable.
- **default_timezone** (*datetime.tzinfo*) – Default timezone to use when value does not have one.

- **force_timezone** – If it is True value will be converted to timezone defined on `default_timezone` parameter. If `default_timezone` is not defined it is ignored.

Type `bool`

convert_value (*value*)

Converts value to field type

Model `whalesong.models.BaseModel` (*data=None, flat=False, *args, **kwargs*)

Bases: `dirty_models.models.BaseModel`

Base model which convert field name from underscore-style to camelCase-style automatically.

property `id` : `str` [*READ ONLY*]

Unique identifier.

class `whalesong.models.ModelFormatterIter` (*model*)

Bases: `dirty_models.utils.ModelFormatterIter`

format_field (*field, value*)

class `whalesong.models.JSONEncoder` (*, *skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None*)

Bases: `dirty_models.utils.JSONEncoder`

Constructor for JSONEncoder, with sensible defaults.

If `skipkeys` is false, then it is a `TypeError` to attempt encoding of keys that are not `str`, `int`, `float` or `None`. If `skipkeys` is True, such items are simply skipped.

If `ensure_ascii` is true, the output is guaranteed to be `str` objects with all incoming non-ASCII characters escaped. If `ensure_ascii` is false, the output can contain non-ASCII characters.

If `check_circular` is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an `OverflowError`). Otherwise, no such check takes place.

If `allow_nan` is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a `ValueError` to encode such floats.

If `sort_keys` is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If `indent` is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, `separators` should be an (`item_separator`, `key_separator`) tuple. The default is (`'`, `'`, `'`: `'`) if `indent` is None and (`'`, `'`, `'`: `'`) otherwise. To get the most compact JSON representation, you should specify (`'`, `'`) to eliminate whitespace.

If specified, `default` is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

default_model_iter

alias of `ModelFormatterIter`

3.4.5 Base Managers

class `whalesong.managers.BaseManager` (*driver, manager_path=""*)

Bases: `object`

Base manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

get_commands ()

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result[~T]*

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

class whalesong.managers.**BaseModelManager** (*driver*, *manager_path*=")

Bases: *whalesong.managers.BaseManager*, *typing.Generic*

Base model manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

classmethod **map_model** (*data*)

Return type *+MODEL_TYPE*

classmethod **get_model_result_class** ()

Return type *Result[+MODEL_TYPE]*

classmethod **get_monitor_result_class** ()

Return type *MonitorResult[+MODEL_TYPE]*

classmethod **get_field_monitor_result_class** (*field*)

Return type *MonitorResult[Dict[str, Any]]*

get_model ()

Get model object

Return type *Result[+MODEL_TYPE]*

Returns Model object

monitor_model()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

monitor_field(*field*)

Monitor any change on a model's field.

Parameters **field**(*str*) – Field to monitor.

Return type *MonitorResult*[Dict[*str*, Any]]

Returns Model monitor

class whalesong.managers.**BaseCollectionManager**(*driver, manager_path=""*)

Bases: *whalesong.managers.BaseManager*, *typing.Generic*

Base collection manager.

Parameters

- **driver**(*BaseWhalesongDriver*) – Whalesong driver
- **manager_path**(*str*) – Manager prefix path.

classmethod **get_monitor_result_class**()

Return type *MonitorResult*[+MODEL_TYPE]

classmethod **get_iterator_result_class**()

Return type *IteratorResult*[+MODEL_TYPE]

classmethod **get_item_result_class**()

Return type *Result*[+MODEL_TYPE]

get_items()

Get all items on collection.

Return type *IteratorResult*[+MODEL_TYPE]

Returns Async iterator

get_length()

Get collection items count.

Return type *Result*[int]

Returns Items count

get_item_by_id(*item_id*)

Get model by identifier.

Parameters **item_id**(*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

find_item_by_id(*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id**(*str*) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters *item_id* (`str`) – Model identifier.

Return type `Result[None]`

get_first ()

Get first item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_last ()

Get last item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

monitor_add ()

Monitor add item collection. Iterate each time a item is added to collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

monitor_remove ()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

monitor_change ()

Monitor change item collection. Iterate each time a item change in collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

monitor_field (*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type `MonitorResult[Dict[str, Any]]`

Returns Model object iterator

get_submanager (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters *name* (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

3.4.6 Contact references

3.4.6.1 Managers

class whalesong.managers.contact.**ContactCollectionManager** (*driver*, *manager_path=""*) *man-*

Bases: *whalesong.managers.BaseCollectionManager*

Contact collection manager. It allows manage contact collection.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result[+MODEL_TYPE]*

Returns Model object.

get_commands ()

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type *Result[+MODEL_TYPE]*

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_result_class()

Return type `Result[+MODEL_TYPE]`

get_items()

Get all items on collection.

Return type `IteratorResult[+MODEL_TYPE]`

Returns Async iterator

get_iterator_result_class()

Return type `IteratorResult[+MODEL_TYPE]`

get_last()

Get last item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_length()

Get collection items count.

Return type `Result[int]`

Returns Items count

get_monitor_result_class()

Return type `MonitorResult[+MODEL_TYPE]`

get_submanager(name)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

monitor_add()

Monitor add item collection. Iterate each time a item is added to collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

monitor_change()

Monitor change item collection. Iterate each time a item change in collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

monitor_field(field)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type `MonitorResult[Dict[str, Any]]`

Returns Model object iterator

monitor_remove()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters *item_id* (*str*) – Model identifier.

Return type *Result*[None]

remove_submanager (*name*)

Remove a submanager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_MANAGER_CLASS

alias of *ContactManager*

resync_contacts ()

Return type *Result*[None]

get_me ()

Return type *Result*[*Contact*]

class whalesong.managers.contact.**ContactManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Contact manager. It allows manage a contact.

profile_pic_thumb

ProfilePictureManager

Contact's picture thumb manager.

__getitem__ (*name*)

Get a submanager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *BaseManager*

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[Dict[str, Any]]

get_model()
Get model object

Return type *Result*[+MODEL_TYPE]

Returns Model object

get_model_result_class()

Return type *Result*[+MODEL_TYPE]

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)
Get a submanager.

Parameters **name** (str) – Field where submanager was stored.

Return type *BaseManager*

map_model(data)

Return type +MODEL_TYPE

monitor_field(field)
Monitor any change on a model's field.

Parameters **field** (str) – Field to monitor.

Return type *MonitorResult*[Dict[str, Any]]

Returns Model monitor

monitor_model()
Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager(name)
Remove a submanager.

Parameters **name** (str) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_CLASS
alias of *Contact*

block()
Block contact.

Return type *Result*[None]

unblock()
Unblock contact.

Return type *Result*[None]

3.4.6.2 Models

Model `whalesong.managers.contact.Contact` (*data=None, flat=False, *args, **kwargs*)
Bases: `whalesong.models.BaseModel`

property name : `str`
Contact name. It will be name on phone contact list.

property pushname : `str`
Contact defined name. It is set by contact on its whatsapp application.

property type : `str`
`i?`

property userhash : `str`
`i?`

property userid : `str`
User identifier. It used to be the phone number.

property status : `Status`
Contact status.

property id : `str` [READ ONLY]
StringIdField field [READ ONLY]

property formattedName : `str`
StringIdField field

Aliases

- `formatted_name`

property shortName : `str`
StringIdField field

Aliases

- `short_name`

property verifiedLevel : `str`
StringIdField field

Aliases

- `verified_level`

property verifiedName : `str`
StringIdField field

Aliases

- `verified_name`

property profilePicThumbObj : `ProfilePicture`
ModelField field (*whalesong.managers.profile_pic_thumb.ProfilePicture*)

Aliases

- `profile_pic_thumb_obj`

property isUser : `bool`
Default value `True`
BooleanField field

Aliases

- `is_user`

property `isBusiness` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_business`

property `isContactBlocked` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_contact_blocked`

property `isEnterprise` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_enterprise`

property `isHighLevelVerified` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_high_level_verified`

property `isMe` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_me`

property `isMyContact` : `bool`

Default value `True`

BooleanField field

Aliases

- `is_my_contact`

property `isPSA` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_psa`

property `isVerified` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_verified`

property `isWAContact` : `bool`

Default value `True`

BooleanField field

Aliases

- `is_wa_contact`

property `plaintextDisabled` : `bool`

Default value `False`

BooleanField field

Aliases

- `plaintext_disabled`

property `statusMute` : `bool`

Default value `False`

BooleanField field

Aliases

- `status_mute`

property `sectionHeader` : `str`

StringIdField field

Aliases

- `section_header`

property `labels` : `List of str`

Default value `[]`

List of contact labels ;?

to_vcard ()

Build vCard from contact.

Return type <function vCard at 0x7f6946120d90>

Returns vCard object of contact

3.4.7 Chat references

3.4.7.1 Managers

class `whalesong.managers.chat.ChatCollectionManager` (*driver*, *manager_path*=")
 Bases: `whalesong.managers.BaseCollectionManager`

Parameters

- **driver** (`BaseWhalesongDriver`) – Whalesong driver
- **manager_path** (`str`) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (`str`) – Field where manager will be stored.
- **submanager** (`BaseManager`) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (`str`) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_commands ()

Get manager available static commands.

Return type `Result[List[str]]`

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (`str`) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_result_class()
Return type *Result*[+MODEL_TYPE]

get_items()
Get all items on collection.
Return type *IteratorResult*[+MODEL_TYPE]
Returns Async iterator

get_iterator_result_class()
Return type *IteratorResult*[+MODEL_TYPE]

get_last()
Get last item in collection.
Return type *Result*[+MODEL_TYPE]
Returns Model object.

get_length()
Get collection items count.
Return type *Result*[int]
Returns Items count

get_monitor_result_class()
Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)
Get a submanager. It could be a explicit submanager or contained model manager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()
Monitor add item collection. Iterate each time a item is added to collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

monitor_change()
Monitor change item collection. Iterate each time a item change in collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

monitor_field(field)
Monitor item's field change. Iterate each time a field changed in any item of collection.
Return type *MonitorResult*[*Dict*[*str*, *Any*]]
Returns Model object iterator

monitor_remove()
Monitor remove item collection. Iterate each time a item is removed from collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[None]

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_MANAGER_CLASS

alias of *ChatManager*

get_active ()

Returns chat selected.

Return type *Result*[*Chat*]

Returns Chat object

resync_messages ()

Resynchronize messages.

Return type *Result*[None]

ensure_chat_with_contact (*contact_id*)

Ensure there is a chat with a given contact. If it does not exist it will be created.

Parameters **contact_id** (*str*) – Contact’s identifier.

Return type *Result*[*Chat*]

Returns Chat object

create_group (*name*, *contact_ids*, *picture=None*, *picture_preview=None*)

Create a new chat group.

Parameters

- **name** (*str*) – Group’s name
- **contact_ids** (*List*[*str*]) – List of contact identifier to invite.
- **picture** (*Optional*[*BytesIO*]) – JPG image.

Return type *Result*[*Chat*]

Returns Chat object.

forward_messages_to_chats (*message_ids*, *chat_ids*)

Forward messages.

Return type *Result*[dict]

Returns Operation result.

class whalesong.managers.chat.**ChatManager** (*driver*, *manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Chat manager. It allows manage a chat.

msgs

MessageCollectionManager

Chat's message collection manager.

msg_load_state

MsgLoadStateManager

Chat's message load state manager.

metadata:

GroupMetadataManager

Chat's group metadata manager.

contact

ContactManager

Chat's contact manager.

live_location

LiveLocationManager

Live location manager. You should call to *find_live_location()* before use it.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *Chat*

send_text (*text*, *quoted_msg_id=None*, *mentions=None*, *link_desc=None*)

Send text message to current chat.

Parameters

- **text** (*str*) – Message to send.
- **quoted_msg_id** (*Optional[str]*) – Quoted message's identifier.
- **mentions** (*Optional[List[str]]*) – List of user ids mentioned.
- **link_desc** – Link description.

Return type *Result[str]*

Returns New message's identifier

send_contact (*contact_id*, *quoted_msg_id=None*)

Send contact to current chat.

Parameters

- **contact_id** (*str*) – Contact identifier to send.
- **quoted_msg_id** (*Optional[str]*) – Quoted message's identifier.

Return type *Result[str]*

Returns New message's identifier

send_contact_phone (*contact_name*, *phone_number*, *quoted_msg_id=None*)

Send contact to current chat using contact name and phone number.

Parameters

- **contact_name** (*str*) – Contact’s name.
- **phone_number** (*str*) – Contact’s phone number.
- **quoted_msg_id** (*Optional[str]*) – Quoted message’s identifier.

Return type *Result[str]*

Returns New message’s identifier

send_media (*media_data*, *content_type=None*, *filename=None*, *caption=None*, *quoted_msg_id=None*, *mentions=None*)

Send media file to current chat.

Parameters

- **media_data** (*BytesIO*) – *io.BytesIO*
- **content_type** (*Optional[str]*) – File content type. It is used by Whatsapp to known how to render it.
- **filename** (*Optional[str]*) – File name.
- **caption** (*Optional[str]*) – Media caption.
- **quoted_msg_id** (*Optional[str]*) – Quoted message’s identifier.
- **mentions** (*Optional[List[str]]*) – List of user ids mentioned.

Return type *Result[str]*

Returns New message’s identifier

leave_group ()

Leave current chat group.

Warning: Only available on group chats.

Return type *Result[None]*

delete_chat ()

Delete chat.

Return type *Result[None]*

send_seen ()

Mark chat as seen.

Return type *Result[None]*

load_earlier_messages ()

Load earlier messages.

Tip: You should monitor chat messages in order to get new messages loaded.

```
async for msg in driver.chat[chat_id].msgs.monitor_add():
    print(msg)
```

Return type `Result[None]`

load_all_earlier_messages()

Load ALL earlier messages.

Tip: You should monitor chat messages in order to get new messages loaded. And perhaps you should remove them after process in order to save memory.

```
async for msg in driver.chat[chat_id].msgs.monitor_add():
    print(msg)
    await driver.chat[chat_id].msgs.remove_item_by_id(item.id)
```

Return type `Result[None]`

set_subject(subject)

Set group subject/title.

Parameters **subject** (`str`) – New group subject/title string

Return type `Result[None]`

Returns None

Model LiveLocation (`data=None, flat=False, *args, **kwargs`)

Bases: `whalesong.models.BaseModel`

Live location model.

clear()

Clears all the data in the object, keeping original data

clear_all()

Clears all the data in the object

clear_modified_data()

Clears only the modified data

copy()

Creates a copy of model

delete_attr_by_path(field_path)

It deletes fields looked up by field path. Field path is dot-formatted string path: `parent_field.child_field`.

Parameters **field_path** (`str`) – field path. It allows `*` as wildcard.

delete_field_value(name)

Mark this field to be deleted

export_data()

Get the results with the modified_data

export_deleted_fields()

Returns a list with any deleted fields form original data. In tree models, deleted fields on children will be appended.

export_modifications()

Returns model modifications.

export_modified_data()

Get the modified data

export_original_data()

Get the original data

flat_data()

Pass all the data from modified_data to original_data

get_1st_attr_by_path(*field_path*, ***kwargs*)

It returns first value looked up by field path. Field path is dot-formatted string path: `parent_field.child_field`.

Parameters

- **field_path** (*str*) – field path. It allows * as wildcard.
- **default** – Default value if field does not exist. If it is not defined `AttributeError` exception will be raised.

Returns value

get_attrs_by_path(*field_path*, *stop_first=False*)

It returns list of values looked up by field path. Field path is dot-formatted string path: `parent_field.child_field`.

Parameters

- **field_path** (*list* or *None*.) – field path. It allows * as wildcard.
- **stop_first** (*bool*) – Stop iteration on first value looked up. Default: False.

Returns A list of values or None it was a invalid path.

Return type `list` or `None`

get_default_data()

Returns a dictionary with default data.

Returns dict

get_field_obj(*name*)

get_field_value(*name*)

Get the field value from the modified data or the original one

get_fields()

Returns used fields of model

get_original_field_value(*name*)

Returns original field value or None

get_parent()

Returns parent model

get_read_only()

Returns whether model could be modified or not

get_real_name(*name*)

get_structure()

Returns a dictionary with model field objects.

Returns dict

import_data(*data*)

Set the fields established in data to the instance

import_deleted_fields(*data*)

Set data fields to deleted

is_locked()

Returns whether model is locked

is_modified()

Returns whether model is modified or not

is_modified_field(*name*)
Returns whether a field is modified or not

lock()
Lock model to avoid modification on read only fields

reset_attr_by_path(*field_path*)
It restores original values for fields looked up by field path. Field path is dot-formatted string path:
parent_field.child_field.
Parameters **field_path** (*str*) – field path. It allows * as wildcard.

reset_field_value(*name*)
Resets value of a field

set_field_value(*name*, *value*)
Set the value to the field modified_data

set_parent(*value*)
Sets parent model

set_read_only(*value*)
Sets whether model could be modified or not

unlock()
Unlock model to be able to write even it's read only

property id : *str* [READ ONLY]
StringIdField field [READ ONLY]

property duration : *timedelta*
TimedeltaField field

property participants : List of *Participant*
Array of ModelField field (*whalesong.managers.live_location.Participant*)

add_submanager(*name*, *submanager*)
Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands()
Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class(*field*)

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

get_model()
Get model object

Return type *Result*[+*MODEL_TYPE*]

Returns Model object

get_model_result_class()

Return type *Result*[+*MODEL_TYPE*]

get_monitor_result_class()
Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)
Get a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *BaseManager*

map_model(data)
Return type +MODEL_TYPE

mark_composing()
Set “typing...” message for 2.5 seconds.
Return type *Result*[None]
Returns None

monitor_field(field)
Monitor any change on a model’s field.
Parameters **field** (*str*) – Field to monitor.
Return type *MonitorResult*[Dict[*str*, Any]]
Returns Model monitor

monitor_model()
Monitor any change on model.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model monitor

remove_submanager(name)
Remove a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *Result*[~T]

mark_recording()
Set “recording audio...” message.
Return type *Result*[None]
Returns None

mark_paused()
Unset “typing...” or “recording audio...” message.
Return type *Result*[None]
Returns None

can_archive()
Check whether chat could be archived.
Return type *Result*[bool]
Returns Whether chat could be archived.

can_send()
Check whether current user could send message in chat.

Return type `Result[bool]`

Returns Whether current user could send message in chat.

can_pin()

Check whether chat could be pinned.

Return type `Result[bool]`

Returns Whether chat could be pinned.

archive()

Archive chat.

Return type `Result[bool]`

Returns Operation result.

unarchive()

Unarchive chat.

Return type `Result[bool]`

Returns Operation result.

pin()

Pin chat.

Return type `Result[bool]`

Returns Operation result.

unpin()

Unpin chat.

Return type `Result[bool]`

Returns Operation result.

set_group_description(description)

Set group description.

Return type `Result[bool]`

Returns Operation result.

star_messages(message_ids)

Star messages.

Return type `Result[bool]`

Returns Operation result.

unstar_messages(message_ids)

Unstar messages.

Return type `Result[bool]`

Returns Operation result.

send_not_spam()

Send not spam report.

Return type `Result[bool]`

Returns Operation result.

send_spam_report()

Send spam report.

Return type *Result[bool]*

Returns Operation result.

find_live_location()

It find chat's live location. If it does not exist it will be created.

Return type *Result[LiveLocation]*

Returns Live location.

class whalesong.managers.chat.**MsgLoadStateManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Message load state manager

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *MsgLoadState*

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands()

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult[Dict[str, Any]]*

get_model()

Get model object

Return type *Result[+MODEL_TYPE]*

Returns Model object

get_model_result_class()
Return type *Result*[+MODEL_TYPE]

get_monitor_result_class()
Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(*name*)
Get a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *BaseManager*

map_model(*data*)
Return type +MODEL_TYPE

monitor_field(*field*)
Monitor any change on a model's field.
Parameters **field** (*str*) – Field to monitor.
Return type *MonitorResult*[*Dict*[*str*, *Any*]]
Returns Model monitor

monitor_model()
Monitor any change on model.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model monitor

remove_submanager(*name*)
Remove a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *Result*[~T]

3.4.7.2 Models

Model `whalesong.managers.chat.Chat` (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

Chat model.

property name : *str*

Chat title.

property pin : *int*

Pin type (i?)

property archive : *bool*

Default value False

Whether chat is archived or not.

property contact : *Contact*

Contact object.

property kind : *str*

i?

property `mute` : `Mute`

Mute information.

property `id` : `str` *[READ ONLY]*

StringIdField field *[READ ONLY]*

property `lastMessageTs` : `datetime`

DateTimeField field

Aliases

- `last_message_ts`
- `t`

property `unreadCount` : `int`

IntegerField field

Aliases

- `unread_count`

property `changeNumberNewJid` : `str`

StringIdField field

Aliases

- `change_number_new_jid`

property `changeNumberOldJid` : `str`

StringIdField field

Aliases

- `change_number_old_jid`

property `groupMetadata` : `GroupMetadata`

ModelField field (*whalesong.managers.group_metadata.GroupMetadata*)

Aliases

- `group_metadata`

property `isAnnounceGrpRestrict` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_announce_grp_restrict`

property `isGroup` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_group`

property `isReadOnly` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_read_only`

property `lastReceivedKey` : `str`
StringIdField field

Aliases

- `last_received_key`

property `modifyTag` : `str`
StringIdField field

Aliases

- `modify_tag`

property `muteExpiration` : `int`
IntegerField field

Aliases

- `mute_expiration`

property `notSpam` : `bool`

Default value `False`

BooleanField field

Aliases

- `not_spam`

Model `whalesong.managers.chat.MsgLoadState` (*data=None, flat=False, *args, **kwargs*)

Bases: `whalesong.models.BaseModel`

property `id` : `str` [READ ONLY]
StringIdField field [READ ONLY]

property `contextLoaded` : `bool`

Default value `False`

BooleanField field

Aliases

- `context_loaded`

property `isLoadingAroundMsgs` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_loading_around_msgs`

property `isLoadingEarlierMsgs` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_loading_earlier_msgs`

```
property isLoadingRecentMsgs : bool
```

Default value False

BooleanField field

Aliases

- `is_loading_recent_msgs`

```
property noEarlierMsgs : bool
```

Default value False

BooleanField field

Aliases

- `no_earlier_msgs`

3.4.8 Message references

3.4.8.1 Managers

```
class whalesong.managers.message.MessageCollectionManager(driver, manager_path="")
```

Bases: *whalesong.managers.BaseCollectionManager*

Message collection manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Walesong driver
- **manager_path** (*str*) – Manager prefix path.

```
__getitem__ (name)
```

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

```
__getattr__ (name)
```

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

```
add_submanager (name, submanager)
```

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

```
find_item_by_id (item_id)
```

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result[+MODEL_TYPE]*

Returns Model object.

get_commands()
Get manager available static commands.
Return type `Result[List[str]]`
Returns Manager static commands.

get_first()
Get first item in collection.
Return type `Result[+MODEL_TYPE]`
Returns Model object.

get_item_by_id(item_id)
Get model by identifier.
Parameters **item_id** (`str`) – Model identifier.
Return type `Result[+MODEL_TYPE]`
Returns Model object.

get_item_result_class()
Return type `Result[+MODEL_TYPE]`

get_items()
Get all items on collection.
Return type `IteratorResult[+MODEL_TYPE]`
Returns Async iterator

get_iterator_result_class()
Return type `IteratorResult[+MODEL_TYPE]`

get_last()
Get last item in collection.
Return type `Result[+MODEL_TYPE]`
Returns Model object.

get_length()
Get collection items count.
Return type `Result[int]`
Returns Items count

get_monitor_result_class()
Return type `MonitorResult[+MODEL_TYPE]`

get_submanager(name)
Get a submanager. It could be a explicit submanager or contained model manager.
Parameters **name** (`str`) – Field where submanager was stored.
Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

monitor_add()
Monitor add item collection. Iterate each time a item is added to collection.
Return type `MonitorResult[+MODEL_TYPE]`
Returns Model object iterator

monitor_change()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field(*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[Dict[str, Any]]

Returns Model object iterator

monitor_remove()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id(*item_id*)

Remove item by identifier.

Parameters *item_id* (str) – Model identifier.

Return type *Result*[None]

remove_submanager(*name*)

Remove a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_MANAGER_CLASS

alias of *MessageManager*

monitor_new()

Monitor new messages.

Return type *MonitorResult*[BaseMessage]

Returns New message monitor.

await download_media(*model*)

Download message's attached media file. It will decrypt media file using key on message object.

Parameters *model* (*MediaMixin*) – MediaMixin

Return type BytesIO

Returns Media stream.

class whalesong.managers.message.**MessageManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Message object manager.

info

MessageInfoManager

Message information manager.

__getitem__(*name*)

Get a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

get_model ()

Get model object

Return type *Result*[+*MODEL_TYPE*]

Returns Model object

get_model_result_class ()

Return type *Result*[+*MODEL_TYPE*]

get_monitor_result_class ()

Return type *MonitorResult*[+*MODEL_TYPE*]

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +*MODEL_TYPE*

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+*MODEL_TYPE*]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_CLASS

alias of *BaseMessage*

await download_media ()

Download message's attached media file. It will decrypt media file using key on message object.

Return type *BytesIO*

Returns Media stream.

fetch_info ()

Fetch message information. It must fetch before try to use message information manager.

Return type *Result*[*MessageInfo*]

Returns Message information (*MessageInfo*)

can_star ()

Check whether message could be starred.

Return type *Result*[*bool*]

Returns Whether message could be starred.

star ()

Star message.

Return type *Result*[*bool*]

unstar ()

Unstar message.

Return type *Result*[*bool*]

can_revoke ()

Check whether message could be revoked (deleted for other).

Return type *Result*[*bool*]

Returns Whether message could be revoked.

revoke (*clear_media=True*)

Revoke message.

Return type *Result*[*str*]

class whalesong.managers.message.**MessageInfoManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Message information object manager.

delivery

MessageAckCollectionManager

Message delivery acknowledgement collection manager.

read
MessageAckCollectionManager
Message read acknowledgement collection manager.

played
MessageAckCollectionManager
Message played acknowledgement collection manager.

__getitem__ (*name*)
Get a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *BaseManager*

__getattr__ (*name*)
Get a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *BaseManager*

add_submanager (*name*, *submanager*)
Add a submanager.
Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()
Get manager available static commands.
Return type *Result[List[str]]*
Returns Manager static commands.

get_field_monitor_result_class (*field*)
Return type *MonitorResult[Dict[str, Any]]*

get_model ()
Get model object
Return type *Result[+MODEL_TYPE]*
Returns Model object

get_model_result_class ()
Return type *Result[+MODEL_TYPE]*

get_monitor_result_class ()
Return type *MonitorResult[+MODEL_TYPE]*

get_submanager (*name*)
Get a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *BaseManager*

map_model (*data*)
Return type *+MODEL_TYPE*

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+*MODEL_TYPE*]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~*T*]

MODEL_CLASS

alias of *MessageInfo*

class whalesong.managers.message.**MessageAckCollectionManager** (*driver*, *manager_path*=")

Bases: *whalesong.managers.BaseCollectionManager*

Message acknowledgement collection manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~*MODEL_MANAGER_TYPE*]

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~*MODEL_MANAGER_TYPE*]

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+*MODEL_TYPE*]

Returns Model object.

get_commands()
Get manager available static commands.
Return type `Result[List[str]]`
Returns Manager static commands.

get_first()
Get first item in collection.
Return type `Result[+MODEL_TYPE]`
Returns Model object.

get_item_by_id(item_id)
Get model by identifier.
Parameters **item_id** (`str`) – Model identifier.
Return type `Result[+MODEL_TYPE]`
Returns Model object.

get_item_result_class()
Return type `Result[+MODEL_TYPE]`

get_items()
Get all items on collection.
Return type `IteratorResult[+MODEL_TYPE]`
Returns Async iterator

get_iterator_result_class()
Return type `IteratorResult[+MODEL_TYPE]`

get_last()
Get last item in collection.
Return type `Result[+MODEL_TYPE]`
Returns Model object.

get_length()
Get collection items count.
Return type `Result[int]`
Returns Items count

get_monitor_result_class()
Return type `MonitorResult[+MODEL_TYPE]`

get_submanager(name)
Get a submanager. It could be a explicit submanager or contained model manager.
Parameters **name** (`str`) – Field where submanager was stored.
Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

monitor_add()
Monitor add item collection. Iterate each time a item is added to collection.
Return type `MonitorResult[+MODEL_TYPE]`
Returns Model object iterator

monitor_change()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field(*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[Dict[str, Any]]

Returns Model object iterator

monitor_remove()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id(*item_id*)

Remove item by identifier.

Parameters *item_id* (str) – Model identifier.

Return type *Result*[None]

remove_submanager(*name*)

Remove a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_MANAGER_CLASS

alias of *MessageAckManager*

class whalesong.managers.message.**MessageAckManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Message acknowledgement object manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (str) – Manager prefix path.

__getitem__(*name*)

Get a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *BaseManager*

__getattr__(*name*)

Get a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *BaseManager*

add_submanager(*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

get_model ()

Get model object

Return type *Result*[+*MODEL_TYPE*]

Returns Model object

get_model_result_class ()

Return type *Result*[+*MODEL_TYPE*]

get_monitor_result_class ()

Return type *MonitorResult*[+*MODEL_TYPE*]

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +*MODEL_TYPE*

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+*MODEL_TYPE*]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~*T*]

MODEL_CLASS

alias of *MessageAck*

3.4.8.2 Models

```

class whalesong.managers.message.MessageTypes
    Bases: enum.Enum

    An enumeration.

    NOTIFICATION_TEMPLATE = 'notification_template'
    GROUP_NOTIFICATION = 'group_notification'
    GP2 = 'gp2'
    BROADCAST_NOTIFICATION = 'broadcast_notification'
    E2E_NOTIFICATION = 'e2e_notification'
    CALL_LOG = 'call_log'
    PROTOCOL = 'protocol'
    CIPHERTEXT = 'ciphertext'
    REVOKED = 'revoked'
    UNKNOWN = 'unknown'
    CHAT = 'chat'
    IMAGE = 'image'
    VCARD = 'vcard'
    MULTI_VCARD = 'multi_vcard'
    LOCATION = 'location'
    PAYMENT = 'payment'
    DOCUMENT = 'document'
    AUDIO = 'audio'
    PTT = 'ptt'
    VIDEO = 'video'
    STICKER = 'sticker'

class whalesong.managers.message.Ack
    Bases: enum.Enum

    An enumeration.

    ERROR = -1
    PENDING = 0
    SERVER = 1
    DEVICE = 2
    READ = 3
    PLAYED = 4

Model whalesong.managers.message.VCardItem(data=None, flat=False, *args, **kwargs)
    Bases: whalesong.models.BaseModel

    vCard item.

```

property `vcard` : `str`

Serialized vCard object.

property `id` : `str` *[READ ONLY]*

StringIdField field *[READ ONLY]*

property `displayName` : `str`

StringIdField field

Aliases

- `display_name`

class `whalesong.managers.message.MessageMetaclass` (*name, bases, classdict*)

Bases: `dirty_models.models.CamelCaseMeta`

Message metaclass. It will build message model according to type.

Model `whalesong.managers.message.BaseMessage` (*data=None, *args, **kwargs*)

Bases: `whalesong.models.BaseModel`

Base message model.

property `type` : `MessageTypes` *[READ ONLY]*

Message type.

property `subtype` : `str`

Message subtype.

property `body` : `str`

Message content.

property `timestamp` : `datetime`

Message timestamp.

Aliases

- `t`

property `to` : `str`

i?

property `author` : `str`

i?

property `sender` : `str`

Sender's contact identifier.

property `self` : `str`

Default value in

i?

property `ack` : `Ack`

Acknowledge state.

property `invis` : `bool`

Default value False

i?

property `star` : `bool`

Default value False

Whether it is starred or not.

property links : **List of str**

List of links of message.

property chat : **Chat**

Chat object where message was sent.

property dir : **str**

¿?

property rtl : **bool**

Default value False

Whether message's content is a right to left text.

property id : **str** *[READ ONLY]*

StringIdField field *[READ ONLY]*

property notifyName : **str**

StringIdField field

Aliases

- **notify_name**

property from : **str**

StringIdField field

Aliases

- **from**

property senderObj : **Contact**

ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- **sender_obj**

property isNewMsg : **bool**

Default value False

BooleanField field

Aliases

- **is_new_msg**

property isForwarded : **bool**

Default value False

BooleanField field

Aliases

- **is_forwarded**

property isGroupMsg : **bool**

Default value False

BooleanField field

Aliases

- `is_group_msg`

property `isStatusV3` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_status_v3`

property `isPSA` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_psa`

property `statusV3TextBg` : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property `isSentByMe` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_sent_by_me`

property `isNotification` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_notification`

property `isGroupNotification` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_group_notification`

property `isBizNotification` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_biz_notification`

property `isMedia` : `bool`

Default value False

BooleanField field

Aliases

- `is_media`

property isLink : `bool`

Default value False

BooleanField field

Aliases

- `is_link`

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- `has_link`

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- `is_doc`

property isMms : `bool`

Default value False

BooleanField field

Aliases

- `is_mms`

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- `is_revoked`

property showForwarded : `bool`

Default value False

BooleanField field

Aliases

- `show_forwarded`

property containsEmoji : `bool`

Default value False

BooleanField field

Aliases

- contains_emoji

property isFailed : bool

Default value False

BooleanField field

Aliases

- is_failed

property isPersistent : bool

Default value False

BooleanField field

Aliases

- is_persistent

property isUserCreatedType : bool

Default value False

BooleanField field

Aliases

- is_user_created_type

property hasPromises : bool

Default value False

BooleanField field

Aliases

- has_promises

Model whalesong.managers.message.QuotedMessageMixin(*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property id : str [READ ONLY]

StringIdField field [READ ONLY]

property quotedMsgObj : BaseMessage

ModelField field (*whalesong.managers.message.BaseMessage*)

Aliases

- quoted_msg_obj

property quotedStanzaID : str

StringIdField field

Aliases

- quoted_stanza_id

property quotedParticipant : str

StringIdField field

Aliases

- **quoted_participant**

property `quotedRemoteJid` : `str`
StringIdField field

Aliases

- **quoted_remote_jid**

Model `whalesong.managers.message.MentionsMixin` (`data=None`, `flat=False`, `*args`, `**kwargs`)

Bases: `whalesong.models.BaseModel`

property `id` : `str` [READ ONLY]
StringIdField field [READ ONLY]

property `mentionedJidList` : List of `str`
Array of StringIdField field

Aliases

- **mentioned_jid_list**

Model `whalesong.managers.message.LinkContentMixin` (`data=None`, `flat=False`, `*args`, `**kwargs`)

Bases: `whalesong.models.BaseModel`

property `description` : `str`
Page description.

property `title` : `str`
Page title.

property `thumbnail` : `None`
Page thumbnail.

property `links` : List of `str`
List of links. `?`

property `id` : `str` [READ ONLY]
StringIdField field [READ ONLY]

property `matchedText` : `str`
StringIdField field

Aliases

- **matched_text**

property `canonicalUrl` : `str`
StringIdField field

Aliases

- **canonical_url**

property `linkPreview` : `bool`

Default value `False`

BooleanField field

Aliases

- **link_preview**

Model whalesong.managers.message.**MediaMixin** (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property type : *MessageTypes* [READ ONLY]

EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property mimetype : *str*

StringIdField field

property caption : *str*

StringIdField field

property filehash : *str*

StringIdField field

property size : *int*

IntegerField field

property id : *str* [READ ONLY]

StringIdField field [READ ONLY]

property clientId : *str*

StringIdField field

Aliases

- *client_url*

property directPath : *str*

StringIdField field

Aliases

- *direct_path*

property mediaKey : *str*

StringIdField field

Aliases

- *media_key*

property isUnsentMedia : *bool*

Default value *False*

BooleanField field

Aliases

- *is_unsent_media*

Model whalesong.managers.message.**MediaFrameMixin** (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property body : *None*

Base64Field field

property height : *int*

IntegerField field

property width : *int*

IntegerField field

property id : *str* [READ ONLY]

StringIdField field [READ ONLY]

Model whalesong.managers.message.**AuthorMixin** (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property **author** : **str**

StringIdField field

property **id** : **str** [READ ONLY]

StringIdField field [READ ONLY]

Model whalesong.managers.message.**MediaStreamMixin** (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property **streamable** : **bool**

Default value False

BooleanField field

property **duration** : **int**

IntegerField field

property **id** : **str** [READ ONLY]

StringIdField field [READ ONLY]

property **isGif** : **bool**

Default value False

BooleanField field

Aliases

- **is_gif**

property **gifAttribution** : **bool**

Default value False

BooleanField field

Aliases

- **gif_attribution**

Model whalesong.managers.message.**TextMessage** (*data=None, *args, **kwargs*)

Bases: *whalesong.managers.message.QuotedMessageMixin, whalesong.managers.message.LinkContentMixin, whalesong.managers.message.MentionsMixin, whalesong.managers.message.BaseMessage*

Text message.

property **font** : **str**

StringIdField field

property **id** : **str** [READ ONLY]

StringIdField field [READ ONLY]

property **quotedMsgObj** : **BaseMessage**

ModelField field (*whalesong.managers.message.BaseMessage*)

Aliases

- **quoted_msg_obj**

property **quotedStanzaID** : **str**

StringIdField field

Aliases

- **quoted_stanza_id**

property quotedParticipant : `str`
StringIdField field

Aliases

- **quoted_participant**

property quotedRemoteJid : `str`
StringIdField field

Aliases

- **quoted_remote_jid**

property matchedText : `str`
StringIdField field

Aliases

- **matched_text**

property canonicalUrl : `str`
StringIdField field

Aliases

- **canonical_url**

property description : `str`
StringIdField field

property title : `str`
StringIdField field

property thumbnail : `None`
Base64Field field

property linkPreview : `bool`

Default value False

BooleanField field

Aliases

- **link_preview**

property links : **List of** `str`
Array of StringIdField field

property mentionedJidList : **List of** `str`
Array of StringIdField field

Aliases

- **mentioned_jid_list**

property type : *MessageTypes* [READ ONLY]
EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property subtype : `str`
StringIdField field


```

property body : str
    StringIdField field

property timestamp : datetime
    DateTimeField field

    Aliases
    • t

property notifyName : str
    StringIdField field

    Aliases
    • notify_name

property from : str
    StringIdField field

    Aliases
    • from

property to : str
    StringIdField field

property author : str
    StringIdField field

property sender : str
    StringIdField field

property senderObj : Contact
    ModelField field (whalesong.managers.contact.Contact)

    Aliases
    • sender_obj

property self : str
    Default value in
    StringIdField field

property ack : Ack
    EnumField field (whalesong.managers.message.Ack)

property invis : bool
    Default value False
    BooleanField field

property isNewMsg : bool
    Default value False
    BooleanField field

    Aliases
    • is_new_msg

property star : bool
    Default value False

```

BooleanField field

property isForwarded : `bool`

Default value False

BooleanField field

Aliases

- `is_forwarded`

property chat : `Chat`

ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_group_msg`

property isStatusV3 : `bool`

Default value False

BooleanField field

Aliases

- `is_status_v3`

property isPSA : `bool`

Default value False

BooleanField field

Aliases

- `is_psa`

property statusV3TextBg : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property isSentByMe : `bool`

Default value False

BooleanField field

Aliases

- `is_sent_by_me`

property isNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_notification`

property isGroupNotification : bool

Default value False

BooleanField field

Aliases

- **is_group_notification**

property isBizNotification : bool

Default value False

BooleanField field

Aliases

- **is_biz_notification**

property isMedia : bool

Default value False

BooleanField field

Aliases

- **is_media**

property isLink : bool

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : bool

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : bool

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : bool

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : bool

Default value False

BooleanField field

Aliases

- **is_revoked**

property showForwarded : **bool**

Default value False

BooleanField field

Aliases

- **show_forwarded**

property containsEmoji : **bool**

Default value False

BooleanField field

Aliases

- **contains_emoji**

property isFailed : **bool**

Default value False

BooleanField field

Aliases

- **is_failed**

property dir : **str**

StringIdField field

property rtl : **bool**

Default value False

BooleanField field

property isPersistent : **bool**

Default value False

BooleanField field

Aliases

- **is_persistent**

property isUserCreatedType : **bool**

Default value False

BooleanField field

Aliases

- **is_user_created_type**

property hasPromises : **bool**

Default value False

BooleanField field

Aliases

- `has_promises`

property `textColor` : `str`

StringIdField field

Aliases

- `text_color`

property `backgroundColor` : `str`

StringIdField field

Aliases

- `background_color`

Model `whalesong.managers.message.ImageMessage` (*data=None, *args, **kwargs*)

Bases: `whalesong.managers.message.QuotedMessageMixin`, `whalesong.managers.message.MentionsMixin`, `whalesong.managers.message.MediaMixin`, `whalesong.managers.message.MediaFrameMixin`, `whalesong.managers.message.BaseMessage`

Image message.

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `quotedMsgObj` : `BaseMessage`

ModelField field (*whalesong.managers.message.BaseMessage*)

Aliases

- `quoted_msg_obj`

property `quotedStanzaID` : `str`

StringIdField field

Aliases

- `quoted_stanza_id`

property `quotedParticipant` : `str`

StringIdField field

Aliases

- `quoted_participant`

property `quotedRemoteJid` : `str`

StringIdField field

Aliases

- `quoted_remote_jid`

property `mentionedJidList` : `List of str`

Array of StringIdField field

Aliases

- `mentioned_jid_list`

property `type` : `MessageTypes` [READ ONLY]

EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property `clientId` : `str`
StringIdField field

Aliases

- `client_url`

property `directPath` : `str`
StringIdField field

Aliases

- `direct_path`

property `mimetype` : `str`
StringIdField field

property `caption` : `str`
StringIdField field

property `filehash` : `str`
StringIdField field

property `size` : `int`
IntegerField field

property `mediaKey` : `str`
StringIdField field

Aliases

- `media_key`

property `isUnsentMedia` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_unsent_media`

property `body` : `None`
Base64Field field

property `height` : `int`
IntegerField field

property `width` : `int`
IntegerField field

property `subtype` : `str`
StringIdField field

property `timestamp` : `datetime`
DateTimeField field

Aliases

- `t`

property `notifyName` : `str`
StringIdField field

Aliases

- `notify_name`

property from : `str`
StringIdField field

Aliases

- `from`

property to : `str`
StringIdField field

property author : `str`
StringIdField field

property sender : `str`
StringIdField field

property senderObj : `Contact`
ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- `sender_obj`

property self : `str`

Default value in

StringIdField field

property ack : `Ack`
EnumField field (*whalesong.managers.message.Ack*)

property invis : `bool`

Default value False

BooleanField field

property isNewMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_new_msg`

property star : `bool`

Default value False

BooleanField field

property isForwarded : `bool`

Default value False

BooleanField field

Aliases

- `is_forwarded`

property links : `List of str`
Array of StringIdField field

property chat : *Chat*
ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : *bool*

Default value False

BooleanField field

Aliases

- *is_group_msg*

property isStatusV3 : *bool*

Default value False

BooleanField field

Aliases

- *is_status_v3*

property isPSA : *bool*

Default value False

BooleanField field

Aliases

- *is_psa*

property statusV3TextBg : *str*

StringIdField field

Aliases

- *status_v3_text_bg*

property isSentByMe : *bool*

Default value False

BooleanField field

Aliases

- *is_sent_by_me*

property isNotification : *bool*

Default value False

BooleanField field

Aliases

- *is_notification*

property isGroupNotification : *bool*

Default value False

BooleanField field

Aliases

- *is_group_notification*

property isBizNotification : *bool*

Default value False

BooleanField field

Aliases

- **is_biz_notification**

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- **is_media**

property isLink : `bool`

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : `bool`

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- **is_revoked**

property showForwarded : `bool`

Default value False

BooleanField field

Aliases

- `show_forwarded`

property `containsEmoji` : `bool`

Default value `False`

BooleanField field

Aliases

- `contains_emoji`

property `isFailed` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_failed`

property `dir` : `str`

StringIdField field

property `rtl` : `bool`

Default value `False`

BooleanField field

property `isPersistent` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value `False`

BooleanField field

Aliases

- `has_promises`

Model `whalesong.managers.message.VideoMessage` (`data=None`, `*args`, `**kwargs`)

Bases: `whalesong.managers.message.QuotedMessageMixin`, `whalesong.managers.message.MentionsMixin`, `whalesong.managers.message.MediaMixin`, `whalesong.managers.message.MediaFrameMixin`, `whalesong.managers.message.MediaStreamMixin`, `whalesong.managers.message.BaseMessage`

Video message.

property id : `str` [READ ONLY]
StringIdField field [READ ONLY]

property quotedMsgObj : `BaseMessage`
ModelField field (*whalesong.managers.message.BaseMessage*)

Aliases

- `quoted_msg_obj`

property quotedStanzaID : `str`
StringIdField field

Aliases

- `quoted_stanza_id`

property quotedParticipant : `str`
StringIdField field

Aliases

- `quoted_participant`

property quotedRemoteJid : `str`
StringIdField field

Aliases

- `quoted_remote_jid`

property mentionedJidList : List of `str`
Array of StringIdField field

Aliases

- `mentioned_jid_list`

property type : `MessageTypes` [READ ONLY]
EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property clientUrl : `str`
StringIdField field

Aliases

- `client_url`

property directPath : `str`
StringIdField field

Aliases

- `direct_path`

property mimetype : `str`
StringIdField field

property caption : `str`
StringIdField field

property filehash : `str`
StringIdField field

property size : `int`

IntegerField field

property mediaKey : `str`

StringIdField field

Aliases

- `media_key`

property isUnsentMedia : `bool`

Default value False

BooleanField field

Aliases

- `is_unsent_media`

property body : `None`

Base64Field field

property height : `int`

IntegerField field

property width : `int`

IntegerField field

property streamable : `bool`

Default value False

BooleanField field

property durantion : `int`

IntegerField field

property isGif : `bool`

Default value False

BooleanField field

Aliases

- `is_gif`

property gifAttribution : `bool`

Default value False

BooleanField field

Aliases

- `gif_attribution`

property subtype : `str`

StringIdField field

property timestamp : `datetime`

DateTimeField field

Aliases

- `t`

property notifyName : **str**
StringIdField field

Aliases

- **notify_name**

property from : **str**
StringIdField field

Aliases

- **from**

property to : **str**
StringIdField field

property author : **str**
StringIdField field

property sender : **str**
StringIdField field

property senderObj : **Contact**
ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- **sender_obj**

property self : **str**

Default value in
StringIdField field

property ack : **Ack**
EnumField field (*whalesong.managers.message.Ack*)

property invis : **bool**

Default value False
BooleanField field

property isNewMsg : **bool**

Default value False
BooleanField field

Aliases

- **is_new_msg**

property star : **bool**

Default value False
BooleanField field

property isForwarded : **bool**

Default value False
BooleanField field

Aliases

- `is_forwarded`

property `links` : `List of str`

Array of StringIdField field

property `chat` : `Chat`

ModelField field (*whalesong.managers.chat.Chat*)

property `isGroupMsg` : `bool`

Default value False

BooleanField field

Aliases

- `is_group_msg`

property `isStatusV3` : `bool`

Default value False

BooleanField field

Aliases

- `is_status_v3`

property `isPSA` : `bool`

Default value False

BooleanField field

Aliases

- `is_psa`

property `statusV3TextBg` : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property `isSentByMe` : `bool`

Default value False

BooleanField field

Aliases

- `is_sent_by_me`

property `isNotification` : `bool`

Default value False

BooleanField field

Aliases

- `is_notification`

property `isGroupNotification` : `bool`

Default value False

BooleanField field

Aliases

- **is_group_notification**

property isBizNotification : `bool`

Default value False

BooleanField field

Aliases

- **is_biz_notification**

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- **is_media**

property isLink : `bool`

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : `bool`

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- `is_revoked`

property `showForwarded` : `bool`

Default value `False`

BooleanField field

Aliases

- `show_forwarded`

property `containsEmoji` : `bool`

Default value `False`

BooleanField field

Aliases

- `contains_emoji`

property `isFailed` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_failed`

property `dir` : `str`

StringIdField field

property `rtl` : `bool`

Default value `False`

BooleanField field

property `isPersistent` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value `False`

BooleanField field

Aliases

- `has_promises`


```

Model whalesong.managers.message.AudioMessage (data=None, *args, **kwargs)
  Bases:   whalesong.managers.message.QuotedMessageMixin, whalesong.managers.
           message.MentionsMixin, whalesong.managers.message.MediaMixin, whalesong.
           managers.message.MediaStreamMixin, whalesong.managers.message.BaseMessage

  Audio message.

  property id : str [READ ONLY]
    StringIdField field [READ ONLY]

  property quotedMsgObj : BaseMessage
    ModelField field (whalesong.managers.message.BaseMessage)

    Aliases
      • quoted_msg_obj

  property quotedStanzaID : str
    StringIdField field

    Aliases
      • quoted_stanza_id

  property quotedParticipant : str
    StringIdField field

    Aliases
      • quoted_participant

  property quotedRemoteJid : str
    StringIdField field

    Aliases
      • quoted_remote_jid

  property mentionedJidList : List of str
    Array of StringIdField field

    Aliases
      • mentioned_jid_list

  property type : MessageTypes [READ ONLY]
    EnumField field [READ ONLY] (whalesong.managers.message.MessageTypes)

  property clientUrl : str
    StringIdField field

    Aliases
      • client_url

  property directPath : str
    StringIdField field

    Aliases
      • direct_path

  property mimetype : str
    StringIdField field

  property caption : str
    StringIdField field

```

property filehash : **str**
StringIdField field

property size : **int**
IntegerField field

property mediaKey : **str**
StringIdField field

Aliases

- **media_key**

property isUnsentMedia : **bool**

Default value False

BooleanField field

Aliases

- **is_unsent_media**

property streamable : **bool**

Default value False

BooleanField field

property duration : **int**
IntegerField field

property isGif : **bool**

Default value False

BooleanField field

Aliases

- **is_gif**

property gifAttribution : **bool**

Default value False

BooleanField field

Aliases

- **gif_attribution**

property subtype : **str**
StringIdField field

property body : **str**
StringIdField field

property timestamp : **datetime**
DateTimeField field

Aliases

- **t**

property notifyName : **str**
StringIdField field

Aliases

- `notify_name`

property from : `str`
StringIdField field

Aliases

- `from`

property to : `str`
StringIdField field

property author : `str`
StringIdField field

property sender : `str`
StringIdField field

property senderObj : `Contact`
ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- `sender_obj`

property self : `str`

Default value in

StringIdField field

property ack : `Ack`
EnumField field (*whalesong.managers.message.Ack*)

property invis : `bool`

Default value False

BooleanField field

property isNewMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_new_msg`

property star : `bool`

Default value False

BooleanField field

property isForwarded : `bool`

Default value False

BooleanField field

Aliases

- `is_forwarded`

property links : `List of str`
Array of StringIdField field

property chat : *Chat*
ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : *bool*

Default value False

BooleanField field

Aliases

- *is_group_msg*

property isStatusV3 : *bool*

Default value False

BooleanField field

Aliases

- *is_status_v3*

property isPSA : *bool*

Default value False

BooleanField field

Aliases

- *is_psa*

property statusV3TextBg : *str*

StringIdField field

Aliases

- *status_v3_text_bg*

property isSentByMe : *bool*

Default value False

BooleanField field

Aliases

- *is_sent_by_me*

property isNotification : *bool*

Default value False

BooleanField field

Aliases

- *is_notification*

property isGroupNotification : *bool*

Default value False

BooleanField field

Aliases

- *is_group_notification*

property isBizNotification : *bool*

Default value False

BooleanField field

Aliases

- **is_biz_notification**

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- **is_media**

property isLink : `bool`

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : `bool`

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- **is_revoked**

property showForwarded : `bool`

Default value False

BooleanField field

Aliases

- `show_forwarded`

property containsEmoji : `bool`

Default value False

BooleanField field

Aliases

- `contains_emoji`

property isFailed : `bool`

Default value False

BooleanField field

Aliases

- `is_failed`

property dir : `str`

StringIdField field

property rtl : `bool`

Default value False

BooleanField field

property isPersistent : `bool`

Default value False

BooleanField field

Aliases

- `is_persistent`

property isUserCreatedType : `bool`

Default value False

BooleanField field

Aliases

- `is_user_created_type`

property hasPromises : `bool`

Default value False

BooleanField field

Aliases

- `has_promises`

Model `whalesong.managers.message.PTTMessage` (*data=None, *args, **kwargs*)

Bases: `whalesong.managers.message.AudioMessage`

Push to talk message.

property id : `str` [READ ONLY]
StringIdField field [READ ONLY]

property quotedMsgObj : `BaseMessage`
ModelField field (*whalesong.managers.message.BaseMessage*)

Aliases

- `quoted_msg_obj`

property quotedStanzaID : `str`
StringIdField field

Aliases

- `quoted_stanza_id`

property quotedParticipant : `str`
StringIdField field

Aliases

- `quoted_participant`

property quotedRemoteJid : `str`
StringIdField field

Aliases

- `quoted_remote_jid`

property mentionedJidList : List of `str`
Array of StringIdField field

Aliases

- `mentioned_jid_list`

property type : `MessageTypes` [READ ONLY]
EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property clientUrl : `str`
StringIdField field

Aliases

- `client_url`

property directPath : `str`
StringIdField field

Aliases

- `direct_path`

property mimetype : `str`
StringIdField field

property caption : `str`
StringIdField field

property filehash : `str`
StringIdField field

property size : `int`
IntegerField field

property mediaKey : `str`
StringIdField field

Aliases

- `media_key`

property isUnsentMedia : `bool`

Default value False

BooleanField field

Aliases

- `is_unsent_media`

property streamable : `bool`

Default value False

BooleanField field

property durantion : `int`
IntegerField field

property isGif : `bool`

Default value False

BooleanField field

Aliases

- `is_gif`

property gifAttribution : `bool`

Default value False

BooleanField field

Aliases

- `gif_attribution`

property subtype : `str`
StringIdField field

property body : `str`
StringIdField field

property timestamp : `datetime`
DateTimeField field

Aliases

- `t`

property notifyName : `str`
StringIdField field

Aliases

- `notify_name`

property from : `str`
StringIdField field

Aliases

- from

property to : `str`
StringIdField field

property author : `str`
StringIdField field

property sender : `str`
StringIdField field

property senderObj : `Contact`
ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- sender_obj

property self : `str`

Default value in

StringIdField field

property ack : `Ack`
EnumField field (*whalesong.managers.message.Ack*)

property invis : `bool`

Default value False

BooleanField field

property isNewMsg : `bool`

Default value False

BooleanField field

Aliases

- is_new_msg

property star : `bool`

Default value False

BooleanField field

property isForwarded : `bool`

Default value False

BooleanField field

Aliases

- is_forwarded

property links : `List of str`
Array of StringIdField field

property chat : `Chat`
ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : `bool`

Default value False

BooleanField field

Aliases

- **is_group_msg**

property isStatusV3 : `bool`

Default value False

BooleanField field

Aliases

- **is_status_v3**

property isPSA : `bool`

Default value False

BooleanField field

Aliases

- **is_psa**

property statusV3TextBg : `str`

StringIdField field

Aliases

- **status_v3_text_bg**

property isSentByMe : `bool`

Default value False

BooleanField field

Aliases

- **is_sent_by_me**

property isNotification : `bool`

Default value False

BooleanField field

Aliases

- **is_notification**

property isGroupNotification : `bool`

Default value False

BooleanField field

Aliases

- **is_group_notification**

property isBizNotification : `bool`

Default value False

BooleanField field

Aliases

- **is_biz_notification**

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- **is_media**

property isLink : `bool`

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : `bool`

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- **is_revoked**

property showForwarded : `bool`

Default value False

BooleanField field

Aliases

- `show_forwarded`

property `containsEmoji` : `bool`

Default value `False`

BooleanField field

Aliases

- `contains_emoji`

property `isFailed` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_failed`

property `dir` : `str`

StringIdField field

property `rtl` : `bool`

Default value `False`

BooleanField field

property `isPersistent` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value `False`

BooleanField field

Aliases

- `has_promises`

Model `whalesong.managers.message.DocumentMessage` (`data=None`, `*args`, `**kwargs`)

Bases: `whalesong.managers.message.QuotedMessageMixin`, `whalesong.managers.message.MentionsMixin`, `whalesong.managers.message.MediaMixin`, `whalesong.managers.message.BaseMessage`

Document message.

property `body` : `None`

Base64Field field

property id : `str` *[READ ONLY]*
StringIdField field [READ ONLY]

property quotedMsgObj : `BaseMessage`
ModelField field (*whalesong.managers.message.BaseMessage*)

Aliases

- `quoted_msg_obj`

property quotedStanzaID : `str`
StringIdField field

Aliases

- `quoted_stanza_id`

property quotedParticipant : `str`
StringIdField field

Aliases

- `quoted_participant`

property quotedRemoteJid : `str`
StringIdField field

Aliases

- `quoted_remote_jid`

property mentionedJidList : `List of str`
Array of StringIdField field

Aliases

- `mentioned_jid_list`

property type : `MessageTypes` *[READ ONLY]*
EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property clientUrl : `str`
StringIdField field

Aliases

- `client_url`

property directPath : `str`
StringIdField field

Aliases

- `direct_path`

property mimetype : `str`
StringIdField field

property caption : `str`
StringIdField field

property filehash : `str`
StringIdField field

property size : `int`
IntegerField field

```

property mediaKey : str
    StringIdField field

    Aliases
        • media_key

property isUnsentMedia : bool

    Default value False

    BooleanField field

    Aliases
        • is_unsent_media

property subtype : str
    StringIdField field

property timestamp : datetime
    DateTimeField field

    Aliases
        • t

property notifyName : str
    StringIdField field

    Aliases
        • notify_name

property from : str
    StringIdField field

    Aliases
        • from

property to : str
    StringIdField field

property author : str
    StringIdField field

property sender : str
    StringIdField field

property senderObj : Contact
    ModelField field (whalesong.managers.contact.Contact)

    Aliases
        • sender_obj

property self : str

    Default value in

    StringIdField field

property ack : Ack
    EnumField field (whalesong.managers.message.Ack)

property invis : bool

```

Default value False

BooleanField field

property isNewMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_new_msg`

property star : `bool`

Default value False

BooleanField field

property isForwarded : `bool`

Default value False

BooleanField field

Aliases

- `is_forwarded`

property links : **List of** `str`

Array of StringIdField field

property chat : `Chat`

ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_group_msg`

property isStatusV3 : `bool`

Default value False

BooleanField field

Aliases

- `is_status_v3`

property isPSA : `bool`

Default value False

BooleanField field

Aliases

- `is_psa`

property statusV3TextBg : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property isSentByMe : `bool`

Default value False

BooleanField field

Aliases

- `is_sent_by_me`

property isNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_notification`

property isGroupNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_group_notification`

property isBizNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_biz_notification`

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- `is_media`

property isLink : `bool`

Default value False

BooleanField field

Aliases

- `is_link`

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- `has_link`

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- `is_doc`

property isMms : `bool`

Default value False

BooleanField field

Aliases

- `is_mms`

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- `is_revoked`

property showForwarded : `bool`

Default value False

BooleanField field

Aliases

- `show_forwarded`

property containsEmoji : `bool`

Default value False

BooleanField field

Aliases

- `contains_emoji`

property isFailed : `bool`

Default value False

BooleanField field

Aliases

- `is_failed`

property dir : `str`

StringIdField field

property rtl : `bool`

Default value False

BooleanField field

property isPersistent : `bool`

Default value False

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value False

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value False

BooleanField field

Aliases

- `has_promises`

property `pageCount` : `int`

IntegerField field

Aliases

- `page_count`

Model `whalesong.managers.message.VCardMessage` (*data=None, *args, **kwargs*)

Bases: `whalesong.managers.message.QuotedMessageMixin`, `whalesong.managers.message.MentionsMixin`, `whalesong.managers.message.BaseMessage`

vCard message.

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `quotedMsgObj` : `BaseMessage`

ModelField field (`whalesong.managers.message.BaseMessage`)

Aliases

- `quoted_msg_obj`

property `quotedStanzaID` : `str`

StringIdField field

Aliases

- `quoted_stanza_id`

property `quotedParticipant` : `str`

StringIdField field

Aliases

- `quoted_participant`

property `quotedRemoteJid` : `str`

StringIdField field

Aliases

- `quoted_remote_jid`

property `mentionedJidList` : `List of str`
 Array of StringIdField field

Aliases

- `mentioned_jid_list`

property `type` : `MessageTypes` [READ ONLY]
 EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property `subtype` : `str`
 StringIdField field

property `body` : `str`
 StringIdField field

property `timestamp` : `datetime`
 DateTimeField field

Aliases

- `t`

property `notifyName` : `str`
 StringIdField field

Aliases

- `notify_name`

property `from` : `str`
 StringIdField field

Aliases

- `from`

property `to` : `str`
 StringIdField field

property `author` : `str`
 StringIdField field

property `sender` : `str`
 StringIdField field

property `senderObj` : `Contact`
 ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- `sender_obj`

property `self` : `str`

Default value in

StringIdField field

property `ack` : `Ack`
 EnumField field (*whalesong.managers.message.Ack*)

property `invis` : `bool`

Default value False

BooleanField field

property isNewMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_new_msg`

property star : `bool`

Default value False

BooleanField field

property isForwarded : `bool`

Default value False

BooleanField field

Aliases

- `is_forwarded`

property links : List of `str`

Array of StringIdField field

property chat : `Chat`

ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_group_msg`

property isStatusV3 : `bool`

Default value False

BooleanField field

Aliases

- `is_status_v3`

property isPSA : `bool`

Default value False

BooleanField field

Aliases

- `is_psa`

property statusV3TextBg : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property isSentByMe : bool

Default value False

BooleanField field

Aliases

- **is_sent_by_me**

property isNotification : bool

Default value False

BooleanField field

Aliases

- **is_notification**

property isGroupNotification : bool

Default value False

BooleanField field

Aliases

- **is_group_notification**

property isBizNotification : bool

Default value False

BooleanField field

Aliases

- **is_biz_notification**

property isMedia : bool

Default value False

BooleanField field

Aliases

- **is_media**

property isLink : bool

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : bool

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : bool

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : `bool`

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- **is_revoked**

property showForwarded : `bool`

Default value False

BooleanField field

Aliases

- **show_forwarded**

property containsEmoji : `bool`

Default value False

BooleanField field

Aliases

- **contains_emoji**

property isFailed : `bool`

Default value False

BooleanField field

Aliases

- **is_failed**

property dir : `str`

StringIdField field

property rtl : `bool`

Default value False

BooleanField field

property isPersistent : `bool`

Default value False

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value `False`

BooleanField field

Aliases

- `has_promises`

Model `whalesong.managers.message.MultiVCardMessage` (*data=None, *args, **kwargs*)
 Bases: `whalesong.managers.message.QuotedMessageMixin`, `whalesong.managers.message.MentionsMixin`, `whalesong.managers.message.BaseMessage`

Multi vCard message.

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `quotedMsgObj` : `BaseMessage`

ModelField field (*whalesong.managers.message.BaseMessage*)

Aliases

- `quoted_msg_obj`

property `quotedStanzaID` : `str`

StringIdField field

Aliases

- `quoted_stanza_id`

property `quotedParticipant` : `str`

StringIdField field

Aliases

- `quoted_participant`

property `quotedRemoteJid` : `str`

StringIdField field

Aliases

- `quoted_remote_jid`

property `mentionedJidList` : `List of str`

Array of StringIdField field

Aliases

- `mentioned_jid_list`

property type : *MessageTypes* [READ ONLY]
EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property subtype : *str*
StringIdField field

property body : *str*
StringIdField field

property timestamp : *datetime*
DateTimeField field

Aliases

- *t*

property notifyName : *str*
StringIdField field

Aliases

- *notify_name*

property from : *str*
StringIdField field

Aliases

- *from*

property to : *str*
StringIdField field

property author : *str*
StringIdField field

property sender : *str*
StringIdField field

property senderObj : *Contact*
ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- *sender_obj*

property self : *str*
Default value in
StringIdField field

property ack : *Ack*
EnumField field (*whalesong.managers.message.Ack*)

property invis : *bool*
Default value False
BooleanField field

property isNewMsg : *bool*
Default value False
BooleanField field

Aliases

- `is_new_msg`

property star : `bool`

Default value `False`

BooleanField field

property isForwarded : `bool`

Default value `False`

BooleanField field

Aliases

- `is_forwarded`

property links : `List of str`

Array of StringIdField field

property chat : `Chat`

ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : `bool`

Default value `False`

BooleanField field

Aliases

- `is_group_msg`

property isStatusV3 : `bool`

Default value `False`

BooleanField field

Aliases

- `is_status_v3`

property isPSA : `bool`

Default value `False`

BooleanField field

Aliases

- `is_psa`

property statusV3TextBg : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property isSentByMe : `bool`

Default value `False`

BooleanField field

Aliases

- `is_sent_by_me`

property isNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_notification`

property isGroupNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_group_notification`

property isBizNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_biz_notification`

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- `is_media`

property isLink : `bool`

Default value False

BooleanField field

Aliases

- `is_link`

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- `has_link`

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- `is_doc`

property isMms : `bool`

Default value False

BooleanField field

Aliases

- `is_mms`

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- `is_revoked`

property showForwarded : `bool`

Default value False

BooleanField field

Aliases

- `show_forwarded`

property containsEmoji : `bool`

Default value False

BooleanField field

Aliases

- `contains_emoji`

property isFailed : `bool`

Default value False

BooleanField field

Aliases

- `is_failed`

property dir : `str`

StringIdField field

property rtl : `bool`

Default value False

BooleanField field

property isPersistent : `bool`

Default value False

BooleanField field

Aliases

- `is_persistent`

property isUserCreatedType : `bool`

Default value False

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value `False`

BooleanField field

Aliases

- `has_promises`

property `vcardList` : List of `VCardItem`

Array of ModelField field (`whalesong.managers.message.VCardItem`)

Aliases

- `vcard_list`

Model `whalesong.managers.message.LocationMessage` (`data=None, *args, **kwargs`)

Bases: `whalesong.managers.message.QuotedMessageMixin`, `whalesong.managers.message.MentionsMixin`, `whalesong.managers.message.BaseMessage`

Location message.

property `body` : `None`

Base64Field field

property `lat` : `float`

FloatField field

property `lng` : `float`

FloatField field

property `loc` : `str`

StringIdField field

property `accuracy` : `int`

IntegerField field

property `speed` : `int`

IntegerField field

property `degrees` : `float`

FloatField field

property `comment` : `str`

StringIdField field

property `sequence` : `int`

IntegerField field

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `quotedMsgObj` : `BaseMessage`

ModelField field (`whalesong.managers.message.BaseMessage`)

Aliases

- `quoted_msg_obj`

property quotedStanzaID : **str**

StringIdField field

Aliases

- quoted_stanza_id

property quotedParticipant : **str**

StringIdField field

Aliases

- quoted_participant

property quotedRemoteJid : **str**

StringIdField field

Aliases

- quoted_remote_jid

property mentionedJidList : **List of str**

Array of StringIdField field

Aliases

- mentioned_jid_list

property type : *MessageTypes* [READ ONLY]

EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property subtype : **str**

StringIdField field

property timestamp : **datetime**

DateTimeField field

Aliases

- t

property notifyName : **str**

StringIdField field

Aliases

- notify_name

property from : **str**

StringIdField field

Aliases

- from

property to : **str**

StringIdField field

property author : **str**

StringIdField field

property sender : **str**

StringIdField field

property senderObj : *Contact*

ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- **sender_obj**

property self : **str**

Default value in

StringIdField field

property ack : **Ack**

EnumField field (*whalesong.managers.message.Ack*)

property invis : **bool**

Default value False

BooleanField field

property isNewMsg : **bool**

Default value False

BooleanField field

Aliases

- **is_new_msg**

property star : **bool**

Default value False

BooleanField field

property isForwarded : **bool**

Default value False

BooleanField field

Aliases

- **is_forwarded**

property links : **List of str**

Array of StringIdField field

property chat : **Chat**

ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : **bool**

Default value False

BooleanField field

Aliases

- **is_group_msg**

property isStatusV3 : **bool**

Default value False

BooleanField field

Aliases

- **is_status_v3**

property isPSA : bool

Default value False

BooleanField field

Aliases

- is_psa

property statusV3TextBg : str

StringIdField field

Aliases

- status_v3_text_bg

property isSentByMe : bool

Default value False

BooleanField field

Aliases

- is_sent_by_me

property isNotification : bool

Default value False

BooleanField field

Aliases

- is_notification

property isGroupNotification : bool

Default value False

BooleanField field

Aliases

- is_group_notification

property isBizNotification : bool

Default value False

BooleanField field

Aliases

- is_biz_notification

property isMedia : bool

Default value False

BooleanField field

Aliases

- is_media

property isLink : bool

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : bool

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : bool

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : bool

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : bool

Default value False

BooleanField field

Aliases

- **is_revoked**

property showForwarded : bool

Default value False

BooleanField field

Aliases

- **show_forwarded**

property containsEmoji : bool

Default value False

BooleanField field

Aliases

- **contains_emoji**

property isFailed : bool

Default value False

BooleanField field

Aliases

- `is_failed`

property `dir` : `str`

StringIdField field

property `rtl` : `bool`

Default value False

BooleanField field

property `isPersistent` : `bool`

Default value False

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value False

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value False

BooleanField field

Aliases

- `has_promises`

property `isLive` : `bool`

Default value False

BooleanField field

Aliases

- `is_live`

property `shareDuration` : `timedelta`

TimedeltaField field

Aliases

- `share_duration`

property `finalThumbnail` : `None`

Base64Field field

Aliases

- `final_thumbnail`

property `finalLat` : `float`

FloatField field

Aliases

- **final_lat**

property finalLng : `float`
FloatField field

Aliases

- **final_lng**

property finalAccuracy : `int`
IntegerField field

Aliases

- **final_accuracy**

property finalSpeed : `int`
IntegerField field

Aliases

- **final_speed**

property finalDegrees : `float`
FloatField field

Aliases

- **final_degrees**

property finalTimeOffset : `timedelta`
TimedeltaField field

Aliases

- **final_time_offset**

property duration : `timedelta`
TimedeltaField field

property text : `str`
StringIdField field

Model `whalesong.managers.message.PaymentMessage` (*data=None, *args, **kwargs*)

Bases: *whalesong.managers.message.BaseMessage*

Payment message.

property id : `str` [READ ONLY]
StringIdField field [READ ONLY]

property type : *MessageTypes* [READ ONLY]
EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property subtype : `str`
StringIdField field

property body : `str`
StringIdField field

property timestamp : `datetime`
DateTimeField field

Aliases

- **t**

property notifyName : **str**
StringIdField field

Aliases

- **notify_name**

property from : **str**
StringIdField field

Aliases

- **from**

property to : **str**
StringIdField field

property author : **str**
StringIdField field

property sender : **str**
StringIdField field

property senderObj : **Contact**
ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- **sender_obj**

property self : **str**
Default value in
StringIdField field

property ack : **Ack**
EnumField field (*whalesong.managers.message.Ack*)

property invis : **bool**
Default value False
BooleanField field

property isNewMsg : **bool**
Default value False
BooleanField field

Aliases

- **is_new_msg**

property star : **bool**
Default value False
BooleanField field

property isForwarded : **bool**
Default value False
BooleanField field

Aliases

- `is_forwarded`

property `links` : `List of str`

Array of StringIdField field

property `chat` : `Chat`

ModelField field (*whalesong.managers.chat.Chat*)

property `isGroupMsg` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_group_msg`

property `isStatusV3` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_status_v3`

property `isPSA` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_psa`

property `statusV3TextBg` : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property `isSentByMe` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_sent_by_me`

property `isNotification` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_notification`

property `isGroupNotification` : `bool`

Default value `False`

BooleanField field

Aliases

- **is_group_notification**

property isBizNotification : `bool`

Default value False

BooleanField field

Aliases

- **is_biz_notification**

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- **is_media**

property isLink : `bool`

Default value False

BooleanField field

Aliases

- **is_link**

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- **has_link**

property isDoc : `bool`

Default value False

BooleanField field

Aliases

- **is_doc**

property isMms : `bool`

Default value False

BooleanField field

Aliases

- **is_mms**

property isRevoked : `bool`

Default value False

BooleanField field

Aliases

- **is_revoked**

property showForwarded : **bool**

Default value False

BooleanField field

Aliases

- **show_forwarded**

property containsEmoji : **bool**

Default value False

BooleanField field

Aliases

- **contains_emoji**

property isFailed : **bool**

Default value False

BooleanField field

Aliases

- **is_failed**

property dir : **str**

StringIdField field

property rtl : **bool**

Default value False

BooleanField field

property isPersistent : **bool**

Default value False

BooleanField field

Aliases

- **is_persistent**

property isUserCreatedType : **bool**

Default value False

BooleanField field

Aliases

- **is_user_created_type**

property hasPromises : **bool**

Default value False

BooleanField field

Aliases

- `has_promises`

Model `whalesong.managers.message.GroupNotificationMessage` (`data=None`, `*args`, `**kwargs`)

Bases: `whalesong.managers.message.BaseMessage`

Notification message.

property `id` : `str` [READ ONLY]
StringIdField field [READ ONLY]

property `type` : `MessageTypes` [READ ONLY]
EnumField field [READ ONLY] (`whalesong.managers.message.MessageTypes`)

property `subtype` : `str`
StringIdField field

property `body` : `str`
StringIdField field

property `timestamp` : `datetime`
DateTimeField field

Aliases

- `t`

property `notifyName` : `str`
StringIdField field

Aliases

- `notify_name`

property `from` : `str`
StringIdField field

Aliases

- `from`

property `to` : `str`
StringIdField field

property `author` : `str`
StringIdField field

property `sender` : `str`
StringIdField field

property `senderObj` : `Contact`
ModelField field (`whalesong.managers.contact.Contact`)

Aliases

- `sender_obj`

property `self` : `str`
Default value in

StringIdField field

property `ack` : `Ack`
EnumField field (`whalesong.managers.message.Ack`)

property `invis` : `bool`

Default value False

BooleanField field

property isNewMsg : **bool**

Default value False

BooleanField field

Aliases

- **is_new_msg**

property star : **bool**

Default value False

BooleanField field

property isForwarded : **bool**

Default value False

BooleanField field

Aliases

- **is_forwarded**

property links : **List of str**

Array of StringIdField field

property chat : *Chat*

ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : **bool**

Default value False

BooleanField field

Aliases

- **is_group_msg**

property isStatusV3 : **bool**

Default value False

BooleanField field

Aliases

- **is_status_v3**

property isPSA : **bool**

Default value False

BooleanField field

Aliases

- **is_psa**

property statusV3TextBg : **str**

StringIdField field

Aliases

- `status_v3_text_bg`

property isSentByMe : `bool`

Default value False

BooleanField field

Aliases

- `is_sent_by_me`

property isNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_notification`

property isGroupNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_group_notification`

property isBizNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_biz_notification`

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- `is_media`

property isLink : `bool`

Default value False

BooleanField field

Aliases

- `is_link`

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- `has_link`

property isDoc : bool

Default value False

BooleanField field

Aliases

- is_doc

property isMms : bool

Default value False

BooleanField field

Aliases

- is_mms

property isRevoked : bool

Default value False

BooleanField field

Aliases

- is_revoked

property showForwarded : bool

Default value False

BooleanField field

Aliases

- show_forwarded

property containsEmoji : bool

Default value False

BooleanField field

Aliases

- contains_emoji

property isFailed : bool

Default value False

BooleanField field

Aliases

- is_failed

property dir : str

StringIdField field

property rtl : bool

Default value False

BooleanField field

property isPersistent : bool

Default value False

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value False

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value False

BooleanField field

Aliases

- `has_promises`

property `urlText` : `str`

StringIdField field

Aliases

- `url_text`

property `urlNumber` : `int`

IntegerField field

Aliases

- `url_number`

property `recipients` : List of `str`

Array of StringIdField field

property `broadcast` : `bool`

Default value False

BooleanField field

property `multicast` : `bool`

Default value False

BooleanField field

Model `whalesong.managers.message.StickerMessage` (`data=None`, `*args`, `**kwargs`)

Bases: `whalesong.managers.message.ImageMessage`

Sticker message.

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `quotedMsgObj` : `BaseMessage`

ModelField field (`whalesong.managers.message.BaseMessage`)

Aliases

- `quoted_msg_obj`

property `quotedStanzaID` : `str`
StringIdField field

Aliases

- `quoted_stanza_id`

property `quotedParticipant` : `str`
StringIdField field

Aliases

- `quoted_participant`

property `quotedRemoteJid` : `str`
StringIdField field

Aliases

- `quoted_remote_jid`

property `mentionedJidList` : List of `str`
Array of StringIdField field

Aliases

- `mentioned_jid_list`

property `type` : `MessageTypes` [READ ONLY]
EnumField field [READ ONLY] (*whalesong.managers.message.MessageTypes*)

property `clientId` : `str`
StringIdField field

Aliases

- `client_url`

property `directPath` : `str`
StringIdField field

Aliases

- `direct_path`

property `mimetype` : `str`
StringIdField field

property `caption` : `str`
StringIdField field

property `filehash` : `str`
StringIdField field

property `size` : `int`
IntegerField field

property `mediaKey` : `str`
StringIdField field

Aliases

- `media_key`

property `isUnsentMedia` : `bool`

Default value False

BooleanField field

Aliases

- **is_unsent_media**

property body : None

Base64Field field

property height : int

IntegerField field

property width : int

IntegerField field

property subtype : str

StringIdField field

property timestamp : datetime

DateTimeField field

Aliases

- **t**

property notifyName : str

StringIdField field

Aliases

- **notify_name**

property from : str

StringIdField field

Aliases

- **from**

property to : str

StringIdField field

property author : str

StringIdField field

property sender : str

StringIdField field

property senderObj : *Contact*

ModelField field (*whalesong.managers.contact.Contact*)

Aliases

- **sender_obj**

property self : str

Default value in

StringIdField field

property ack : *Ack*

EnumField field (*whalesong.managers.message.Ack*)

property invis : bool

Default value False

BooleanField field

property isNewMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_new_msg`

property star : `bool`

Default value False

BooleanField field

property isForwarded : `bool`

Default value False

BooleanField field

Aliases

- `is_forwarded`

property links : `List of str`

Array of StringIdField field

property chat : `Chat`

ModelField field (*whalesong.managers.chat.Chat*)

property isGroupMsg : `bool`

Default value False

BooleanField field

Aliases

- `is_group_msg`

property isStatusV3 : `bool`

Default value False

BooleanField field

Aliases

- `is_status_v3`

property isPSA : `bool`

Default value False

BooleanField field

Aliases

- `is_psa`

property statusV3TextBg : `str`

StringIdField field

Aliases

- `status_v3_text_bg`

property isSentByMe : `bool`

Default value False

BooleanField field

Aliases

- `is_sent_by_me`

property isNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_notification`

property isGroupNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_group_notification`

property isBizNotification : `bool`

Default value False

BooleanField field

Aliases

- `is_biz_notification`

property isMedia : `bool`

Default value False

BooleanField field

Aliases

- `is_media`

property isLink : `bool`

Default value False

BooleanField field

Aliases

- `is_link`

property hasLink : `bool`

Default value False

BooleanField field

Aliases

- `has_link`

property isDoc : bool

Default value False

BooleanField field

Aliases

- is_doc

property isMms : bool

Default value False

BooleanField field

Aliases

- is_mms

property isRevoked : bool

Default value False

BooleanField field

Aliases

- is_revoked

property showForwarded : bool

Default value False

BooleanField field

Aliases

- show_forwarded

property containsEmoji : bool

Default value False

BooleanField field

Aliases

- contains_emoji

property isFailed : bool

Default value False

BooleanField field

Aliases

- is_failed

property dir : str

StringIdField field

property rtl : bool

Default value False

BooleanField field

property isPersistent : bool

Default value False

BooleanField field

Aliases

- `is_persistent`

property `isUserCreatedType` : `bool`

Default value False

BooleanField field

Aliases

- `is_user_created_type`

property `hasPromises` : `bool`

Default value False

BooleanField field

Aliases

- `has_promises`

Model `whalesong.managers.message.MessageAck` (`data=None`, `flat=False`, `*args`, `**kwargs`)

Bases: `whalesong.models.BaseModel`

Message acknowledgement.

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `timestamp` : `datetime`

Ack timestamp.

Aliases

- `t`

Model `whalesong.managers.message.MessageInfo` (`data=None`, `flat=False`, `*args`, `**kwargs`)

Bases: `whalesong.models.BaseModel`

Message information.

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `deliveryRemaining` : `int`

Default value 0

IntegerField field

Aliases

- `delivery_remaining`

property `isPtt` : `bool`

Default value False

BooleanField field

Aliases

- `is_ptt`

property `playedRemaining` : `int`

Default value 0

IntegerField field

Aliases

- `played_remaining`

property `readRemaining` : `int`

Default value 0

IntegerField field

Aliases

- `read_remaining`

property `delivery` : List of *MessageAck*

Delivery message acknowledgement list.

property `played` : List of *MessageAck*

Played message acknowledgement list.

property `read` : List of *MessageAck*

Read message acknowledgement list.

await `whalesong.managers.message.download_media(driver, model)`

Download message's attached media file. It will decrypt media file using key on message object.

Parameters

- **driver** (*BaseWhalesongDriver*) –
- **model** (*MediaMixin*) – *MediaMixin*

Return type `BytesIO`

Returns Media stream.

3.4.9 Group metadata references

3.4.9.1 Managers

class `whalesong.managers.group_metadata.GroupMetadataCollectionManager` (*driver*,
manager_path=")

Bases: *whalesong.managers.BaseCollectionManager*

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters `name (str)` – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters `item_id (str)` – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_commands ()

Get manager available static commands.

Return type `Result[List[str]]`

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters `item_id (str)` – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_result_class ()

Return type `Result[+MODEL_TYPE]`

get_items ()

Get all items on collection.

Return type `IteratorResult[+MODEL_TYPE]`

Returns Async iterator

get_iterator_result_class ()

Return type `IteratorResult[+MODEL_TYPE]`

get_last ()

Get last item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_length ()

Get collection items count.

Return type *Result*[int]

Returns Items count

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()

Monitor add item collection. Iterate each time a item is added to collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_change()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field(*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model object iterator

monitor_remove()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id(*item_id*)

Remove item by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[None]

remove_submanager(*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_MANAGER_CLASS

alias of *GroupMetadataManager*

class whalesong.managers.group_metadata.**GroupMetadataManager**(*driver*, *manager_path*=")

Bases: *whalesong.managers.BaseModelManager*

Group metadata manager. It allows manage groups, further than a chat.

participants*ParticipantCollectionManager*

Group's participants collection manager.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***__getattr__** (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***add_submanager** (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]**Returns** Manager static commands.**get_field_monitor_result_class** (*field*)**Return type** *MonitorResult*[*Dict*[*str*, *Any*]]**get_model** ()

Get model object

Return type *Result*[+*MODEL_TYPE*]**Returns** Model object**get_model_result_class** ()**Return type** *Result*[+*MODEL_TYPE*]**get_monitor_result_class** ()**Return type** *MonitorResult*[+*MODEL_TYPE*]**get_submanager** (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***map_model** (*data*)**Return type** +*MODEL_TYPE***monitor_field** (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[Dict[str, Any]]

Returns Model monitor

monitor_model()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager(name)

Remove a submanager.

Parameters **name** (str) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_CLASS

alias of *GroupMetadata*

group_invite_code()

Return type *Result*[None]

revoke_group_invite()

Return type *Result*[None]

class whalesong.managers.group_metadata.**ParticipantCollectionManager** (*driver*,
manager_path=")

Bases: *whalesong.managers.BaseCollectionManager*

Participant collection manager. It allows manage group participants.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (str) – Manager prefix path.

__getitem__(name)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (str) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

__getattr__(name)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (str) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

MODEL_MANAGER_CLASS

alias of *ParticipantManager*

add_participants(contact_ids)

Return type *Result*[None]

can_add(contact_id)

Return type *Result*[bool]

remove_participants(contact_ids)

Return type `Result[None]`

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

can_remove (*contact_id*)

Return type `Result[bool]`

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_commands ()

Get manager available static commands.

Return type `Result[List[str]]`

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_result_class ()

Return type `Result[+MODEL_TYPE]`

get_items ()

Get all items on collection.

Return type `IteratorResult[+MODEL_TYPE]`

Returns Async iterator

get_iterator_result_class ()

Return type `IteratorResult[+MODEL_TYPE]`

get_last ()

Get last item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_length()
Get collection items count.
Return type `Result[int]`
Returns Items count

get_monitor_result_class()
Return type `MonitorResult[+MODEL_TYPE]`

get_submanager(name)
Get a submanager. It could be a explicit submanager or contained model manager.
Parameters **name** (`str`) – Field where submanager was stored.
Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

monitor_add()
Monitor add item collection. Iterate each time a item is added to collection.
Return type `MonitorResult[+MODEL_TYPE]`
Returns Model object iterator

monitor_change()
Monitor change item collection. Iterate each time a item change in collection.
Return type `MonitorResult[+MODEL_TYPE]`
Returns Model object iterator

monitor_field(field)
Monitor item's field change. Iterate each time a field changed in any item of collection.
Return type `MonitorResult[Dict[str, Any]]`
Returns Model object iterator

monitor_remove()
Monitor remove item collection. Iterate each time a item is removed from collection.
Return type `MonitorResult[+MODEL_TYPE]`
Returns Model object iterator

remove_item_by_id(item_id)
Remove item by identifier.
Parameters **item_id** (`str`) – Model identifier.
Return type `Result[None]`

remove_submanager(name)
Remove a submanager.
Parameters **name** (`str`) – Field where submanager was stored.
Return type `Result[~T]`

promote_participants(contact_ids)
Return type `Result[None]`

can_promote(contact_id)
Return type `Result[bool]`

demote_participants(contact_ids)

Return type *Result*[None]

can_demote (*contact_id*)

Return type *Result*[bool]

class whalesong.managers.group_metadata.**ParticipantManager** (*driver*, *manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Participant manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Walesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *Participant*

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[List[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[Dict[*str*, Any]]

get_model ()

Get model object

Return type *Result*[+MODEL_TYPE]

Returns Model object

get_model_result_class ()

Return type *Result*[+MODEL_TYPE]

get_monitor_result_class ()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +MODEL_TYPE

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

3.4.9.2 Models

Model whalesong.managers.group_metadata.**Participant** (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property **id** : **str** [READ ONLY]

StringIdField field [READ ONLY]

property **isAdmin** : **bool**

Default value False

BooleanField field

Aliases

- **is_admin**

property **isSuperAdmin** : **bool**

BooleanField field

Aliases

- **is_super_admin**

Model whalesong.managers.group_metadata.**GroupMetadata** (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

Group metadata model.

```

property announce : str
    i?

property creation : datetime
    Group creation timestamp.

property desc : str
    Group description.

property owner : str
    Who made group.

property participants : List of Participant
    List of participants.

property restrict : str
    i?

property id : str [READ ONLY]
    StringIdField field [READ ONLY]

property descOwner : str
    StringIdField field

    Aliases
    • desc_owner

property descTime : datetime
    DateTimeField field

    Aliases
    • desc_time

property groupInviteLink : str
    StringIdField field

    Aliases
    • group_invite_link

property inviteCode : str
    StringIdField field

    Aliases
    • invite_code

```

3.4.10 Wap references

3.4.10.1 Managers

```

class whalesong.managers.wap.WapManager(driver, manager_path="")
    Bases: whalesong.managers.BaseModelManager

    Entry point to request data to phone.

    Parameters
    • driver (BaseWhalesongDriver) – Whalesong driver
    • manager_path (str) – Manager prefix path.

```

__getitem__ (*name*)
Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)
Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS
alias of *Wap*

query_exist (*contact_id*)
Check whether a contact identifier exists on Whatsapp or not.

Parameters **contact_id** (*str*) – Contact identifier

Return type *Result*[*bool*]

Returns *Bool*

add_submanager (*name, submanager*)
Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()
Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[*Dict*[*str, Any*]]

get_model ()
Get model object

Return type *Result*[+*MODEL_TYPE*]

Returns Model object

get_model_result_class ()

Return type *Result*[+*MODEL_TYPE*]

get_monitor_result_class ()

Return type *MonitorResult*[+*MODEL_TYPE*]

get_submanager (*name*)
Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type `+MODEL_TYPE`

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (`str`) – Field to monitor.

Return type `MonitorResult[Dict[str, Any]]`

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Result[~T]`

3.4.10.2 Models

Model `whalesong.managers.wap.Wap(*args, **kwargs)`

Bases: `dirty_models.models.FastDynamicModel`

3.4.11 Sticker pack references

3.4.11.1 Managers

class `whalesong.managers.sticker_pack.StickerPackCollectionManager` (*driver*,
manager_path="")

Bases: `whalesong.managers.BaseCollectionManager`

Sticker pack collection manager. It allows manage sticker pack collection.

Parameters

- **driver** (`BaseWhalesongDriver`) – Whalesong driver
- **manager_path** (`str`) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

MODEL_MANAGER_CLASS

alias of `StickerPackManager`

fetch_page (*page*)

Fetch a sticker pack page.

Return type *Result*[~T]

fetch_all_pages ()

Fetch all sticker pack pages.

Return type *Result*[bool]

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_commands ()

Get manager available static commands.

Return type *Result*[List[*str*]]

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_result_class ()

Return type *Result*[+MODEL_TYPE]

get_items ()

Get all items on collection.

Return type *IteratorResult*[+MODEL_TYPE]

Returns Async iterator

get_iterator_result_class ()

Return type *IteratorResult*[+MODEL_TYPE]

get_last ()

Get last item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_length()
Get collection items count.

Return type *Result*[int]

Returns Items count

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)
Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()
Monitor add item collection. Iterate each time a item is added to collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_change()
Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field(field)
Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model object iterator

monitor_remove()
Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id(item_id)
Remove item by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[None]

remove_submanager(name)
Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

reset()
Reset sticker pack collection.

Return type *Result*[~T]

get_item_by_name (*name*)

Get sticker pack by name.

Parameters **name** (*str*) – Sticker pack name.

Return type *Result[StickerPack]*

Returns Sticker pack object.

class whalesong.managers.sticker_pack.**StickerPackManager** (*driver*, *manager_path=""*) *man-*

Bases: *whalesong.managers.BaseModelManager*

Sticker pack manager. It allows manage a sticker pack.

stickers

StickerCollectionManager

Sticker collection manager.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *StickerPack*

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult[Dict[str, Any]]*

get_model ()

Get model object

Return type *Result[+MODEL_TYPE]*

Returns Model object

get_model_result_class ()

Return type *Result[+MODEL_TYPE]*

get_monitor_result_class ()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +MODEL_TYPE

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[Dict[*str*, Any]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

class whalesong.managers.sticker_pack.**StickerCollectionManager** (*driver*, *manager_path*=")

Bases: *whalesong.managers.BaseCollectionManager*

Sticker collection manager. It allows manage sticker collection.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type Union[*BaseManager*, ~MODEL_MANAGER_TYPE]

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type Union[*BaseManager*, ~MODEL_MANAGER_TYPE]

MODEL_MANAGER_CLASS

alias of *StickerManager*

fetch ()

Fetch all stickers. You must fetch stickers before try to us them.

Return type *Result*[None]

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_result_class ()

Return type *Result*[+MODEL_TYPE]

get_items ()

Get all items on collection.

Return type *IteratorResult*[+MODEL_TYPE]

Returns Async iterator

get_iterator_result_class ()

Return type *IteratorResult*[+MODEL_TYPE]

get_last ()

Get last item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_length ()

Get collection items count.

Return type *Result*[*int*]

Returns Items count

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()

Monitor add item collection. Iterate each time a item is added to collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_change()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field (*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model object iterator

monitor_remove()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[None]

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

class whalesong.managers.sticker_pack.**StickerManager** (*driver*, *manager_path*=")

Bases: *whalesong.managers.BaseModelManager*

Sticker manager. It allows manage a sticker.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *Sticker*

send_to_chat (*chat_id*, *quoted_msg_id=None*)

Send this sticker to a chat.

Parameters

- **chat_id** (*str*) – Chat identifier where to send sticker.
- **quoted_msg_id** (*Optional[str]*) – Quoted message identifier.

Return type *Result[str]*

Returns Message identifier.

await download_image ()

Download sticker's image file. It will decrypt image file using key on sticker object.

Return type *BytesIO*

Returns Image stream

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult[Dict[str, Any]]*

get_model ()

Get model object

Return type *Result[+MODEL_TYPE]*

Returns Model object

get_model_result_class ()

Return type *Result[+MODEL_TYPE]*

get_monitor_result_class ()

Return type *MonitorResult[+MODEL_TYPE]*

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +MODEL_TYPE

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

3.4.11.2 Models

Model `whalesong.managers.sticker_pack.Sticker` (*data=None, flat=False, *args, **kwargs*)

Bases: `whalesong.managers.message.MediaMixin`, `whalesong.models.BaseModel`

property type : *MessageTypes* [READ ONLY]

Default value `MessageTypes.STICKER`

EnumField field [READ ONLY] (`whalesong.managers.message.MessageTypes`)

property id : *str* [READ ONLY]

StringIdField field [READ ONLY]

property clientId : *str*

StringIdField field

Aliases

- `client_url`

property directPath : *str*

StringIdField field

Aliases

- `direct_path`

property mimeType : *str*

StringIdField field

property caption : *str*

StringIdField field

property `filehash` : `str`
StringIdField field

property `size` : `int`
IntegerField field

property `mediaKey` : `str`
StringIdField field

Aliases

- `media_key`

property `isUnsentMedia` : `bool`

Default value `False`

BooleanField field

Aliases

- `is_unsent_media`

Model `whalesong.managers.sticker_pack.StickerPack` (`data=None`, `flat=False`, `*args`,
`**kwargs`)

Bases: `whalesong.models.BaseModel`

property `name` : `str`
Sticker pack name.

property `url` : `str`
Sticker pack url image.

property `id` : `str` [READ ONLY]
StringIdField field [READ ONLY]

3.4.12 Connection references

3.4.12.1 Managers

class `whalesong.managers.conn.ConnManager` (`driver`, `manager_path=""`)

Bases: `whalesong.managers.BaseModelManager`

Parameters

- **driver** (`BaseWhalesongDriver`) – Whalesong driver
- **manager_path** (`str`) – Manager prefix path.

__getitem__ (`name`)
Get a submanager.

Parameters `name` (`str`) – Field where submanager was stored.

Return type `BaseManager`

__getattr__ (`name`)
Get a submanager.

Parameters `name` (`str`) – Field where submanager was stored.

Return type `BaseManager`

add_submanager (`name`, `submanager`)
Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]**Returns** Manager static commands.**get_field_monitor_result_class** (*field*)**Return type** *MonitorResult*[*Dict*[*str*, *Any*]]**get_model** ()

Get model object

Return type *Result*[+*MODEL_TYPE*]**Returns** Model object**get_model_result_class** ()**Return type** *Result*[+*MODEL_TYPE*]**get_monitor_result_class** ()**Return type** *MonitorResult*[+*MODEL_TYPE*]**get_submanager** (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***map_model** (*data*)**Return type** +*MODEL_TYPE***monitor_field** (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.**Return type** *MonitorResult*[*Dict*[*str*, *Any*]]**Returns** Model monitor**monitor_model** ()

Monitor any change on model.

Return type *MonitorResult*[+*MODEL_TYPE*]**Returns** Model monitor**remove_submanager** (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *Result*[~*T*]**MODEL_CLASS**alias of *Conn*

update_pushname (*name*)

Update user's push name.

Parameters **name** (*str*) – New push name

Return type *Result*[None]

can_update_pushname ()

Whether it is possible to update push name or not.

Return type *Result*[bool]

Returns Whether it is possible to update push name or not.

3.4.12.2 Models

Model `whalesong.managers.conn.PhoneDescription` (*data=None*, *flat=False*, **args*,
***kwargs*)

Bases: `whalesong.models.BaseModel`

property `mcc` : *str*

i?

property `mnc` : *str*

i?

property `id` : *str* [READ ONLY]

StringIdField field [READ ONLY]

property `whatsappVersion` : *str*

StringIdField field

Aliases

- `whatsapp_version`
- `wa_version`

property `osVersion` : *str*

StringIdField field

Aliases

- `os_version`

property `deviceManufacturer` : *str*

StringIdField field

Aliases

- `device_manufacturer`

property `deviceModel` : *str*

StringIdField field

Aliases

- `device_model`

property `osBuildNumber` : *str*

StringIdField field

Aliases

- `os_build_number`

Model `whalesong.managers.conn.Conn` (*data=None, flat=False, *args, **kwargs*)

Bases: `whalesong.models.BaseModel`

property `ref` : `str`

Client token reference. Used on QR.

property `refTTL` : `int`

Ref time to live

property `connected` : `bool`

Whether is connected or not.

property `me` : `str`

Whatsapp user identifier.

property `id` : `str` [READ ONLY]

StringIdField field [READ ONLY]

property `whatsappId` : `str`

StringIdField field

Aliases

- `whatsapp_id`
- `wid`

property `protoVersion` : List of `int`

Array of IntegerField field

Aliases

- `proto_version`

property `clientToken` : `str`

StringIdField field

Aliases

- `client_token`

property `serverToken` : `str`

StringIdField field

Aliases

- `server_token`

property `isResponse` : `bool`

BooleanField field

Aliases

- `is_response`

property `smbTos` : `int`

IntegerField field

Aliases

- `smb_tos`

property `battery` : `int`

Phone battery level, in percentage.

property `plugged` : `bool`

Whether phone is plugged to charger or not.

property locale : `str`

Phone locale.

Aliases

- `lc`

property language : `str`

Phone language.

Aliases

- `lg`

property locales : `str`

Phone locale-language.

property is_24h : `bool`

Whether time must be in 24h format.

Aliases

- `is24h`

property platform : `str`

Platform (android, iphone, wp7, etc...)

property phone : *PhoneDescription*

Phone description.

property tos : `int`

?

property pushname : `str`

Current user's push name.

3.4.13 Stream references

3.4.13.1 Managers

class whalesong.managers.stream.**StreamManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (`str`) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type *BaseManager*

add_submanager (*name*, *submanager*)
Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()
Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

get_model ()
Get model object

Return type *Result*[+*MODEL_TYPE*]

Returns Model object

get_model_result_class ()

Return type *Result*[+*MODEL_TYPE*]

get_monitor_result_class ()

Return type *MonitorResult*[+*MODEL_TYPE*]

get_submanager (*name*)
Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +*MODEL_TYPE*

monitor_field (*field*)
Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()
Monitor any change on model.

Return type *MonitorResult*[+*MODEL_TYPE*]

Returns Model monitor

remove_submanager (*name*)
Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~*T*]

MODEL_CLASSalias of *Stream***poke()**

Refresh ref field. It is used to refresh QR image when it expires.

Return type *Result*[None]**takeover()**

Refresh login. It is used to take session again when other browser has been started session.

Return type *Result*[None]**logout()**

Logs out of currently logged in session

Return type *Result*[None]

3.4.13.2 Models

Model `whalesong.managers.stream.Stream(data=None, flat=False, *args, **kwargs)`Bases: *whalesong.models.BaseModel*

Connection stream model.

class StateBases: `enum.Enum`

Connection states.

OPENING = 'OPENING'

Opening stream.

PAIRING = 'PAIRING'

Pairing WhatsappWeb with a phone.

UNPAIRED = 'UNPAIRED'

Unpaired WhatsappWeb with a phone. QR is available.

UNPAIRED_IDLE = 'UNPAIRED_IDLE'

Unpaired WhatsappWeb with a phone. QR is not available.

CONNECTED = 'CONNECTED'

WhatsappWeb is connected to a phone.

TIMEOUT = 'TIMEOUT'

WhatsappWeb connection to a phone is timeout.

CONFLICT = 'CONFLICT'

Other browser has initiated WhatsappWeb with same phone.

UNLAUNCHED = 'UNLAUNCHED'

WhatsappWeb application has not been launched.

PROXYBLOCK = 'PROXYBLOCK'

Proxy is blocking connection.

TOS_BLOCK = 'TOS_BLOCK'

?

SMB_TOS_BLOCK = 'SMB_TOS_BLOCK'

?

```

class Stream
    Bases: enum.Enum

    An enumeration.

    DISCONNECTED = 'DISCONNECTED'
        Stream disconnected.

    SYNCING = 'SYNCING'
        Synchronizing data with phone.

    RESUMING = 'RESUMING'
        Resuming connection with phone.

    CONNECTED = 'CONNECTED'
        Connected to phone.

    property id : str [READ ONLY]
        StringIdField field [READ ONLY]

    property backoffGeneration : int
        IntegerField field

        Aliases
        • backoff_generation

    property canSend : bool
        BooleanField field

        Aliases
        • can_send

    property hasSynced : bool
        BooleanField field

        Aliases
        • has_synced

    property isIncognito : bool
        BooleanField field

        Aliases
        • is_incognito

    property retryTimestamp : datetime
        DateTimeField field

        Aliases
        • retry_timestamp

    property syncTag : str
        StringIdField field

        Aliases
        • sync_tag

    launch_generation = None
        ⓘ?

    property launched : bool
        Whether it has been launched.

```

property state : *State*
Current stream connection state.

property stream : *Stream*
Current stream state

3.4.14 Presence references

3.4.14.1 Managers

class whalesong.managers.presence.**PresenceCollectionManager** (*driver*, *manager_path=""*)
Bases: *whalesong.managers.BaseCollectionManager*
Presence collection manager.

Note: Be aware Whatsapp has some limitations about presence announcement.

- You must be available in order to receive presence announcements. Look at *display information*.
 - After a while, other peer must see Whalesong user available in order to send presence announcement. But if it is not available Whalesong user is not going to announce its presence. So, presence announcements are made after first message.
-

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)
Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

__getattr__ (*name*)
Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

MODEL_MANAGER_CLASS
alias of *PresenceManager*

add_submanager (*name, submanager*)
Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)
Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_commands()

Get manager available static commands.

Return type *Result*[List[str]]

Returns Manager static commands.

get_first()

Get first item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_by_id(item_id)

Get model by identifier.

Parameters *item_id* (str) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_result_class()

Return type *Result*[+MODEL_TYPE]

get_items()

Get all items on collection.

Return type *IteratorResult*[+MODEL_TYPE]

Returns Async iterator

get_iterator_result_class()

Return type *IteratorResult*[+MODEL_TYPE]

get_last()

Get last item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_length()

Get collection items count.

Return type *Result*[int]

Returns Items count

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters *name* (str) – Field where submanager was stored.

Return type Union[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()

Monitor add item collection. Iterate each time a item is added to collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_change ()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field (*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[Dict[str, Any]]

Returns Model object iterator

monitor_remove ()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters *item_id* (str) – Model identifier.

Return type *Result*[None]

remove_submanager (*name*)

Remove a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *Result*[~T]

class whalesong.managers.presence.**PresenceManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Presence manager.

__getitem__ (*name*)

Get a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *Presence*

subscribe ()

Return type *Result*[*Presence*]

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]**Returns** Manager static commands.**get_field_monitor_result_class** (*field*)**Return type** *MonitorResult*[*Dict*[*str*, *Any*]]**get_model** ()

Get model object

Return type *Result*[+*MODEL_TYPE*]**Returns** Model object**get_model_result_class** ()**Return type** *Result*[+*MODEL_TYPE*]**get_monitor_result_class** ()**Return type** *MonitorResult*[+*MODEL_TYPE*]**get_submanager** (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***map_model** (*data*)**Return type** +*MODEL_TYPE***monitor_field** (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.**Return type** *MonitorResult*[*Dict*[*str*, *Any*]]**Returns** Model monitor**monitor_model** ()

Monitor any change on model.

Return type *MonitorResult*[+*MODEL_TYPE*]**Returns** Model monitor**remove_submanager** (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *Result*[~*T*]

```
class whalesong.managers.presence.ChatStateCollectionManager (driver, manager_path="")
```

Bases: *whalesong.managers.BaseCollectionManager*

Chat state collection manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

```
__getitem__ (name)
```

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

```
__getattr__ (name)
```

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

MODEL_MANAGER_CLASS

alias of *ChatStateManager*

```
add_submanager (name, submanager)
```

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

```
find_item_by_id (item_id)
```

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result[+MODEL_TYPE]*

Returns Model object.

```
get_commands ()
```

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

```
get_first ()
```

Get first item in collection.

Return type *Result[+MODEL_TYPE]*

Returns Model object.

```
get_item_by_id (item_id)
```

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result[+MODEL_TYPE]*

Returns Model object.

get_item_result_class()
 Return type *Result*[+MODEL_TYPE]

get_items()
 Get all items on collection.
 Return type *IteratorResult*[+MODEL_TYPE]
 Returns Async iterator

get_iterator_result_class()
 Return type *IteratorResult*[+MODEL_TYPE]

get_last()
 Get last item in collection.
 Return type *Result*[+MODEL_TYPE]
 Returns Model object.

get_length()
 Get collection items count.
 Return type *Result*[int]
 Returns Items count

get_monitor_result_class()
 Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)
 Get a submanager. It could be a explicit submanager or contained model manager.
 Parameters **name** (*str*) – Field where submanager was stored.
 Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()
 Monitor add item collection. Iterate each time a item is added to collection.
 Return type *MonitorResult*[+MODEL_TYPE]
 Returns Model object iterator

monitor_change()
 Monitor change item collection. Iterate each time a item change in collection.
 Return type *MonitorResult*[+MODEL_TYPE]
 Returns Model object iterator

monitor_field(field)
 Monitor item's field change. Iterate each time a field changed in any item of collection.
 Return type *MonitorResult*[*Dict*[*str*, *Any*]]
 Returns Model object iterator

monitor_remove()
 Monitor remove item collection. Iterate each time a item is removed from collection.
 Return type *MonitorResult*[+MODEL_TYPE]
 Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[None]

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

class whalesong.managers.presence.**ChatStateManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Chat state manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *ChatState*

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[List[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[Dict[*str*, Any]]

get_model ()

Get model object

Return type *Result*[+MODEL_TYPE]

Returns Model object

get_model_result_class()
 Return type `Result[+MODEL_TYPE]`

get_monitor_result_class()
 Return type `MonitorResult[+MODEL_TYPE]`

get_submanager(name)
 Get a submanager.
 Parameters **name** (`str`) – Field where submanager was stored.
 Return type `BaseManager`

map_model(data)
 Return type `+MODEL_TYPE`

monitor_field(field)
 Monitor any change on a model's field.
 Parameters **field** (`str`) – Field to monitor.
 Return type `MonitorResult[Dict[str, Any]]`
 Returns Model monitor

monitor_model()
 Monitor any change on model.
 Return type `MonitorResult[+MODEL_TYPE]`
 Returns Model monitor

remove_submanager(name)
 Remove a submanager.
 Parameters **name** (`str`) – Field where submanager was stored.
 Return type `Result[~T]`

3.4.14.2 Models

Model `whalesong.managers.presence.ChatState` (`data=None, flat=False, *args, **kwargs`)

Bases: `whalesong.models.BaseModel`

class Type
 Bases: `enum.Enum`
 An enumeration.
AVAILABLE = 'available'
UNAVAILABLE = 'unavailable'
COMPOSING = 'composing'
RECORDING = 'recording'

property deny : `bool`
 BooleanField field

property timestamp : `datetime`
 DateTimeField field

Aliases

- `t`

property type : `Type`

EnumField field (`whalesong.managers.presence.Type`)

property id : `str` [READ ONLY]

StringIdField field [READ ONLY]

property isState : `bool`

BooleanField field

Aliases

- `is_state`

property updateTime : `datetime`

DateTimeField field

Aliases

- `update_time`

Model `whalesong.managers.presence.Presence` (`data=None, flat=False, *args, **kwargs`)

Bases: `whalesong.models.BaseModel`

Presence model.

property id : `str` [READ ONLY]

StringIdField field [READ ONLY]

property chatActive : `bool`

BooleanField field

Aliases

- `chat_active`

property chatState : `ChatState`

ModelField field (`whalesong.managers.presence.ChatState`)

Aliases

- `chat_state`

property chatStates : List of `ChatState`

Array of ModelField field (`whalesong.managers.presence.ChatState`)

Aliases

- `chat_states`

- `chatstates`

property hasData : `bool`

BooleanField field

Aliases

- `has_data`

property isGroup : `bool`

BooleanField field

Aliases

- `is_group`

property isUser : `bool`
BooleanField field

Aliases

- `is_user`

property isOnline : `bool`
BooleanField field

Aliases

- `is_online`

property isSubscribed : `bool`
BooleanField field

Aliases

- `is_subscribed`

3.4.15 Profile pictures references

3.4.15.1 Managers

class `whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager` (*driver*, *manager_path*=")

Bases: `whalesong.managers.BaseCollectionManager`

Profile picture collection manager.

Parameters

- **driver** (`BaseWhalesongDriver`) – Whalesong driver
- **manager_path** (`str`) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (`str`) – Field where manager will be stored.
- **submanager** (`BaseManager`) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (`str`) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_commands()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_first()

Get first item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_by_id(item_id)

Get model by identifier.

Parameters *item_id* (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_result_class()

Return type *Result*[+MODEL_TYPE]

get_items()

Get all items on collection.

Return type *IteratorResult*[+MODEL_TYPE]

Returns Async iterator

get_iterator_result_class()

Return type *IteratorResult*[+MODEL_TYPE]

get_last()

Get last item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_length()

Get collection items count.

Return type *Result*[*int*]

Returns Items count

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()

Monitor add item collection. Iterate each time a item is added to collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_change ()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field (*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[Dict[str, Any]]

Returns Model object iterator

monitor_remove ()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters *item_id* (str) – Model identifier.

Return type *Result*[None]

remove_submanager (*name*)

Remove a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_MANAGER_CLASS

alias of *ProfilePictureManager*

class whalesong.managers.profile_pic_thumb.**ProfilePictureManager** (*driver*, *manager_path*=")

Bases: *whalesong.managers.BaseModelManager*

Profile picture manager. It allows manage a contact picture.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (str) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASSalias of *ProfilePicture***can_set()**

Whether can set a new picture or not.

Return type *Result*[bool]**set_picture**(*picture*, *picture_preview*=None)

Set a new picture.

Return type *Result*[bool]**can_delete()**

Whether can delete picture or not.

Return type *Result*[bool]**add_submanager**(*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

delete_picture()

Delete current picture.

Return type *Result*[bool]**get_commands()**

Get manager available static commands.

Return type *Result*[List[*str*]]**Returns** Manager static commands.**get_field_monitor_result_class**(*field*)**Return type** *MonitorResult*[Dict[*str*, Any]]**get_model()**

Get model object

Return type *Result*[+MODEL_TYPE]**Returns** Model object**get_model_result_class()****Return type** *Result*[+MODEL_TYPE]**get_monitor_result_class()****Return type** *MonitorResult*[+MODEL_TYPE]**get_submanager**(*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***map_model**(*data*)**Return type** +MODEL_TYPE

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+*MODEL_TYPE*]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~*T*]

3.4.15.2 Models

Model `whalesong.managers.profile_pic_thumb.ProfilePicture` (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property eurl : *str*

Url to contact picture.

property raw : *str*

?

property tag : *str*

?

Note: I guess it is used to know when contact picture changed.

property id : *str* [*READ ONLY*]

StringIdField field [*READ ONLY*]

3.4.16 Status references

3.4.16.1 Managers

class `whalesong.managers.status.StatusCollectionManager` (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseCollectionManager*

Status collection manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

MODEL_MANAGER_CLASS

alias of *StatusManager*

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_result_class ()

Return type *Result*[+MODEL_TYPE]

get_items ()

Get all items on collection.

Return type *IteratorResult*[+MODEL_TYPE]

Returns Async iterator

get_iterator_result_class ()

Return type *IteratorResult*[+MODEL_TYPE]

get_last()
Get last item in collection.
Return type *Result*[+MODEL_TYPE]
Returns Model object.

get_length()
Get collection items count.
Return type *Result*[int]
Returns Items count

get_monitor_result_class()
Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)
Get a submanager. It could be a explicit submanager or contained model manager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()
Monitor add item collection. Iterate each time a item is added to collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

monitor_change()
Monitor change item collection. Iterate each time a item change in collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

monitor_field(field)
Monitor item's field change. Iterate each time a field changed in any item of collection.
Return type *MonitorResult*[*Dict*[*str*, *Any*]]
Returns Model object iterator

monitor_remove()
Monitor remove item collection. Iterate each time a item is removed from collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

remove_item_by_id(item_id)
Remove item by identifier.
Parameters **item_id** (*str*) – Model identifier.
Return type *Result*[None]

remove_submanager(name)
Remove a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *Result*[~T]

set_my_status (*new_status=None*)

Set current user status.

Return type *Result[bool]*

Returns Operation result.

class whalesong.managers.status.**StatusManager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Status manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *Status*

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult[Dict[str, Any]]*

get_model ()

Get model object

Return type *Result[+MODEL_TYPE]*

Returns Model object

get_model_result_class ()

Return type *Result[+MODEL_TYPE]*

get_monitor_result_class ()

Return type *MonitorResult[+MODEL_TYPE]*

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +MODEL_TYPE

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

3.4.16.2 Models

Model *whalesong.managers.status.Status* (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

property status : *str*

Contact status.

property id : *str* [READ ONLY]

StringIdField field [READ ONLY]

3.4.17 Live location references

3.4.17.1 Managers

class *whalesong.managers.live_location.LiveLocationCollectionManager* (*driver,*
manager_path="")

Bases: *whalesong.managers.BaseCollectionManager*

Live locations collection manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_commands ()

Get manager available static commands.

Return type `Result[List[str]]`

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_result_class ()

Return type `Result[+MODEL_TYPE]`

get_items ()

Get all items on collection.

Return type `IteratorResult[+MODEL_TYPE]`

Returns Async iterator

get_iterator_result_class ()

Return type `IteratorResult[+MODEL_TYPE]`

get_last()
Get last item in collection.
Return type *Result*[+MODEL_TYPE]
Returns Model object.

get_length()
Get collection items count.
Return type *Result*[int]
Returns Items count

get_monitor_result_class()
Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(name)
Get a submanager. It could be a explicit submanager or contained model manager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()
Monitor add item collection. Iterate each time a item is added to collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

monitor_change()
Monitor change item collection. Iterate each time a item change in collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

monitor_field(field)
Monitor item's field change. Iterate each time a field changed in any item of collection.
Return type *MonitorResult*[*Dict*[*str*, *Any*]]
Returns Model object iterator

monitor_remove()
Monitor remove item collection. Iterate each time a item is removed from collection.
Return type *MonitorResult*[+MODEL_TYPE]
Returns Model object iterator

remove_item_by_id(item_id)
Remove item by identifier.
Parameters **item_id** (*str*) – Model identifier.
Return type *Result*[None]

remove_submanager(name)
Remove a submanager.
Parameters **name** (*str*) – Field where submanager was stored.
Return type *Result*[~T]

MODEL_MANAGER_CLASSalias of *LiveLocationManager*

```
class whalesong.managers.live_location.LiveLocationManager (driver, manager_path="")
```

Bases: *whalesong.managers.BaseModelManager*

Live location manager.

participants*ParticipantCollectionManager*

Live location's participants collection manager.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***__getattr__** (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager***add_submanager** (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result[List[str]]***Returns** Manager static commands.**get_field_monitor_result_class** (*field*)**Return type** *MonitorResult[Dict[str, Any]]***get_model** ()

Get model object

Return type *Result[+MODEL_TYPE]***Returns** Model object**get_model_result_class** ()**Return type** *Result[+MODEL_TYPE]***get_monitor_result_class** ()**Return type** *MonitorResult[+MODEL_TYPE]***get_submanager** (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.**Return type** *BaseManager*

map_model (*data*)

Return type +MODEL_TYPE

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_CLASS

alias of *LiveLocation*

subscribe ()

Subscribe to a live location. It is needed in order to receive updates.

Return type *Result*[None]

unsubscribe ()

Unsubscribe to a live location.

Return type *Result*[None]

stop_my_live_location ()

Stop to share current user live location in this chat.

Return type *Result*[None]

class whalesong.managers.live_location.**ParticipantCollectionManager** (*driver*,
manager_path=")

Bases: *whalesong.managers.BaseCollectionManager*

Participant collection manager. It allows manage live location participants.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters `name (str)` – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

add_submanager (*name, submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters `item_id (str)` – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_commands ()

Get manager available static commands.

Return type `Result[List[str]]`

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters `item_id (str)` – Model identifier.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_item_result_class ()

Return type `Result[+MODEL_TYPE]`

get_items ()

Get all items on collection.

Return type `IteratorResult[+MODEL_TYPE]`

Returns Async iterator

get_iterator_result_class ()

Return type `IteratorResult[+MODEL_TYPE]`

get_last ()

Get last item in collection.

Return type `Result[+MODEL_TYPE]`

Returns Model object.

get_length ()

Get collection items count.

Return type *Result*[int]

Returns Items count

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~MODEL_MANAGER_TYPE]

monitor_add()

Monitor add item collection. Iterate each time a item is added to collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_change()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field(*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model object iterator

monitor_remove()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id(*item_id*)

Remove item by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[None]

remove_submanager(*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_MANAGER_CLASS

alias of *ParticipantManager*

class whalesong.managers.live_location.**ParticipantManager**(*driver*, *manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

Participant manager.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver

- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

get_model ()

Get model object

Return type *Result*[+*MODEL_TYPE*]

Returns Model object

get_model_result_class ()

Return type *Result*[+*MODEL_TYPE*]

get_monitor_result_class ()

Return type *MonitorResult*[+*MODEL_TYPE*]

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +*MODEL_TYPE*

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager(name)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

MODEL_CLASS

alias of *Participant*

3.4.17.2 Models

Model whalesong.managers.live_location.**Participant** (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

Live location participant model.

property accuracy : int

Location accuracy in meters.

property comment : str

Comment sent with live location message.

property degrees : float

User direction in degrees.

property expiration : datetime

When live location will expire.

property lat : float

Latitude.

property lng : float

Longitude.

property msg : BaseMessage

Message used to start live location.

property contact : Contact

Live location owner.

property sequence : int

Sequence number.

property speed : int

User speed.

property id : str [READ ONLY]

StringIdField field [READ ONLY]

property lastUpdate : datetime

DateTimeField field

Aliases

- `last_update`

Model `whalesong.managers.live_location.LiveLocation` (`data=None`, `flat=False`, `*args`, `**kwargs`)

Bases: `whalesong.models.BaseModel`

Live location model.

property `duration` : `timedelta`
Live location duration.

property `id` : `str` [READ ONLY]
StringIdField field [READ ONLY]

property `participants` : `List of Participant`
List of participants.

3.4.18 Mute references

3.4.18.1 Managers

class `whalesong.managers.mute.MuteCollectionManager` (`driver`, `manager_path=""`)
Bases: `whalesong.managers.BaseCollectionManager`

Mutes collection manager. It allows manage global mute as well.

Parameters

- **driver** (`BaseWhalesongDriver`) – Whalesong driver
- **manager_path** (`str`) – Manager prefix path.

`__getitem__` (`name`)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters `name` (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

`__getattr__` (`name`)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters `name` (`str`) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

MODEL_MANAGER_CLASS

alias of `MuteManager`

`get_global_notifications` ()

Return type `Result[bool]`

`set_global_notifications` (`state`)

Return type `Result[None]`

`get_global_sounds` ()

Return type `Result[bool]`

`add_submanager` (`name`, `submanager`)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_first ()

Get first item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_by_id (*item_id*)

Get model by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_item_result_class ()

Return type *Result*[+MODEL_TYPE]

get_items ()

Get all items on collection.

Return type *IteratorResult*[+MODEL_TYPE]

Returns Async iterator

get_iterator_result_class ()

Return type *IteratorResult*[+MODEL_TYPE]

get_last ()

Get last item in collection.

Return type *Result*[+MODEL_TYPE]

Returns Model object.

get_length ()

Get collection items count.

Return type *Result*[*int*]

Returns Items count

get_monitor_result_class ()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type `Union[BaseManager, ~MODEL_MANAGER_TYPE]`

monitor_add ()

Monitor add item collection. Iterate each time a item is added to collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

monitor_change ()

Monitor change item collection. Iterate each time a item change in collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

monitor_field (*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type `MonitorResult[Dict[str, Any]]`

Returns Model object iterator

monitor_remove ()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters **item_id** (*str*) – Model identifier.

Return type `Result[None]`

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type `Result[~T]`

set_global_sounds (*state*)

Return type `Result[None]`

get_global_previews ()

Return type `Result[bool]`

set_global_previews (*state*)

Return type `Result[None]`

class whalesong.managers.mute.**MuteManager** (*driver, manager_path=""*)

Bases: `whalesong.managers.BaseModelManager`

Mute manager. It allows manage chat mute.

Parameters

- **driver** (`BaseWhalesongDriver`) – Whalesong driver

- **manager_path**(*str*) – Manager prefix path.

__getitem__(*name*)
Get a submanager.

Parameters **name**(*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__(*name*)
Get a submanager.

Parameters **name**(*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS
alias of *Mute*

mute(*expiration*)

Return type *Result*[None]

can_mute()

Return type *Result*[bool]

unmute()

Return type *Result*[None]

add_submanager(*name*, *submanager*)
Add a submanager.

Parameters

- **name**(*str*) – Field where manager will be stored.
- **submanager**(*BaseManager*) – Submanager

get_commands()
Get manager available static commands.

Return type *Result*[List[*str*]]

Returns Manager static commands.

get_field_monitor_result_class(*field*)

Return type *MonitorResult*[Dict[*str*, Any]]

get_model()
Get model object

Return type *Result*[+MODEL_TYPE]

Returns Model object

get_model_result_class()

Return type *Result*[+MODEL_TYPE]

get_monitor_result_class()

Return type *MonitorResult*[+MODEL_TYPE]

get_submanager(*name*)
Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +MODEL_TYPE

monitor_field (*field*)

Monitor any change on a model's field.

Parameters **field** (*str*) – Field to monitor.

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model monitor

remove_submanager (*name*)

Remove a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Result*[~T]

3.4.18.2 Models

Model `whalesong.managers.mute.Mute` (*data=None, flat=False, *args, **kwargs*)

Bases: *whalesong.models.BaseModel*

Mute model.

property **expiration** : **datetime**

Expiration date time.

property **id** : **str** [*READ ONLY*]

StringIdField field [*READ ONLY*]

property **isMuted** : **bool**

BooleanField field

Aliases

- **is_muted**

property **isState** : **bool**

BooleanField field

Aliases

- **is_state**

3.4.19 Display information references

3.4.19.1 Managers

class whalesong.managers.display_info.**DisplayInfoManager** (*driver*, *manager_path=""*) *man-*

Bases: *whalesong.managers.BaseModelManager*

Manage display information.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result[List[str]]*

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult[Dict[str, Any]]*

get_model ()

Get model object

Return type *Result[+MODEL_TYPE]*

Returns Model object

get_model_result_class ()

Return type *Result[+MODEL_TYPE]*

get_monitor_result_class ()

Return type *MonitorResult[+MODEL_TYPE]*

get_submanager (*name*)

Get a submanager.

Parameters `name (str)` – Field where submanager was stored.

Return type `BaseManager`

map_model (`data`)

Return type `+MODEL_TYPE`

monitor_field (`field`)

Monitor any change on a model's field.

Parameters `field (str)` – Field to monitor.

Return type `MonitorResult[Dict[str, Any]]`

Returns Model monitor

monitor_model ()

Monitor any change on model.

Return type `MonitorResult[+MODEL_TYPE]`

Returns Model monitor

remove_submanager (`name`)

Remove a submanager.

Parameters `name (str)` – Field where submanager was stored.

Return type `Result[~T]`

MODEL_CLASS

alias of `DisplayInfo`

mark_available ()

Mark current user as available. It is need to get presence from other users.

Return type `Result[None]`

mark_unavailable ()

Mark current user as unavailable.

Return type `Result[None]`

unobscure ()

Unobscure display.

Return type `Result[None]`

set_available_permanent ()

Set user available permanently. It starts a loop in order to set availability each 30 seconds.

unset_available_permanent ()

Unset user available permanently. It stops permanent availability loop.

is_available_permanent ()

Checks whether permanent availability loop is running.

Returns Permanent availability loop state.

3.4.19.2 Models

Model `whalesong.managers.display_info.DisplayInfo (data=None, flat=False, *args, **kwargs)`

Bases: `whalesong.models.BaseModel`

Connection stream model.

```
class StreamInfo
```

Bases: `enum.Enum`

Stream information.

```
OFFLINE = 'OFFLINE'
```

Offline.

```
OPENING = 'OPENING'
```

Opening connection.

```
PAIRING = 'PAIRING'
```

Pairing phone.

```
SYNCING = 'SYNCING'
```

Synchronizing data.

```
RESUMING = 'RESUMING'
```

Resuming connection.

```
CONNECTING = 'CONNECTING'
```

Connecting.

```
NORMAL = 'NORMAL'
```

Normal.

```
TIMEOUT = 'TIMEOUT'
```

Connection timeout.

```
class StreamMode
```

Bases: `enum.Enum`

Stream mode.

```
QR = 'QR'
```

Wait for QR scan.

```
MAIN = 'MAIN'
```

Main.

```
SYNCING = 'SYNCING'
```

Synchronizing data.

```
OFFLINE = 'OFFLINE'
```

Not connected.

```
CONFLICT = 'CONFLICT'
```

Other browser has opened session.

```
PROXYBLOCK = 'PROXYBLOCK'
```

Proxy blocks connection.

```
TOS_BLOCK = 'TOS_BLOCK'
```

i?

```
SMB_TOS_BLOCK = 'SMB_TOS_BLOCK'
```

i?

```
DEPRECATED_VERSION = 'DEPRECATED_VERSION'
```

Using a deprecated version.

```
class DisplayState
    Bases: enum.Enum

    Display state.

    SHOW = 'SHOW'
        Display showing.

    OBSCURE = 'OBSCURE'
        Display obscured.

    HIDE = 'HIDE'
        Display hidden.

property available : bool
    Default value False

    Whether current user is available.

property id : str [READ ONLY]
    StringIdField field [READ ONLY]

property clientExpired : bool
    BooleanField field

    Aliases
        • client_expired

property couldForce : bool
    BooleanField field

    Aliases
        • could_force

property displayInfo : StreamInfo
    EnumField field (whalesong.managers.display_info.StreamInfo)

    Aliases
        • display_info

property isState : bool
    BooleanField field

    Aliases
        • is_state

property phoneAuthed : bool
    BooleanField field

    Aliases
        • phone_authed

property uiActive : bool
    BooleanField field

    Aliases
        • ui_active

property resumeCount : int
    IntegerField field
```


Aliases

- **resume_count**

hard_expired = None

?

property info : *StreamInfo*

Same than display info?

property obscurity : *DisplayState*

Current display state.

property mode : *StreamMode*

Stream mode.

3.4.20 Status V3 references

3.4.20.1 Managers

class whalesong.managers.status_v3.**StatusV3CollectionManager** (*driver*, *manager_path*=")

Bases: *whalesong.managers.BaseCollectionManager*

Manage a collection of StatusV3.

Parameters

- **driver** (*BaseWhalesongDriver*) – Whalesong driver
- **manager_path** (*str*) – Manager prefix path.

__getitem__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

__getattr__ (*name*)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *Union[BaseManager, ~MODEL_MANAGER_TYPE]*

MODEL_MANAGER_CLASS

alias of *StatusV3Manager*

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

find_item_by_id (*item_id*)

Find model by identifier. If item is not in collection it will try to load it.

Parameters **item_id** (*str*) – Model identifier.

Return type *Result[+MODEL_TYPE]*

Returns Model object.

get_commands()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_first()

Get first item in collection.

Return type *Result*[+*MODEL_TYPE*]

Returns Model object.

get_item_by_id(item_id)

Get model by identifier.

Parameters *item_id* (*str*) – Model identifier.

Return type *Result*[+*MODEL_TYPE*]

Returns Model object.

get_item_result_class()

Return type *Result*[+*MODEL_TYPE*]

get_items()

Get all items on collection.

Return type *IteratorResult*[+*MODEL_TYPE*]

Returns Async iterator

get_iterator_result_class()

Return type *IteratorResult*[+*MODEL_TYPE*]

get_last()

Get last item in collection.

Return type *Result*[+*MODEL_TYPE*]

Returns Model object.

get_length()

Get collection items count.

Return type *Result*[*int*]

Returns Items count

get_monitor_result_class()

Return type *MonitorResult*[+*MODEL_TYPE*]

get_submanager(name)

Get a submanager. It could be a explicit submanager or contained model manager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *Union*[*BaseManager*, ~*MODEL_MANAGER_TYPE*]

monitor_add()

Monitor add item collection. Iterate each time a item is added to collection.

Return type *MonitorResult*[+*MODEL_TYPE*]

Returns Model object iterator

monitor_change ()

Monitor change item collection. Iterate each time a item change in collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

monitor_field (*field*)

Monitor item's field change. Iterate each time a field changed in any item of collection.

Return type *MonitorResult*[Dict[str, Any]]

Returns Model object iterator

monitor_remove ()

Monitor remove item collection. Iterate each time a item is removed from collection.

Return type *MonitorResult*[+MODEL_TYPE]

Returns Model object iterator

remove_item_by_id (*item_id*)

Remove item by identifier.

Parameters *item_id* (str) – Model identifier.

Return type *Result*[None]

remove_submanager (*name*)

Remove a submanager.

Parameters *name* (str) – Field where submanager was stored.

Return type *Result*[~T]

get_unexpired (*unread=True*)

Get the read or unread StatusV3 collection

Parameters *unread* (bool) – List read or unread statuses

Return type *IteratorResult*[StatusV3]

Returns List of StatusV3

sync ()

Sync Statuses

Return type *Result*[None]

Returns None

get_my_status ()

Get the own user StatusV3

Return type *Result*[StatusV3]

Returns StatusV3 object

class whalesong.managers.status_v3.**StatusV3Manager** (*driver, manager_path=""*)

Bases: *whalesong.managers.BaseModelManager*

StatusV3 manager. Allow to manage a WhatsApp status.

__getitem__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

__getattr__ (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

MODEL_CLASS

alias of *StatusV3*

send_read_status (*message_id*)

Mark a statusV3 as read.

Parameters **message_id** (*str*) – Message serialized ID to be marked

Return type *Result*[*bool*]

add_submanager (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (*str*) – Field where manager will be stored.
- **submanager** (*BaseManager*) – Submanager

get_commands ()

Get manager available static commands.

Return type *Result*[*List*[*str*]]

Returns Manager static commands.

get_field_monitor_result_class (*field*)

Return type *MonitorResult*[*Dict*[*str*, *Any*]]

get_model ()

Get model object

Return type *Result*[+*MODEL_TYPE*]

Returns Model object

get_model_result_class ()

Return type *Result*[+*MODEL_TYPE*]

get_monitor_result_class ()

Return type *MonitorResult*[+*MODEL_TYPE*]

get_submanager (*name*)

Get a submanager.

Parameters **name** (*str*) – Field where submanager was stored.

Return type *BaseManager*

map_model (*data*)

Return type +*MODEL_TYPE*

monitor_field (*field*)

Monitor any change on a model's field.

Parameters `field` (`str`) – Field to monitor.

Return type `MonitorResult`[`Dict`[`str`, `Any`]]

Returns Model monitor

`monitor_model()`

Monitor any change on model.

Return type `MonitorResult`[`+MODEL_TYPE`]

Returns Model monitor

`remove_submanager` (`name`)

Remove a submanager.

Parameters `name` (`str`) – Field where submanager was stored.

Return type `Result`[`~T`]

3.4.20.2 Models

Model `whalesong.managers.status_v3.StatusV3` (`data=None`, `flat=False`, `*args`, `**kwargs`)

Bases: `whalesong.models.BaseModel`

StatusV3 model

property `contact` : `Contact`

Contact object

property `id` : `str` [`READ ONLY`]

StringIdField field [`READ ONLY`]

property `unreadCount` : `int`

IntegerField field

Aliases

- `unread_count`

property `expireTs` : `datetime`

DateTimeField field

Aliases

- `expire_ts`

property `lastReceivedKey` : `str`

StringIdField field

Aliases

- `last_received_key`

property `readKeys` : `HashMapModel` hash map which values are `str`

HashMapField field (`dirty_models.models.HashMapModel`)

Aliases

- `read_keys`

3.4.21 Storage references

3.4.21.1 Managers

class `whalesong.managers.storage.StorageManager` (*driver*, *manager_path*=")

Bases: `whalesong.managers.BaseManager`

Local storage manager. It allows manage browser local storage.

Parameters

- **driver** (`BaseWhalesongDriver`) – Whalesong driver
- **manager_path** (`str`) – Manager prefix path.

`__getitem__` (*name*)

Get a submanager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `BaseManager`

`__getattr__` (*name*)

Get a submanager.

Parameters **name** (`str`) – Field where submanager was stored.

Return type `BaseManager`

`get_storage` ()

Return type `Result[Dict[str, Any]]`

`get_item` (*key*)

Return type `Result[Any]`

`set_item` (*key*, *value*)

Return type `Result[None]`

`set_storage` (*data*)

Return type `Result[None]`

`monitor_storage` ()

Return type `MonitorResult[Dict[str, Any]]`

`monitor_item_storage` ()

Return type `MonitorResult[Any]`

`add_submanager` (*name*, *submanager*)

Add a submanager.

Parameters

- **name** (`str`) – Field where manager will be stored.
- **submanager** (`BaseManager`) – Submanager

`get_commands` ()

Get manager available static commands.

Return type `Result[List[str]]`

Returns Manager static commands.

get_submanager (*name*)

Get a submanager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *BaseManager*

remove_submanager (*name*)

Remove a submanager.

Parameters *name* (*str*) – Field where submanager was stored.

Return type *Result*[~T]

3.4.22 Whalesong Errors

exception whalesong.errors.**WhalesongException**

Bases: *Exception*

exception whalesong.errors.**ManagerNotFound**

Bases: *whalesong.errors.WholesongException*

exception whalesong.errors.**UnknownError**

Bases: *whalesong.errors.WholesongException*

exception whalesong.errors.**ChatNotFound**

Bases: *whalesong.errors.WholesongException*

exception whalesong.errors.**ContactNotFound**

Bases: *whalesong.errors.WholesongException*

exception whalesong.errors.**StopMonitor**

Bases: *StopAsyncIteration*, *whalesong.errors.WholesongException*

exception whalesong.errors.**StopIterator**

Bases: *StopAsyncIteration*, *whalesong.errors.WholesongException*

exception whalesong.errors.**RequiredExecutionId**

Bases: *whalesong.errors.WholesongException*

exception whalesong.errors.**RequiredCommandName**

Bases: *whalesong.errors.WholesongException*

exception whalesong.errors.**ModelNotFound**

Bases: *whalesong.errors.WholesongException*

3.4.23 Firefox Profile

class whalesong.firefox_profile.**FirefoxProfile** (*profile_directory=None*)

Bases: *selenium.webdriver.firefox.firefox_profile.FirefoxProfile*

3.5 How to develop

3.5.1 Development requirements

- node (only for development)
- npm (only for development)

- make (only for development)

3.5.2 Install library requirements

```
$ make requirements
```

3.5.3 Build Javascript scriptlet

You have to rebuild scriptlet after any change if you want to use in Python code.

```
$ make build-js
```

3.5.4 Beautify code

You must to beautify code before make a pull request. Ugly code will not be accepted.

```
$ make beautify
```

3.6 Changelog

3.6.1 Version 0.9.1 (Work in progress)

- Fix [Issue #106](#)

3.6.2 Version 0.9.0

- Added Whatsapp Stories management (StatusV3). Thx to @jabolina.
- Added logout feature. Thx to @parthibd.
- Fixed stop/start driver (see [Issue #82](#)). Thx to @parthibd.
- Fixed Firefox driver initialization in order to allow more than one process. Thx to @parthibd.
- Fixed *wait_until_stop* driver method.
- Examples now stop gracefully.

3.6.3 Version 0.8.4

- Fixed requirements installation out of virtual enviornment. Thx to @Theblood.
- Enabled media codecs on Firefox profile template.
- Fixed problems with live location. [Issue #73](#)
- Fixed problem monitoring models.
- Added new examples: *getlivelocations-chromium.py* and *getlivelocations.py*

3.6.4 Version 0.8.2

- Fixed QR screenshot on Firefox.

3.6.5 Version 0.8.1

- Fixed Wap object discovery. Thx to @jabolina.
- Fixed screenshot method.
- Added Chromium version of some examples.

3.6.6 Version 0.8.0

- Live locations management.
- Mutes management.
- Revoke messages. Delete messages for others.
- Minimized scriptlet.

3.6.7 Version 0.7.2

- Capability to manage display information. It allows to mark current user as available in other to get presences from other users.

There are some limitations. Look at *documentation*.

- Minor fixes.
- Remove some debug messages.
- Fix issues with presence.
- Added manual ping-pong for Chromium backend.
- Modified *presencemonitor-chromium.py* and *presencemonitor.py* examples in order to get presences permanently.

3.6.8 Version 0.7.1

- Fixed Chromium driver.
- Forced Websockets 6.0
- Added new example *presencemonitor-chromium.py*. It is same than *presencemonitor.py* but using Chromium.

3.6.9 Version 0.7.0

- Added support for Chromium.
- Added support for backends that are able to push results.
- Defined Connection model.
- Added new connection's method *updatePushname*.

- Defined Stream model and enumerations.
- Added Presence manager.
- Added new example: *presencemonitor.py*. It monitor user presences.
- Simplified some code.
- Fields' value are mapped on monitors.
- Profile and group's picture management.
- New chat methods: *pin*, *unpin*, *archive*, *unarchive*, *set_group_description*, *star_messages*, *unstar_messages*, *send_not_spam*, *send_spam_report*, *can_archive*, *can_send* and *can_pin*.
- New chat collection method: *forward_messages_to_chats*.
- New message methods: *can_star*, *star* and *unstar*.
- Status (old one) management.
- New commands in *minibot.py* example: */status* and */pushname*.

3.6.10 Version 0.6.0

- Ability to get message information, it includes message acks (with timestamps). In addition, it is possible to monitor ack changes.

You must fetch message info before be able to monitor it.

```
msg_info: MessageInfo = await driver.messages[message_id].fetch_info()
```

And in order to monitor acks (message information changes):

```
async for event in driver.messages[message_id].info.monitor_model():
    print(event)
```

- Added Sticker message.
- Load and send stickers.
- Better type hinting.
- Better documentation.
- Added new command to *minibot.py* example: */sticker*.
- Added new example: *getstickers.py*. It downloads all stickers registered.

3.6.11 Version 0.5.3

- Add *query_exist* method to wap manager in order to get whether a contact identifier exists or not.
- When send a text message with an url it will try to get link preview and attach to message. It's not compatible with quoted messages.
- Added two new commands to *minibot.py* example: */link* and */exist*.

3.6.12 Version 0.5.2

- Fixed bug when sending docs. Thx to @jabolina.
- Added *set_subject* method to chat manager in order to be able to change group title.
- Added *mark_composing* method to chat manager in order to show “typing...” message.
- Added *mark_recording* method to chat manager in order to show “recording audio...” message.
- Added *mark_paused* method to chat manager in order to remove “typing...” or “recording audio...” message.

3.6.13 Version 0.5.1

- Fixed bug with user chats.
- Added new command */send* to minibot example.

3.6.14 Version 0.5.0

- Added *ensure_chat_with_contact* to chat collection manager. Ensure chat with a whatsapp user, if it does not exist it will be created. (Be careful with SPAM)
- Added *create_group* to chat collection manager.
- Added *block* and *unblock* methods to contact manager.
- Added group participants management: add, remove, promote, demote.
- Added group link management.

3.6.15 Version 0.4.4

- Allow extra options for Firefox driver.
- Added *leave_group*. Thx to @jabolina.
- Added *delete_chat*. Thx to @jabolina.

3.6.16 Version 0.4.0

- Removed *send_vcard* on chats. It is not possible now because WhatsAppWeb changes.
- Added *send_contact* and *send_contact_phone* in order to send contacts using contact id or contact name and phone.
- Small changes and refactors.

3.6.17 Version 0.3.0

- Reduce Firefox footprint.
- Message classes.
- Improved getMessages example. Now, it downloads media files.
- Package published at Pypi.

3.6.18 Version 0.2.0

Warning: Command separator changed from . to |.

- Simplified code to manage models.
- Added *remove_item_by_id*, *get_length*, *get_first* and *get_last* methods to collection managers.
- Added *load_earlier_messages* and *load_all_earlier_messages* methods to chat manager.

CHAPTER 4

Related projects

Port of Whalesong to Node.js.

Author: @jabolina

Repository: <https://github.com/jabolina/whalesong-js>

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

CHAPTER 6

Legal

This code is in no way affiliated with, authorized, maintained, sponsored or endorsed by WhatsApp or any of its affiliates or subsidiaries. This is an independent and unofficial software. Use at your own risk.

W

- `whalesong.errors`, [201](#)
- `whalesong.firefox_profile`, [201](#)
- `whalesong.managers`, [22](#)
- `whalesong.managers.chat`, [44](#)
- `whalesong.managers.conn`, [154](#)
- `whalesong.managers.contact`, [30](#)
- `whalesong.managers.display_info`, [192](#)
- `whalesong.managers.group_metadata`, [140](#)
- `whalesong.managers.live_location`, [185](#)
- `whalesong.managers.message`, [57](#)
- `whalesong.managers.mute`, [190](#)
- `whalesong.managers.presence`, [167](#)
- `whalesong.managers.profile_pic_thumb`,
[173](#)
- `whalesong.managers.status`, [177](#)
- `whalesong.managers.status_v3`, [199](#)
- `whalesong.managers.sticker_pack`, [151](#)
- `whalesong.managers.stream`, [158](#)
- `whalesong.managers.wap`, [143](#)
- `whalesong.models`, [21](#)
- `whalesong.results`, [19](#)

Symbols

<code>__getattr__()</code> (<i>whalesong.managers.chat.ChatCollectionManager</i> method), 50	<code>__getattr__()</code> (<i>whalesong.managers.message.MessageManager</i> method), 50
<code>__getattr__()</code> (<i>whalesong.managers.chat.ChatCollectionManager</i> method), 33	<code>__getattr__()</code> (<i>whalesong.managers.mute.MuteCollectionManager</i> method), 186
<code>__getattr__()</code> (<i>whalesong.managers.chat.ChatManager</i> method), 36	<code>__getattr__()</code> (<i>whalesong.managers.mute.MuteManager</i> method), 189
<code>__getattr__()</code> (<i>whalesong.managers.chat.MsgLoadStateManager</i> method), 43	<code>__getattr__()</code> (<i>whalesong.managers.presence.ChatStateCollectionManager</i> method), 164
<code>__getattr__()</code> (<i>whalesong.managers.conn.ConnManager</i> method), 152	<code>__getattr__()</code> (<i>whalesong.managers.presence.ChatStateManager</i> method), 166
<code>__getattr__()</code> (<i>whalesong.managers.contact.ContactCollectionManager</i> method), 26	<code>__getattr__()</code> (<i>whalesong.managers.presence.PresenceCollectionManager</i> method), 160
<code>__getattr__()</code> (<i>whalesong.managers.contact.ContactManager</i> method), 28	<code>__getattr__()</code> (<i>whalesong.managers.presence.PresenceManager</i> method), 162
<code>__getattr__()</code> (<i>whalesong.managers.display_info.DisplayInfoManager</i> method), 191	<code>__getattr__()</code> (<i>whalesong.managers.profile_pic_thumb.ProfilePictureManager</i> method), 169
<code>__getattr__()</code> (<i>whalesong.managers.group_metadata.GroupMetadataCollectionManager</i> method), 132	<code>__getattr__()</code> (<i>whalesong.managers.profile_pic_thumb.ProfilePictureManager</i> method), 171
<code>__getattr__()</code> (<i>whalesong.managers.group_metadata.GroupMetadataManager</i> method), 135	<code>__getattr__()</code> (<i>whalesong.managers.status.StatusCollectionManager</i> method), 174
<code>__getattr__()</code> (<i>whalesong.managers.group_metadata.ParticipantCollectionManager</i> method), 136	<code>__getattr__()</code> (<i>whalesong.managers.status.StatusManager</i> method), 176
<code>__getattr__()</code> (<i>whalesong.managers.group_metadata.ParticipantManager</i> method), 139	<code>__getattr__()</code> (<i>whalesong.managers.status_v3.StatusV3CollectionManager</i> method), 195
<code>__getattr__()</code> (<i>whalesong.managers.live_location.LiveLocationCollectionManager</i> method), 178	<code>__getattr__()</code> (<i>whalesong.managers.status_v3.StatusV3Manager</i> method), 198
<code>__getattr__()</code> (<i>whalesong.managers.live_location.LiveLocationManager</i> method), 180	<code>__getattr__()</code> (<i>whalesong.managers.sticker_pack.StickerCollectionManager</i> method), 147
<code>__getattr__()</code> (<i>whalesong.managers.live_location.ParticipantCollectionManager</i> method), 181	<code>__getattr__()</code> (<i>whalesong.managers.sticker_pack.StickerManager</i> method), 150
<code>__getattr__()</code> (<i>whalesong.managers.live_location.ParticipantManager</i> method), 184	<code>__getattr__()</code> (<i>whalesong.managers.sticker_pack.StickerPackCollectionManager</i> method), 143
<code>__getattr__()</code> (<i>whalesong.managers.message.MessageAckCollectionManager</i> method), 53	<code>__getattr__()</code> (<i>whalesong.managers.sticker_pack.StickerPackManager</i> method), 146
<code>__getattr__()</code> (<i>whalesong.managers.message.MessageAckManager</i> method), 55	<code>__getattr__()</code> (<i>whalesong.managers.storage.StorageManager</i> method), 200
<code>__getattr__()</code> (<i>whalesong.managers.message.MessageCollectionManager</i> method), 47	<code>__getattr__()</code> (<i>whalesong.managers.stream.StreamManager</i> method), 156
<code>__getattr__()</code> (<i>whalesong.managers.message.MessageInfoManager</i> method), 52	<code>__getattr__()</code> (<i>whalesong.managers.wap.WapManager</i> method), 142

`__getitem__()` (`whalesong.managers.chat.ChatCollectionManager` method), 33
`__getitem__()` (`whalesong.managers.chat.ChatManager` method), 36
`__getitem__()` (`whalesong.managers.chat.MsgLoadStateManager` method), 43
`__getitem__()` (`whalesong.managers.conn.ConnManager` method), 152
`__getitem__()` (`whalesong.managers.contact.ContactCollectionManager` method), 26
`__getitem__()` (`whalesong.managers.contact.ContactManager` method), 28
`__getitem__()` (`whalesong.managers.display_info.DisplayInfoManager` method), 191
`__getitem__()` (`whalesong.managers.group_metadata.GroupMetadataCollectionManager` method), 132
`__getitem__()` (`whalesong.managers.group_metadata.GroupMetadataManager` method), 135
`__getitem__()` (`whalesong.managers.group_metadata.ParticipantCollectionManager` method), 136
`__getitem__()` (`whalesong.managers.group_metadata.ParticipantManager` method), 139
`__getitem__()` (`whalesong.managers.live_location.LiveLocationCollectionManager` method), 177
`__getitem__()` (`whalesong.managers.live_location.LiveLocationManager` method), 180
`__getitem__()` (`whalesong.managers.live_location.ParticipantCollectionManager` method), 181
`__getitem__()` (`whalesong.managers.live_location.ParticipantManager` method), 184
`__getitem__()` (`whalesong.managers.message.MessageAckCollectionManager` method), 53
`__getitem__()` (`whalesong.managers.message.MessageAckManager` method), 55
`__getitem__()` (`whalesong.managers.message.MessageCollectionManager` method), 47
`__getitem__()` (`whalesong.managers.message.MessageInfoManager` method), 52
`__getitem__()` (`whalesong.managers.message.MessageManager` method), 49
`__getitem__()` (`whalesong.managers.mute.MuteCollectionManager` method), 186
`__getitem__()` (`whalesong.managers.mute.MuteManager` method), 189
`__getitem__()` (`whalesong.managers.presence.ChatStateCollectionManager` method), 164
`__getitem__()` (`whalesong.managers.presence.ChatStateManager` method), 166
`__getitem__()` (`whalesong.managers.presence.PresenceCollectionManager` method), 160
`__getitem__()` (`whalesong.managers.presence.PresenceManager` method), 162
`__getitem__()` (`whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager` method), 169
`__getitem__()` (`whalesong.managers.profile_pic_thumb.ProfilePictureManager` method), 171
`__getitem__()` (`whalesong.managers.status.StatusCollectionManager` method), 173
`__getitem__()` (`whalesong.managers.status.StatusManager` method), 176
`__getitem__()` (`whalesong.managers.status_v3.StatusV3CollectionManager` method), 195
`__getitem__()` (`whalesong.managers.status_v3.StatusV3Manager` method), 197
`__getitem__()` (`whalesong.managers.sticker_pack.StickerCollectionManager` method), 147
`__getitem__()` (`whalesong.managers.sticker_pack.StickerManager` method), 149
`__getitem__()` (`whalesong.managers.sticker_pack.StickerPackCollectionManager` method), 143
`__getitem__()` (`whalesong.managers.sticker_pack.StickerPackManager` method), 146
`__getitem__()` (`whalesong.managers.storage.StorageManager` method), 200
`__getitem__()` (`whalesong.managers.stream.StreamManager` method), 156
`__getitem__()` (`whalesong.managers.wap.WapManager` method), 141

A

`ack` (`whalesong.managers.live_location.ParticipantManager` property), 185
`ack` (`whalesong.managers.message.LocationMessage` property), 110
`ack` (`whalesong.managers.message.AudioMessage` property), 85
`ack` (`whalesong.managers.message.BaseMessage` property), 85
`ack` (`whalesong.managers.message.DocumentMessage` property), 96
`ack` (`whalesong.managers.message.GroupNotificationMessage` property), 121
`ack` (`whalesong.managers.message.ImageMessage` property), 73
`ack` (`whalesong.managers.message.LocationMessage` property), 112
`ack` (`whalesong.managers.message.MultiVCardMessage` property), 106
`ack` (`whalesong.managers.message.PaymentMessage` property), 117
`ack` (`whalesong.managers.message.PTTMessage` property), 109
`ack` (`whalesong.managers.message.StickerMessage` property), 127
`ack` (`whalesong.managers.message.TextMessage` property), 109

ack (*whalesong.managers.message.VCardMessage property*), 101

ack (*whalesong.managers.message.VideoMessage property*), 79

add_callback() (*whalesong.results.MonitorResult method*), 20

add_participants() (*whalesong.managers.group_metadata.ParticipantCollectionManager method*), 136

add_submanager() (*whalesong.managers.BaseManager method*), 23

add_submanager() (*whalesong.managers.chat.ChatCollectionManager method*), 33

add_submanager() (*whalesong.managers.chat.ChatManager method*), 40

add_submanager() (*whalesong.managers.chat.MsgLoadStateManager method*), 43

add_submanager() (*whalesong.managers.conn.ConnManager method*), 152

add_submanager() (*whalesong.managers.contact.ContactCollectionManager method*), 26

add_submanager() (*whalesong.managers.contact.ContactManager method*), 28

add_submanager() (*whalesong.managers.display_info.DisplayInfoManager method*), 191

add_submanager() (*whalesong.managers.group_metadata.GroupMetadataCollectionManager method*), 133

add_submanager() (*whalesong.managers.group_metadata.GroupMetadataManager method*), 135

add_submanager() (*whalesong.managers.group_metadata.ParticipantCollectionManager method*), 137

add_submanager() (*whalesong.managers.group_metadata.ParticipantManager method*), 139

add_submanager() (*whalesong.managers.live_location.LiveLocationCollectionManager method*), 178

add_submanager() (*whalesong.managers.live_location.LiveLocationManager method*), 180

add_submanager() (*whalesong.managers.live_location.ParticipantCollectionManager method*), 182

add_submanager() (*whalesong.managers.live_location.ParticipantManager method*), 184

add_submanager() (*whalesong.managers.message.MessageAckCollectionManager method*), 53

add_submanager() (*whalesong.managers.message.MessageAckManager method*), 55

add_submanager() (*whalesong.managers.message.MessageCollectionManager method*), 47

add_submanager() (*whalesong.managers.message.MessageInfoManager method*), 52

add_submanager() (*whalesong.managers.message.MessageManager method*), 50

add_submanager() (*whalesong.managers.mute.MuteCollectionManager method*), 186

add_submanager() (*whalesong.managers.mute.MuteManager method*), 189

add_submanager() (*whalesong.managers.presence.ChatStateCollectionManager method*), 164

add_submanager() (*whalesong.managers.presence.ChatStateManager method*), 166

add_submanager() (*whalesong.managers.presence.PresenceCollectionManager method*), 160

add_submanager() (*whalesong.managers.presence.PresenceManager method*), 162

add_submanager() (*whalesong.managers.profile_pic_thumb.ProfilePicThumbCollectionManager method*), 169

add_submanager() (*whalesong.managers.profile_pic_thumb.ProfilePicThumbManager method*), 172

add_submanager() (*whalesong.managers.status.StatusCollectionManager method*), 174

add_submanager() (*whalesong.managers.status.StatusManager method*), 176

add_submanager() (*whalesong.managers.status_v3.StatusV3CollectionManager method*), 195

add_submanager() (*whalesong.managers.status_v3.StatusV3Manager method*), 198

add_submanager() (*whalesong.managers.sticker_pack.StickerCollectionManager method*), 148

add_submanager() (*whalesong.managers.sticker_pack.StickerManager method*), 150

add_submanager() (*whalesong.managers.sticker_pack.StickerPackCollectionManager method*), 144

add_submanager() (*whalesong.managers.sticker_pack.StickerPackManager method*), 146

add_submanager() (*whalesong.managers.storage.StorageManager method*), 200

add_submanager() (*whalesong.managers.stream.StreamManager method*), 156

add_submanager() (*whalesong.managers.wap.WapManager method*), 142

add_submanager() (*whalesong.managers.group_metadata.GroupMetadata property*), 140

add_submanager() (*whalesong.managers.live_location.ParticipantCollectionManager method*), 44

archive() (*whalesong.managers.chat.ChatManager method*), 42

AUDIO (*whalesong.managers.message.MessageTypes attribute*), 82

AudioMessage (*whalesong.managers.message property*), 82

author (*whalesong.managers.message.AudioMessage property*), 82

author (*whalesong.managers.message.AuthorMixin property*), 65

author (*whalesong.managers.message.BaseMessage property*), 58

author (*whalesong.managers.message.DocumentMessage property*), 96

author (*whalesong.managers.message.GroupNotificationMessage property*), 121

- ul style="list-style-type: none; padding-left: 0;">
- author (*whalesong.managers.message.ImageMessage* property), 73
- author (*whalesong.managers.message.LocationMessage* property), 111
- author (*whalesong.managers.message.MultiVCardMessage* property), 106
- author (*whalesong.managers.message.PaymentMessage* property), 117
- author (*whalesong.managers.message.PTTMessage* property), 91
- author (*whalesong.managers.message.StickerMessage* property), 127
- author (*whalesong.managers.message.TextMessage* property), 67
- author (*whalesong.managers.message.VCardMessage* property), 101
- author (*whalesong.managers.message.VideoMessage* property), 79
- AuthorMixin (*whalesong.managers.message.Model*), 65
- available (*whalesong.managers.display_info.DisplayInfo* property), 194
- AVAILABLE (*whalesong.managers.presence.ChatState.Type* attribute), 167
- ## B
- backgroundColor (*whalesong.managers.message.TextMessage* property), 71
 - backoffGeneration (*whalesong.managers.stream.Stream* property), 159
 - Base64Field (*class in whalesong.models*), 21
 - BaseCollectionManager (*class in whalesong.managers*), 24
 - BaseIteratorResult (*class in whalesong.results*), 20
 - BaseManager (*class in whalesong.managers*), 22
 - BaseMessage (*whalesong.managers.message.Model*), 58
 - BaseModel (*whalesong.models.Model*), 22
 - BaseModelManager (*class in whalesong.managers*), 23
 - BasePartialResult (*class in whalesong.results*), 19
 - BaseResultMixin (*class in whalesong.results*), 19
 - BaseWhalesongDriver (*class in whalesong.driver*), 18
 - battery (*whalesong.managers.conn.Conn* property), 155
 - block() (*whalesong.managers.contact.ContactManager* method), 29
 - body (*whalesong.managers.message.AudioMessage* property), 84
 - body (*whalesong.managers.message.BaseMessage* property), 58
 - body (*whalesong.managers.message.DocumentMessage* property), 94
 - body (*whalesong.managers.message.GroupNotificationMessage* property), 121
 - body (*whalesong.managers.message.ImageMessage* property), 72
 - body (*whalesong.managers.message.LocationMessage* property), 110
 - body (*whalesong.managers.message.MediaFrameMixin* property), 64
 - body (*whalesong.managers.message.MultiVCardMessage* property), 106
 - body (*whalesong.managers.message.PaymentMessage* property), 116
 - body (*whalesong.managers.message.PTTMessage* property), 90
 - body (*whalesong.managers.message.StickerMessage* property), 127
 - body (*whalesong.managers.message.TextMessage* property), 66
 - body (*whalesong.managers.message.VCardMessage* property), 101
 - body (*whalesong.managers.message.VideoMessage* property), 78
 - broadcast (*whalesong.managers.message.GroupNotificationMessage* property), 125
- ## C
- BROADCAST_NOTIFICATION (*whalesong.managers.message.MessageTypes* attribute), 57
 - CALL_LOG (*whalesong.managers.message.MessageTypes* attribute), 57
 - can_add() (*whalesong.managers.group_metadata.ParticipantCollection* method), 136
 - can_archive() (*whalesong.managers.chat.ChatManager* method), 41
 - can_delete() (*whalesong.managers.profile_pic_thumb.ProfilePictureManager* method), 172
 - can_demote() (*whalesong.managers.group_metadata.ParticipantCollection* method), 139
 - can_mute() (*whalesong.managers.mute.MuteManager* method), 189
 - can_pin() (*whalesong.managers.chat.ChatManager* method), 42
 - can_promote() (*whalesong.managers.group_metadata.ParticipantCollection* method), 138
 - can_remove() (*whalesong.managers.group_metadata.ParticipantCollection* method), 137
 - can_revoke() (*whalesong.managers.message.MessageManager* method), 51
 - can_send() (*whalesong.managers.chat.ChatManager* method), 41

`can_set()` (`whattlesong.managers.profile_pic_thumb.ProfilePictureManager` method), 172
`can_star()` (`whattlesong.managers.message.MessageManager` method), 51
`can_update_pushname()` (`whattlesong.managers.conn.ConnManager` method), 154
`cancel()` (`whattlesong.results.BasePartialResult` method), 20
`cancel_all()` (`whattlesong.results.ResultManager` method), 21
`cancel_iterators()` (`whattlesong.driver.BaseWattlesongDriver` method), 18
`cancel_iterators()` (`whattlesong.Wattlesong` method), 17
`cancel_monitors()` (`whattlesong.driver.BaseWattlesongDriver` method), 18
`cancel_result()` (`whattlesong.results.ResultManager` method), 21
`canonicalUrl` (`whattlesong.managers.message.LinkContentMixin` property), 63
`canonicalUrl` (`whattlesong.managers.message.TextMessage` property), 66
`canSend` (`whattlesong.managers.stream.Stream` property), 159
`caption` (`whattlesong.managers.message.AudioMessage` property), 83
`caption` (`whattlesong.managers.message.DocumentMessage` property), 95
`caption` (`whattlesong.managers.message.ImageMessage` property), 72
`caption` (`whattlesong.managers.message.MediaMixin` property), 64
`caption` (`whattlesong.managers.message.PTTMessage` property), 89
`caption` (`whattlesong.managers.message.StickerMessage` property), 126
`caption` (`whattlesong.managers.message.VideoMessage` property), 77
`caption` (`whattlesong.managers.sticker_pack.Sticker` property), 151
`changeNumberNewJid` (`whattlesong.managers.chat.Chat` property), 45
`changeNumberOldJid` (`whattlesong.managers.chat.Chat` property), 45
`Chat` (`whattlesong.managers.chat` Model), 44
`chat` (`whattlesong.managers.message.AudioMessage` property), 85
`chat` (`whattlesong.managers.message.BaseMessage` property), 59
`chat` (`whattlesong.managers.message.DocumentMessage` property), 97
`chat` (`whattlesong.managers.message.GroupNotificationMessage` property), 122
`chat` (`whattlesong.managers.message.ImageMessage` property), 73
`chat` (`whattlesong.managers.message.LocationMessage` property), 112
`CHAT` (`whattlesong.managers.message.MessageTypes` attribute), 57
`chat` (`whattlesong.managers.message.MultiVCardMessage` property), 107
`chat` (`whattlesong.managers.message.PaymentMessage` property), 118
`chat` (`whattlesong.managers.message.PTTMessage` property), 91
`chat` (`whattlesong.managers.message.StickerMessage` property), 128
`chat` (`whattlesong.managers.message.TextMessage` property), 68
`chat` (`whattlesong.managers.message.VCardMessage` property), 102
`chat` (`whattlesong.managers.message.VideoMessage` property), 80
`chatActive` (`whattlesong.managers.presence.Presence` property), 168
`ChatCollectionManager` (class in `whattlesong.managers.chat`), 33
`ChatManager` (class in `whattlesong.managers.chat`), 35
`ChatManager.LiveLocation` (`whattlesong.managers.chat` Model), 38
`ChatNotFoundError`, 201
`chats` (`whattlesong.Wattlesong` attribute), 16
`ChatState` (`whattlesong.managers.presence` Model), 167
`chatState` (`whattlesong.managers.presence.Presence` property), 168
`ChatState.Type` (class in `whattlesong.managers.presence`), 167
`ChatStateCollectionManager` (class in `whattlesong.managers.presence`), 163
`ChatStateManager` (class in `whattlesong.managers.presence`), 166
`chatStates` (`whattlesong.managers.presence.Presence` property), 168
`CIPHERTEXT` (`whattlesong.managers.message.MessageTypes` attribute), 57
`clear()` (`whattlesong.managers.chat.ChatManager.LiveLocation` method), 38
`clear_all()` (`whattlesong.managers.chat.ChatManager.LiveLocation` method), 38
`clear_modified_data()` (`whattlesong.managers.chat.ChatManager.LiveLocation` method), 38

[clientExpired \(whalesong.managers.display_info.DisplayInfo property\), 194](#)
[clientToken \(whalesong.managers.conn.Conn property\), 155](#)
[clientId \(whalesong.managers.message.AudioMessage property\), 83](#)
[clientId \(whalesong.managers.message.DocumentMessage property\), 95](#)
[clientId \(whalesong.managers.message.ImageMessage property\), 71](#)
[clientId \(whalesong.managers.message.MediaMixIn property\), 64](#)
[clientId \(whalesong.managers.message.PTTMessage property\), 89](#)
[clientId \(whalesong.managers.message.StickerMessage property\), 126](#)
[clientId \(whalesong.managers.message.VideoMessage property\), 77](#)
[clientId \(whalesong.managers.sticker_pack.Sticker property\), 151](#)
[close \(\) \(whalesong.driver.BaseWhalesongDriver method\), 18](#)
[comment \(whalesong.managers.live_location.Participant property\), 185](#)
[comment \(whalesong.managers.message.LocationMessage property\), 110](#)
[COMPOSING \(whalesong.managers.presence.ChatState.Type attribute\), 167](#)
[CONFLICT \(whalesong.managers.display_info.DisplayInfo.StreamMode attribute\), 193](#)
[CONFLICT \(whalesong.managers.stream.Stream.State attribute\), 158](#)
[Conn \(whalesong.managers.conn Model\), 154](#)
[conn \(whalesong.Wholesong attribute\), 16](#)
[connect \(\) \(whalesong.driver.BaseWhalesongDriver method\), 18](#)
[connect \(\) \(whalesong.driver_chromium.WholesongDriver method\), 19](#)
[connect \(\) \(whalesong.driver_firefox.WholesongDriver method\), 18](#)
[connected \(whalesong.managers.conn.Conn property\), 155](#)
[CONNECTED \(whalesong.managers.stream.Stream.State attribute\), 158](#)
[CONNECTED \(whalesong.managers.stream.Stream.Stream attribute\), 159](#)
[CONNECTING \(whalesong.managers.display_info.DisplayInfo.StreamMode attribute\), 193](#)
[ConnManager \(class in whalesong.managers.conn\), 152](#)
[contact \(whalesong.managers.chat.Chat property\), 44](#)
[contact \(whalesong.managers.chat.ChatManager attribute\), 36](#)
[Contact \(whalesong.managers.contact Model\), 30](#)
[ContactInfo \(whalesong.managers.live_location.Participant property\), 185](#)
[contact \(whalesong.managers.status_v3.StatusV3 property\), 199](#)
[ContactCollectionManager \(class in whalesong.managers.contact\), 26](#)
[ContactManager \(class in whalesong.managers.contact\), 28](#)
[ContactNotFoundError, 201](#)
[contacts \(whalesong.Wholesong attribute\), 16](#)
[containsEmoji \(whalesong.managers.message.AudioMessage property\), 88](#)
[containsEmoji \(whalesong.managers.message.BaseMessage property\), 61](#)
[containsEmoji \(whalesong.managers.message.DocumentMessage property\), 99](#)
[containsEmoji \(whalesong.managers.message.GroupNotificationMessage property\), 124](#)
[containsEmoji \(whalesong.managers.message.ImageMessage property\), 76](#)
[containsEmoji \(whalesong.managers.message.LocationMessage property\), 114](#)
[containsEmoji \(whalesong.managers.message.MultiVCardMessage property\), 109](#)
[containsEmoji \(whalesong.managers.message.PaymentMessage property\), 120](#)
[containsEmoji \(whalesong.managers.message.PTTMessage property\), 94](#)
[containsEmoji \(whalesong.managers.message.StickerMessage property\), 130](#)
[containsEmoji \(whalesong.managers.message.TextMessage property\), 70](#)
[containsEmoji \(whalesong.managers.message.VCardMessage property\), 104](#)
[containsEmoji \(whalesong.managers.message.VideoMessage property\), 82](#)
[contextLoaded \(whalesong.managers.chat.MsgLoadState property\), 46](#)
[convert_value \(\) \(whalesong.models.Base64Field method\), 21](#)
[convert_value \(\) \(whalesong.models.DateTimeField method\), 22](#)
[copy \(\) \(whalesong.managers.chat.ChatManager.LiveLocation method\), 38](#)
[couldForce \(whalesong.managers.display_info.DisplayInfo property\), 194](#)
[createGroup \(\) \(whalesong.managers.chat.ChatCollectionManager method\), 35](#)
[creation \(whalesong.managers.group_metadata.GroupMetadata property\), 141](#)

D

[DateTimeField \(class in whalesong.models\), 21](#)

[default_model_iter](#) ([whalesong.models.JSONEncoder](#) attribute), 22
[degrees](#) ([whalesong.managers.live_location.Participant](#) property), 185
[degrees](#) ([whalesong.managers.message.LocationMessage](#) dir property), 110
[delete_attr_by_path\(\)](#) ([whalesong.managers.chat.ChatManager.LiveLocation](#) method), 38
[delete_chat\(\)](#) ([whalesong.managers.chat.ChatManager](#) method), 37
[delete_field_value\(\)](#) ([whalesong.managers.chat.ChatManager.LiveLocation](#) method), 38
[delete_picture\(\)](#) ([whalesong.managers.profile_pic_thumb.ProfilePictureManager](#) method), 172
[delivery](#) ([whalesong.managers.message.MessageInfo](#) dir property), 132
[delivery](#) ([whalesong.managers.message.MessageInfoManager](#) attribute), 51
[deliveryRemaining](#) ([whalesong.managers.message.MessageInfo](#) property), 131
[demote_participants\(\)](#) ([whalesong.managers.group_metadata.ParticipantCollectionManager](#) method), 138
[deny](#) ([whalesong.managers.presence.ChatState](#) property), 167
[DEPRECATED_VERSION](#) ([whalesong.managers.display_info.DisplayInfo.StreamMode](#) attribute), 193
[desc](#) ([whalesong.managers.group_metadata.GroupMetadata](#) property), 141
[descOwner](#) ([whalesong.managers.group_metadata.GroupMetadata](#) property), 141
[description](#) ([whalesong.managers.message.LinkContentMixin](#) property), 63
[description](#) ([whalesong.managers.message.TextMessage](#) property), 66
[descTime](#) ([whalesong.managers.group_metadata.GroupMetadata](#) property), 141
[DEVICE](#) ([whalesong.managers.message.Ack](#) attribute), 57
[deviceManufacturer](#) ([whalesong.managers.conn.PhoneDescription](#) property), 154
[deviceModel](#) ([whalesong.managers.conn.PhoneDescription](#) property), 154
[dir](#) ([whalesong.managers.message.AudioMessage](#) property), 88
[dir](#) ([whalesong.managers.message.BaseMessage](#) property), 59
[dir](#) ([whalesong.managers.message.DocumentMessage](#) property), 99
[dir](#) ([whalesong.managers.message.GroupNotificationMessage](#) property), 124
[dir](#) ([whalesong.managers.message.ImageMessage](#) property), 76
[dir](#) ([whalesong.managers.message.LocationMessage](#) property), 115
[dir](#) ([whalesong.managers.message.MultiVCardMessage](#) property), 109
[dir](#) ([whalesong.managers.message.PaymentMessage](#) property), 120
[dir](#) ([whalesong.managers.message.PTTMessage](#) property), 94
[dir](#) ([whalesong.managers.message.StickerMessage](#) property), 130
[dir](#) ([whalesong.managers.message.TextMessage](#) property), 70
[dir](#) ([whalesong.managers.message.VCardMessage](#) property), 104
[dir](#) ([whalesong.managers.message.VideoMessage](#) property), 82
[directPath](#) ([whalesong.managers.message.AudioMessage](#) property), 83
[directPath](#) ([whalesong.managers.message.DocumentMessage](#) property), 95
[directPath](#) ([whalesong.managers.message.ImageMessage](#) property), 72
[directPath](#) ([whalesong.managers.message.MediaMixin](#) property), 64
[directPath](#) ([whalesong.managers.message.PTTMessage](#) property), 89
[directPath](#) ([whalesong.managers.message.StickerMessage](#) property), 126
[directPath](#) ([whalesong.managers.message.VideoMessage](#) property), 77
[directPath](#) ([whalesong.managers.sticker_pack.Sticker](#) property), 151
[DISCONNECTED](#) ([whalesong.managers.stream.Stream.Stream](#) attribute), 159
[display_info](#) ([whalesong.Wholesong](#) attribute), 17
[displayInfo](#) ([whalesong.managers.display_info](#) Model), 192
[displayInfo](#) ([whalesong.managers.display_info.DisplayInfo](#) property), 194
[DisplayInfo.DisplayState](#) (class in [whalesong.managers.display_info](#)), 193
[DisplayInfo.StreamInfo](#) (class in [whalesong.managers.display_info](#)), 193
[DisplayInfo.StreamMode](#) (class in [whalesong.managers.display_info](#)), 193
[DisplayInfoManager](#) (class in [whalesong.managers.display_info](#)), 191
[displayName](#) ([whalesong.managers.message.VCardItem](#) property), 58

DOCUMENT (*whalesong.managers.message.MessageTypes* *export_modifications()* *attribute*), 57 (*whalesong.managers.chat.ChatManager.LiveLocation* *method*), 38

DocumentMessage (*whalesong.managers.message* *Model*), 94 *export_modified_data()*

download_file() (*whalesong.driver.BaseWhalesongDriver* *method*), 18 (*whalesong.managers.chat.ChatManager.LiveLocation* *method*), 38

download_file() (*whalesong.Whalesong* *method*), 17 *export_original_data()* (*whalesong.managers.chat.ChatManager.LiveLocation* *method*), 38

download_image() (*whalesong.managers.sticker_pack.StickerManager* *method*), 150

download_media() (*in module F* *whalesong.managers.message*), 132 *fetch()* (*whalesong.managers.sticker_pack.StickerCollectionManager* *method*), 49

download_media() (*whalesong.managers.message.MessageCollectionManager* *method*), 49 *fetch_all_pages()* (*whalesong.managers.sticker_pack.StickerPackCollectionManager* *method*), 144

download_media() (*whalesong.managers.message.MessageManager* *method*), 51 *fetch_info()* (*whalesong.managers.message.MessageManager* *method*), 51

duration (*whalesong.managers.message.AudioMessage* *property*), 84 *fetch_page()* (*whalesong.managers.sticker_pack.StickerPackCollectionManager* *method*), 143

duration (*whalesong.managers.message.MediaStreamMixin* *property*), 65

duration (*whalesong.managers.message.PTTMessage* *property*), 90 *filehash* (*whalesong.managers.message.AudioMessage* *property*), 83

duration (*whalesong.managers.message.VideoMessage* *property*), 78 *filehash* (*whalesong.managers.message.DocumentMessage* *property*), 95

duration (*whalesong.managers.chat.ChatManager.LiveLocation* *property*), 40 *filehash* (*whalesong.managers.message.ImageMessage* *property*), 72

duration (*whalesong.managers.live_location.LiveLocation* *property*), 186 *filehash* (*whalesong.managers.message.MediaMixin* *property*), 64

duration (*whalesong.managers.message.LocationMessage* *property*), 116 *filehash* (*whalesong.managers.message.PTTMessage* *property*), 89

E *filehash* (*whalesong.managers.message.StickerMessage* *property*), 126

E2E_NOTIFICATION (*whalesong.managers.message.MessageTypes* *attribute*), 57 *filehash* (*whalesong.managers.message.VideoMessage* *property*), 77

ensure_chat_with_contact() (*whalesong.managers.chat.ChatCollectionManager* *method*), 35 *filehash* (*whalesong.managers.sticker_pack.Sticker* *property*), 151

ERROR (*whalesong.managers.message.Ack* *attribute*), 57 *finalAccuracy* (*whalesong.managers.message.LocationMessage* *property*), 116

eurl (*whalesong.managers.profile_pic_thumb.ProfilePicture* *property*), 173 *finalDegrees* (*whalesong.managers.message.LocationMessage* *property*), 116

execute_command() (*whalesong.driver.BaseWhalesongDriver* *method*), 18 *finalLat* (*whalesong.managers.message.LocationMessage* *property*), 115

expiration (*whalesong.managers.live_location.Participant* *property*), 185 *finalLng* (*whalesong.managers.message.LocationMessage* *property*), 116

expiration (*whalesong.managers.mute.Mute* *property*), 190 *finalSpeed* (*whalesong.managers.message.LocationMessage* *property*), 116

expireTs (*whalesong.managers.status_v3.StatusV3* *property*), 199 *finalThumbnail* (*whalesong.managers.message.LocationMessage* *property*), 115

export_data() (*whalesong.managers.chat.ChatManager.LiveLocation* *method*), 38 *finalTimeOffset* (*whalesong.managers.message.LocationMessage* *property*), 116

export_deleted_fields() (*whalesong.managers.chat.ChatManager.LiveLocation* *method*), 38 *find_item_by_id()* (*whalesong.managers.BaseCollectionManager* *method*), 24

find_item_by_id()

(whalesong.managers.chat.ChatCollectionManager method), 33
 find_item_by_id() (whalesong.managers.contact.ContactCollectionManager method), 22
 find_item_by_id() (whalesong.managers.group_metadata.GroupMetadataCollectionManager method), 133
 find_item_by_id() (whalesong.managers.group_metadata.ParticipantCollectionManager method), 137
 find_item_by_id() (whalesong.managers.live_location.LiveLocationCollectionManager method), 178
 find_item_by_id() (whalesong.managers.live_location.ParticipantCollectionManager method), 182
 find_item_by_id() (whalesong.managers.message.MessageAckCollectionManager method), 53
 find_item_by_id() (whalesong.managers.message.MessageCollectionManager method), 47
 find_item_by_id() (whalesong.managers.mute.MuteCollectionManager method), 187
 find_item_by_id() (whalesong.managers.presence.ChatStateCollectionManager method), 164
 find_item_by_id() (whalesong.managers.presence.PresenceCollectionManager method), 160
 find_item_by_id() (whalesong.managers.profile_pic_thumb.ProfilePicThumbCollectionManager method), 169
 find_item_by_id() (whalesong.managers.status.StatusCollectionManager method), 174
 find_item_by_id() (whalesong.managers.status_v3.StatusV3CollectionManager method), 195
 find_item_by_id() (whalesong.managers.sticker_pack.StickerCollectionManager method), 148
 find_item_by_id() (whalesong.managers.sticker_pack.StickerPackCollectionManager method), 144
 find_live_location() (whalesong.managers.chat.ChatManager method), 43
 FirefoxProfile (class in whalesong.firefox_profile), 201
 flat_data() (whalesong.managers.chat.ChatManager.LiveLocation method), 39
 front (whalesong.managers.message.TextMessage property), 65
 format_field() (whalesong.models.ModelFormatterIter method), 22
 formattedName (whalesong.managers.contact.Contact property), 30
 forward_to_chats() (whalesong.managers.chat.ChatCollectionManager method), 35
 from (whalesong.driver_firefox.WholesongDriver method), 18
 from (whalesong.managers.message.AudioMessage property), 85
 from (whalesong.managers.message.BaseMessage property), 59
 from (whalesong.managers.message.DocumentMessage property), 96
 from (whalesong.managers.message.GroupNotificationMessage property), 121
 from (whalesong.managers.message.ImageMessage property), 73
 from (whalesong.managers.message.LocationMessage property), 111
 from (whalesong.managers.message.MultiVCardMessage property), 106
 from (whalesong.managers.message.PaymentMessage property), 117
 from (whalesong.managers.message.PTTMessage property), 90
 from (whalesong.managers.message.StickerMessage property), 127
 from (whalesong.managers.message.TextMessage property), 67
 from (whalesong.managers.message.VCardMessage property), 101
 from (whalesong.managers.message.VideoMessage property), 79

G

get_attr_by_path() (whalesong.managers.chat.ChatManager.LiveLocation method), 39
 get_resive() (whalesong.managers.chat.ChatCollectionManager method), 35
 get_attrs_by_path() (whalesong.managers.chat.ChatManager.LiveLocation method), 39
 get_commands() (whalesong.managers.BaseManager method), 23
 get_commands() (whalesong.managers.chat.ChatCollectionManager method), 33
 get_commands() (whalesong.managers.chat.ChatManager method), 40


```

get_commands() (whalesong.managers.chat.MsgLoadStateManager method), 43
get_commands() (whalesong.managers.chat.MsgLoadStateManager method), 176
get_commands() (whalesong.managers.conn.ConnManager method), 153
get_commands() (whalesong.managers.conn.ConnManager method), 196
get_commands() (whalesong.managers.contact.ContactCollectionManager method), 26
get_commands() (whalesong.managers.contact.ContactCollectionManager method), 198
get_commands() (whalesong.managers.contact.ContactManager method), 28
get_commands() (whalesong.managers.contact.ContactManager method), 148
get_commands() (whalesong.managers.display_info.DisplayInfoManager method), 191
get_commands() (whalesong.managers.display_info.DisplayInfoManager method), 150
get_commands() (whalesong.managers.group_metadata.GroupMetadataCollectionManager method), 133
get_commands() (whalesong.managers.group_metadata.GroupMetadataCollectionManager method), 144
get_commands() (whalesong.managers.group_metadata.GroupMetadataManager method), 135
get_commands() (whalesong.managers.group_metadata.GroupMetadataManager method), 146
get_commands() (whalesong.managers.group_metadata.ParticipantCollectionManager method), 137
get_commands() (whalesong.managers.group_metadata.ParticipantCollectionManager method), 200
get_commands() (whalesong.managers.group_metadata.ParticipantManager method), 139
get_commands() (whalesong.managers.group_metadata.ParticipantManager method), 157
get_commands() (whalesong.managers.live_location.LiveLocationCollectionManager method), 178
get_commands() (whalesong.managers.live_location.LiveLocationCollectionManager method), 142
get_commands() (whalesong.managers.live_location.LiveLocationManager method), 180
get_commands() (whalesong.managers.live_location.LiveLocationManager method), 182
get_commands() (whalesong.managers.live_location.ParticipantCollectionManager method), 184
get_commands() (whalesong.managers.live_location.ParticipantCollectionManager method), 23
get_commands() (whalesong.managers.message.MessageAckCollectionManager method), 54
get_commands() (whalesong.managers.message.MessageAckCollectionManager method), 40
get_commands() (whalesong.managers.message.MessageAckManager method), 56
get_commands() (whalesong.managers.message.MessageAckManager method), 40
get_commands() (whalesong.managers.message.MessageCollectionManager method), 48
get_commands() (whalesong.managers.message.MessageCollectionManager method), 43
get_commands() (whalesong.managers.message.MessageInfoManager method), 52
get_commands() (whalesong.managers.message.MessageInfoManager method), 52
get_commands() (whalesong.managers.message.MessageManager method), 50
get_commands() (whalesong.managers.message.MessageManager method), 50
get_commands() (whalesong.managers.mute.MuteCollectionManager method), 187
get_commands() (whalesong.managers.mute.MuteCollectionManager method), 28
get_commands() (whalesong.managers.mute.MuteManager method), 189
get_commands() (whalesong.managers.mute.MuteManager method), 189
get_commands() (whalesong.managers.presence.ChatStateCollectionManager method), 164
get_commands() (whalesong.managers.presence.ChatStateCollectionManager method), 164
get_commands() (whalesong.managers.presence.ChatStateManager method), 166
get_commands() (whalesong.managers.presence.ChatStateManager method), 135
get_commands() (whalesong.managers.presence.PresenceCollectionManager method), 161
get_commands() (whalesong.managers.presence.PresenceCollectionManager method), 161
get_commands() (whalesong.managers.presence.PresenceManager method), 163
get_commands() (whalesong.managers.presence.PresenceManager method), 139
get_commands() (whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager method), 170
get_commands() (whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager method), 180
get_commands() (whalesong.managers.profile_pic_thumb.ProfilePictureManager method), 172
get_commands() (whalesong.managers.profile_pic_thumb.ProfilePictureManager method), 172
get_commands() (whalesong.managers.status.StatusCollectionManager method), 174
get_commands() (whalesong.managers.status.StatusCollectionManager method), 174

```

225

[illegible]

<code>(whalesong.managers.chat.ChatCollectionManager</code>	<code>method), 27</code>
<code>method), 34</code>	<code>get_last() (whalesong.managers.group_metadata.GroupMetadataColl</code>
<code>get_iterator_result_class()</code>	<code>method), 133</code>
<code>(whalesong.managers.contact.ContactCollectionManager</code>	<code>get_last() (whalesong.managers.group_metadata.ParticipantCollection</code>
<code>method), 27</code>	<code>method), 137</code>
<code>get_iterator_result_class()</code>	<code>get_last() (whalesong.managers.live_location.LiveLocationCollection</code>
<code>(whalesong.managers.group_metadata.GroupMetadataCollectionManager</code>	<code>method), 133</code>
<code>method), 133</code>	<code>get_last() (whalesong.managers.live_location.ParticipantCollectionM</code>
<code>get_iterator_result_class()</code>	<code>method), 182</code>
<code>(whalesong.managers.group_metadata.ParticipantCollectionManager</code>	<code>get_last() (whalesong.managers.message.MessageAckCollectionManag</code>
<code>method), 137</code>	<code>method), 54</code>
<code>get_iterator_result_class()</code>	<code>get_last() (whalesong.managers.message.MessageCollectionManager</code>
<code>(whalesong.managers.live_location.LiveLocationCollectionManager</code>	<code>method), 48</code>
<code>method), 178</code>	<code>get_last() (whalesong.managers.mute.MuteCollectionManager</code>
<code>get_iterator_result_class()</code>	<code>method), 187</code>
<code>(whalesong.managers.live_location.ParticipantCollectionManager</code>	<code>get_last() (whalesong.managers.presence.ChatStateCollectionManag</code>
<code>method), 182</code>	<code>method), 165</code>
<code>get_iterator_result_class()</code>	<code>get_last() (whalesong.managers.presence.PresenceCollectionManager</code>
<code>(whalesong.managers.message.MessageAckCollectionManager</code>	<code>method), 161</code>
<code>method), 54</code>	<code>get_last() (whalesong.managers.profile_pic_thumb.ProfilePictureColl</code>
<code>get_iterator_result_class()</code>	<code>method), 170</code>
<code>(whalesong.managers.message.MessageCollectionManager</code>	<code>get_last() (whalesong.managers.status.StatusCollectionManager</code>
<code>method), 48</code>	<code>method), 174</code>
<code>get_iterator_result_class()</code>	<code>get_last() (whalesong.managers.status_v3.StatusV3CollectionManag</code>
<code>(whalesong.managers.mute.MuteCollectionManager</code>	<code>method), 196</code>
<code>method), 187</code>	<code>get_last() (whalesong.managers.sticker_pack.StickerCollectionManag</code>
<code>get_iterator_result_class()</code>	<code>method), 148</code>
<code>(whalesong.managers.presence.ChatStateCollectionManager</code>	<code>get_last() (whalesong.managers.sticker_pack.StickerPackCollectionM</code>
<code>method), 165</code>	<code>method), 144</code>
<code>get_iterator_result_class()</code>	<code>get_length() (whalesong.managers.BaseCollectionManager</code>
<code>(whalesong.managers.presence.PresenceCollectionManager</code>	<code>method), 24</code>
<code>method), 161</code>	<code>get_length() (whalesong.managers.chat.ChatCollectionManager</code>
<code>get_iterator_result_class()</code>	<code>method), 34</code>
<code>(whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager</code>	<code>method), 27</code>
<code>method), 170</code>	<code>get_length() (whalesong.managers.group_metadata.GroupMetadataC</code>
<code>get_iterator_result_class()</code>	<code>method), 133</code>
<code>(whalesong.managers.status.StatusCollectionManager</code>	<code>method), 174</code>
<code>method), 174</code>	<code>get_length() (whalesong.managers.group_metadata.ParticipantCollec</code>
<code>get_iterator_result_class()</code>	<code>method), 137</code>
<code>(whalesong.managers.status_v3.StatusV3CollectionManager</code>	<code>get_length() (whalesong.managers.live_location.LiveLocationCollect</code>
<code>method), 196</code>	<code>method), 179</code>
<code>get_iterator_result_class()</code>	<code>get_length() (whalesong.managers.live_location.ParticipantCollection</code>
<code>(whalesong.managers.sticker_pack.StickerCollectionManager</code>	<code>method), 182</code>
<code>method), 148</code>	<code>get_length() (whalesong.managers.message.MessageAckCollectionMa</code>
<code>get_iterator_result_class()</code>	<code>method), 54</code>
<code>(whalesong.managers.sticker_pack.StickerPackCollectionManager</code>	<code>get_length() (whalesong.managers.message.MessageCollectionManag</code>
<code>method), 144</code>	<code>method), 48</code>
<code>get_iterators() (whalesong.results.ResultManager</code>	<code>get_length() (whalesong.managers.mute.MuteCollectionManager</code>
<code>method), 21</code>	<code>method), 187</code>
<code>get_last() (whalesong.managers.BaseCollectionManager</code>	<code>get_length() (whalesong.managers.presence.ChatStateCollectionMan</code>
<code>method), 25</code>	<code>method), 165</code>
<code>get_last() (whalesong.managers.chat.ChatCollectionManager</code>	<code>get_length() (whalesong.managers.presence.PresenceCollectionMan</code>
<code>method), 34</code>	<code>method), 161</code>
<code>get_last() (whalesong.managers.contact.ContactCollectionManager</code>	<code>get_length() (whalesong.managers.profile_pic_thumb.ProfilePictureC</code>

method), 170

get_length() (whalesong.managers.status.StatusCollectionManager method), 175

get_length() (whalesong.managers.status_v3.StatusV3CollectionManager method), 196

get_length() (whalesong.managers.sticker_pack.StickerCollectionManager method), 148

get_length() (whalesong.managers.sticker_pack.StickerPackCollectionManager method), 145

get_me() (whalesong.managers.contact.ContactCollectionManager method), 28

get_model() (whalesong.managers.BaseModelManager method), 23

get_model() (whalesong.managers.chat.ChatManager method), 40

get_model() (whalesong.managers.chat.MsgLoadStateManager method), 43

get_model() (whalesong.managers.conn.ConnManager method), 153

get_model() (whalesong.managers.contact.ContactManager method), 29

get_model() (whalesong.managers.display_info.DisplayInfoManager method), 191

get_model() (whalesong.managers.group_metadata.GroupMetadataManager method), 135

get_model() (whalesong.managers.group_metadata.ParticipantManager method), 139

get_model() (whalesong.managers.live_location.LiveLocationManager method), 180

get_model() (whalesong.managers.live_location.ParticipantManager method), 184

get_model() (whalesong.managers.message.MessageAckManager method), 56

get_model() (whalesong.managers.message.MessageInfoManager method), 52

get_model() (whalesong.managers.message.MessageManager method), 50

get_model() (whalesong.managers.mute.MuteManager method), 189

get_model() (whalesong.managers.presence.ChatStateManager method), 166

get_model() (whalesong.managers.presence.PresenceManager method), 163

get_model() (whalesong.managers.profile_pic_thumb.ProfilePictureManager method), 172

get_model() (whalesong.managers.status.StatusManager method), 176

get_model() (whalesong.managers.status_v3.StatusV3Manager method), 198

get_model() (whalesong.managers.sticker_pack.StickerManager method), 150

get_model() (whalesong.managers.sticker_pack.StickerPackManager method), 146

get_model() (whalesong.managers.stream.StreamManager method), 157

get_model() (whalesong.managers.wap.WapManager method), 142

get_model_result_class() (whalesong.managers.BaseModelManager method), 23

get_model_result_class() (whalesong.managers.chat.ChatManager method), 40

get_model_result_class() (whalesong.managers.chat.MsgLoadStateManager method), 43

get_model_result_class() (whalesong.managers.conn.ConnManager method), 153

get_model_result_class() (whalesong.managers.contact.ContactManager method), 29

get_model_result_class() (whalesong.managers.display_info.DisplayInfoManager method), 191

get_model_result_class() (whalesong.managers.group_metadata.GroupMetadataManager method), 135

get_model_result_class() (whalesong.managers.group_metadata.ParticipantManager method), 139

get_model_result_class() (whalesong.managers.live_location.LiveLocationManager method), 180

get_model_result_class() (whalesong.managers.live_location.ParticipantManager method), 184

get_model_result_class() (whalesong.managers.message.MessageAckManager method), 56

get_model_result_class() (whalesong.managers.message.MessageInfoManager method), 52

get_model_result_class() (whalesong.managers.message.MessageManager method), 50

get_model_result_class() (whalesong.managers.mute.MuteManager method), 189

get_model_result_class() (whalesong.managers.presence.ChatStateManager method), 166

get_model_result_class() (whalesong.managers.presence.PresenceManager method), 163

get_model_result_class() (whalesong.managers.profile_pic_thumb.ProfilePictureManager method), 172

get_model_result_class() (whalesong.managers.status.StatusManager method), 176

get_model_result_class() (whalesong.managers.status_v3.StatusV3Manager method), 198

get_model_result_class() (whalesong.managers.sticker_pack.StickerManager method), 150

get_model_result_class() (whalesong.managers.sticker_pack.StickerPackManager method), 146

get_model_result_class() (whalesong.managers.stream.StreamManager method), 157

```

get_model_result_class()
    (wattlesong.managers.status.StatusManager
     method), 176
get_model_result_class()
    (wattlesong.managers.status_v3.StatusV3Manager
     method), 198
get_model_result_class()
    (wattlesong.managers.sticker_pack.StickerManager
     method), 150
get_model_result_class()
    (wattlesong.managers.sticker_pack.StickerPackManager
     method), 146
get_model_result_class()
    (wattlesong.managers.stream.StreamManager
     method), 157
get_model_result_class()
    (wattlesong.managers.wap.WapManager
     method), 142
get_monitor_result_class()
    (wattlesong.managers.BaseCollectionManager
     method), 24
get_monitor_result_class()
    (wattlesong.managers.BaseModelManager
     method), 23
get_monitor_result_class()
    (wattlesong.managers.chat.ChatCollectionManager
     method), 34
get_monitor_result_class()
    (wattlesong.managers.chat.ChatManager
     method), 40
get_monitor_result_class()
    (wattlesong.managers.chat.MsgLoadStateManager
     method), 44
get_monitor_result_class()
    (wattlesong.managers.conn.ConnManager
     method), 153
get_monitor_result_class()
    (wattlesong.managers.contact.ContactCollectionManager
     method), 27
get_monitor_result_class()
    (wattlesong.managers.contact.ContactManager
     method), 29
get_monitor_result_class()
    (wattlesong.managers.display_info.DisplayInfoManager
     method), 191
get_monitor_result_class()
    (wattlesong.managers.group_metadata.GroupMetadataCollectionManager
     method), 134
get_monitor_result_class()
    (wattlesong.managers.group_metadata.GroupMetadataManager
     method), 135
get_monitor_result_class()
    (wattlesong.managers.group_metadata.ParticipantCollectionManager
     method), 138
get_monitor_result_class()
    (wattlesong.managers.group_metadata.ParticipantManager
     method), 139
get_monitor_result_class()
    (wattlesong.managers.live_location.LiveLocationCollectionManager
     method), 179
get_monitor_result_class()
    (wattlesong.managers.live_location.LiveLocationManager
     method), 180
get_monitor_result_class()
    (wattlesong.managers.live_location.ParticipantCollectionManager
     method), 183
get_monitor_result_class()
    (wattlesong.managers.live_location.ParticipantManager
     method), 184
get_monitor_result_class()
    (wattlesong.managers.message.MessageAckCollectionManager
     method), 54
get_monitor_result_class()
    (wattlesong.managers.message.MessageAckManager
     method), 56
get_monitor_result_class()
    (wattlesong.managers.message.MessageCollectionManager
     method), 48
get_monitor_result_class()
    (wattlesong.managers.message.MessageInfoManager
     method), 52
get_monitor_result_class()
    (wattlesong.managers.message.MessageManager
     method), 50
get_monitor_result_class()
    (wattlesong.managers.mute.MuteCollectionManager
     method), 187
get_monitor_result_class()
    (wattlesong.managers.mute.MuteManager
     method), 189
get_monitor_result_class()
    (wattlesong.managers.presence.ChatStateCollectionManager
     method), 165
get_monitor_result_class()
    (wattlesong.managers.presence.ChatStateManager
     method), 167
get_monitor_result_class()
    (wattlesong.managers.presence.PresenceCollectionManager
     method), 161
get_monitor_result_class()
    (wattlesong.managers.presence.PresenceManager
     method), 163
get_monitor_result_class()
    (wattlesong.managers.profile_pic_thumb.ProfilePictureCollectionManager
     method), 170
get_monitor_result_class()
    (wattlesong.managers.profile_pic_thumb.ProfilePictureManager
     method), 172

```

[get_monitor_result_class\(\)](#) (*whalesong.managers.status.StatusCollectionManager* method), 175
[get_monitor_result_class\(\)](#) (*whalesong.managers.status.StatusManager* method), 176
[get_monitor_result_class\(\)](#) (*whalesong.managers.status_v3.StatusV3CollectionManager* method), 196
[get_monitor_result_class\(\)](#) (*whalesong.managers.status_v3.StatusV3Manager* method), 198
[get_monitor_result_class\(\)](#) (*whalesong.managers.sticker_pack.StickerCollectionManager* method), 148
[get_monitor_result_class\(\)](#) (*whalesong.managers.sticker_pack.StickerManager* method), 150
[get_monitor_result_class\(\)](#) (*whalesong.managers.sticker_pack.StickerPackCollectionManager* method), 145
[get_monitor_result_class\(\)](#) (*whalesong.managers.sticker_pack.StickerPackManager* method), 146
[get_monitor_result_class\(\)](#) (*whalesong.managers.stream.StreamManager* method), 157
[get_monitor_result_class\(\)](#) (*whalesong.managers.wap.WapManager* method), 142
[get_monitors\(\)](#) (*whalesong.results.ResultManager* method), 21
[get_my_status\(\)](#) (*whalesong.managers.status_v3.StatusV3CollectionManager* method), 197
[get_next_id\(\)](#) (*whalesong.results.ResultManager* method), 20
[get_original_field_value\(\)](#) (*whalesong.managers.chat.ChatManager.LiveLocation* method), 39
[get_parent\(\)](#) (*whalesong.managers.chat.ChatManager.LiveLocation* method), 39
[get_read_only\(\)](#) (*whalesong.managers.chat.ChatManager.LiveLocation* method), 39
[get_real_name\(\)](#) (*whalesong.managers.chat.ChatManager.LiveLocation* method), 39
[get_storage\(\)](#) (*whalesong.managers.storage.StorageManager* method), 200
[get_structure\(\)](#) (*whalesong.managers.chat.ChatManager.LiveLocation* method), 39
[get_submanager\(\)](#) (*whalesong.managers.BaseCollectionManager* method), 25
[get_submanager\(\)](#) (*whalesong.managers.BaseManager* method), 23
[get_submanager\(\)](#) (*whalesong.managers.chat.ChatCollectionManager* method), 34
[get_submanager\(\)](#) (*whalesong.managers.chat.ChatManager* method), 41
[get_submanager\(\)](#) (*whalesong.managers.chat.MsgLoadStateManager* method), 44
[get_submanager\(\)](#) (*whalesong.managers.conn.ConnManager* method), 153
[get_submanager\(\)](#) (*whalesong.managers.contact.ContactCollectionManager* method), 27
[get_submanager\(\)](#) (*whalesong.managers.contact.ContactManager* method), 29
[get_submanager\(\)](#) (*whalesong.managers.display_info.DisplayInfoManager* method), 191
[get_submanager\(\)](#) (*whalesong.managers.group_metadata.GroupMetadataManager* method), 134
[get_submanager\(\)](#) (*whalesong.managers.group_metadata.GroupMetadataManager* method), 135
[get_submanager\(\)](#) (*whalesong.managers.group_metadata.ParticipantCollectionManager* method), 138
[get_submanager\(\)](#) (*whalesong.managers.group_metadata.ParticipantCollectionManager* method), 139
[get_submanager\(\)](#) (*whalesong.managers.live_location.LiveLocationCollectionManager* method), 179
[get_submanager\(\)](#) (*whalesong.managers.live_location.LiveLocationManager* method), 180
[get_submanager\(\)](#) (*whalesong.managers.live_location.ParticipantCollectionManager* method), 183
[get_submanager\(\)](#) (*whalesong.managers.live_location.ParticipantCollectionManager* method), 184
[get_submanager\(\)](#) (*whalesong.managers.message.MessageAckCollectionManager* method), 54
[get_submanager\(\)](#) (*whalesong.managers.message.MessageAckManager* method), 54
[get_submanager\(\)](#) (*whalesong.managers.message.MessageCollectionManager* method), 48
[get_submanager\(\)](#) (*whalesong.managers.message.MessageInfoManager* method), 52
[get_submanager\(\)](#) (*whalesong.managers.message.MessageManager* method), 50
[get_submanager\(\)](#) (*whalesong.managers.mute.MuteCollectionManager* method), 187
[get_submanager\(\)](#) (*whalesong.managers.mute.MuteManager* method), 189
[get_submanager\(\)](#) (*whalesong.managers.presence.ChatStateCollectionManager* method), 165
[get_submanager\(\)](#) (*whalesong.managers.presence.ChatStateManager* method), 167
[get_submanager\(\)](#) (*whalesong.managers.presence.PresenceCollectionManager* method), 161
[get_submanager\(\)](#) (*whalesong.managers.presence.PresenceManager* method), 163
[get_submanager\(\)](#) (*whalesong.managers.profile_pic_thumb.ProfilePicThumbCollectionManager* method), 170
[get_submanager\(\)](#) (*whalesong.managers.profile_pic_thumb.ProfilePicThumbManager* method), 170

H

- `method`), 172
- `get_submanager()` (`whalesong.managers.status.StatusCollectionManager`
`method`), 175
- `get_submanager()` (`whalesong.managers.status.StatusManager`
`method`), 176
- `get_submanager()` (`whalesong.managers.status_v3.StatusV3CollectionManager`
`method`), 196
- `get_submanager()` (`whalesong.managers.status_v3.StatusV3Manager`
`method`), 198
- `get_submanager()` (`whalesong.managers.sticker_pack.StickerCollectionManager`
`method`), 149
- `get_submanager()` (`whalesong.managers.sticker_pack.StickerManager`
`method`), 150
- `get_submanager()` (`whalesong.managers.sticker_pack.StickerPackCollectionManager`
`method`), 145
- `get_submanager()` (`whalesong.managers.sticker_pack.StickerPackManager`
`method`), 147
- `get_submanager()` (`whalesong.managers.storage.StorageManager`
`method`), 200
- `get_submanager()` (`whalesong.managers.stream.StreamManager`
`method`), 157
- `get_submanager()` (`whalesong.managers.wap.WapManager`
`method`), 142
- `get_unexpired()` (`whalesong.managers.status_v3.StatusV3CollectionManager`
`method`), 197
- `gifAttribution` (`whalesong.managers.message.AudioMessage`
`property`), 84
- `gifAttribution` (`whalesong.managers.message.MediaStreamMixin`
`property`), 65
- `gifAttribution` (`whalesong.managers.message.PTTMessage`
`property`), 90
- `gifAttribution` (`whalesong.managers.message.VideoMessage`
`property`), 78
- GP2 (`whalesong.managers.message.MessageTypes` `attribute`), 57
- `group_invite_code()`
(`whalesong.managers.group_metadata.GroupMetadataManager`
`method`), 136
- GROUP_NOTIFICATION
(`whalesong.managers.message.MessageTypes`
`attribute`), 57
- `groupInviteLink` (`whalesong.managers.group_metadata.GroupMetadata`
`property`), 141
- `groupMetadata` (`whalesong.managers.chat.Chat`
`property`), 45
- `GroupMetadata` (`whalesong.managers.group_metadata`
`Model`), 140
- `GroupMetadataCollectionManager` (`class` `in`
`whalesong.managers.group_metadata`), 132
- `GroupMetadataManager` (`class` `in`
`whalesong.managers.group_metadata`), 134
- `GroupNotificationMessage`
(`whalesong.managers.message` `Model`), 121
- `hard_expired` (`whalesong.managers.display_info.DisplayInfo`
`attribute`), 195
- `hasData` (`whalesong.managers.presence.Presence`
`property`), 168
- `hasLink` (`whalesong.managers.message.AudioMessage`
`property`), 87
- `hasLink` (`whalesong.managers.message.BaseMessage`
`property`), 61
- `hasLink` (`whalesong.managers.message.DocumentMessage`
`property`), 98
- `hasLink` (`whalesong.managers.message.GroupNotificationMessage`
`property`), 123
- `hasLink` (`whalesong.managers.message.ImageMessage`
`property`), 75
- `hasLink` (`whalesong.managers.message.LocationMessage`
`property`), 114
- `hasLink` (`whalesong.managers.message.MultiVCardMessage`
`property`), 108
- `hasLink` (`whalesong.managers.message.PaymentMessage`
`property`), 119
- `hasLink` (`whalesong.managers.message.PTTMessage`
`property`), 93
- `hasLink` (`whalesong.managers.message.StickerMessage`
`property`), 129
- `hasLink` (`whalesong.managers.message.TextMessage`
`property`), 69
- `hasLink` (`whalesong.managers.message.VCardMessage`
`property`), 103
- `hasLink` (`whalesong.managers.message.VideoMessage`
`property`), 81
- `hasPromises` (`whalesong.managers.message.AudioMessage`
`property`), 88
- `hasPromises` (`whalesong.managers.message.BaseMessage`
`property`), 62
- `hasPromises` (`whalesong.managers.message.DocumentMessage`
`property`), 100
- `hasPromises` (`whalesong.managers.message.GroupNotificationMessage`
`property`), 125
- `hasPromises` (`whalesong.managers.message.ImageMessage`
`property`), 76
- `hasPromises` (`whalesong.managers.message.LocationMessage`
`property`), 115
- `hasPromises` (`whalesong.managers.message.MultiVCardMessage`
`property`), 110
- `hasPromises` (`whalesong.managers.message.PaymentMessage`
`property`), 120
- `hasPromises` (`whalesong.managers.message.PTTMessage`
`property`), 94
- `hasPromises` (`whalesong.managers.message.StickerMessage`
`property`), 131
- `hasPromises` (`whalesong.managers.message.TextMessage`
`property`), 70
- `hasPromises` (`whalesong.managers.message.VCardMessage`
`property`), 103

- [property](#)), 105
 - [hasPromises \(whalesong.managers.message.VideoMessage property\)](#), 82
 - [hasSynced \(whalesong.managers.stream.Stream property\)](#), 159
 - [height \(whalesong.managers.message.ImageMessage property\)](#), 72
 - [height \(whalesong.managers.message.MediaFrameMixin property\)](#), 64
 - [height \(whalesong.managers.message.StickerMessage property\)](#), 127
 - [height \(whalesong.managers.message.VideoMessage property\)](#), 78
 - [HIDE \(whalesong.managers.display_info.DisplayInfo.DisplayState attribute\)](#), 194
- I**
- [id \(whalesong.managers.chat.Chat property\)](#), 45
 - [id \(whalesong.managers.chat.ChatManager.LiveLocation property\)](#), 40
 - [id \(whalesong.managers.chat.MsgLoadState property\)](#), 46
 - [id \(whalesong.managers.conn.Conn property\)](#), 155
 - [id \(whalesong.managers.conn.PhoneDescription property\)](#), 154
 - [id \(whalesong.managers.contact.Contact property\)](#), 30
 - [id \(whalesong.managers.display_info.DisplayInfo property\)](#), 194
 - [id \(whalesong.managers.group_metadata.GroupMetadata property\)](#), 141
 - [id \(whalesong.managers.group_metadata.Participant property\)](#), 140
 - [id \(whalesong.managers.live_location.LiveLocation property\)](#), 186
 - [id \(whalesong.managers.live_location.Participant property\)](#), 185
 - [id \(whalesong.managers.message.AudioMessage property\)](#), 83
 - [id \(whalesong.managers.message.AuthorMixin property\)](#), 65
 - [id \(whalesong.managers.message.BaseMessage property\)](#), 59
 - [id \(whalesong.managers.message.DocumentMessage property\)](#), 94
 - [id \(whalesong.managers.message.GroupNotificationMessage property\)](#), 121
 - [id \(whalesong.managers.message.ImageMessage property\)](#), 71
 - [id \(whalesong.managers.message.LinkContentMixin property\)](#), 63
 - [id \(whalesong.managers.message.LocationMessage property\)](#), 110
 - [id \(whalesong.managers.message.MediaFrameMixin property\)](#), 64
 - [id \(whalesong.managers.message.MediaMixin property\)](#), 64
 - [id \(whalesong.managers.message.MediaStreamMixin property\)](#), 65
 - [id \(whalesong.managers.message.MentionsMixin property\)](#), 63
 - [id \(whalesong.managers.message.MessageAck property\)](#), 131
 - [id \(whalesong.managers.message.MessageInfo property\)](#), 131
 - [id \(whalesong.managers.message.MultiVCardMessage property\)](#), 105
 - [id \(whalesong.managers.message.PaymentMessage property\)](#), 116
 - [id \(whalesong.managers.message.PTTMessage property\)](#), 88
 - [id \(whalesong.managers.message.QuotedMessageMixin property\)](#), 62
 - [id \(whalesong.managers.message.StickerMessage property\)](#), 125
 - [id \(whalesong.managers.message.TextMessage property\)](#), 65
 - [id \(whalesong.managers.message.VCardItem property\)](#), 58
 - [id \(whalesong.managers.message.VCardMessage property\)](#), 100
 - [id \(whalesong.managers.message.VideoMessage property\)](#), 77
 - [id \(whalesong.managers.mute.Mute property\)](#), 190
 - [id \(whalesong.managers.presence.ChatState property\)](#), 168
 - [id \(whalesong.managers.presence.Presence property\)](#), 168
 - [id \(whalesong.managers.profile_pic_thumb.ProfilePicture property\)](#), 173
 - [id \(whalesong.managers.status.Status property\)](#), 177
 - [id \(whalesong.managers.status_v3.StatusV3 property\)](#), 199
 - [id \(whalesong.managers.sticker_pack.Sticker property\)](#), 151
 - [id \(whalesong.managers.sticker_pack.StickerPack property\)](#), 152
 - [id \(whalesong.managers.stream.Stream property\)](#), 159
 - [id \(whalesong.models.BaseModel property\)](#), 22
 - [IMAGE \(whalesong.managers.message.MessageTypes attribute\)](#), 57
 - [ImageMessage \(whalesong.managers.message.Model\)](#), 71
 - [import_data\(\) \(whalesong.managers.chat.ChatManager.LiveLocation method\)](#), 39
 - [import_deleted_fields\(\) \(whalesong.managers.chat.ChatManager.LiveLocation method\)](#), 39
 - [info \(whalesong.managers.display_info.DisplayInfo](#)

property), 195
 info (whalesong.managers.message.MessageManager
 attribute), 49
 invis (whalesong.managers.message.AudioMessage
 property), 85
 invis (whalesong.managers.message.BaseMessage
 property), 58
 invis (whalesong.managers.message.DocumentMessage
 property), 96
 invis (whalesong.managers.message.GroupNotificationMessage
 property), 121
 invis (whalesong.managers.message.ImageMessage
 property), 73
 invis (whalesong.managers.message.LocationMessage
 property), 112
 invis (whalesong.managers.message.MultiVCardMessage
 property), 106
 invis (whalesong.managers.message.PaymentMessage
 property), 117
 invis (whalesong.managers.message.PTTMessage
 property), 91
 invis (whalesong.managers.message.StickerMessage
 property), 127
 invis (whalesong.managers.message.TextMessage
 property), 67
 invis (whalesong.managers.message.VCardMessage
 property), 101
 invis (whalesong.managers.message.VideoMessage
 property), 79
 inviteCode (whalesong.managers.group_metadata.GroupMetadata
 property), 141
 is_24h (whalesong.managers.conn.Conn property),
 156
 is_available_permanent ()
 (whalesong.managers.display_info.DisplayInfoManager
 method), 192
 is_locked () (whalesong.managers.chat.ChatManager.LiveLocation
 method), 39
 is_modified () (whalesong.managers.chat.ChatManager.LiveLocation
 method), 39
 is_modified_field ()
 (whalesong.managers.chat.ChatManager.LiveLocation
 method), 39
 isAdmin (whalesong.managers.group_metadata.Participants
 property), 140
 isAnnounceGrpRestrict
 (whalesong.managers.chat.Chat property),
 45
 isBizNotification
 (whalesong.managers.message.AudioMessage
 property), 86
 isBizNotification
 (whalesong.managers.message.BaseMessage
 property), 60
 isBizNotification
 (whalesong.managers.message.DocumentMessage
 property), 98
 isBizNotification
 (whalesong.managers.message.GroupNotificationMessage
 property), 123
 isBizNotification
 (whalesong.managers.message.ImageMessage
 property), 74
 isBizNotification
 (whalesong.managers.message.LocationMessage
 property), 113
 isBizNotification
 (whalesong.managers.message.MultiVCardMessage
 property), 108
 isBizNotification
 (whalesong.managers.message.PaymentMessage
 property), 119
 isBizNotification
 (whalesong.managers.message.PTTMessage
 property), 92
 isBizNotification
 (whalesong.managers.message.StickerMessage
 property), 129
 isBizNotification
 (whalesong.managers.message.TextMessage
 property), 69
 isBizNotification
 (whalesong.managers.message.VCardMessage
 property), 103
 isBizNotification
 (whalesong.managers.message.VideoMessage
 property), 81
 isBusiness (whalesong.managers.contact.Contact
 property), 31
 isContactBlocked (whalesong.managers.contact.Contact
 property), 31
 isDoc (whalesong.managers.message.AudioMessage
 property), 87
 isDoc (whalesong.managers.message.BaseMessage
 property), 61
 isDoc (whalesong.managers.message.DocumentMessage
 property), 98
 isDoc (whalesong.managers.message.GroupNotificationMessage
 property), 123
 isDoc (whalesong.managers.message.ImageMessage
 property), 75
 isDoc (whalesong.managers.message.LocationMessage
 property), 114
 isDoc (whalesong.managers.message.MultiVCardMessage
 property), 108
 isDoc (whalesong.managers.message.PaymentMessage
 property), 119
 isDoc (whalesong.managers.message.PTTMessage

	<i>property</i>), 93		<i>property</i>), 91
isDoc	(<i>whalesong.managers.message.StickerMessage</i> <i>property</i>), 129	isForwarded	(<i>whalesong.managers.message.StickerMessage</i> <i>property</i>), 128
isDoc	(<i>whalesong.managers.message.TextMessage</i> <i>property</i>), 69	isForwarded	(<i>whalesong.managers.message.TextMessage</i> <i>property</i>), 68
isDoc	(<i>whalesong.managers.message.VCardMessage</i> <i>property</i>), 103	isForwarded	(<i>whalesong.managers.message.VCardMessage</i> <i>property</i>), 102
isDoc	(<i>whalesong.managers.message.VideoMessage</i> <i>property</i>), 81	isForwarded	(<i>whalesong.managers.message.VideoMessage</i> <i>property</i>), 79
isEnterprise	(<i>whalesong.managers.contact.Contact</i> <i>property</i>), 31	isGif	(<i>whalesong.managers.message.AudioMessage</i> <i>property</i>), 84
isFailed	(<i>whalesong.managers.message.AudioMessage</i> <i>property</i>), 88	isGif	(<i>whalesong.managers.message.MediaStreamMixin</i> <i>property</i>), 65
isFailed	(<i>whalesong.managers.message.BaseMessage</i> <i>property</i>), 62	isGif	(<i>whalesong.managers.message.PTTMessage</i> <i>property</i>), 90
isFailed	(<i>whalesong.managers.message.DocumentMessage</i> <i>property</i>), 99	isGif	(<i>whalesong.managers.message.VideoMessage</i> <i>property</i>), 78
isFailed	(<i>whalesong.managers.message.GroupNotificationMessage</i> <i>property</i>), 124	isGroup	(<i>whalesong.managers.chat.Chat</i> <i>property</i>), 45
isFailed	(<i>whalesong.managers.message.ImageMessage</i> <i>property</i>), 76	isGroup	(<i>whalesong.managers.presence.Presence</i> <i>property</i>), 168
isFailed	(<i>whalesong.managers.message.LocationMessage</i> <i>property</i>), 114	isGroupMsg	(<i>whalesong.managers.message.AudioMessage</i> <i>property</i>), 86
isFailed	(<i>whalesong.managers.message.MultiVCardMessage</i> <i>property</i>), 109	isGroupMsg	(<i>whalesong.managers.message.BaseMessage</i> <i>property</i>), 59
isFailed	(<i>whalesong.managers.message.PaymentMessage</i> <i>property</i>), 120	isGroupMsg	(<i>whalesong.managers.message.DocumentMessage</i> <i>property</i>), 97
isFailed	(<i>whalesong.managers.message.PTTMessage</i> <i>property</i>), 94	isGroupMsg	(<i>whalesong.managers.message.GroupNotificationMessage</i> <i>property</i>), 122
isFailed	(<i>whalesong.managers.message.StickerMessage</i> <i>property</i>), 130	isGroupMsg	(<i>whalesong.managers.message.ImageMessage</i> <i>property</i>), 74
isFailed	(<i>whalesong.managers.message.TextMessage</i> <i>property</i>), 70	isGroupMsg	(<i>whalesong.managers.message.LocationMessage</i> <i>property</i>), 112
isFailed	(<i>whalesong.managers.message.VCardMessage</i> <i>property</i>), 104	isGroupMsg	(<i>whalesong.managers.message.MultiVCardMessage</i> <i>property</i>), 107
isFailed	(<i>whalesong.managers.message.VideoMessage</i> <i>property</i>), 82	isGroupMsg	(<i>whalesong.managers.message.PaymentMessage</i> <i>property</i>), 118
isForwarded	(<i>whalesong.managers.message.AudioMessage</i> <i>property</i>), 85	isGroupMsg	(<i>whalesong.managers.message.PTTMessage</i> <i>property</i>), 91
isForwarded	(<i>whalesong.managers.message.BaseMessage</i> <i>property</i>), 59	isGroupMsg	(<i>whalesong.managers.message.StickerMessage</i> <i>property</i>), 128
isForwarded	(<i>whalesong.managers.message.DocumentMessage</i> <i>property</i>), 97	isGroupMsg	(<i>whalesong.managers.message.TextMessage</i> <i>property</i>), 68
isForwarded	(<i>whalesong.managers.message.GroupNotificationMessage</i> <i>property</i>), 122	isGroupMsg	(<i>whalesong.managers.message.VCardMessage</i> <i>property</i>), 102
isForwarded	(<i>whalesong.managers.message.ImageMessage</i> <i>property</i>), 73	isGroupMsg	(<i>whalesong.managers.message.VideoMessage</i> <i>property</i>), 80
isForwarded	(<i>whalesong.managers.message.LocationMessage</i> <i>property</i>), 112	isGroupNotification	
isForwarded	(<i>whalesong.managers.message.MultiVCardMessage</i> <i>property</i>), 107		(<i>whalesong.managers.message.AudioMessage</i> <i>property</i>), 86
isForwarded	(<i>whalesong.managers.message.PaymentMessage</i> <i>property</i>), 117	isGroupNotification	
isForwarded	(<i>whalesong.managers.message.PTTMessage</i> <i>property</i>), 117		(<i>whalesong.managers.message.BaseMessage</i> <i>property</i>), 60
		isGroupNotification	
			(<i>whalesong.managers.message.DocumentMessage</i> <i>property</i>), 60

property), 98
 isGroupNotification (wattlesong.managers.message.GroupNotificationMessage property), 123
 isGroupNotification (wattlesong.managers.message.ImageMessage property), 74
 isGroupNotification (wattlesong.managers.message.LocationMessage property), 113
 isGroupNotification (wattlesong.managers.message.MultiVCardMessage property), 108
 isGroupNotification (wattlesong.managers.message.PaymentMessage property), 118
 isGroupNotification (wattlesong.managers.message.PTTMessage property), 92
 isGroupNotification (wattlesong.managers.message.StickerMessage property), 129
 isGroupNotification (wattlesong.managers.message.TextMessage property), 68
 isGroupNotification (wattlesong.managers.message.VCardMessage property), 103
 isGroupNotification (wattlesong.managers.message.VideoMessage property), 80
 isHighLevelVerified (wattlesong.managers.contact.Contact property), 31
 isIncognito (wattlesong.managers.stream.Stream property), 159
 isLink (wattlesong.managers.message.AudioMessage property), 87
 isLink (wattlesong.managers.message.BaseMessage property), 61
 isLink (wattlesong.managers.message.DocumentMessage property), 98
 isLink (wattlesong.managers.message.GroupNotificationMessage property), 123
 isLink (wattlesong.managers.message.ImageMessage property), 75
 isLink (wattlesong.managers.message.LocationMessage property), 113
 isLink (wattlesong.managers.message.MultiVCardMessage property), 108
 isLink (wattlesong.managers.message.PaymentMessage property), 119
 isLink (wattlesong.managers.message.PTTMessage property), 93
 isLink (wattlesong.managers.message.StickerMessage property), 129
 isLink (wattlesong.managers.message.TextMessage property), 69
 isLink (wattlesong.managers.message.VCardMessage property), 103
 isLink (wattlesong.managers.message.VideoMessage property), 81
 isLive (wattlesong.managers.message.LocationMessage property), 115
 isLoadingAroundMsgs (wattlesong.managers.chat.MsgLoadState property), 46
 isLoadingEarlierMsgs (wattlesong.managers.chat.MsgLoadState property), 46
 isLoadingRecentMsgs (wattlesong.managers.chat.MsgLoadState property), 46
 isMe (wattlesong.managers.contact.Contact property), 31
 isMedia (wattlesong.managers.message.AudioMessage property), 87
 isMedia (wattlesong.managers.message.BaseMessage property), 60
 isMedia (wattlesong.managers.message.DocumentMessage property), 98
 isMedia (wattlesong.managers.message.GroupNotificationMessage property), 123
 isMedia (wattlesong.managers.message.ImageMessage property), 75
 isMedia (wattlesong.managers.message.LocationMessage property), 113
 isMedia (wattlesong.managers.message.MultiVCardMessage property), 108
 isMedia (wattlesong.managers.message.PaymentMessage property), 119
 isMedia (wattlesong.managers.message.PTTMessage property), 93
 isMedia (wattlesong.managers.message.StickerMessage property), 129
 isMedia (wattlesong.managers.message.TextMessage property), 69
 isMedia (wattlesong.managers.message.VCardMessage property), 103
 isMedia (wattlesong.managers.message.VideoMessage property), 81
 isMms (wattlesong.managers.message.AudioMessage property), 87
 isMms (wattlesong.managers.message.BaseMessage property), 61
 isMms (wattlesong.managers.message.DocumentMessage property), 99
 isMms (wattlesong.managers.message.GroupNotificationMessage

<code>property)</code> , 124	<code>property)</code> , 98
<code>isMms (whalesong.managers.message.ImageMessage property)</code> , 75	<code>isNotification (whalesong.managers.message.GroupNotificationMessage property)</code> , 123
<code>isMms (whalesong.managers.message.LocationMessage property)</code> , 114	<code>isNotification (whalesong.managers.message.ImageMessage property)</code> , 74
<code>isMms (whalesong.managers.message.MultiVCardMessage property)</code> , 108	<code>isNotification (whalesong.managers.message.LocationMessage property)</code> , 113
<code>isMms (whalesong.managers.message.PaymentMessage property)</code> , 119	<code>isNotification (whalesong.managers.message.MultiVCardMessage property)</code> , 107
<code>isMms (whalesong.managers.message.PTTMessage property)</code> , 93	<code>isNotification (whalesong.managers.message.PaymentMessage property)</code> , 118
<code>isMms (whalesong.managers.message.StickerMessage property)</code> , 130	<code>isNotification (whalesong.managers.message.PTTMessage property)</code> , 92
<code>isMms (whalesong.managers.message.TextMessage property)</code> , 69	<code>isNotification (whalesong.managers.message.StickerMessage property)</code> , 129
<code>isMms (whalesong.managers.message.VCardMessage property)</code> , 104	<code>isNotification (whalesong.managers.message.TextMessage property)</code> , 68
<code>isMms (whalesong.managers.message.VideoMessage property)</code> , 81	<code>isNotification (whalesong.managers.message.VCardMessage property)</code> , 103
<code>isMuted (whalesong.managers.mute.Mute property)</code> , 190	<code>isNotification (whalesong.managers.message.VideoMessage property)</code> , 80
<code>isMyContact (whalesong.managers.contact.Contact property)</code> , 31	<code>isOnline (whalesong.managers.presence.Presence property)</code> , 169
<code>isNewMsg (whalesong.managers.message.AudioMessage property)</code> , 85	<code>isPersistent (whalesong.managers.message.AudioMessage property)</code> , 88
<code>isNewMsg (whalesong.managers.message.BaseMessage property)</code> , 59	<code>isPersistent (whalesong.managers.message.BaseMessage property)</code> , 62
<code>isNewMsg (whalesong.managers.message.DocumentMessage property)</code> , 97	<code>isPersistent (whalesong.managers.message.DocumentMessage property)</code> , 99
<code>isNewMsg (whalesong.managers.message.GroupNotificationMessage property)</code> , 122	<code>isPersistent (whalesong.managers.message.GroupNotificationMessage property)</code> , 124
<code>isNewMsg (whalesong.managers.message.ImageMessage property)</code> , 73	<code>isPersistent (whalesong.managers.message.ImageMessage property)</code> , 76
<code>isNewMsg (whalesong.managers.message.LocationMessage property)</code> , 112	<code>isPersistent (whalesong.managers.message.LocationMessage property)</code> , 115
<code>isNewMsg (whalesong.managers.message.MultiVCardMessage property)</code> , 106	<code>isPersistent (whalesong.managers.message.MultiVCardMessage property)</code> , 109
<code>isNewMsg (whalesong.managers.message.PaymentMessage property)</code> , 117	<code>isPersistent (whalesong.managers.message.PaymentMessage property)</code> , 120
<code>isNewMsg (whalesong.managers.message.PTTMessage property)</code> , 91	<code>isPersistent (whalesong.managers.message.PTTMessage property)</code> , 94
<code>isNewMsg (whalesong.managers.message.StickerMessage property)</code> , 128	<code>isPersistent (whalesong.managers.message.StickerMessage property)</code> , 130
<code>isNewMsg (whalesong.managers.message.TextMessage property)</code> , 67	<code>isPersistent (whalesong.managers.message.TextMessage property)</code> , 70
<code>isNewMsg (whalesong.managers.message.VCardMessage property)</code> , 102	<code>isPersistent (whalesong.managers.message.VCardMessage property)</code> , 104
<code>isNewMsg (whalesong.managers.message.VideoMessage property)</code> , 79	<code>isPersistent (whalesong.managers.message.VideoMessage property)</code> , 82
<code>isNotification (whalesong.managers.message.AudioMessage property)</code> , 86	<code>isMyContact (whalesong.managers.contact.Contact property)</code> , 31
<code>isNotification (whalesong.managers.message.BaseMessage property)</code> , 60	<code>isNewMsg (whalesong.managers.message.AudioMessage property)</code> , 86
<code>isNotification (whalesong.managers.message.DocumentMessage property)</code> , 97	<code>isNewMsg (whalesong.managers.message.BaseMessage property)</code> , 59

property), 60

isPSA (whalesong.managers.message.DocumentMessage property), 97

isPSA (whalesong.managers.message.GroupNotificationMessage property), 122

isPSA (whalesong.managers.message.ImageMessage property), 74

isPSA (whalesong.managers.message.LocationMessage property), 112

isPSA (whalesong.managers.message.MultiVCardMessage property), 107

isPSA (whalesong.managers.message.PaymentMessage property), 118

isPSA (whalesong.managers.message.PTTMessage property), 92

isPSA (whalesong.managers.message.StickerMessage property), 128

isPSA (whalesong.managers.message.TextMessage property), 68

isPSA (whalesong.managers.message.VCardMessage property), 102

isPSA (whalesong.managers.message.VideoMessage property), 80

isPtt (whalesong.managers.message.MessageInfo property), 131

isReadOnly (whalesong.managers.chat.Chat property), 45

isResponse (whalesong.managers.conn.Conn property), 155

isRevoked (whalesong.managers.message.AudioMessage property), 87

isRevoked (whalesong.managers.message.BaseMessage property), 61

isRevoked (whalesong.managers.message.DocumentMessage property), 99

isRevoked (whalesong.managers.message.GroupNotificationMessage property), 124

isRevoked (whalesong.managers.message.ImageMessage property), 75

isRevoked (whalesong.managers.message.LocationMessage property), 114

isRevoked (whalesong.managers.message.MultiVCardMessage property), 109

isRevoked (whalesong.managers.message.PaymentMessage property), 119

isRevoked (whalesong.managers.message.PTTMessage property), 93

isRevoked (whalesong.managers.message.StickerMessage property), 130

isRevoked (whalesong.managers.message.TextMessage property), 69

isRevoked (whalesong.managers.message.VCardMessage property), 104

isRevoked (whalesong.managers.message.VideoMessage property), 81

isSentByMe (whalesong.managers.message.AudioMessage property), 86

isSentByMe (whalesong.managers.message.BaseMessage property), 60

isSentByMe (whalesong.managers.message.DocumentMessage property), 98

isSentByMe (whalesong.managers.message.GroupNotificationMessage property), 123

isSentByMe (whalesong.managers.message.ImageMessage property), 74

isSentByMe (whalesong.managers.message.LocationMessage property), 113

isSentByMe (whalesong.managers.message.MultiVCardMessage property), 107

isSentByMe (whalesong.managers.message.PaymentMessage property), 118

isSentByMe (whalesong.managers.message.PTTMessage property), 92

isSentByMe (whalesong.managers.message.StickerMessage property), 129

isSentByMe (whalesong.managers.message.TextMessage property), 68

isSentByMe (whalesong.managers.message.VCardMessage property), 102

isSentByMe (whalesong.managers.message.VideoMessage property), 80

isState (whalesong.managers.display_info.DisplayInfo property), 194

isState (whalesong.managers.mute.Mute property), 190

isState (whalesong.managers.presence.ChatState property), 168

isStatusV3 (whalesong.managers.message.AudioMessage property), 86

isStatusV3 (whalesong.managers.message.BaseMessage property), 60

isStatusV3 (whalesong.managers.message.DocumentMessage property), 97

isStatusV3 (whalesong.managers.message.GroupNotificationMessage property), 122

isStatusV3 (whalesong.managers.message.ImageMessage property), 74

isStatusV3 (whalesong.managers.message.LocationMessage property), 112

isStatusV3 (whalesong.managers.message.MultiVCardMessage property), 107

isStatusV3 (whalesong.managers.message.PaymentMessage property), 118

isStatusV3 (whalesong.managers.message.PTTMessage property), 92

isStatusV3 (whalesong.managers.message.StickerMessage property), 128

isStatusV3 (whalesong.managers.message.TextMessage property), 68

property), 68
isStatusV3 (whalesong.managers.message.VCardMessage property), 102
isStatusV3 (whalesong.managers.message.VideoMessage property), 80
isSubscribed (whalesong.managers.presence.Presence property), 169
isSuperAdmin (whalesong.managers.group_metadata.Participant property), 140
isUnsentMedia (whalesong.managers.message.AudioMessage property), 84
isUnsentMedia (whalesong.managers.message.DocumentMessage property), 96
isUnsentMedia (whalesong.managers.message.ImageMessage property), 72
isUnsentMedia (whalesong.managers.message.MediaMixin property), 64
isUnsentMedia (whalesong.managers.message.PTTMessage property), 90
isUnsentMedia (whalesong.managers.message.StickerMessage property), 126
isUnsentMedia (whalesong.managers.message.VideoMessage property), 78
isUnsentMedia (whalesong.managers.sticker_pack.Sticker property), 152
isUser (whalesong.managers.contact.Contact property), 30
isUser (whalesong.managers.presence.Presence property), 168
isUserCreatedType (whalesong.managers.message.AudioMessage property), 88
isUserCreatedType (whalesong.managers.message.BaseMessage property), 62
isUserCreatedType (whalesong.managers.message.DocumentMessage property), 100
isUserCreatedType (whalesong.managers.message.GroupNotificationMessage property), 125
isUserCreatedType (whalesong.managers.message.ImageMessage property), 76
isUserCreatedType (whalesong.managers.message.LocationMessage property), 115
isUserCreatedType (whalesong.managers.message.MultiVCardMessage property), 109
isUserCreatedType (whalesong.managers.message.PaymentMessage property), 120
isUserCreatedType (whalesong.managers.message.PTTMessage property), 94
isUserCreatedType (whalesong.managers.message.StickerMessage property), 131
isUserCreatedType (whalesong.managers.message.TextMessage property), 70
isUserCreatedType (whalesong.managers.message.VCardMessage property), 105
isUserCreatedType (whalesong.managers.message.VideoMessage property), 82
isVerified (whalesong.managers.contact.Contact property), 32
isWAContact (whalesong.managers.contact.Contact property), 32
IteratorResult (class in whalesong.results), 20
J
JSONEncoder (class in whalesong.models), 22
K
kind (whalesong.managers.chat.Chat property), 44
L
labels (whalesong.managers.contact.Contact property), 32
language (whalesong.managers.conn.Conn property), 156
lastMessageTs (whalesong.managers.chat.Chat property), 45
lastReceivedKey (whalesong.managers.chat.Chat property), 46
lastReceivedKey (whalesong.managers.status_v3.StatusV3 property), 199
lastUpdate (whalesong.managers.live_location.Participant property), 185
lat (whalesong.managers.live_location.Participant property), 185
lat (whalesong.managers.message.LocationMessage property), 110
launch_generation (whalesong.managers.stream.Stream attribute), 159
launched (whalesong.managers.stream.Stream property), 159
leave_group() (whalesong.managers.chat.ChatManager method), 37
LinkContentMixin (whalesong.managers.message Model), 63
linkPreview (whalesong.managers.message.LinkContentMixin property), 63

[linkPreview \(whalesong.managers.message.TextMessage property\), 66](#)
[links \(whalesong.managers.message.AudioMessage property\), 85](#)
[links \(whalesong.managers.message.BaseMessage property\), 59](#)
[links \(whalesong.managers.message.DocumentMessage property\), 97](#)
[links \(whalesong.managers.message.GroupNotificationMessage property\), 122](#)
[links \(whalesong.managers.message.ImageMessage property\), 73](#)
[links \(whalesong.managers.message.LinkContentMixin property\), 63](#)
[links \(whalesong.managers.message.LocationMessage property\), 112](#)
[links \(whalesong.managers.message.MultiVCardMessage property\), 107](#)
[links \(whalesong.managers.message.PaymentMessage property\), 118](#)
[links \(whalesong.managers.message.PTTMessage property\), 91](#)
[links \(whalesong.managers.message.StickerMessage property\), 128](#)
[links \(whalesong.managers.message.TextMessage property\), 66](#)
[links \(whalesong.managers.message.VCardMessage property\), 102](#)
[links \(whalesong.managers.message.VideoMessage property\), 80](#)
[live_location \(whalesong.managers.chat.ChatManager attribute\), 36](#)
[live_locations \(whalesong.Walesong attribute\), 17](#)
[LiveLocation \(whalesong.managers.live_location.Model\), 186](#)
[LiveLocationCollectionManager \(class in whalesong.managers.live_location\), 177](#)
[LiveLocationManager \(class in whalesong.managers.live_location\), 180](#)
[lng \(whalesong.managers.live_location.Participant property\), 185](#)
[lng \(whalesong.managers.message.LocationMessage property\), 110](#)
[load_all_earlier_messages \(\) \(whalesong.managers.chat.ChatManager method\), 38](#)
[load_earlier_messages \(\) \(whalesong.managers.chat.ChatManager method\), 37](#)
[loc \(whalesong.managers.message.LocationMessage property\), 110](#)
[locale \(whalesong.managers.conn.Conn property\), 155](#)
[locales \(whalesong.managers.conn.Conn property\), 156](#)
[LOCATION \(whalesong.managers.message.MessageTypes attribute\), 57](#)
[LocationMessage \(whalesong.managers.message.Model\), 110](#)
[lock \(\) \(whalesong.managers.chat.ChatManager.LiveLocation method\), 40](#)
[logout \(\) \(whalesong.managers.stream.StreamManager method\), 158](#)
[loop \(whalesong.Walesong attribute\), 17](#)

M

[MAIN \(whalesong.managers.display_info.DisplayInfo.StreamMode attribute\), 193](#)
[ManagerNotFound, 201](#)
[map \(\) \(whalesong.results.BaseResultMixin method\), 19](#)
[map \(\) \(whalesong.results.IteratorResult method\), 20](#)
[map_model \(\) \(whalesong.managers.BaseModelManager method\), 23](#)
[map_model \(\) \(whalesong.managers.chat.ChatManager method\), 41](#)
[map_model \(\) \(whalesong.managers.chat.MsgLoadStateManager method\), 44](#)
[map_model \(\) \(whalesong.managers.conn.ConnManager method\), 153](#)
[map_model \(\) \(whalesong.managers.contact.ContactManager method\), 29](#)
[map_model \(\) \(whalesong.managers.display_info.DisplayInfoManager method\), 192](#)
[map_model \(\) \(whalesong.managers.group_metadata.GroupMetadataManager method\), 135](#)
[map_model \(\) \(whalesong.managers.group_metadata.ParticipantManager method\), 140](#)
[map_model \(\) \(whalesong.managers.live_location.LiveLocationManager method\), 181](#)
[map_model \(\) \(whalesong.managers.live_location.ParticipantManager method\), 184](#)
[map_model \(\) \(whalesong.managers.message.MessageAckManager method\), 56](#)
[map_model \(\) \(whalesong.managers.message.MessageInfoManager method\), 52](#)
[map_model \(\) \(whalesong.managers.message.MessageManager method\), 50](#)
[map_model \(\) \(whalesong.managers.mute.MuteManager method\), 190](#)
[map_model \(\) \(whalesong.managers.presence.ChatStateManager method\), 167](#)
[map_model \(\) \(whalesong.managers.presence.PresenceManager method\), 163](#)
[map_model \(\) \(whalesong.managers.profile_pic_thumb.ProfilePictureManager method\), 172](#)
[map_model \(\) \(whalesong.managers.status.StatusManager method\), 177](#)

[map_model\(\) \(whalesong.managers.status_v3.StatusV3Manager method\), 198](#)
[map_model\(\) \(whalesong.managers.sticker_pack.StickerManager method\), 151](#)
[map_model\(\) \(whalesong.managers.sticker_pack.StickerPackManager method\), 147](#)
[map_model\(\) \(whalesong.managers.stream.StreamManager method\), 157](#)
[map_model\(\) \(whalesong.managers.wap.WapManager method\), 142](#)
[mark_available\(\) \(whalesong.managers.display_info.DisplayInfoManager method\), 192](#)
[mark_composing\(\) \(whalesong.managers.chat.ChatManager method\), 41](#)
[mark_paused\(\) \(whalesong.managers.chat.ChatManager method\), 41](#)
[mark_recording\(\) \(whalesong.managers.chat.ChatManager method\), 41](#)
[mark_unavailable\(\) \(whalesong.managers.display_info.DisplayInfoManager method\), 192](#)
[matchedText \(whalesong.managers.message.LinkContentMixin property\), 63](#)
[matchedText \(whalesong.managers.message.TextMessage property\), 66](#)
[mcc \(whalesong.managers.conn.PhoneDescription property\), 154](#)
[me \(whalesong.managers.conn.Conn property\), 155](#)
[MediaFrameMixin \(whalesong.managers.message Model\), 64](#)
[mediaKey \(whalesong.managers.message.AudioMessage property\), 84](#)
[mediaKey \(whalesong.managers.message.DocumentMessage property\), 95](#)
[mediaKey \(whalesong.managers.message.ImageMessage property\), 72](#)
[mediaKey \(whalesong.managers.message.MediaMixin property\), 64](#)
[mediaKey \(whalesong.managers.message.PTTMessage property\), 89](#)
[mediaKey \(whalesong.managers.message.StickerMessage property\), 126](#)
[mediaKey \(whalesong.managers.message.VideoMessage property\), 78](#)
[mediaKey \(whalesong.managers.sticker_pack.Sticker property\), 152](#)
[MediaMixin \(whalesong.managers.message Model\), 63](#)
[MediaStreamMixin \(whalesong.managers.message Model\), 65](#)
[mentionedJidList \(whalesong.managers.message.AudioMessage property\), 83](#)
[mentionedJidList \(whalesong.managers.message.DocumentMessage property\), 95](#)
[mentionedJidList \(whalesong.managers.message.ImageMessage property\), 71](#)
[mentionedJidList \(whalesong.managers.message.LocationMessage property\), 111](#)
[mentionedJidList \(whalesong.managers.message.MentionsMixin property\), 63](#)
[mentionedJidList \(whalesong.managers.message.MultiVCardMessage property\), 105](#)
[mentionedJidList \(whalesong.managers.message.PTTMessage property\), 89](#)
[mentionedJidList \(whalesong.managers.message.StickerMessage property\), 126](#)
[mentionedJidList \(whalesong.managers.message.TextMessage property\), 66](#)
[mentionedJidList \(whalesong.managers.message.VCardMessage property\), 101](#)
[mentionedJidList \(whalesong.managers.message.VideoMessage property\), 77](#)
[MentionsMixin \(whalesong.managers.message Model\), 63](#)
[MessageAck \(whalesong.managers.message Model\), 131](#)
[MessageAckCollectionManager \(class in whalesong.managers.message\), 53](#)
[MessageAckManager \(class in whalesong.managers.message\), 55](#)
[MessageCollectionManager \(class in whalesong.managers.message\), 47](#)
[MessageInfo \(whalesong.managers.message Model\), 131](#)
[MessageInfoManager \(class in whalesong.managers.message\), 51](#)
[MessageManager \(class in whalesong.managers.message\), 49](#)
[MessageMetaclass \(class in whalesong.managers.message\), 58](#)
[messages \(whalesong.Wholesong attribute\), 16](#)
[MessageTypes \(class in whalesong.managers.message\), 57](#)
[mimetype \(whalesong.managers.message.AudioMessage property\), 83](#)
[mimetype \(whalesong.managers.message.DocumentMessage property\), 95](#)
[mimetype \(whalesong.managers.message.ImageMessage property\), 72](#)
[mimetype \(whalesong.managers.message.MediaMixin property\), 64](#)
[mimetype \(whalesong.managers.message.PTTMessage property\), 89](#)
[mimetype \(whalesong.managers.message.StickerMessage property\), 126](#)
[mimetype \(whalesong.managers.message.VideoMessage property\), 77](#)
[mimetype \(whalesong.managers.sticker_pack.Sticker property\), 152](#)

property), 151
 mnc (whalesong.managers.conn.PhoneDescription prop-
 erty), 154
 mode (whalesong.managers.display_info.DisplayInfo
 property), 195
 MODEL_CLASS (whalesong.managers.chat.ChatManager
 attribute), 36
 MODEL_CLASS (whalesong.managers.chat.MsgLoadStateManager
 attribute), 43
 MODEL_CLASS (whalesong.managers.conn.ConnManager
 attribute), 153
 MODEL_CLASS (whalesong.managers.contact.ContactManager
 attribute), 29
 MODEL_CLASS (whalesong.managers.display_info.DisplayInfoManager
 attribute), 192
 MODEL_CLASS (whalesong.managers.group_metadata.GroupMetadataManager
 attribute), 136
 MODEL_CLASS (whalesong.managers.group_metadata.ParticipantManager
 attribute), 139
 MODEL_CLASS (whalesong.managers.live_location.LiveLocationManager
 attribute), 181
 MODEL_CLASS (whalesong.managers.live_location.ParticipantManager
 attribute), 185
 MODEL_CLASS (whalesong.managers.message.MessageAckManager
 attribute), 56
 MODEL_CLASS (whalesong.managers.message.MessageInfoManager
 attribute), 53
 MODEL_CLASS (whalesong.managers.message.MessageManager
 attribute), 51
 MODEL_CLASS (whalesong.managers.mute.MuteManager
 attribute), 189
 MODEL_CLASS (whalesong.managers.presence.ChatStateManager
 attribute), 166
 MODEL_CLASS (whalesong.managers.presence.PresenceManager
 attribute), 162
 MODEL_CLASS (whalesong.managers.profile_pic_thumb.ProfilePictureManager
 attribute), 171
 MODEL_CLASS (whalesong.managers.status.StatusManager
 attribute), 176
 MODEL_CLASS (whalesong.managers.status_v3.StatusV3Manager
 attribute), 198
 MODEL_CLASS (whalesong.managers.sticker_pack.StickerManager
 attribute), 150
 MODEL_CLASS (whalesong.managers.sticker_pack.StickerPackManager
 attribute), 146
 MODEL_CLASS (whalesong.managers.stream.StreamManager
 attribute), 157
 MODEL_CLASS (whalesong.managers.wap.WapManager
 attribute), 142
 MODEL_MANAGER_CLASS
 (whalesong.managers.chat.ChatCollectionManager
 attribute), 35
 MODEL_MANAGER_CLASS
 (whalesong.managers.contact.ContactCollectionManager
 attribute), 27
 attribute), 28
 MODEL_MANAGER_CLASS
 (whalesong.managers.group_metadata.GroupMetadataCollectionManager
 attribute), 134
 MODEL_MANAGER_CLASS
 (whalesong.managers.group_metadata.ParticipantCollectionManager
 attribute), 136
 MODEL_MANAGER_CLASS
 (whalesong.managers.live_location.LiveLocationCollectionManager
 attribute), 179
 MODEL_MANAGER_CLASS
 (whalesong.managers.live_location.ParticipantCollectionManager
 attribute), 183
 MODEL_MANAGER_CLASS
 (whalesong.managers.message.MessageAckCollectionManager
 attribute), 55
 MODEL_MANAGER_CLASS
 (whalesong.managers.message.MessageCollectionManager
 attribute), 49
 MODEL_MANAGER_CLASS
 (whalesong.managers.mute.MuteCollectionManager
 attribute), 186
 MODEL_MANAGER_CLASS
 (whalesong.managers.presence.ChatStateCollectionManager
 attribute), 164
 MODEL_MANAGER_CLASS
 (whalesong.managers.presence.PresenceCollectionManager
 attribute), 160
 MODEL_MANAGER_CLASS
 (whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager
 attribute), 171
 MODEL_MANAGER_CLASS
 (whalesong.managers.status.StatusCollectionManager
 attribute), 174
 MODEL_MANAGER_CLASS
 (whalesong.managers.status_v3.StatusV3CollectionManager
 attribute), 195
 MODEL_MANAGER_CLASS
 (whalesong.managers.sticker_pack.StickerCollectionManager
 attribute), 147
 MODEL_MANAGER_CLASS
 (whalesong.managers.sticker_pack.StickerPackCollectionManager
 attribute), 143
 matterIter (class in whalesong.models),
 22
 ModelNotFound, 201
 modifyTag (whalesong.managers.chat.Chat property),
 46
 monitor_add() (whalesong.managers.BaseCollectionManager
 method), 25
 monitor_add() (whalesong.managers.chat.ChatCollectionManager
 method), 34
 monitor_add() (whalesong.managers.contact.ContactCollectionManager
 method), 27

`monitor_add()` (`whalesong.managers.group_metadata.GroupMetadataCollectionManager` method), 134
`monitor_add()` (`whalesong.managers.group_metadata.ParticipantCollectionManager` method), 138
`monitor_add()` (`whalesong.managers.live_location.LiveLocationCollectionManager` method), 179
`monitor_add()` (`whalesong.managers.live_location.ParticipantCollectionManager` method), 183
`monitor_add()` (`whalesong.managers.message.MessageAckCollectionManager` method), 54
`monitor_add()` (`whalesong.managers.message.MessageCollectionManager` method), 48
`monitor_add()` (`whalesong.managers.mute.MuteCollectionManager` method), 188
`monitor_add()` (`whalesong.managers.presence.ChatStateCollectionManager` method), 165
`monitor_add()` (`whalesong.managers.presence.PresenceCollectionManager` method), 161
`monitor_add()` (`whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager` method), 170
`monitor_add()` (`whalesong.managers.status.StatusCollectionManager` method), 175
`monitor_add()` (`whalesong.managers.status_v3.StatusV3CollectionManager` method), 196
`monitor_add()` (`whalesong.managers.sticker_pack.StickerCollectionManager` method), 149
`monitor_add()` (`whalesong.managers.sticker_pack.StickerPackCollectionManager` method), 145
`monitor_change()` (`whalesong.managers.BaseCollectionManager` method), 25
`monitor_change()` (`whalesong.managers.chat.ChatCollectionManager` method), 34
`monitor_change()` (`whalesong.managers.contact.ContactCollectionManager` method), 27
`monitor_change()` (`whalesong.managers.group_metadata.GroupMetadataCollectionManager` method), 134
`monitor_change()` (`whalesong.managers.group_metadata.ParticipantCollectionManager` method), 138
`monitor_change()` (`whalesong.managers.live_location.LiveLocationCollectionManager` method), 179
`monitor_change()` (`whalesong.managers.live_location.ParticipantCollectionManager` method), 183
`monitor_change()` (`whalesong.managers.message.MessageAckCollectionManager` method), 55
`monitor_change()` (`whalesong.managers.message.MessageCollectionManager` method), 49
`monitor_change()` (`whalesong.managers.message.MessageInfoManager` method), 52
`monitor_change()` (`whalesong.managers.message.MessageManager` method), 50
`monitor_change()` (`whalesong.managers.mute.MuteCollectionManager` method), 188


```
monitor_field() (whalesong.managers.mute.MuteManager method), 181  
    monitor_model() (whalesong.managers.live_location.ParticipantManager method), 190  
monitor_field() (whalesong.managers.presence.ChatStateCollectionManager method), 165  
    monitor_model() (whalesong.managers.message.MessageAckManager method), 167  
monitor_field() (whalesong.managers.presence.ChatStateManager method), 56  
    monitor_model() (whalesong.managers.message.MessageInfoManager method), 162  
monitor_field() (whalesong.managers.presence.PresenceCollectionManager method), 162  
    monitor_model() (whalesong.managers.message.MessageManager method), 50  
monitor_field() (whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager method), 171  
    monitor_model() (whalesong.managers.presence.ChatStateManager method), 172  
monitor_field() (whalesong.managers.profile_pic_thumb.ProfilePictureManager method), 172  
    monitor_model() (whalesong.managers.presence.PresenceManager method), 163  
monitor_field() (whalesong.managers.status.StatusCollectionManager method), 173  
    monitor_model() (whalesong.managers.profile_pic_thumb.ProfilePictureManager method), 177  
monitor_field() (whalesong.managers.status.StatusManager method), 177  
    monitor_model() (whalesong.managers.status.StatusManager method), 199  
monitor_field() (whalesong.managers.status_v3.StatusV3CollectionManager method), 197  
    monitor_model() (whalesong.managers.status_v3.StatusV3Manager method), 198  
monitor_field() (whalesong.managers.sticker_pack.StickerCollectionManager method), 149  
    monitor_model() (whalesong.managers.sticker_pack.StickerPackManager method), 147  
monitor_field() (whalesong.managers.sticker_pack.StickerManager method), 151  
    monitor_model() (whalesong.managers.stream.StreamManager method), 145  
monitor_field() (whalesong.managers.sticker_pack.StickerPackCollectionManager method), 147  
    monitor_model() (whalesong.managers.wap.WapManager method), 143  
monitor_field() (whalesong.managers.sticker_pack.StickerPackManager method), 147  
    monitor_new() (whalesong.managers.message.MessageCollectionManager method), 49  
monitor_field() (whalesong.managers.stream.StreamManager method), 157  
    monitor_remove() (whalesong.managers.BaseCollectionManager method), 25  
monitor_field() (whalesong.managers.wap.WapManager method), 143  
    monitor_remove() (whalesong.managers.chat.ChatCollectionManager method), 34  
monitor_item_storage() (whalesong.managers.storage.StorageManager method), 200  
    monitor_remove() (whalesong.managers.contact.ContactCollectionManager method), 27  
monitor_model() (whalesong.managers.BaseModelManager method), 24  
    monitor_remove() (whalesong.managers.group_metadata.GroupMetadataManager method), 134  
monitor_model() (whalesong.managers.chat.ChatManager method), 41  
    monitor_remove() (whalesong.managers.group_metadata.ParticipantManager method), 138  
monitor_model() (whalesong.managers.chat.MsgLoadStateManager method), 44  
    monitor_remove() (whalesong.managers.live_location.LiveLocationCollectionManager method), 179  
monitor_model() (whalesong.managers.conn.ConnManager method), 153  
    monitor_remove() (whalesong.managers.live_location.ParticipantCollectionManager method), 183  
monitor_model() (whalesong.managers.contact.ContactManager method), 29  
    monitor_remove() (whalesong.managers.message.MessageAckCollectionManager method), 55  
monitor_model() (whalesong.managers.display_info.DisplayInfoManager method), 192  
    monitor_remove() (whalesong.managers.message.MessageCollectionManager method), 49  
monitor_model() (whalesong.managers.group_metadata.GroupMetadataManager method), 136  
    monitor_remove() (whalesong.managers.mute.MuteCollectionManager method), 188  
monitor_model() (whalesong.managers.group_metadata.ParticipantManager method), 140  
    monitor_remove() (whalesong.managers.presence.ChatStateCollectionManager method), 165  
monitor_model() (whalesong.managers.live_location.LiveLocationManager method), 165
```

- [method\), 162](#)
 - [monitor_remove\(\) \(whalesong.managers.profile_pic_thumbnail.ProfilePicThumbnailManager message.AudioMessage method\), 171](#)
 - [monitor_remove\(\) \(whalesong.managers.status.StatusCollectionManager message.BaseMessage method\), 175](#)
 - [monitor_remove\(\) \(whalesong.managers.status_v3.StatusV3CollectionManager message.DocumentMessage method\), 197](#)
 - [monitor_remove\(\) \(whalesong.managers.sticker_pack.StickerCollectionManager message.GroupNotificationMessage method\), 149](#)
 - [monitor_remove\(\) \(whalesong.managers.sticker_pack.StickerPackCollectionManager message.ImageMessage method\), 145](#)
 - [monitor_storage\(\) \(whalesong.managers.storage.StorageManager method\), 200](#)
 - [MonitorResult \(class in whalesong.results\), 20](#)
 - [msg \(whalesong.managers.live_location.Participant property\), 185](#)
 - [msg_load_state \(whalesong.managers.chat.ChatManager attribute\), 36](#)
 - [MsgLoadState \(whalesong.managers.chat Model\), 46](#)
 - [MsgLoadStateManager \(class in whalesong.managers.chat\), 43](#)
 - [msgs \(whalesong.managers.chat.ChatManager attribute\), 35](#)
 - [MULTI_VCARD \(whalesong.managers.message.MessageTypes attribute\), 57](#)
 - [multicast \(whalesong.managers.message.GroupNotificationMessage property\), 125](#)
 - [MultiVCardMessage \(whalesong.managers.message Model\), 105](#)
 - [mute \(whalesong.managers.chat.Chat property\), 44](#)
 - [Mute \(whalesong.managers.mute Model\), 190](#)
 - [mute\(\) \(whalesong.managers.mute.MuteManager method\), 189](#)
 - [MuteCollectionManager \(class in whalesong.managers.mute\), 186](#)
 - [muteExpiration \(whalesong.managers.chat.Chat property\), 46](#)
 - [MuteManager \(class in whalesong.managers.mute\), 188](#)
 - [mutes \(whalesong.Wholesong attribute\), 17](#)
- ## N
- [name \(whalesong.managers.chat.Chat property\), 44](#)
 - [name \(whalesong.managers.contact.Contact property\), 30](#)
 - [name \(whalesong.managers.sticker_pack.StickerPack property\), 152](#)
 - [noEarlierMsgs \(whalesong.managers.chat.MsgLoadState property\), 47](#)
 - [NORMAL \(whalesong.managers.display_info.DisplayInfo.StreamInfo attribute\), 193](#)
 - [NOTIFICATION_TEMPLATE \(whalesong.managers.message.MessageTypes attribute\), 57](#)
- ## O
- [OBSCURE \(whalesong.managers.display_info.DisplayInfo.DisplayState attribute\), 194](#)
 - [obscurity \(whalesong.managers.display_info.DisplayInfo property\), 195](#)
 - [OFFLINE \(whalesong.managers.display_info.DisplayInfo.StreamInfo attribute\), 193](#)
 - [OFFLINE \(whalesong.managers.display_info.DisplayInfo.StreamMode attribute\), 193](#)
 - [OPENING \(whalesong.managers.display_info.DisplayInfo.StreamInfo attribute\), 193](#)
 - [OPENING \(whalesong.managers.stream.Stream.State attribute\), 158](#)
 - [osBuildNumber \(whalesong.managers.conn.PhoneDescription property\), 154](#)
 - [osVersion \(whalesong.managers.conn.PhoneDescription property\), 154](#)
 - [owner \(whalesong.managers.group_metadata.GroupMetadata property\), 141](#)
- ## P
- [pageCount \(whalesong.managers.message.DocumentMessage property\), 100](#)
 - [PAIRING \(whalesong.managers.display_info.DisplayInfo.StreamInfo attribute\), 193](#)

PAIRING (*whalesong.managers.stream.Stream.State* attribute), 158

Participant (*whalesong.managers.group_metadata.Model*), 140

Participant (*whalesong.managers.live_location.Model*), 185

ParticipantCollectionManager (class in *whalesong.managers.group_metadata*), 136

ParticipantCollectionManager (class in *whalesong.managers.live_location*), 181

ParticipantManager (class in *whalesong.managers.group_metadata*), 139

ParticipantManager (class in *whalesong.managers.live_location*), 183

participants (*whalesong.managers.chat.ChatManager.LiveLocation* attribute), 40

participants (*whalesong.managers.group_metadata.GroupMetadata* attribute), 141

participants (*whalesong.managers.group_metadata.GroupMetadataManager* attribute), 134

participants (*whalesong.managers.live_location.LiveLocation* attribute), 186

participants (*whalesong.managers.live_location.LiveLocationManager* attribute), 180

PAYMENT (*whalesong.managers.message.MessageTypes* attribute), 57

PaymentMessage (*whalesong.managers.message.Model*), 116

PENDING (*whalesong.managers.message.Ack* attribute), 57

phone (*whalesong.managers.conn.Conn* property), 156

phoneAuthed (*whalesong.managers.display_info.DisplayInfo* property), 194

PhoneDescription (*whalesong.managers.conn.Conn* attribute), 154

pin (*whalesong.managers.chat.Chat* property), 44

pin() (*whalesong.managers.chat.ChatManager* method), 42

plaintextDisabled (*whalesong.managers.contact.Contact* property), 32

platform (*whalesong.managers.conn.Conn* property), 156

PLAYED (*whalesong.managers.message.Ack* attribute), 57

played (*whalesong.managers.message.MessageInfo* property), 132

played (*whalesong.managers.message.MessageInfoManager* attribute), 52

playedRemaining (*whalesong.managers.message.MessageInfo* property), 131

plugged (*whalesong.managers.conn.Conn* property), 155

poke() (*whalesong.managers.stream.StreamManager* method), 158

poll() (*whalesong.driver_firefox.WholesongDriver* method), 19

Presence (*whalesong.managers.presence.Model*), 168

PresenceCollectionManager (class in *whalesong.managers.presence*), 160

PresenceManager (class in *whalesong.managers.presence*), 162

process_result() (*whalesong.driver.BaseWholesongDriver* method), 18

process_result_sync() (*whalesong.driver.BaseWholesongDriver* method), 18

profile_pic_thumb (*whalesong.managers.contact.ContactManager* attribute), 28

profile_pic_thumb_obj (*whalesong.managers.contact.Contact* property), 28

ProfilePicture (*whalesong.managers.profile_pic_thumb.Model*), 173

ProfilePictureCollectionManager (class in *whalesong.managers.profile_pic_thumb*), 169

ProfilePictureManager (class in *whalesong.managers.profile_pic_thumb*), 171

promote_participants() (*whalesong.managers.group_metadata.ParticipantCollectionManager* method), 138

PROTOCOL (*whalesong.managers.message.MessageTypes* attribute), 57

InfoToVersion (*whalesong.managers.conn.Conn* property), 155

PROXYBLOCK (*whalesong.managers.display_info.DisplayInfo.StreamMode* attribute), 193

PROXYBLOCK (*whalesong.managers.stream.Stream.State* attribute), 158

PTT (*whalesong.managers.message.MessageTypes* attribute), 57

PTTMessage (*whalesong.managers.message.Model*), 88

pushname (*whalesong.managers.conn.Conn* property), 156

pushname (*whalesong.managers.contact.Contact* property), 30

Q

QR (*whalesong.managers.display_info.DisplayInfo.StreamMode* attribute), 193

qr (*whalesong.Wholesong* method), 17

query_exist() (*whalesong.managers.wap.WapManager* method), 142

QuotedMessageMixin (*whalesong.managers.message.Model*), 62

`quotedMsgObj` (*whalesong.managers.message.AudioMessage* property), 77
 (*property*), 83
`quotedMsgObj` (*whalesong.managers.message.DocumentMessage* property), 83
 (*property*), 95
`quotedMsgObj` (*whalesong.managers.message.ImageMessage* property), 95
 (*property*), 71
`quotedMsgObj` (*whalesong.managers.message.LocationMessage* property), 71
 (*property*), 110
`quotedMsgObj` (*whalesong.managers.message.MultiVCardMessage* property), 111
 (*property*), 105
`quotedMsgObj` (*whalesong.managers.message.PTTMessage* property), 105
 (*property*), 89
`quotedMsgObj` (*whalesong.managers.message.QuotedMessageMix* property), 89
 (*property*), 62
`quotedMsgObj` (*whalesong.managers.message.StickerMessage* property), 63
 (*property*), 125
`quotedMsgObj` (*whalesong.managers.message.TextMessage* property), 126
 (*property*), 65
`quotedMsgObj` (*whalesong.managers.message.VCardMessage* property), 66
 (*property*), 100
`quotedMsgObj` (*whalesong.managers.message.VideoMessage* property), 100
 (*property*), 77
`quotedParticipant`
 (*whalesong.managers.message.AudioMessage* property), 83
`quotedParticipant`
 (*whalesong.managers.message.DocumentMessage* property), 95
`quotedParticipant`
 (*whalesong.managers.message.ImageMessage* property), 71
`quotedParticipant`
 (*whalesong.managers.message.LocationMessage* property), 111
`quotedParticipant`
 (*whalesong.managers.message.MultiVCardMessage* property), 105
`quotedParticipant`
 (*whalesong.managers.message.PTTMessage* property), 89
`quotedParticipant`
 (*whalesong.managers.message.QuotedMessageMix* property), 62
`quotedParticipant`
 (*whalesong.managers.message.StickerMessage* property), 126
`quotedParticipant`
 (*whalesong.managers.message.TextMessage* property), 65
`quotedParticipant`
 (*whalesong.managers.message.VCardMessage* property), 100
`quotedParticipant`
 (*whalesong.managers.message.VideoMessage* property), 77
`quotedRemoteJid` (*whalesong.managers.message.AudioMessage* property), 83
`quotedRemoteJid` (*whalesong.managers.message.DocumentMessage* property), 95
`quotedRemoteJid` (*whalesong.managers.message.ImageMessage* property), 71
`quotedRemoteJid` (*whalesong.managers.message.LocationMessage* property), 110
`quotedRemoteJid` (*whalesong.managers.message.MultiVCardMessage* property), 105
`quotedRemoteJid` (*whalesong.managers.message.PTTMessage* property), 89
`quotedRemoteJid` (*whalesong.managers.message.QuotedMessageMix* property), 62
`quotedRemoteJid` (*whalesong.managers.message.StickerMessage* property), 126
`quotedRemoteJid` (*whalesong.managers.message.TextMessage* property), 65
`quotedRemoteJid` (*whalesong.managers.message.VCardMessage* property), 100
`quotedRemoteJid` (*whalesong.managers.message.VideoMessage* property), 77
`quotedStanzaID` (*whalesong.managers.message.AudioMessage* property), 83
`quotedStanzaID` (*whalesong.managers.message.DocumentMessage* property), 95
`quotedStanzaID` (*whalesong.managers.message.ImageMessage* property), 71
`quotedStanzaID` (*whalesong.managers.message.LocationMessage* property), 110
`quotedStanzaID` (*whalesong.managers.message.MultiVCardMessage* property), 105
`quotedStanzaID` (*whalesong.managers.message.PTTMessage* property), 89
`quotedStanzaID` (*whalesong.managers.message.QuotedMessageMix* property), 62
`quotedStanzaID` (*whalesong.managers.message.StickerMessage* property), 126
`quotedStanzaID` (*whalesong.managers.message.TextMessage* property), 65
`quotedStanzaID` (*whalesong.managers.message.VCardMessage* property), 100
`quotedStanzaID` (*whalesong.managers.message.VideoMessage* property), 77

R

`raw` (*whalesong.managers.profile_pic_thumb.ProfilePicture* property), 173
`READ` (*whalesong.managers.message.Ack* attribute), 57
`read` (*whalesong.managers.message.MessageInfo* property), 132
`read` (*whalesong.managers.message.MessageInfoManager* attribute), 51

readKeys (whalesong.managers.status_v3.StatusV3 property), 199
 readRemaining (whalesong.managers.message.MessageAckCollectionManager property), 132
 recipients (whalesong.managers.message.GroupNotificationMessageCollectionManager property), 125
 RECORDING (whalesong.managers.presence.ChatState.Type attribute), 167
 ref (whalesong.managers.conn.Conn property), 155
 refresh () (whalesong.driver.BaseWhealsongDriver method), 18
 refresh () (whalesong.driver_chromium.WhealsongDriver method), 19
 refresh () (whalesong.driver_firefox.WhealsongDriver method), 19
 refTTL (whalesong.managers.conn.Conn property), 155
 remove_item_by_id () (whalesong.managers.BaseCollectionManager method), 25
 remove_item_by_id () (whalesong.managers.chat.ChatCollectionManager method), 34
 remove_item_by_id () (whalesong.managers.contact.ContactCollectionManager method), 28
 remove_item_by_id () (whalesong.managers.group_metadata.GroupMetadataCollectionManager method), 134
 remove_item_by_id () (whalesong.managers.group_metadata.ParticipantCollectionManager method), 138
 remove_item_by_id () (whalesong.managers.live_location.LiveLocationCollectionManager method), 179
 remove_item_by_id () (whalesong.managers.live_location.ParticipantCollectionManager method), 183
 remove_item_by_id () (whalesong.managers.message.MessageAckCollectionManager method), 55
 remove_item_by_id () (whalesong.managers.message.MessageCollectionManager method), 49
 remove_item_by_id () (whalesong.managers.mute.MuteCollectionManager method), 188
 remove_item_by_id () (whalesong.managers.presence.ChatStateCollectionManager method), 165
 remove_item_by_id () (whalesong.managers.presence.PresenceCollectionManager method), 162
 remove_item_by_id () (whalesong.managers.profile_pic_thumb.ProfilePictureCollectionManager method), 171
 remove_item_by_id () (whalesong.managers.status.StatusCollectionManager method), 175
 remove_item_by_id () (whalesong.managers.status_v3.StatusV3CollectionManager method), 197
 remove_item_by_id () (whalesong.managers.sticker_pack.StickerCollectionManager method), 149
 remove_item_by_id () (whalesong.managers.sticker_pack.StickerPackCollectionManager method), 145
 remove_participants () (whalesong.managers.group_metadata.ParticipantCollectionManager method), 136
 remove_result () (whalesong.results.ResultManager method), 20
 remove_submanager () (whalesong.managers.BaseManager method), 23
 remove_submanager () (whalesong.managers.chat.ChatCollectionManager method), 35
 remove_submanager () (whalesong.managers.chat.ChatManager method), 44
 remove_submanager () (whalesong.managers.conn.ConnManager method), 153
 remove_submanager () (whalesong.managers.contact.ContactCollectionManager method), 28
 remove_submanager () (whalesong.managers.contact.ContactManager method), 29
 remove_submanager () (whalesong.managers.display_info.DisplayInfoManager method), 192
 remove_submanager () (whalesong.managers.group_metadata.GroupMetadataCollectionManager method), 134
 remove_submanager () (whalesong.managers.group_metadata.GroupMetadataManager method), 136
 remove_submanager () (whalesong.managers.group_metadata.ParticipantCollectionManager method), 138
 remove_submanager () (whalesong.managers.group_metadata.ParticipantManager method), 138

property), 194
 RESUMING (wattlesong.managers.display_info.DisplayInfo.StreamInfo attribute), 193
 RESUMING (wattlesong.managers.stream.Stream.Stream attribute), 159
 resync_contacts() (wattlesong.managers.contact.ContactCollectionManager method), 28
 resync_messages() (wattlesong.managers.chat.ChatCollectionManager method), 35
 retryTimestamp (wattlesong.managers.stream.Stream property), 159
 revoke() (wattlesong.managers.message.MessageManager method), 51
 revoke_group_invite() (wattlesong.managers.group_metadata.GroupMetadataManager method), 136
 REVOKED (wattlesong.managers.message.MessageTypes attribute), 57
 rtl (wattlesong.managers.message.AudioMessage property), 88
 rtl (wattlesong.managers.message.BaseMessage property), 59
 rtl (wattlesong.managers.message.DocumentMessage property), 99
 rtl (wattlesong.managers.message.GroupNotificationMessage property), 124
 rtl (wattlesong.managers.message.ImageMessage property), 76
 rtl (wattlesong.managers.message.LocationMessage property), 115
 rtl (wattlesong.managers.message.MultiVCardMessage property), 109
 rtl (wattlesong.managers.message.PaymentMessage property), 120
 rtl (wattlesong.managers.message.PTTMessage property), 94
 rtl (wattlesong.managers.message.StickerMessage property), 130
 rtl (wattlesong.managers.message.TextMessage property), 70
 rtl (wattlesong.managers.message.VCardMessage property), 104
 rtl (wattlesong.managers.message.VideoMessage property), 82
 run_scriptlet() (wattlesong.driver.BaseWattlesongDriver method), 18
 screenshot_element() (wattlesong.driver.BaseWattlesongDriver method), 18
 sectionHeader (wattlesong.managers.contact.Contact property), 32
 self (wattlesong.managers.message.AudioMessage property), 85
 self (wattlesong.managers.message.BaseMessage property), 58
 self (wattlesong.managers.message.DocumentMessage property), 96
 self (wattlesong.managers.message.GroupNotificationMessage property), 121
 self (wattlesong.managers.message.ImageMessage property), 73
 self (wattlesong.managers.message.LocationMessage property), 112
 self (wattlesong.managers.message.MultiVCardMessage property), 106
 self (wattlesong.managers.message.PaymentMessage property), 117
 self (wattlesong.managers.message.PTTMessage property), 91
 self (wattlesong.managers.message.StickerMessage property), 127
 self (wattlesong.managers.message.TextMessage property), 67
 self (wattlesong.managers.message.VCardMessage property), 101
 self (wattlesong.managers.message.VideoMessage property), 79
 send_contact() (wattlesong.managers.chat.ChatManager method), 36
 send_contact_phone() (wattlesong.managers.chat.ChatManager method), 36
 send_media() (wattlesong.managers.chat.ChatManager method), 37
 send_not_spam() (wattlesong.managers.chat.ChatManager method), 42
 send_read_status() (wattlesong.managers.status_v3.StatusV3Manager method), 198
 send_seen() (wattlesong.managers.chat.ChatManager method), 37
 send_spam_report() (wattlesong.managers.chat.ChatManager method), 42
 send_text() (wattlesong.managers.chat.ChatManager method), 36
 send_to_chat() (wattlesong.managers.sticker_pack.StickerManager method), 150
 sender (wattlesong.managers.message.AudioMessage property), 85

S

sender (whalesong.managers.message.BaseMessage SERVER (whalesong.managers.message.Ack attribute),
 property), 58 57
 sender (whalesong.managers.message.DocumentMessage serverToken (whalesong.managers.conn.Conn prop-
 property), 96 erty), 155
 sender (whalesong.managers.message.GroupNotificationMessage available_permanent ()
 property), 121 (whalesong.managers.display_info.DisplayInfoManager
 method), 192
 sender (whalesong.managers.message.ImageMessage set_error_result ()
 property), 73 (whalesong.results.BaseResultMixin method),
 19
 sender (whalesong.managers.message.LocationMessage set_error_result ()
 property), 111 (whalesong.results.ResultManager method), 21
 sender (whalesong.managers.message.MultiVCardMessage set_field_value ()
 property), 106 (whalesong.managers.chat.ChatManager.LiveLocation
 method), 40
 sender (whalesong.managers.message.PTTMessage set_final_result ()
 property), 91 (whalesong.results.BaseResultMixin method),
 19
 sender (whalesong.managers.message.StickerMessage set_final_result ()
 property), 127 (whalesong.results.ResultManager method), 21
 sender (whalesong.managers.message.TextMessage set_global_notifications ()
 property), 67 (whalesong.managers.mute.MuteCollectionManager
 method), 186
 sender (whalesong.managers.message.VCardMessage set_global_previews ()
 property), 101 (whalesong.managers.mute.MuteCollectionManager
 method), 188
 senderObj (whalesong.managers.message.AudioMessage set_global_sounds ()
 property), 85 (whalesong.managers.mute.MuteCollectionManager
 method), 188
 senderObj (whalesong.managers.message.BaseMessage set_group_description ()
 property), 59 (whalesong.managers.mute.MuteCollectionManager
 method), 188
 senderObj (whalesong.managers.message.DocumentMessage set_group_description ()
 property), 96 (whalesong.managers.chat.ChatManager
 method), 42
 senderObj (whalesong.managers.message.GroupNotificationMessage set_item () (whalesong.managers.storage.StorageManager
 property), 121 method), 200
 senderObj (whalesong.managers.message.ImageMessage get_my_status () (whalesong.managers.status.StatusCollectionManager
 property), 73 method), 175
 senderObj (whalesong.managers.message.LocationMessage set_parent () (whalesong.managers.chat.ChatManager.LiveLocation
 property), 111 method), 40
 senderObj (whalesong.managers.message.MultiVCardMessage set_partial_result ()
 property), 106 (whalesong.results.BasePartialResult method),
 20
 senderObj (whalesong.managers.message.PaymentMessage set_partial_result ()
 property), 117 (whalesong.results.ResultManager method), 21
 senderObj (whalesong.managers.message.PTTMessage set_picture () (whalesong.managers.profile_pic_thumb.ProfilePicture
 property), 91 method), 172
 senderObj (whalesong.managers.message.StickerMessage set_read_only () (whalesong.managers.chat.ChatManager.LiveLocati
 property), 127 method), 40
 senderObj (whalesong.managers.message.TextMessage set_storage () (whalesong.managers.storage.StorageManager
 property), 67 method), 200
 senderObj (whalesong.managers.message.VCardMessage set_subject () (whalesong.managers.chat.ChatManager
 property), 101 method), 38
 senderObj (whalesong.managers.message.VideoMessage shareDuration (whalesong.managers.message.LocationMessage
 property), 79 property), 115
 sequence (whalesong.managers.live_location.Participant shortName (whalesong.managers.contact.Contact
 property), 185
 sequence (whalesong.managers.message.LocationMessage

property), 30
 SHOW (wattlesong.managers.display_info.DisplayInfo.DisplayState attribute), 194
 showForwarded (wattlesong.managers.message.AudioMessage property), 87
 showForwarded (wattlesong.managers.message.BaseMessage property), 61
 showForwarded (wattlesong.managers.message.DocumentMessage property), 99
 showForwarded (wattlesong.managers.message.GroupNotificationMessage property), 124
 showForwarded (wattlesong.managers.message.ImageMessage property), 75
 showForwarded (wattlesong.managers.message.LocationMessage property), 114
 showForwarded (wattlesong.managers.message.MultiVCardMessage property), 109
 showForwarded (wattlesong.managers.message.PaymentMessage property), 120
 showForwarded (wattlesong.managers.message.PTTMessage property), 93
 showForwarded (wattlesong.managers.message.StickerMessage property), 130
 showForwarded (wattlesong.managers.message.TextMessage property), 70
 showForwarded (wattlesong.managers.message.VCardMessage property), 104
 showForwarded (wattlesong.managers.message.VideoMessage property), 82
 size (wattlesong.managers.message.AudioMessage property), 84
 size (wattlesong.managers.message.DocumentMessage property), 95
 size (wattlesong.managers.message.ImageMessage property), 72
 size (wattlesong.managers.message.MediaMixin property), 64
 size (wattlesong.managers.message.PTTMessage property), 89
 size (wattlesong.managers.message.StickerMessage property), 126
 size (wattlesong.managers.message.VideoMessage property), 77
 size (wattlesong.managers.sticker_pack.Sticker property), 152
 SMB_TOS_BLOCK (wattlesong.managers.display_info.DisplayInfo.StreamMode attribute), 193
 SMB_TOS_BLOCK (wattlesong.managers.stream.Stream.State attribute), 158
 smbTos (wattlesong.managers.conn.Conn property), 155
 speed (wattlesong.managers.live_location.Participant property), 185
 speed (wattlesong.managers.message.LocationMessage property), 110
 (wattlesong.managers.message.AudioMessage property), 85
 (wattlesong.managers.message.BaseMessage property), 58
 (wattlesong.managers.message.DocumentMessage property), 97
 (wattlesong.managers.message.GroupNotificationMessage property), 122
 (wattlesong.managers.message.ImageMessage property), 73
 (wattlesong.managers.message.LocationMessage property), 112
 (wattlesong.managers.message.MultiVCardMessage property), 107
 (wattlesong.managers.message.PaymentMessage property), 117
 (wattlesong.managers.message.PTTMessage property), 91
 (wattlesong.managers.message.StickerMessage property), 128
 (wattlesong.managers.message.TextMessage property), 67
 (wattlesong.managers.message.VCardMessage property), 102
 (wattlesong.managers.message.VideoMessage property), 79
 () (wattlesong.managers.message.MessageManager method), 51
 star_messages () (wattlesong.managers.chat.ChatManager method), 42
 start () (wattlesong.Wattlesong method), 17
 start_driver () (wattlesong.driver.BaseWattlesongDriver method), 18
 start_monitor () (wattlesong.results.MonitorResult method), 20
 state (wattlesong.managers.stream.Stream property), 159
 status (wattlesong.managers.contact.Contact property), 30
 Status (wattlesong.managers.status Model), 177
 status (wattlesong.managers.status.Status property), 177
 status (wattlesong.Wattlesong attribute), 17
 status_v3 (wattlesong.Wattlesong attribute), 17
 StatusManager (class in wattlesong.managers.status), 173
 StatusManager (class in wattlesong.managers.status), 176
 statusMute (wattlesong.managers.contact.Contact property), 32
 StatusV3 (wattlesong.managers.status_v3 Model), 199
 StatusV3CollectionManager (class in wattlesong.managers.status_v3), 195

StatusV3Manager (class in storage (whalesong.Wholesong attribute), 16
 whalesong.managers.status_v3), 197
 statusV3TextBg (whalesong.managers.message.AudioMessage whalesong.managers.storage), 200
 property), 86
 statusV3TextBg (whalesong.managers.message.BaseMessage whalesong.managers.stream.Model), 158
 property), 60
 statusV3TextBg (whalesong.managers.message.DocumentMessage whalesong.Wholesong attribute), 16
 property), 97
 statusV3TextBg (whalesong.managers.message.GroupNotificationMessage
 property), 122
 statusV3TextBg (whalesong.managers.message.ImageMessage whalesong.managers.stream), 158
 property), 74
 statusV3TextBg (whalesong.managers.message.LocationMessage whalesong.managers.stream), 158
 property), 84
 statusV3TextBg (whalesong.managers.message.MultiVCardMessage whalesong.managers.stream), 158
 property), 65
 statusV3TextBg (whalesong.managers.message.PaymentMessage whalesong.managers.stream), 158
 property), 90
 statusV3TextBg (whalesong.managers.message.PTTMessage whalesong.managers.stream), 158
 property), 78
 statusV3TextBg (whalesong.managers.message.StickerMessage whalesong.managers.stream), 156
 property), 128
 statusV3TextBg (whalesong.managers.message.TextMessage whalesong.managers.stream), 156
 property), 68
 statusV3TextBg (whalesong.managers.message.VCardMessage whalesong.managers.stream), 156
 property), 102
 statusV3TextBg (whalesong.managers.message.VideoMessage whalesong.managers.stream), 156
 property), 80
 STICKER (whalesong.managers.message.MessageTypes attribute), 57
 Sticker (whalesong.managers.sticker_pack Model), 151
 sticker_packs (whalesong.Wholesong attribute), 16
 StickerCollectionManager (class in whalesong.managers.sticker_pack), 147
 StickerManager (class in whalesong.managers.sticker_pack), 149
 StickerMessage (whalesong.managers.message Model), 125
 StickerPack (whalesong.managers.sticker_pack Model), 152
 StickerPackCollectionManager (class in whalesong.managers.sticker_pack), 143
 StickerPackManager (class in whalesong.managers.sticker_pack), 146
 stickers (whalesong.managers.sticker_pack.StickerPackManager whalesong.managers.sticker_pack attribute), 146
 stop () (whalesong.Wholesong method), 17
 stop_monitor () (whalesong.Wholesong method), 17
 stop_my_live_location () (whalesong.managers.live_location.LiveLocationManager method), 181
 StopIterator, 201
 StopMonitor, 201
 StorageManager (class in storage (whalesong.Wholesong attribute), 16
 StorageManager (class in storage (whalesong.Wholesong attribute), 16
 Stream (whalesong.managers.stream.Model), 158
 StreamManager (class in storage (whalesong.Wholesong attribute), 16
 Stream.State (class in whalesong.managers.stream),
 Stream.Stream (class in storage (whalesong.Wholesong attribute), 16
 streamable (whalesong.managers.message.AudioMessage
 streamable (whalesong.managers.message.MediaStreamMixin
 streamable (whalesong.managers.message.PTTMessage
 streamable (whalesong.managers.message.VideoMessage
 subscribe () (whalesong.managers.live_location.LiveLocationManager
 subscribe () (whalesong.managers.presence.PresenceManager
 subtype (whalesong.managers.message.AudioMessage
 subtype (whalesong.managers.message.BaseMessage
 subtype (whalesong.managers.message.DocumentMessage
 subtype (whalesong.managers.message.GroupNotificationMessage
 subtype (whalesong.managers.message.ImageMessage
 subtype (whalesong.managers.message.LocationMessage
 subtype (whalesong.managers.message.MultiVCardMessage
 subtype (whalesong.managers.message.PaymentMessage
 subtype (whalesong.managers.message.PTTMessage
 subtype (whalesong.managers.message.StickerMessage
 subtype (whalesong.managers.message.TextMessage
 subtype (whalesong.managers.message.VCardMessage
 subtype (whalesong.managers.message.VideoMessage
 sync () (whalesong.managers.status_v3.StatusV3CollectionManager
 method), 197
 SYNCING (whalesong.managers.display_info.DisplayInfo.StreamInfo

- attribute), 193
- SYNCING (whalesong.managers.display_info.DisplayInfo.StreamMode property), 78
- attribute), 193
- SYNCING (whalesong.managers.stream.Stream.Stream attribute), 159
- syncTag (whalesong.managers.stream.Stream property), 159
- ## T
- tag (whalesong.managers.profile_pic_thumb.ProfilePicture property), 173
- takeover() (whalesong.managers.stream.StreamManager method), 158
- text (whalesong.managers.message.LocationMessage property), 116
- textColor (whalesong.managers.message.TextMessage property), 71
- TextMessage (whalesong.managers.message Model), 65
- thumbnail (whalesong.managers.message.LinkContentMixin property), 63
- thumbnail (whalesong.managers.message.TextMessage property), 66
- TIMEOUT (whalesong.managers.display_info.DisplayInfo.StreamInfo property), 117
- attribute), 193
- TIMEOUT (whalesong.managers.stream.Stream.State attribute), 158
- timestamp (whalesong.managers.message.AudioMessage property), 84
- timestamp (whalesong.managers.message.BaseMessage property), 58
- timestamp (whalesong.managers.message.DocumentMessage property), 96
- timestamp (whalesong.managers.message.GroupNotificationMessage property), 121
- timestamp (whalesong.managers.message.ImageMessage property), 72
- timestamp (whalesong.managers.message.LocationMessage property), 111
- timestamp (whalesong.managers.message.MessageAck property), 131
- timestamp (whalesong.managers.message.MultiVCardMessage property), 106
- timestamp (whalesong.managers.message.PaymentMessage property), 116
- timestamp (whalesong.managers.message.PTTMessage property), 90
- timestamp (whalesong.managers.message.StickerMessage property), 127
- timestamp (whalesong.managers.message.TextMessage property), 67
- timestamp (whalesong.managers.message.VCardMessage property), 101
- timestamp (whalesong.managers.message.VideoMessage property), 85
- timestamp (whalesong.managers.presence.ChatState property), 167
- title (whalesong.managers.message.LinkContentMixin property), 63
- title (whalesong.managers.message.TextMessage property), 66
- to (whalesong.managers.message.AudioMessage property), 85
- to (whalesong.managers.message.BaseMessage property), 58
- to (whalesong.managers.message.DocumentMessage property), 96
- to (whalesong.managers.message.GroupNotificationMessage property), 121
- to (whalesong.managers.message.ImageMessage property), 73
- to (whalesong.managers.message.LocationMessage property), 111
- to (whalesong.managers.message.MultiVCardMessage property), 106
- to (whalesong.managers.message.PaymentMessage property), 117
- to (whalesong.managers.message.PTTMessage property), 91
- to (whalesong.managers.message.StickerMessage property), 127
- to (whalesong.managers.message.TextMessage property), 67
- to (whalesong.managers.message.VCardMessage property), 101
- to (whalesong.managers.message.VideoMessage property), 79
- to_vcard() (whalesong.managers.contact.Contact method), 32
- tos (whalesong.managers.conn.Conn property), 156
- TOS_BLOCK (whalesong.managers.display_info.DisplayInfo.StreamMode attribute), 193
- TOS_BLOCK (whalesong.managers.stream.Stream.State attribute), 158
- type (whalesong.managers.contact.Contact property), 30
- type (whalesong.managers.message.AudioMessage property), 83
- type (whalesong.managers.message.BaseMessage property), 58
- type (whalesong.managers.message.DocumentMessage property), 95
- type (whalesong.managers.message.GroupNotificationMessage property), 121
- type (whalesong.managers.message.ImageMessage property), 71
- type (whalesong.managers.message.LocationMessage

- property*), 111
 - type (*whalesong.managers.message.MediaMixin property*), 64
 - type (*whalesong.managers.message.MultiVCardMessage property*), 105
 - type (*whalesong.managers.message.PaymentMessage property*), 116
 - type (*whalesong.managers.message.PTTMessage property*), 89
 - type (*whalesong.managers.message.StickerMessage property*), 126
 - type (*whalesong.managers.message.TextMessage property*), 66
 - type (*whalesong.managers.message.VCardMessage property*), 101
 - type (*whalesong.managers.message.VideoMessage property*), 77
 - type (*whalesong.managers.presence.ChatState property*), 168
 - type (*whalesong.managers.sticker_pack.Sticker property*), 151
- U**
- uiActive (*whalesong.managers.display_info.DisplayInfo property*), 194
 - unarchive () (*whalesong.managers.chat.ChatManager method*), 42
 - UNAVAILABLE (*whalesong.managers.presence.ChatState.Type attribute*), 167
 - unblock () (*whalesong.managers.contact.ContactManager method*), 29
 - UNKNOWN (*whalesong.managers.message.MessageTypes attribute*), 57
 - UnknownError, 201
 - UNLAUNCHED (*whalesong.managers.stream.Stream.State attribute*), 158
 - unlock () (*whalesong.managers.chat.ChatManager.LiveLocation method*), 40
 - unmute () (*whalesong.managers.mute.MuteManager method*), 189
 - unobscure () (*whalesong.managers.display_info.DisplayInfoManager method*), 192
 - UNPAIRED (*whalesong.managers.stream.Stream.State attribute*), 158
 - UNPAIRED_IDLE (*whalesong.managers.stream.Stream.State attribute*), 158
 - unpin () (*whalesong.managers.chat.ChatManager method*), 42
 - unreadCount (*whalesong.managers.chat.Chat property*), 45
 - unreadCount (*whalesong.managers.status_v3.StatusV3 property*), 199
 - unset_available_permanent () (*whalesong.managers.display_info.DisplayInfoManager method*), 192
 - unstar () (*whalesong.managers.message.MessageManager method*), 51
 - unstar_messages () (*whalesong.managers.chat.ChatManager method*), 42
 - unsubscribe () (*whalesong.managers.live_location.LiveLocationManager method*), 181
 - update_pushname () (*whalesong.managers.conn.ConnManager method*), 153
 - updateTime (*whalesong.managers.presence.ChatState property*), 168
 - url (*whalesong.managers.sticker_pack.StickerPack property*), 152
 - urlNumber (*whalesong.managers.message.GroupNotificationMessage property*), 125
 - urlText (*whalesong.managers.message.GroupNotificationMessage property*), 125
 - userhash (*whalesong.managers.contact.Contact property*), 30
 - userid (*whalesong.managers.contact.Contact property*), 30
- V**
- VCARD (*whalesong.managers.message.MessageTypes attribute*), 57
 - VCard (*whalesong.managers.message.VCardItem property*), 57
 - VCardItem (*whalesong.managers.message.Model*), 57
 - vcardList (*whalesong.managers.message.MultiVCardMessage property*), 110
 - VCardMessage (*whalesong.managers.message.Model*), 100
 - verifiedLevel (*whalesong.managers.contact.Contact property*), 30
 - verifiedName (*whalesong.managers.contact.Contact property*), 30
 - VIDEO (*whalesong.managers.message.MessageTypes attribute*), 57
 - VinfoManager (*whalesong.managers.message.Model*), 76
- W**
- wait_until_stop () (*whalesong.driver.BaseWhalesongDriver method*), 18
 - wait_until_stop () (*whalesong.Wholesong method*), 17
 - Wap (*whalesong.managers.wap.Model*), 143
 - wap (*whalesong.Wholesong attribute*), 16
 - WapManager (*class in whalesong.managers.wap*), 141
 - Whalesong (*class in whalesong*), 16
 - whalesong.errors (*module*), 201

[whalesong.firefox_profile \(module\)](#), 201
[whalesong.managers \(module\)](#), 22
[whalesong.managers.chat \(module\)](#), 44
[whalesong.managers.conn \(module\)](#), 154
[whalesong.managers.contact \(module\)](#), 30
[whalesong.managers.display_info \(module\)](#), 192
[whalesong.managers.group_metadata \(module\)](#), 140
[whalesong.managers.live_location \(module\)](#), 185
[whalesong.managers.message \(module\)](#), 57
[whalesong.managers.mute \(module\)](#), 190
[whalesong.managers.presence \(module\)](#), 167
[whalesong.managers.profile_pic_thumb \(module\)](#), 173
[whalesong.managers.status \(module\)](#), 177
[whalesong.managers.status_v3 \(module\)](#), 199
[whalesong.managers.sticker_pack \(module\)](#), 151
[whalesong.managers.stream \(module\)](#), 158
[whalesong.managers.wap \(module\)](#), 143
[whalesong.models \(module\)](#), 21
[whalesong.results \(module\)](#), 19
[WhalesongDriver \(class in whalesong.driver_chromium\)](#), 19
[WhalesongDriver \(class in whalesong.driver_firefox\)](#), 18
[WhalesongException](#), 201
[whatsappId \(whalesong.managers.conn.Conn property\)](#), 155
[whatsappVersion \(whalesong.managers.conn.PhoneDescription property\)](#), 154
[width \(whalesong.managers.message.ImageMessage property\)](#), 72
[width \(whalesong.managers.message.MediaFrameMixin property\)](#), 64
[width \(whalesong.managers.message.StickerMessage property\)](#), 127
[width \(whalesong.managers.message.VideoMessage property\)](#), 78