smsc Documentation

Release 0.1.1

Author

May 30, 2017

Contents:

CHAPTER 1

SMSC

SMSC.ru HTTP API Library.

Installation

Install smsc package from PyPI:

```
$ pip install smsc
```

Getting started

Basic usage sample:

```
>>> from smsc.messages import SMSMessage
>>> from smsc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.send(to='79999999999', message=SMSMessage(text='Hello, World!'))
>>> res.count
1
>>> res.cost
1.44
```

Documentation

Documentation is available at Read the Docs.

Links

- SMSC.ru HTTP API

Release History

0.1.1 (2017-05-30)

- Default charset set to UTF-8.

0.1.0 (2017-05-29)

- Birth!

CHAPTER 2

The API Documentation

If you are looking for information on a specific function, class, or method, this part of the documentation is for you.

smsc package

Submodules

smsc.api module

smsc.api module.

This module implements the SMSC.ru HTTP API.

copyright

3. 2017 by Alexey Shevchenko.

license MIT, see LICENSE for more details.

class `smsc.api.`**`SMSC`**(*login: str*, *password: str*, *sender: typing.Union[str, NoneType] = None*) → None
Bases: `object`

Class for interaction with smsc.ru API.

Usage:

```
>>> from smsc.api import SMSC
>>> client = SMSC(login="alexey", password="psw")
>>> client
<SMSC login='alexey' sender='SMSC.ru'>
```

Parameters

- **`login`** (*str*) – Account login name
- **`password`** (*str*) – Password or MD5 hash of password in lower case

get_balance() → smsc.responses.BalanceResponse
Get current account balance.

Usage:

```
>>> from smsc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.get_balance()
>>> res
<BalanceResponse balance=100.01 credit=None currency=RUR>
```

Returns Returns the API answer wrapped in the *BalanceResponse* object

Return type *BalanceResponse*

get_cost(*to: typing.Union[str, typing.List[str]], message: smsc.messages.Message*) → smsc.responses.CostResponse
Retrieve cost of the message.

Usage:

```
>>> from smsc.messages import SMSMessage
>>> from smsc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.get_cost(to='79999999999', message=SMSMessage(text='Hello,␣
↪World!'))
>>> res.count
1
>>> res.cost
1.44
```

Parameters

- **to** (*str|List[str]*) – Phone number or list of phone numbers
- **message** (Message) – Concrete message instance for measure cost

Returns Returns the API answer wrapped in the *CostResponse* object

Return type *CostResponse*

get_status(*to: typing.Union[str, typing.List[str]], msg_id: typing.Union[str, typing.List[str]]*) → typing.List[smsc.responses.StatusResponse]
Get current status of sent message.

Usage:

```
>>> from smsc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.get_status(to='79999999999', msg_id='1')
>>> res[0].status
<Status status=1 name=>
```

Parameters

- **to** (*str|List[str]*) – Phone number or list of phone numbers
- **msg_id** (*str|List[str]*) – Identification of sent message or list of them

Returns Returns the API answer wrapped in the list of *StatusResponse* objects

Return type List[*StatusResponse*]

send(*to: typing.Union[str, typing.List[str]], message: smsc.messages.Message*) → smsc.responses.SendResponse

Send the message.

Usage:

```
>>> from smsc.messages import SMSMessage
>>> from smsc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.send(to='79999999999', message=SMSMessage(text='Hello, World!
↪'))
>>> res.count
1
>>> res.cost
1.44
```

Parameters

- **to** (*str|List[str]*) – Phone number or list of phone numbers
- **message** (Message) – Concrete message instance for sending

Returns Returns the API answer wrapped in the *SendResponse* object

Return type *SendResponse*

smsc.exceptions module

smsc.exceptions module.

This module contains the set of smsc' exceptions.

copyright

3. 2017 by Alexey Shevchenko.

license MIT, see LICENSE for more details.

exception smsc.exceptions.**GetBalanceError**

Bases: *smsc.exceptions.SMSCException*

A Get Balance error occurred.

exception smsc.exceptions.**GetCostError**

Bases: *smsc.exceptions.SMSCException*

A Get Cost error occurred.

exception smsc.exceptions.**GetStatusError**

Bases: *smsc.exceptions.SMSCException*

A Get Status error occurred.

exception smsc.exceptions.**SMSCException**

Bases: Exception

There was an ambiguous exception that occurred while handling your SMSC API request.

exception smsc.exceptions.**SendError**
Bases: *smsc.exceptions.SMSCException*

A Send Message error occurred.

smsc.messages module

smsc.messages module.

This module contains the messages objects that power smsc library.

copyright

3. 2017 by Alexey Shevchenko.

license MIT, see LICENSE for more details.

class smsc.messages.**FlashMessage**(*text: str*, ***kwargs: dict*) → None
Bases: *smsc.messages.Message*

Flash-SMS message type.

Parameters

- **text** (*str*) – Text of the message
- **kwargs** (*dict*) – Dictionary for optional API parameters

Usage:

```
>>> from smsc import FlashMessage
>>> m = FlashMessage(text="Hello, World!")
>>> m
<FlashMessage text=Hello, World! format=flash>
```

class smsc.messages.**Message**(*text: str*, *msg_format: typing.Union[str, NoneType] = None*, ***kwargs: dict*) → None
Bases: object

Basic class for messages of ant type.

Preferred for internal usage.

Parameters

- **text** (*str*) – Text of the message
- **msg_format** (*Optional[str]*) – Message format. If None or empty - default, SMS message
- **kwargs** (*dict*) – Dictionary for optional API parameters

encode() → typing.Dict[str, typing.Any]
Message parameters in dict, prepared for the API.

format
Format of the message. Default is SMS Message.

text
Text of the message.

class smsc.messages.**SMSMessage**(*text: str*, ***kwargs: dict*) → None
Bases: *smsc.messages.Message*

SMS message type.

Parameters

- **text** (*str*) – Text of the message
- **kwargs** (*dict*) – Dictionary for optional API parameters

Usage:

```
>>> from smsc import SMSMessage

>>> m = SMSMessage(text="Hello, World!")
>>> m
<SMSMessage text=Hello, World! format=None>
```

class smsc.messages.**ViberMessage**(*text: str*, ***kwargs: dict*) → None

Bases: *smsc.messages.Message*

Viber messenger message type.

Note: Seems currently not working now.

Parameters

- **text** (*str*) – Text of the message
- **kwargs** (*dict*) – Dictionary for optional API parameters

Usage:

```
>>> from smsc import ViberMessage
>>> m = ViberMessage(text="Hello, World!")
>>> m
<ViberMessage text=Hello, World! format=viber>
```

smsc.responses module

smsc.responses module.

This module contains the responses objects wrapping SMSC.ru API answers.

copyright

3. 2017 by Alexey Shevchenko.

license MIT, see LICENSE for more details.

class smsc.responses.**BalanceResponse**(*obj: typing.Dict[str, typing.Any]*) → None

Bases: *smsc.responses.Response*

Response for get account balance API command.

Parameters **obj** (*dict*) – Dictionary from API JSON response

balance
Actual account balance.

credit
Available credit of account (if applied).

currency
Currency for current account.

class smsc.responses.**CostResponse**(*obj: typing.Dict[str, typing.Any]*) → None
Bases: *smsc.responses.Response*

Response for get cost (send variation) API command.

Parameters **obj** (*dict*) – Dictionary from API JSON response

cost
Cost of message.

count
Count of billed message parts.

class smsc.responses.**Response**(*obj: typing.Dict[str, typing.Any]*) → None
Bases: object

Basic class for response wrappers.

Parameters **obj** (*dict*) – Dictionary from API JSON response

error
Error in response, if present.

class smsc.responses.**SMSCError**
Bases: *smsc.responses.SMSCError*

Named tuple for error description with code.

Parameters

- **code** (*int*) – Code of error
- **error** (*str*) – Description of error

class smsc.responses.**SendResponse**(*obj: typing.Dict[str, typing.Any]*) → None
Bases: *smsc.responses.Response*

Response for send API command.

Parameters **obj** (*dict*) – Dictionary from API JSON response

cost
Cost of sent message.

count
Count of billed message parts.

message_id
Id of sent message.

class smsc.responses.**Status**(*obj: typing.Dict[str, typing.Any]*) → None
Bases: object

Message delivery status with identification.

Parameters **obj** (*dict*) – Dictionary from API JSON response

name
Delivery status description.

status_id
Id of delivery status.

class smsc.responses.**StatusResponse**(*obj: typing.Dict[str, typing.Any]*) → None
Bases: *smsc.responses.Response*

Response for get status API command.

Parameters **obj** (*dict*) – Dictionary from API JSON response

data
Delivery status detailed data.

status
Message delivery status with identification.

Module contents

SMSC.ru API Library.

Smsc is an library to send messages through SMSC.ru HTTP API. Basic usage:

```
>>> from smsc.messages import SMSMessage
>>> from smsc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.send(to='79999999999', message=SMSMessage(text='Hello, World!'))
>>> res.count
1
>>> res.cost
1.44
```

The some other API methods are supported - see *smsc.api*. Full documentation is at Read the Docs.

copyright

3. 2017 by Alexey Shevchenko.

license MIT, see LICENSE for more details.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index