
smc Documentation

Release 0.1.1

Author

May 30, 2017

Contents:

1	SMSC	1
1.1	Installation	1
1.2	Getting started	1
1.3	Documentation	1
1.4	Links	2
1.5	Release History	2
1.5.1	0.1.1 (2017-05-30)	2
1.5.2	0.1.0 (2017-05-29)	2
2	The API Documentation	3
2.1	smc package	3
2.1.1	Submodules	3
2.1.2	smc.api module	3
2.1.3	smc.exceptions module	5
2.1.4	smc.messages module	6
2.1.5	smc.responses module	7
2.1.6	Module contents	9
3	Indices and tables	11
	Python Module Index	13

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!

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() → `smc.responses.BalanceResponse`

Get current account balance.

Usage:

```
>>> from smc.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: smc.messages.Message) → smc.responses.CostResponse

Retrieve cost of the message.

Usage:

```
>>> from smc.messages import SMSMessage
>>> from smc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.get_cost(to='7999999999', 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[smc.responses.StatusResponse]

Get current status of sent message.

Usage:

```
>>> from smc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.get_status(to='7999999999', 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: smc.messages.Message) → smc.responses.SendResponse
Send the message.

Usage:

```
>>> from smc.messages import SMSMessage
>>> from smc.api import SMSC
>>> client = SMSC(login='alexey', password='psw')
>>> res = client.send(to='7999999999', 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*

smc.exceptions module

smc.exceptions module.

This module contains the set of smc' exceptions.

copyright

3. 2017 by Alexey Shevchenko.

license MIT, see LICENSE for more details.

exception smc.exceptions.**GetBalanceError**

Bases: *smc.exceptions.SMSCException*

A Get Balance error occurred.

exception smc.exceptions.**GetCostError**

Bases: *smc.exceptions.SMSCException*

A Get Cost error occurred.

exception smc.exceptions.**GetStatusError**

Bases: *smc.exceptions.SMSCException*

A Get Status error occurred.

exception smc.exceptions.**SMSCException**

Bases: Exception

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

exception `smc.exceptions.SendError`

Bases: `smc.exceptions.SMSCException`

A Send Message error occurred.

smc.messages module

smc.messages module.

This module contains the messages objects that power smc library.

copyright

3. 2017 by Alexey Shevchenko.

license MIT, see LICENSE for more details.

class `smc.messages.FlashMessage` (*text: str, **kwargs: dict*) → None

Bases: `smc.messages.Message`

Flash-SMS message type.

Parameters

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

Usage:

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

class `smc.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 `smc.messages.SMSMessage` (*text: str, **kwargs: dict*) → None

Bases: `smc.messages.Message`

SMS message type.

Parameters

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

Usage:

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

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

Bases: `smc.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 smc import ViberMessage
>>> m = ViberMessage(text="Hello, World!")
>>> m
<ViberMessage text=Hello, World! format=viber>
```

smc.responses module

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

Bases: `smc.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 `smc.responses.CostResponse` (*obj: typing.Dict[str, typing.Any]*) → None
 Bases: `smc.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 `smc.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 `smc.responses.SMSCError`
 Bases: `smc.responses.SMSCError`

Named tuple for error description with code.

Parameters

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

class `smc.responses.SendResponse` (*obj: typing.Dict[str, typing.Any]*) → None
 Bases: `smc.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 `smc.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 `smc.responses.StatusResponse` (*obj: typing.Dict[str, typing.Any]*) → None
 Bases: `smc.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='7999999999', 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`

S

- `smc`, 9
- `smc.api`, 3
- `smc.exceptions`, 5
- `smc.messages`, 6
- `smc.responses`, 7

B

balance (smmc.responses.BalanceResponse attribute), 7
BalanceResponse (class in smmc.responses), 7

C

cost (smmc.responses.CostResponse attribute), 8
cost (smmc.responses.SendResponse attribute), 8
CostResponse (class in smmc.responses), 7
count (smmc.responses.CostResponse attribute), 8
count (smmc.responses.SendResponse attribute), 8
credit (smmc.responses.BalanceResponse attribute), 7
currency (smmc.responses.BalanceResponse attribute), 7

D

data (smmc.responses.StatusResponse attribute), 9

E

encode() (smmc.messages.Message method), 6
error (smmc.responses.Response attribute), 8

F

FlashMessage (class in smmc.messages), 6
format (smmc.messages.Message attribute), 6

G

get_balance() (smmc.api.SMSC method), 4
get_cost() (smmc.api.SMSC method), 4
get_status() (smmc.api.SMSC method), 4
GetBalanceError, 5
GetCostError, 5
GetStatusError, 5

M

Message (class in smmc.messages), 6
message_id (smmc.responses.SendResponse attribute), 8

N

name (smmc.responses.Status attribute), 8

R

Response (class in smmc.responses), 8

S

send() (smmc.api.SMSC method), 5
SendError, 5
SendResponse (class in smmc.responses), 8
SMSC (class in smmc.api), 3
smmc (module), 9
smmc.api (module), 3
smmc.exceptions (module), 5
smmc.messages (module), 6
smmc.responses (module), 7
SMSCErrors (class in smmc.responses), 8
SMSCErrorsException, 5
SMSMessage (class in smmc.messages), 6
Status (class in smmc.responses), 8
status (smmc.responses.StatusResponse attribute), 9
status_id (smmc.responses.Status attribute), 8
StatusResponse (class in smmc.responses), 8

T

text (smmc.messages.Message attribute), 6

V

ViberMessage (class in smmc.messages), 7