PyTS3 Documentation

Release 1.0.11

Benedikt Schmitt

Sep 08, 2018

Contents

Hint: You are viewing the documentation for v1. You should consider to use v2 for new projects as it supports also the TS3 client query interface and SSH. v1 will only receive bug fixes but no more enhancements.

CHAPTER 1

Content

1.1 Installation

ts3 is registered on PyPi, so you are done with:

```
$ pip3 install ts3
```

You can update the library then with:

```
$ pip3 install --upgrade ts3
```

1.2 API

1.2.1 `common`

This module contains common functions, classes and exceptions used in almost all modules of the ts3 package.

exception `ts3.common.`**`TS3Error`**

Bases: `Exception`

This is the base class for all exceptions in this package.

1.2.2 `escape`

This module contains classes and functions used to build valid query strings and to unescape responses.

class `ts3.escape.`**`TS3Escape`**

Bases: `object`

Provides methods to escape a string properly and to build query strings.

classmethod escape(*raw*)

Escapes the value of *raw*.

```
>>> TS3Escape.escape(None)
''
>>> TS3Escape.escape(2)
'2'
>>> TS3Escape.escape(True)
'1'
>>> TS3Escape.escape('Hello World')
'Hello\sWorld'
```

Parameters **raw** (*None, str, bool, int or RawParameter*) – The value to escape.

Returns The escaped value of *raw*

Return type string

Raises **TypeError** – If *raw* has an unsupported type.

classmethod unescape(*txt*)

Unescapes the str *txt*.

```
>>> TS3Escape.unescape('Hello\sWorld')
'Hello World'
```

Parameters **txt** (*string*) – The string to escape.

Raises **TypeError** – If *txt* is not a string.

classmethod escape_parameters(*parameters*)

Escapes the parameters of a TS3 query and encodes it as a part of a valid ts3 query string.

```
>>> # None
>>> TS3Escape.escape_parameters(None)
''
>>> # key -> str
>>> TS3Escape.escape_parameters({'virtualserver_name': 'foo bar'})
'virtualserver_name=foo\\sbar'
>>> # key -> None
>>> TS3Escape.escape_parameters({"permsid": None})
''
>>> # Of course, you can mix them:
>>> TS3Escape.escape_parameters(
...     {'virtualserver_name': 'foo bar',
...      'permsid': None}
...     )
'virtualserver_name=foo\\sbar'
```

Parameters **parameters** (*dictionary*) – The dictionary with the key value pairs.

classmethod escape_parameterlist(*parameterslist*)

Escapes each parameter dictionary in the parameterslist and encodes the list as a part of a valid ts3 query string.

```
>>> TS3Escape.escape_parameterlist(None)
''
>>> TS3Escape.escape_parameterlist(
...     [{"permid": 17276, "permvalue": 50, "permskip": 1},
...      {"permid": 21415, "permvalue": 20, "permskip": 0}]
...     )
'permid=17276 permvalue=50 permskip=1|permid=21415 permvalue=20 permskip=0'
```

Note, that the order of the parameters might change, when you use the built-in dictionary, that does not care about the order.

Parameters **`parameterslist`** (*`None or a list of dictionaries`*) – A list of parameters.

`classmethod escape_options`(*options*)

Escapes the items in the *options* list and prepends a '-' if necessairy. If *options* is None, the empty string will be returned.

```
>>> TS3Escape.options_to_str(None)
''
>>> TS3Escape.options_to_str([None, 'permsid', '-virtual'])
'-permsid -virtual'
```

Parameters **`options`** (*`None or a list of strings.`*) – A list with the options.

1.2.3 `response`

This module contains the classes to parse a TeamSpeak 3 Server Query response and to structure the data.

`exception` `ts3.response.`**`TS3ParserError`**(*resp*, *exc=None*)

Bases: *`ts3.common.TS3Error`*, `ValueError`

Raised, if the data could not be parsed.

`resp = None`

The TS3Response object, that has thrown the exception.

`exc = None`

The original exception, if the parsing failed due to an exception like UnicodeDecodeError.

`class` `ts3.response.`**`TS3Response`**(*data*)

Bases: `object`

Parses **ONE** response and stores it's data. If you init an instance with the data of more than one response, parsing will fail.

Note, that this class is **lazy**. This means, that the response is only parsed, if you request an attribute, that requires a parsed version of the data.

For convenience, this class supports container emualtion, so these calls are equal:

```
>>> ts3resp.parsed[0]["client_nickname"] == ts3resp[0]["client_nickname"]
True
```

Parameters **`data`** (*`bytes`*) – The byte string received from the server.

`data`

Getter The list of lines from the original received response.

Type list of bytes

`data_bytestr`

Getter The raw response as bytestring.

Type bytes

`parsed`

Getter The parsed response as a list of dictionaries.

Type list of dictionaries [str->str]

Raises *`TS3ParserError`* – If the response could not be parsed.

class `ts3.response.`**`TS3QueryResponse`**(*data*)

Bases: *`ts3.response.TS3Response`*

The same as *`TS3Response`*, but the *error* attribute is public.

`error`

Getter A dictionary, that contains the error id and message.

Type dict

Raises *`TS3ParserError`* – If the response could not be parsed.

class `ts3.response.`**`TS3Event`**(*data*)

Bases: *`ts3.response.TS3Response`*

The same as *`TS3Response`*, but the *event* attribute is public.

`event`

Getter A dictionary with the information about the event.

Type dict

Raises *`TS3ParserError`* – If the response could not be parsed.

1.2.4 commands

class `ts3.commands.`**`TS3Commands`**

Bases: `object`

Provides a convenient interface to build the parameters for `query.TS3BaseConnection.send()`.

Hint: Some query commands accept multiple parameters like *`channeladdperm()`* (*permid*, *permsid*). I did not found a nice way to implement that feature. If you think you know a nice way how to overload the methods, so that they can handle this feature too, you're welcome to create a issue or pull request on GitHub.

Note: All methods in this class accept only **keyword arguments** to improve the readability and to avoid wrong parameter orders compared to the official documentation.

`banadd`(**, ip=None, name=None, uid=None, time=None, banreason=None*)

Usage:

```
banadd [ip={regexp}] [name={regexp}] [uid={clientUID}] [time={timeInSeconds}]␣
↪[banreason={text}]
```

Adds a new ban rule on the selected virtual server. All parameters are optional but at least one of the following must be set: ip, name, or uid.

Example:

```
banadd ip=1.2.3.4 banreason=just\s4\sfun
banid=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.banadd(ip="127.0.0.1")
...
>>> ts3cmd.banadd(name="Ben")
...
>>> ts3cmd.banadd(name="Ben", time=3600, banreason="I hate you!")
```

banclient (*, *clid*, *time=None*, *banreason=None*)

Usage:

```
banclient clid={clientID} [time={timeInSeconds}] [banreason={text}]
```

Bans the client specified with ID clid from the server. Please note that this will create two separate ban rules for the targeted clients IP address and his unique identifier.

Example:

```
banclient clid=4 time=3600
banid=2
banid=3
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.banclient(clid=42, time=900, banreason="HAHA")
...
```

bandel (*, *banid*)

Usage:

```
bandel banid={banID}
```

Deletes the ban rule with ID banid from the server.

Example:

```
bandel banid=3
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.bandel(8)
...
```

bandelall ()

Usage:

```
bandelall
```

Deletes all active ban rules from the server.

Example:

```
bandelall
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.bandelall()
...
```

banlist()

Usage:

```
banlist
```

Displays a list of active bans on the selected virtual server.

Example:

```
banlist
banid=7 ip=1.2.3.4 created=1259444002242 invokername=Sven invokercldbid=56
invokeruid=oHhi9WzXLNEFQOwAu4JYKGU+C+c= reason enforcements=0
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.banlist()
...
```

bindinglist()

Usage:

```
bindinglist
```

Displays a list of IP addresses used by the server instance on multi-homed machines.

Example:

```
bindinglist
ip=0.0.0.0
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.bindinglist()
...
```

channeladdperm(**, cid, permvalue, permid=None, permsid=None*)

Usage:

```
channeladdperm cid={channelID} ( permid={permID}|permsid={permName} permvalue=
↪{permValue} )...
```

Adds a set of specified permissions to a channel. Multiple permissions can be added by providing the two parameters of each permission. A permission can be specified by permid or permsid.

Example:

```
channeladdperm cid=16 permsid=i_client_needed_join_power permvalue=50
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channeladdperm(
...     cid=12, permsid="i_client_needed_join_power", permvalue=50)
...
```

channelclientaddperm(***, *cid*, *cldbid*, *permvalue*, *permid=None*, *permsid=None*)

Usage:

```
channelclientaddperm cid={channelID} cldbid={clientDBID} ( permid={permID}
↪|permsid={permName} permvalue={permValue} )...
```

Adds a set of specified permissions to a client in a specific channel. Multiple permissions can be added by providing the two parameters of each permission. A permission can be specified by permid or permsid.

Example:

```
channelclientaddperm cid=12 cldbid=3 permsid=i_icon_id permvalue=100
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelclientaddperm(
...     cid=12, cldbid=3, permsid="i_icon_id", permvalue=100)
...
```

channelclientdelperm(***, *cid*, *cldbid*, *permsid=None*, *permid=None*)

Usage:

```
channelclientdelperm cid={channelID} cldbid={clientDBID} permid={permID}
↪|permsid={permName}...
```

Removes a set of specified permissions from a client in a specific channel. Multiple permissions can be removed at once. A permission can be specified by permid or permsid.

Example:

```
channelclientdelperm cid=12 cldbid=3 permsid=i_icon_id|permsid=b_icon_manage
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelclientdelperm(
...     cid=12, cldbid=3, permsid="i_icon_id")
...
>>> ts3cmd.channelclientdelperm(
...     cid=12, cldbid=3, permsid="b_icon_manage")
...
```

channelclientpermlist(***, *cid*, *cldbid*, *permsid=False*)

Usage:

```
channelclientpermlist cid={channelID} cldbid={clientDBID} [-permsid]
```

Displays a list of permissions defined for a client in a specific channel.

Example:

```
channelclientpermlist cid=12 cldbid=3
cid=12 cldbid=3 permid=4353 permvalue=1 permnegated=0 permskip=0|permid=17276␣
↪permvalue=50 permnegated=0 permskip=0|permid=21415 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelclientpermlist(cid=12, cldbid=3)
...
>>> ts3cmd.channelclientpermlist(cid=12, cldbid=2, permsid=True)
...
```

channelcreate(**, channel_name, **channel_properties*)

Usage:

```
channelcreate channel_name={channelName} [channel_properties...]
```

Creates a new channel using the given properties and displays its ID.

Example:

```
channelcreate channel_name=My\sChannel channel_topic=My\sTopic
cid=16
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelcreate(
...     channel_name="My Channel", channel_topic="My Topic")
...
```

channeldelete(**, cid, force*)

Usage:

```
channeldelete cid={channelID} force={1|0}
```

Deletes an existing channel by ID. If force is set to 1, the channel will be deleted even if there are clients within.

Example:: channeldelete cid=16 force=1 error id=0 msg=ok

Example:

```
>>> ts3cmd.channeldelete(cid=16, force=True)
...
```

channeldelperm(**, cid, permsid=None, permid=None*)

Usage:

```
channeldelperm cid=123 permid={permID}|permsid={permName}...
```

Removes a set of specified permissions from a channel. Multiple permissions can be removed at once. A permission can be specified by permid or permsid.

Example:

```
channeldelperm cid=16 permsid=i_icon_id|permsid=i_client_needed_talk_power
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channeldelperm(cid=16, permsid="i_icon_id")
...
>>> ts3cmd.channeldelperm(
...     cid=16, permsid="i_client_needed_talk_power")
...
```

channeledit (*, *cid*, ***channel_properties*)

Usage:

```
channeledit cid={channelID} [channel_properties...]
```

Changes a channels configuration using given properties.

Example:

```
channeledit cid=15 channel_codec_quality=3 channel_description=My\stext
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channeledit(
...     cid=15,
...     channel_codec_quality=3,
...     channel_description="My text"
...     )
...
```

channelfind (*, *pattern=None*)

Usage:

```
channelfind [pattern={channelName}]
```

Displays a list of channels matching a given name pattern.

Example:

```
channelfind pattern=default
cid=15 channel_name=Default\sChannel
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelfind(pattern="default")
...
```

channelgroupadd (*, *name*, *type_=None*)

Usage:

```
channelgroupadd name={groupName} [type={groupDbType}]
```

Creates a new channel group using a given name and displays its ID. The optional type parameter can be used to create ServerQuery groups and template groups.

Example:

```
channelgroupadd name=Channel\sAdmin
cgid=13
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgroupadd(name="Channel Admin")
...
```

channelgroupaddperm(**, cgid, permvalue, permid=None, permsid=None*)

Usage:

```
channelgroupaddperm cgid={groupID} permid={permID} permvalue={permValue}
channelgroupaddperm cgid={groupID} permsid={permName} permvalue={permValue}
```

Adds a set of specified permissions to a channel group. Multiple permissions can be added by providing the two parameters of each permission. A permission can be specified by permid or permsid.

Example:

```
channelgroupaddperm cgid=78 permsid=b_icon_manage permvalue=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgroupaddperm(
...     cgid=78, permsid="b_icon_manage", permvalue=1)
...
```

channelgroupclientlist(**, cid=None, cldbid=None, cgid=None*)

Usage:

```
channelgroupclientlist [cid={channelID}] [cldbid={clientDBID}] [cgid={groupID}
↪]
```

Displays all the client and/or channel IDs currently assigned to channel groups. All three parameters are optional so you're free to choose the most suitable combination for your requirements.

Example:

```
channelgroupclientlist cid=2 cgid=9
cid=2 cldbid=9 cgid=9|cid=2 cldbid=24 cgid=9|cid=2 cldbid=47 cgid=9
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgroupclientlist(cid=2, cgid=9)
...
```

channelgroupcopy(**, scgid, tcgid, name, type_*)

Usage:

```
channelgroupcopy scgid={sourceGroupID} tcgid={targetGroupID} name={groupName}␣
↪type={groupDbType}
```

Creates a copy of the channel group specified with ssgid. If tsgid is set to 0, the server will create a new group. To overwrite an existing group, simply set tsgid to the ID of a designated target group. If a target group is set, the name parameter will be ignored.

The type parameter can be used to create ServerQuery and template groups.

Example:

```
channelgroupcopy scgid=4 tcgid=0 name=My\sGroup\s(Copy) type=1
cgid=13
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgroupcopy(
...     scgid=4, tcgid=0, name="My Group (Copy)", type_=1)
...
```

channelgroupdel(***, *cgid*, *force*)

Usage:

```
channelgroupdel cgid={groupID} force={1|0}
```

Deletes a channel group by ID. If force is set to 1, the channel group will be deleted even if there are clients within.

Example:

```
channelgroupdel cgid=13
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgroupdel(cgid=13, force=True)
...
```

channelgroupdelperm(***, *cgid*, *permid=None*, *permsid=None*)

Usage:

```
channelgroupdelperm cgid={groupID} permid={permID}|...
channelgroupdelperm cgid={groupID} permsid={permName}|...
```

Removes a set of specified permissions from the channel group. Multiple permissions can be removed at once. A permission can be specified by permid or permsid.

Example:

```
channelgroupdelperm cgid=16 permid=17276|permid=21415
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgroupdelperm(
...     cgid=16, permsid="i_ft_needed_file_upload_power")
...
```

channelgrouplist()

Usage:

```
channelgrouplist
```

Displays a list of channel groups available on the selected virtual server.

Example:

```
channelgrouplist
cgid=1 name=Channel\sAdmin type=2 iconid=100 savedb=1|cgid=2 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgrouplist()
...
```

channelgrouppermlist(*, *cgid*, *permsid=False*)

Usage:

```
channelgrouppermlist cgid={groupID} [-permsid]
```

Displays a list of permissions assigned to the channel group specified with cgid.

Example:

```
channelgrouppermlist cgid=13
permid=8470 permvalue=1 permnegated=0 permskip=0|permid=8475 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgrouppermlist(cgid=13, permsid=False)
...
```

channelgrouprename(*, *cgid*, *name*)

Usage:

```
channelgrouprename cgid={groupID} name={groupName}
```

Changes the name of a specified channel group.

Example:

```
channelgrouprename cgid=13 name=New\sName
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelgrouprename(cgid=13, name="New name")
...
```

channelinfo(*, *cid*)

Usage:

```
channelinfo cid={channelID}
```

Displays detailed configuration information about a channel including ID, topic, description, etc.

Example:

```
channelinfo cid=1
channel_name=Default\sChannel channel_topic=No\s[b]topic[\/b]\shere channel_
↪description=Welcome ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelinfo(cid=1)
...
```

channellist(***, *topic=False*, *flags=False*, *voice=False*, *limits=False*, *icon=False*, *secondsempty=False*)

Usage:

```
channellist [-topic] [-flags] [-voice] [-limits] [-icon] [-secondsempty]
```

Displays a list of channels created on a virtual server including their ID, order, name, etc. The output can be modified using several command options.

Example:

```
channellist -topic
cid=15 pid=0 channel_order=0 channel_name=Default\sChannel channel_
↪topic=No\s[b]topic[\/b] total_clients=2|cid=16 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channellist(topic=True)
...
```

channelmove(***, *cid*, *cpid*, *order=None*)

Usage:

```
channelmove cid={channelID} cpid={channelParentID} [order={channelSortOrder}]
```

Moves a channel to a new parent channel with the ID cpid. If order is specified, the channel will be sorted right under the channel with the specified ID. If order is set to 0, the channel will be sorted right below the new parent.

Example:: channelmove cid=16 cpid=1 order=0 error id=0 msg=ok

Example:

```
>>> ts3cmd.channelmove(cid=16, cpid=1, order=0)
...
```

channelpermlist(***, *cid*, *permsid=False*)

Usage:

```
channelpermlist cid={channelID} [-permsid]
```

Displays a list of permissions defined for a channel.

Example:

```
channelpermlist cid=2
cid=2 permid=4353 permvalue=1 permnegated=0 permskip=0|permid=17276...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.channelpermlist(cid=2)
...
>>> ts3cmd.channelpermlist(cid=2, permsid=True)
...
```

clientaddperm (*, *cldbid*, *permvalue*, *permskip*, *permid=None*, *permsid=None*)

Usage:

```
clientaddperm cldbid={clientDBID} permid={permID} permvalue={permValue}␣
↪permskip={1|0}|...
clientaddperm cldbid={clientDBID} permsid={permName} permvalue={permValue}␣
↪permskip={1|0}|...
```

Adds a set of specified permissions to a client. Multiple permissions can be added by providing the three parameters of each permission. A permission can be specified by permid or permsid.

Example:

```
clientaddperm cldbid=16 permsid=i_client_talk_power permvalue=5 permskip=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientaddperm(
...     cldbid=16, permsid="i_client_talk_power", permvalue=5,
...     permskip=True)
...
```

clientdbdelete (*, *cldbid*)

Usage:

```
clientdbdelete cldbid={clientDBID}
```

Deletes a clients properties from the database.

Example:

```
clientdbdelete cldbid=56
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientdbdelete(cldbid=56)
...
```

clientdbedit (*, *cldbid*, ***client_properties*)

Usage:

```
clientdbedit cldbid={clientDBID} [client_properties...]
```

Changes a clients settings using given properties.

Example:

```
clientdbedit cldbid=56 client_description=Best\sguy\sever!
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientdbedit(
...     cldbid=56, client_description="Best guy ever!")
...
```

clientdbfind(*, *pattern*, *uid=False*)

Usage:

```
clientdbfind pattern={clientName|clientUID} [-uid]
```

Displays a list of client database IDs matching a given pattern. You can either search for a clients last known nickname or his unique identity by using the -uid option.

Example:

```
clientdbfind pattern=sven
cldbid=56
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientdbfind("sven")
...
>>> ts3cmd.clientdbfind("sven", uid=True)
...
```

clientdbinfo(*, *cldbid*)

Usage:

```
clientdbinfo cldbid={clientDBID}
```

Displays detailed database information about a client including unique ID, creation date, etc.

Example:

```
clientdbinfo cldbid=2
client_unique_identifier=5rRxyxEjd+Kk/MvPRfqZdSI0teA= client_
↪nickname=dante696 client_database_id=2 client_created=1279002103 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientdbinfo(cldbid=2)
...
```

clientdblist(*, *start=None*, *duration=None*, *count=False*)

Usage:

```
clientdblist [start={offset}] [duration={limit}] [-count]
```

Displays a list of client identities known by the server including their database ID, last nickname, etc.

Example:

```
clientdblist
cldbid=7 client_unique_identifier=DZhdQU58qyooEK4Fr8Ly738hEmc=
client_nickname=MuhChy client_created=1259147468 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientdblist()
...
>>> ts3cmd.clientdblist(count=True)
...
```

clientdelperm(**, cldbid, permid=None, permsid=None*)

Usage:

```
channeldelperm cldbid={clientDBID} permid={permID}|permsid={permName}...
```

Removes a set of specified permissions from a client. Multiple permissions can be removed at once. A permission can be specified by permid or permsid.

Example:

```
clientdelperm cldbid=16 permsid=i_icon_id|permsid=b_icon_manage
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientdelperm(cldbid=16, permsid="i_icon_id")
...
>>> ts3cmd.clientdelperm(cldbid=16, permsid="b_icon_manage")
...
```

clientedit(**, clid, **client_properties*)

Usage:

```
clientedit clid={clientID} [client_properties...]
```

Changes a clients settings using given properties.

Example:

```
clientedit clid=10 client_description=Best\sguy\sever!
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientedit(clid=10, client_description="Best guy ever!")
...
```

clientfind(**, pattern*)

Usage:

```
clientfind pattern={clientName}
```

Displays a list of clients matching a given name pattern.

Example:

```
clientfind pattern=sven
clid=7 client_nickname=Sven
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientfind(pattern="sven")
...
```

clientgetdbidfromuid(*, *cluid*)

Usage:

```
clientgetdbidfromuid cluid={clientUID}
```

Displays the database ID matching the unique identifier specified by cluid.

Example:

```
clientgetdbidfromuid cluid=dyjxkshZP6bz0n3bnwFQ1CkwZOM=
cluid=dyjxkshZP6bz0n3bnwFQ1CkwZOM= cldbid=32
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientgetdbidfromuid(cluid="dyjxkshZP6bz0n3bnwFQ1CkwZOM")
...
```

clientgetids(*, *cluid*)

Usage:

```
clientgetids cluid={clientUID}
```

Displays all client IDs matching the unique identifier specified by cluid.

Example:

```
clientgetids cluid=dyjxkshZP6bz0n3bnwFQ1CkwZOM=
cluid=dyjxkshZP6bz0n3bnwFQ1CkwZOM= clid=1 name=Janko
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientgetids(cluid="dyjxkshZP6bz0n3bnwFQ1CkwZOM")
...
```

clientgetnamefromdbid(*, *cldbid*)

Usage:

```
clientgetnamefromdbid cldbid={clientDBID}
```

Displays the unique identifier and nickname matching the database ID specified by cldbid.

Example:

```
clientgetnamefromdbid cldbid=32
cluid=dyjxkshZP6bz0n3bnwFQ1CkwZOM= cldbid=32 name=Janko
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientgetnamefromdbid(cldbid=32)
...
```

clientgetnamefromuid(*, *cluid*)

Usage:

```
clientgetnamefromuid cluid={clientUID}
```

Displays the database ID and nickname matching the unique identifier specified by cluid.

Example:

```
clientgetnamefromuid cluid=dyjxkshZP6bz0n3bnwFQ1CkwZOM=
cluid=dyjxkshZP6bz0n3bnwFQ1CkwZOM= cldbid=32 name=Janko
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientgetnamefromuid(
...     cluid="dyjxkshZP6bz0n3bnwFQ1CkwZOM")
...
```

clientgetuidfromclid(*, *clid*)

Usage:

```
clientgetuidfromclid clid={clientID}
```

Displays the unique identifier matching the clientID specified by clid.

Example:

```
clientgetuidfromclid clid=8
clid=8 cluid=yXM6PUfbCcPU+joxIFek1xOQwwQ= nickname=MuhChy1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientgetuidfromclid(clid=8)
...
```

clientinfo(*, *clid*)

Usage:

```
clientinfo clid={clientID}
```

Displays detailed configuration information about a client including unique ID, nickname, client version, etc.

Example:

```
clientinfo clid=6
client_unique_identifier=P5H2hrN6+gpQI4n\/dXp3p17vtY0= client_nickname=Rabe
client_version=3.0.0-alpha24\s[Build:\s8785]...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientinfo(clid=6)
...
```

clientkick(*, *clid*, *reasonid*, *reasonmsg=None*)

Usage:

```
clientkick reasonid={4|5} [reasonmsg={text}] clid={clientID}...
```

Kicks one or more clients specified with clid from their currently joined channel or from the server, depending on reasonid. The reasonmsg parameter specifies a text message sent to the kicked clients. This parameter is optional and may only have a maximum of 40 characters.

Available reasonid values are:

- 4: Kick the client from his current channel into the default channel
- 5: Kick the client from the server

Example:

```
clientkick reasonid=4 reasonmsg=Go\saway! clid=5|clid=6
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientkick(reasonid=4, reasonmsg="Go away!", clid=5)
...
>>> ts3cmd.clientkick(reasonid=4, reasonmsg="Go away!", clid=6)
...
```

clientlist(***, *uid=False*, *away=False*, *voice=False*, *times=False*, *groups=False*, *info=False*, *country=False*, *ip=False*)

Usage:

```
clientlist [-uid] [-away] [-voice] [-times] [-groups] [-info] [-country] [-ip]
```

Displays a list of clients online on a virtual server including their ID, nickname, status flags, etc. The output can be modified using several command options. Please note that the output will only contain clients which are currently in channels you're able to subscribe to.

Example:

```
clientlist -away
clid=5 cid=7 client_database_id=40 client_nickname=ScP client_type=0
client_away=1 client_away_message=not\shere|clid=6...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientlist(away=True)
...
```

clientmove(***, *clid*, *cid*, *cpw=None*)

Usage:

```
clientmove cid={channelID} [cpw={channelPassword}] clid={clientID}...
```

Moves one or more clients specified with clid to the channel with ID cid. If the target channel has a password, it needs to be specified with cpw. If the channel has no password, the parameter can be omitted.

Example:

```
clientmove cid=3 clid=5|clid=6
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientmove(cid=3, clid=5)
...
>>> ts3cmd.clientmove(cid=3, clid=6)
...
```

clientpermlist(*, *cldbid*, *permsid=False*)

Usage:

```
clientpermlist cldbid={clientDBID} [-permsid]
```

Displays a list of permissions defined for a client.

Example:: clientpermlist cldbid=2 cldbid=2 permid=4353 permvalue=1 permnegated=0 permskip=0|permid=17276... error id=0 msg=ok

Example:

```
>>> ts3example.clientpermlist(cldbid=2)
...
```

clientpoke(*, *msg*, *clid*)

Usage:

```
clientpoke msg={txt} clid={clientID}
```

Sends a poke message to the client specified with clid.

Example:

```
clientpoke msg=Wake\sup! clid=5
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientpoke(msg="Wake up", clid=5)
...
```

clientsetserverquerylogin(*, *client_login_name*)

Usage:

```
clientsetserverquerylogin client_login_name={username}
```

Updates your own ServerQuery login credentials using a specified username. The password will be auto-generated.

Example:

```
clientsetserverquerylogin client_login_name=admin
client_login_password=+r\/TQqvR
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientsetserverquerylogin(client_login_name="admin")
...
```

clientupdate(***client_properties*)

Usage:

```
clientupdate [client_properties...]
```

Change your ServerQuery clients settings using given properties.

Example:

```
clientupdate client_nickname=ScP\s(query)
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.clientupdate(client_nickname="ScP (query)")
...
```

complainadd(**, tcldbid, message*)

Usage:

```
complainadd tcldbid={targetClientDBID} message={text}
```

Submits a complaint about the client with database ID tcldbid to the server.

Example:

```
complainadd tcldbid=3 message=Bad\sguy!
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.complainadd(tcldbid=3, message="Bad guy!")
...
```

complaindel(**, tcldbid, fcldbid*)

Usage:

```
complaindel tcldbid={targetClientDBID} fcldbid={fromClientDBID}
```

Deletes the complaint about the client with database ID tcldbid submitted by the client with database ID fcldbid from the server.

Example:

```
complaindel tcldbid=3 fcldbid=4
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.complaindel(tcldbid=3, fcldbid=4)
...
```

complaindelall(**, tcldbid*)

Usage:

```
complaindelall tcldbid={targetClientDBID}
```

Deletes all complaints about the client with database ID tcldbid from the server.

Example:

```
complaindelall tcldbid=3
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.complaindelall(tcldbid=3)
...
```

complainlist (***, *tcldbid=None*)

Usage:

```
complainlist [tcldbid={targetClientDBID}]
```

Displays a list of complaints on the selected virtual server. If tcldbid is specified, only complaints about the targeted client will be shown.

Example:

```
complainlist tcldbid=3
tcldbid=3 tname=Julian fcldbid=56 fname=Sven message=Bad\sguy!...
error id=0 msg=ok
```

Example:

```
>>> ts3.complainlist(tcldbid=3)
...
```

custominfo (***, *cldbid*)

Usage:

```
custominfo cldbid={clientDBID}
```

Displays a list of custom properties for the client specified with cldbid.

Example:

```
custominfo cldbid=3
cldbid=3 ident=forum_account value=ScP|ident=forum_id value=123
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.custominfo(cldbid=3)
...
```

customsearch (***, *ident*, *pattern*)

Usage:

```
customsearch ident={ident} pattern={pattern}
```

Searches for custom client properties specified by ident and value. The value parameter can include regular characters and SQL wildcard characters (e.g. %).

Example:

```
customsearch ident=forum_account pattern=%ScP%
cldbid=2 ident=forum_account value=ScP
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.customsearch(ident="forum_account", pattern="%ScP")
...
```

ftcreatedir(*, *cid*, *dirname*, *cpw=None*)

Usage:

```
ftcreatedir cid={channelID} cpw={channelPassword} dirname={dirPath}
```

Creates new directory in a channels file repository.

Example:

```
ftcreatedir cid=2 cpw= dirname=\/My\sDirectory
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftcreatedir(cid=2, dirname="/My Directory")
...
```

ftdeletefile(*, *cid*, *name*, *cpw=None*)

Usage:

```
ftdeletefile cid={channelID} cpw={channelPassword} name={filePath}...
```

Deletes one or more files stored in a channels file repository.

Example:

```
ftdeletefile cid=2 cpw= name=\/Pic1.PNG|name=\/Pic2.PNG
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftdeletefile(cid=2, name="/Pic1.PNG")
...
>>> ts3cmd.ftdeletefile(cid=2, name="/Pic2.PNG")
...
```

ftgetfileinfo(*, *name*, *cid*, *cpw=None*)

Usage:

```
ftgetfileinfo cid={channelID} cpw={channelPassword} name={filePath}...
```

Displays detailed information about one or more specified files stored in a channels file repository.

Example:

```
ftgetfileinfo cid=2 cpw= name=\/Pic1.PNG|cid=2 cpw= name=\/Pic2.PNG
cid=2 path=\/ name=Stuff size=0 datetime=1259415210 type=0|name=Pic1.PNG
size=563783 datetime=1259425462 type=1|name=Pic2.PNG...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftgetfileinfo(cid=2, name="/Pic1.PNG")
...
```

ftgetfilelist(***, *path*, *cid*, *cpw=None*)

Usage:

```
ftgetfilelist cid={channelID} cpw={channelPassword} path={filePath}
```

Displays a list of files and directories stored in the specified channels file repository.

Example:

```
ftgetfilelist cid=2 cpw= path=\/
cid=2 path=\/ name=Stuff size=0 datetime=1259415210 type=0|name=Pic1.PNG
size=563783 datetime=1259425462 type=1|name=Pic2.PNG...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftgetfilelist(cid=2, path="/")
...
```

ftinitdownload(***, *clientftfid*, *name*, *seekpos*, *cid*, *cpw=None*)

Usage:

```
ftinitdownload clientftfid={clientFileTransferID} name={filePath}
               cid={channelID} cpw={channelPassword}
               seekpos={seekPosition}
```

Initializes a file transfer download. clientftfid is an arbitrary ID to identify the file transfer on client-side. On success, the server generates a new ftkey which is required to start downloading the file through TeamSpeak 3's file transfer interface.

Example:

```
ftinitdownload clientftfid=1 name=\/image.iso cid=5 cpw= seekpos=0
clientftfid=1 serverftfid=7 ftkey=NrOga\/4d2GpYC5oKgxuclTO37X83ca\/1 port=...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftinitdownload(
...     clientftfid=1, name="/image.iso", cid=5, seekpos=0)
...
```

ftinitupload(***, *clientftfid*, *name*, *cid*, *size*, *overwrite*, *resume*, *cpw=None*)

Usage:

```
ftinitupload clientftfid={clientFileTransferID} name={filePath}
             cid={channelID} cpw={channelPassword} size={fileSize}
             overwrite={1|0} resume={1|0}
```

Initializes a file transfer upload. clientftfid is an arbitrary ID to identify the file transfer on client-side. On success, the server generates a new ftkey which is required to start uploading the file through TeamSpeak 3's file transfer interface.

Example:

```
ftinitupload clientftfid=1 name=\/image.iso cid=5 cpw= size=673460224␣
↪overwrite=1 resume=0
clientftfid=1 serverftfid=6 ftkey=itRNdsIOvcBiBg\/Xj4Ge51ZSrsShHuid port=...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftinitupload(
...     clientftfid=1, name="/image.iso", cid=5, size=673460224,
...     overwrite=1, resume=0)
...
```

ftlist()

Usage:

```
ftlist
```

Displays a list of running file transfers on the selected virtual server. The output contains the path to which a file is uploaded to, the current transfer rate in bytes per second, etc.

Example:

```
ftlist
clid=2 path=files\/virtualserver_1\/channel_5 name=image.iso size=673460224
sizedone=450756 clientftfid=2 serverftfid=6 sender=0 status=1 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftlist()
...
```

ftrenamefile(*, *cid*, *oldname*, *newname*, *cpw=None*, *tcid=None*, *tcpw=None*)

Usage:

```
ftrenamefile cid={channelID} cpw={channelPassword}
             [tcid={targetChannelID}] [tcpw={targetChannelPassword}]
             oldname={oldFilePath} newname={newFilePath}
```

Renames a file in a channels file repository. If the two parameters tcid and tcpw are specified, the file will be moved into another channels file repository.

Example:

```
ftrenamefile cid=2 cpw= tcid=3 tcpw= oldname=\/Pic3.PNG newname=\/Pic3.PNG
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftrenamefile(
...     cid=2, tcid=3, oldname="/Pic3.PNG", newname="/Pic3.PNG")
...
```

ftstop(*, *serverftfid*, *delete*)

Usage:

```
ftstop serverftfid={serverFileTransferID} delete={1|0}
```

Stops the running file transfer with server-side ID serverftfid.

Example:

```
ftstop serverftfid=2 delete=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.ftstop(serverftfid=2, delete=1)
...
```

gm (*, *msg*)

Usage:

```
gm msg={text}
```

Sends a text message to all clients on all virtual servers in the TeamSpeak 3 Server instance.

Example:

```
gm msg=Hello\sWorld!
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.gm(msg="Hello World")
...
```

help (*, *cmd=None*)

TeamSpeak 3 Server :: ServerQuery (c) TeamSpeak Systems GmbH

ServerQuery is a command-line interface built into the TeamSpeak 3 Server which allows powerful scripting and automation tools to be built based on the exact same instruction set and functionality provided by the TeamSpeak 3 Client. For example, you can use scripts to automate the management of virtual servers or nightly backups. In short, you can perform operations more efficiently by using ServerQuery scripts than you can by using a user interface.

Command Overview:

```
help                           | read help files
login                          | authenticate with the server
logout                         | deselect virtual server and log out
quit                           | close connection
use                            | select virtual server
banadd                         | create a ban rule
banclient                      | ban a client
bandelall                      | delete all ban rules
bandel                         | delete a ban rule
banlist                        | list ban rules on a virtual server
bindinglist                    | list IP addresses used by the server instance
channeladdperm                 | assign permission to channel
channelclientaddperm           | assign permission to channel-client combi
channelclientdelperm           | remove permission from channel-client combi
channelclientpermlist          | list channel-client specific permissions
channelcreate                  | create a channel
channeldelete                  | delete a channel
channeldelperm                 | remove permission from channel
channeledit                    | change channel properties
channelfind                    | find channel by name
channelgroupadd                | create a channel group
channelgroupaddperm            | assign permission to channel group
channelgroupclientlist         | find channel groups by client ID
channelgroupcopy               | copy a channel group
channelgroupdel                | delete a channel group
```

(continues on next page)

(continued from previous page)

```
channelgroupdelperm          | remove permission from channel group
channelgrouplist             | list channel groups
channelgrouppermlist         | list channel group permissions
channelgrouprename           | rename a channel group
channelinfo                  | display channel properties
channellist                  | list channels on a virtual server
channelmove                  | move channel to new parent
channelpermlist              | list channel specific permissions
clientaddperm                | assign permission to client
clientdbdelete               | delete client database properties
clientdbedit                 | change client database properties
clientdbfind                 | find client database ID by nickname or UID
clientdbinfo                 | display client database properties
clientdblist                 | list known client UIDs
clientdelperm                | remove permission from client
clientedit                   | change client properties
clientfind                   | find client by nickname
clientgetdbidfromuid         | find client database ID by UID
clientgetids                 | find client IDs by UID
clientgetnamefromdbid        | find client nickname by database ID
clientgetnamefromuid         | find client nickname by UID
clientgetuidfromclid         | find client UID by client ID
clientinfo                   | display client properties
clientkick                   | kick a client
clientlist                   | list clients online on a virtual server
clientmove                   | move a client
clientpermlist               | list client specific permissions
clientpoke                   | poke a client
clientsetserverquerylogin    | set own login credentials
clientupdate                 | set own properties
complainadd                  | create a client complaint
complaindelall               | delete all client complaints
complaindel                  | delete a client complaint
complainlist                 | list client complaints on a virtual server
custominfo                   | display custom client properties
customsearch                 | search for custom client properties
ftcreatedir                  | create a directory
ftdeletefile                 | delete a file
ftgetfileinfo                | display details about a file
ftgetfilelist                | list files stored in a channel filebase
ftinitdownload               | init a file download
ftinitupload                 | init a file upload
ftlist                       | list active file transfers
ftrenamefile                 | rename a file
ftstop                       | stop a file transfer
gm                           | send global text message
hostinfo                     | display server instance connection info
instanceedit                 | change server instance properties
instanceinfo                 | display server instance properties
logadd                       | add custom entry to log
logview                      | list recent log entries
messageadd                   | send an offline message
messagedel                   | delete an offline message from your inbox
messageget                   | display an offline message from your inbox
messagelist                  | list offline messages from your inbox
messageupdateflag            | mark an offline message as read
permfind                     | find permission assignments by ID
```

(continues on next page)

(continued from previous page)

```
permget                         | display client permission value for yourself
permidgetbyname                 | find permission ID by name
permissionlist                  | list permissions available
permoverview                    | display client permission overview
permreset                       | delete all server and channel groups and␣
↪restore default permissions
privilegekeyadd                 | creates a new privilege key
privilegekeydelete              | delete an existing privilege key
privilegekeylist                | list all existing privilege keys on this server
privilegekeyuse                 | use a privilege key
sendtextmessage                 | send text message
servercreate                    | create a virtual server
serverdelete                    | delete a virtual server
serveredit                      | change virtual server properties
servergroupaddclient            | add client to server group
servergroupadd                  | create a server group
servergroupaddperm              | assign permissions to server group
servergroupautoaddperm          | globally assign permissions to server groups
servergroupbyclientid           | get all server groups of specified client
servergroupclientlist           | list clients in a server group
servergroupcopy                 | create a copy of an existing server group
servergroupdelclient            | remove client from server group
servergroupdel                  | delete a server group
servergroupdelperm              | remove permissions from server group
servergroupautodelperm          | globally remove permissions from server group
servergrouplist                 | list server groups
servergrouppermlist             | list server group permissions
servergrouprename               | rename a server group
servergroupsbyclientid          | find server groups by client ID
serveridgetbyport               | find database ID by virtual server port
serverinfo                      | display virtual server properties
serverlist                      | list virtual servers
servernotifyregister            | register for event notifications
servernotifyunregister          | unregister from event notifications
serverprocessstop               | shutdown server process
serverrequestconnectioninfo     | display virtual server connection info
serversnapshotcreate            | create snapshot of a virtual server
serversnapshotdeploy            | deploy snapshot of a virtual server
serverstart                     | start a virtual server
servertemppasswordadd           | create a new temporary server password
servertemppassworddel           | delete an existing temporary server password
servertemppasswordlist          | list all existing temporary server passwords
serverstop                      | stop a virtual server
setclientchannelgroup           | set a clients channel group
tokenadd                        | create a privilege key (token)
tokendelete                     | delete a privilege key (token)
tokenlist                       | list privilege keys (tokens) available
tokenuse                        | use a privilege key (token)
version                         | display version information
whoami                          | display current session info
```

Example:

```
>>> ts3cmd.help()
...
>>> ts3cmd.help(cmd="whoami")
...
```

hostinfo()

Usage:

```
hostinfo
```

Displays detailed configuration information about the server instance including uptime, number of virtual servers online, traffic information, etc.

Example:

```
hostinfo
virtualservers_running_total=3 virtualservers_total_maxclients=384 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.hostinfo()
...
```

instanceedit(***instance_properties*)

Usage:

```
instanceedit [instance_properties...]
```

Changes the server instance configuration using given properties.

Example:

```
instanceedit serverinstance_filetransfer_port=1337
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.instanceedit(serverinstance_filetransfer_port=1337)
...
```

instanceinfo()

Usage:

```
instanceinfo
```

Displays the server instance configuration including database revision number, the file transfer port, default group IDs, etc.

Example:

```
instanceinfo
serverinstance_database_version=12 serverinstance_filetransfer_port=30033
serverinstance_template_guest_serverquery_group=1...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.instanceinfo()
...
```

logadd(**, loglevel, logmsg*)

Usage:

```
logadd loglevel={1-4} logmsg={text}
```

Writes a custom entry into the servers log. Depending on your permissions, you'll be able to add entries into the server instance log and/or your virtual servers log. The loglevel parameter specifies the type of the entry.

Example:

```
logadd loglevel=4 logmsg=Informational\smessage!
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.logadd(loglevel=4, logmsg="Informational message!")
...
```

login (*, *client_login_name*, *client_login_password*)

Usage:

login client_login_name={username} client_login_password={password} login {username} {password}

Authenticates with the TeamSpeak 3 Server instance using given ServerQuery login credentials.

Example:: login client_login_name=xyz client_login_password=xyz error id=0 msg=ok

login xyz xyz error id=0 msg=ok

Example:

```
>>> ts3cmd.login(
...     client_login_name="xyz", client_login_password="xyz")
...
```

logout ()

Usage:

```
logout
```

Deselects the active virtual server and logs out from the server instance.

Example:

```
logout
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.logout()
...
```

logview (*, *lines=None*, *reverse=None*, *instance=None*, *begin_pos=None*)

Usage:

```
logview [lines={1-100}] [reverse={1|0}] [instance={1|0}] [begin_pos={n}]
```

Displays a specified number of entries from the servers logfile. If instance is set to 1, the server will return lines from the master logfile (ts3server_0) instead of the selected virtual server logfile.

Example:

```
logview
last_pos=403788 file_size=411980 l=\p\slistening\son\s0.0.0.0:9987 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.logview()
...
>>> ts3cmd.logview(lines=100, begin_pos=10)
...
```

messageadd (*, *cliud*, *subject*, *message*)

Usage:

```
messageadd cluid={clientUID} subject={subject} message={text}
```

Sends an offline message to the client specified by cluid.

Example:

```
messageadd cluid=oHhi9WzXLNEFQOwAu4JYKGU+C+c= subject=Hi! message=Hello?!?
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.messageadd(
...     cluid="oHhi9WzXLNEFQOwAu4JYKGU+C+c=", subject="Hi!",
...     message="Hello?!?")
...
```

messagedel (*, *msgid*)

Usage:

```
messagedel msgid={messageID}
```

Deletes an existing offline message with ID msgid from your inbox.

Example:

```
messagedel msgid=4
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.messagedel(msgid=4)
...
```

messageget (*, *msgid*)

Usage:

```
messageget msgid={messageID}
```

Displays an existing offline message with ID msgid from your inbox. Please note that this does not automatically set the flag_read property of the message.

Example:

```
messageget msgid=4
msgid=4 cluid=xwEzb5ENOaglVHu9oelK++reUyE= subject=Hi! message=Hello?!?
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.messageget(msgid=4)
...
```

messagelist()

Usage:

```
messagelist
```

Displays a list of offline messages you've received. The output contains the senders unique identifier, the messages subject, etc.

Example:

```
messagelist
msgid=4 cluid=xwEzb5ENOaglVHu9oelK++reUyE= subject=Test flag_read=0...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.messagelist()
...
```

messageupdateflag(*, *msgid*, *flag*)

Usage:

```
messageupdateflag msgid={messageID} flag={1|0}
```

Updates the flag_read property of the offline message specified with msgid. If flag is set to 1, the message will be marked as read.

Example:

```
messageupdateflag msgid=4 flag=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.messageupdateflag(msgid=4, flag=1)
...
```

permfind(*, *permid*)

Usage:

```
permfind permid={permID}
```

Displays detailed information about all assignments of the permission specified with permid. The output is similar to permoverview which includes the type and the ID of the client, channel or group associated with the permission.

Example:

```
permfind permid=4353
t=0 id1=1 id2=0 p=4353|t=0 id1=2 id2=0 p=4353
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.permfind(permid=4353)
...
```

permget (*, *permid=None*, *permsid=None*)

Usage:

```
permget permid={permID}
permget permsid={permName}
```

Displays the current value of the permission specified with permid or permsid for your own connection. This can be useful when you need to check your own privileges.

Example:

```
permget permid=21174
permsid=i_client_move_power permid=21174 permvalue=100
error id=0 msg=ok

permget permsid=i_client_move_power
permsid=i_client_move_power permid=21174 permvalue=100
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.permget(permid=21174)
...
>>> ts3cmd.permget(permsid="i_client_move_power")
...
```

permidgetbyname (*, *permsid=None*)

Usage:

```
permidgetbyname permsid={permName}|permsid={permName}|...
```

Displays the database ID of one or more permissions specified by permsid.

Example:

```
permidgetbyname permsid=b_serverinstance_help_view
permsid=b_serverinstance_help_view permid=4353
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.permidgetbyname(permsid="b_serverinstance_help_view")
...
```

permissionlist ()

Usage:

```
permissionlist
```

Displays a list of permissions available on the server instance including ID, name and description.

Example:

```
permissionlist
permid=21413 permname=b_client_channel_textmessage_send permdesc=Send\ste...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.permissionlist()
...
```

permoverview(***, *cid*, *cldbid*, *permid*)

Usage:

```
permoverview cid={channelID} cldbid={clientDBID} permid={permID}
```

Displays all permissions assigned to a client for the channel specified with cid. If permid is set to 0, all permissions will be displayed. The output follows the following format:

t={permType} id1={id1} id2={id2} p={permID} v={permValue} n={permNegated} s={permSkip}|t={permType} id1={id1} id2={id2} p={permID} v={permValue} n={permNegated} s={permSkip}|...

The possible values for t, id1 and id2 are:

0: Server Group; => id1={serverGroupID}, id2=0 1: Global Client; => id1={clientDBID}, id2=0 2: Channel; => id1={channelID}, id2=0 3: Channel Group; => id1={channelID}, id2={channelGroupID} 4: Channel Client; => id1={channelID}, id2={clientDBID}

Example:

```
permoverview cldbid=57 cid=74 permid=0
t=0 id1=5 id2=0 p=37 v=1 n=0 s=0|t=0 id1=5 id2=0 p=38 v=1 n=0 s=0|...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.permoverview(cldbid=57, cid=74, permid=0)
...
```

permreset()

Usage:

```
permreset
```

Restores the default permission settings on the selected virtual server and creates a new initial administrator token. Please note that in case of an error during the permreset call - e.g. when the database has been modified or corrupted - the virtual server will be deleted from the database.

Example:

```
permreset
token=MqQbPLLm6jLC+x8j31jUL7GkME1UY0GaDYK+XG5e
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.permreset()
...
```

privilegekeyadd(***, *tokentype*, *tokenid1*, *tokenid2*, *tokendescription=None*, *tokencustomset=None*)

Usage:

```
privilegekeyadd tokentype={1|0} tokenid1={groupID}
                tokenid2={channelID} [tokendescription={description}]
                [tokencustomset={customFieldSet}]
```

Create a new token. If tokentype is set to 0, the ID specified with tokenid1 will be a server group ID. Otherwise, tokenid1 is used as a channel group ID and you need to provide a valid channel ID using tokenid2.

The tokencustomset parameter allows you to specify a set of custom client properties. This feature can be used when generating tokens to combine a website account database with a TeamSpeak user. The syntax of the value needs to be escaped using the ServerQuery escape patterns and has to follow the general syntax of:

ident=ident1 value=value1|ident=ident2 value=value2|ident=ident3 value=value3

Example:

```
privilegekeyadd tokentype=0 tokenid1=6 tokenid2=0 tokendescription=Test
 tokencustomset=ident=forum_user\svalue=dante\pident=forum_id\svalue=123
token=1ayoQOxG8r5Re78zgChvLYBWWaFWCoty0Uh+pUFk
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.privilegekeyadd(
...     tokentype=0, tokenid1=6, tokenid2=0,
...     tokendescription="Test",
...     tokencustomset="ident=forum_user\svalue=dante\pident=forum_
↪id\svalue=123"
...     )
...
```

privilegekeydelete(***, *token*)

Usage:

```
privilegekeydelete token={tokenKey}
```

Deletes an existing token matching the token key specified with token.

Example:

```
privilegekeydelete token=eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.privilegekeydelete(
...     token="eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW")
...
```

privilegekeylist()

Usage:

```
privilegekeylist
```

Displays a list of tokens available including their type and group IDs. Tokens can be used to gain access to specified server or channel groups.

A token is similar to a client with administrator privileges that adds you to a certain permission group, but without the necessity of a such a client with administrator privileges to actually exist. It is a long (random looking) string that can be used as a ticket into a specific server group.

Example:

```
privilegekeylist
token=88CVUg\/zkujt+y+WfHdko79UcM4R6uyCL6nEfy3B token_type=0 token_id1=9...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.privilegekeylist()
...
```

privilegekeyuse(*, *token*)

Usage:

```
privilegekeyuse token={tokenKey}
```

Use a token key gain access to a server or channel group. Please note that the server will automatically delete the token after it has been used.

Example:

```
privilegekeyuse token=eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.privilegekeyuse(
...     token="eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW")
...
```

quit()

Usage:

```
quit
```

Closes the ServerQuery connection to the TeamSpeak 3 Server instance.

Example:

```
quit
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.quit()
...
```

sendtextmessage(*, *targetmode*, *target*, *msg*)

Usage:

```
sendtextmessage targetmode={1-3}
                target={serverID|channelID|clientID} msg={text}
```

Sends a text message a specified target. The type of the target is determined by targetmode while target specifies the ID of the recipient, whether it be a virtual server, a channel or a client.

Example:

```
sendtextmessage targetmode=2 target=1 msg=Hello\sWorld!
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.sendtextmessage(
...     targetmode=2, target=1, msg="Hello World!")
```

servercreate(***virtualserver_properties*)

Usage:

```
servercreate [virtualserver_properties...]
```

Creates a new virtual server using the given properties and displays its ID and initial administrator token. If virtualserver_port is not specified, the server will test for the first unused UDP port.

Example:

```
servercreate virtualserver_name=TeamSpeak\s]\p[\sServer
 virtualserver_port=9988 virtualserver_maxclients=32
sid=7 token=HhPbcMAMdAHGUip1yOma2Tl3sN0DN7B3Y0JVzYv6 virtualserver_port=9988
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servercreate(
...     virtualserver_name="TeamSpeak ]|[ Server",
...     virtualserver_port=9988, virtualserver_maxclients=32)
...
```

serverdelete(**, sid*)

Usage:

```
serverdelete sid={serverID}
```

Deletes the virtual server specified with sid. Please note that only virtual servers in stopped state can be deleted.

Example:

```
serverdelete sid=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serverdelete(sid=1)
...
```

serveredit(***virtualserver_properties*)

Usage:

```
serveredit [virtualserver_properties...]
```

Changes the selected virtual servers configuration using given properties.

Example:

```
serveredit virtualserver_name=TeamSpeak\sServer virtualserver_maxclients=32
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serveredit(
...     virtualserver_name="TeamSpeak Server",
...     virtualserver_maxclients=32)
...
```

servergroupadd(*, *name*, *type_=None*)

Usage:

```
servergroupadd name={groupName} [type={groupDbType}]
```

Creates a new server group using the name specified with name and displays its ID. The optional type parameter can be used to create ServerQuery groups and template groups.

Example:

```
servergroupadd name=Server\sAdmin
sgid=13
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupadd(name="Server Admin")
...
```

servergroupaddclient(*, *sgid*, *cldbid*)

Usage:

```
servergroupaddclient sgid={groupID} cldbid={clientDBID}
```

Adds a client to the server group specified with sgid. Please note that a client cannot be added to default groups or template groups.

Example:

```
servergroupaddclient sgid=16 cldbid=3
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupaddclient(sgid=16, cldbid=3)
...
```

servergroupaddperm(*, *sgid*, *permnegated*, *permskip*, *permid=None*, *permsid=None*, *permvalue=None*)

Usage:

```
servergroupaddperm sgid={groupID} permid={permID}
                   permvalue={permValue} permnegated={1|0}
                   permskip={1|0}|...
servergroupaddperm sgid={groupID} permsid={permName}
                   permvalue={permValue} permnegated={1|0}
                   permskip={1|0}|...
```

Adds a set of specified permissions to the server group specified with sgid. Multiple permissions can be added by providing the four parameters of each permission. A permission can be specified by permid or permsid.

Example:

```
servergroupaddperm sgid=13 permid=8470 permvalue=1 permnegated=0
 permskip=0|permid=8475 permvalue=0 permnegated=1 permskip=0
error id=0 msg=ok

servergroupaddperm sgid=13 permsid=i_icon_id permvalue=123
 permnegated=0 permskip=0|permsid=b_virtualserver_stop permvalue=0
 permnegated=1 permskip=0
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupaddperm(
...     sgid=13, permid=8470, permvalue=1, permnegated=0,
...     permskip=0)
...
>>> ts3cmd.servergroupaddperm(
...     sgid=13, permsid="i_icon_id", permvalue=123, permnegated=0,
...     permskip=0)
...
```

servergroupautoaddperm(**, sgtype, permvalue, permnegated, permskip, permid=None, permsid=None*)

Usage:

```
servergroupautoaddperm sgtype={type} permid={permID}
                       permvalue={permValue} permnegated={1|0}
                       permskip={1|0}|...
servergroupautoaddperm sgtype={type} permsid={permName}
                       permvalue={permValue} permnegated={1|0}
                       permskip={1|0}|...
```

Adds a set of specified permissions to ALL regular server groups on all virtual servers. The target groups will be identified by the value of their i_group_auto_update_type permission specified with sgtype. Multiple permissions can be added at once. A permission can be specified by permid or permsid.

The known values for sgtype are:

10: Channel Guest 15: Server Guest 20: Query Guest 25: Channel Voice 30: Server Normal 35: Channel Operator 40: Channel Admin 45: Server Admin 50: Query Admin

Example:

```
servergroupautoaddperm sgtype=45 permid=8470 permvalue=1 permnegated=0
 permskip=0|permid=8475 permvalue=0 permnegated=1 permskip=0
error id=0 msg=ok
```

(continues on next page)

(continued from previous page)

```

servergroupautoaddperm sgtype=45 permsid=i_icon_id permvalue=123
 permnegated=0 permskip=0|permsid=b_virtualserver_stop permvalue=0
 permnegated=1 permskip=0
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupautoaddperm(
...     sgtype=45, permid=8470, permvalue=1, permnegated=0,
...     permskip=0)
...
>>> ts3cmd.servergroupautoaddperm(
...     sgtype=45, permsid="i_icon_id", permvalue=123,
...     permnegated=0, permskip=0)
...
```

servergroupautodelperm(**, sgtype, permid=None, permsid=None*)

Usage:

```
servergroupautodelperm sgtype={type} permid={permID}|permid={permID}|...
servergroupautodelperm sgtype={type} permsid={permName}|...
```

Removes a set of specified permissions from ALL regular server groups on all virtual servers. The target groups will be identified by the value of their i_group_auto_update_type permission specified with sgtype. Multiple permissions can be removed at once. A permission can be specified by permid or permsid.

The known values for sgtype are:

10: Channel Guest 15: Server Guest 20: Query Guest 25: Channel Voice 30: Server Normal 35: Channel Operator 40: Channel Admin 45: Server Admin 50: Query Admin

Examples:

```
servergroupautodelperm sgtype=45 permid=8470|permid=8475
error id=0 msg=ok

servergroupautodelperm sgtype=45 permsid=b_virtualserver_modify_maxclients
error id=0 msg=ok
```

Examples:

```
>>> ts3cmd.servergroupautodelperm(sgtype=45, permid=8470)
...
>>> ts3cmd.servergroupautodelperm(
...     sgtype=45, permsid="b_virtualserver_modify_maxclients")
...
```

servergroupbyclientid(**, cldbid*)

Usage:

```
servergroupsbyclientid cldbid={clientDBID}
```

Displays all server groups the client specified with cldbid is currently residing in.

Example:

```
servergroupsbyclientid cldbid=18
name=Server\sAdmin sgid=6 cldbid=18
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupbyclientid(cldbid=18)
...
```

servergroupclientlist(*, *sgid*, *names=False*)
Usage:

```
servergroupclientlist sgid={groupID} [-names]
```

Displays the IDs of all clients currently residing in the server group specified with sgid. If you're using the -names option, the output will also contain the last known nickname and the unique identifier of the clients.

Example:

```
servergroupclientlist sgid=16
cldbid=7|cldbid=8|cldbid=9|cldbid=11|cldbid=13|cldbid=16|cldbid=18|...
error id=0 msg=ok

servergroupclientlist sgid=8 -names
cldbid=4 client_nickname=ScP client_unique_identifier=FPMPSC6MXqXq7...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupclientlist(sgid=16)
...
>>> ts3cmd.servergroupclientlist(sgid=8, names=True)
...
```

servergroupcopy(*, *ssgid*, *tsgid*, *name*, *type_*)
Usage:

```
servergroupcopy ssgid={sourceGroupID} tsgid={targetGroupID}
                name={groupName} type={groupDbType}
```

Creates a copy of the server group specified with ssgid. If tsgid is set to 0, the server will create a new group. To overwrite an existing group, simply set tsgid to the ID of a designated target group. If a target group is set, the name parameter will be ignored.

The type parameter can be used to create ServerQuery and template groups.

Example:

```
servergroupcopy ssgid=6 tsgid=0 name=My\sGroup\s(Copy) type=1
sgid=21
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupcopy(
...     ssgid=6, tsgid=0, name="My Group (Copy)", type_=1)
...
```

servergroupdel (**, sgid, force=False*)

Usage:

```
servergroupdel sgid={groupID} force={1|0}
```

Deletes the server group specified with sgid. If force is set to 1, the server group will be deleted even if there are clients within.

Example:

```
servergroupdel sgid=13
error id=0 msg=ok

servergroupdel sgid=14 force=1
error id=0 msg=ok
```

Examples:

```
>>> ts3cmd.servergroupdel(sgid=13)
...
>>> ts3cmd.servergroupdel(sgid=14, force=True)
...
```

servergroupdelclient (**, sgid, cldbid*)

Usage:

```
servergroupdelclient sgid={groupID} cldbid={clientDBID}
```

Removes a client from the server group specified with sgid.

Example:

```
servergroupdelclient sgid=16 cldbid=3
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupdelclient(sgid=16, cldbid=3)
...
```

servergroupdelperm (**, sgid, permid=None, permsid=None*)

Usage:

```
servergroupdelperm sgid={groupID} permid={permID}|permid={permID}
servergroupdelperm sgid={groupID} permsid={permName}
```

Removes a set of specified permissions from the server group specified with sgid. Multiple permissions can be removed at once. A permission can be specified by permid or permsid.

Examples:

```
servergroupdelperm sgid=16 permid=8470|permid=8475
error id=0 msg=ok

servergroupdelperm sgid=16 permsid=i_channel_join_power
error id=0 msg=ok
```

Examples:

```
>>> ts3cmd.servergroupdelperm(sgid=16, permid=8470)
...
>>> ts3cmd.servergroupdelperm(sgid=16, permid=8475)
...
>>> ts3cmd.servergroupdelperm(
...     sgid=16, permsid="i_channel_join_power")
...
```

servergrouplist()

Usage:

```
servergrouplist
```

Displays a list of server groups available. Depending on your permissions, the output may also contain global ServerQuery groups and template groups.

Example:

```
servergrouplist
sgid=9 name=Server\sAdmin type=1 iconid=300 savedb=1|sgid=10 name=Normal t...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergrouplist()
...
```

servergrouppermlist(*, *sgid*, *permsid=False*)

Usage:

```
servergrouppermlist sgid={groupID} [-permsid]
```

Displays a list of permissions assigned to the server group specified with sgid. The optional -permsid parameter can be used to get the permission names instead of their internal ID.

Example: servergrouppermlist sgid=13 permid=8470 permvalue=1 permnegated=0 permskip=0|permid=8475 permvalue=1|. . . error id=0 msg=ok

servergrouppermlist sgid=14 -permsid permsid=b_virtualserver_info_view permvalue=1 permnegated=0 permskip=0|. . . error id=0 msg=ok

Example:

```
>>> ts3cmd.servergrouppermlist(sgid=13)
...
>>> ts3cmd.servergrouppermlist(sgid=14, permsid=True)
...
```

servergrouprename(*, *sgid*, *name*)

Usage:

```
servergrouprename sgid={groupID} name={groupName}
```

Changes the name of the server group specified with sgid.

Example:

```
servergrouprename sgid=13 name=New\sName
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergrouprename(sgid=13, name="New Name")
...
```

servergroupsbyclientid(*, *cldbid*)

Usage:

```
servergroupsbyclientid cldbid={clientDBID}
```

Displays all server groups the client specified with cldbid is currently residing in.

Example:

```
servergroupsbyclientid cldbid=18
name=Server\sAdmin sgid=6 cldbid=18
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servergroupsbyclientid(cldbid=18)
...
```

serveridgetbyport(*, *virtualserver_port*)

Usage:

```
serveridgetbyport virtualserver_port={serverPort}
```

Displays the database ID of the virtual server running on the UDP port specified by virtualserver_port.

Example:

```
serveridgetbyport virtualserver_port=9987
server_id=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serveridgetbyport(virtualserver_port=9987)
...
```

serverinfo()

Usage:

```
serverinfo
```

Displays detailed configuration information about the selected virtual server including unique ID, number of clients online, configuration, etc.

Example:

```
serverinfo
virtualserver_port=9987 virtualserver_name=TeamSpeak\s]I[\sServer virtua...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serverinfo()
...
```

serverlist(*, *uid=False*, *all_=False*, *short=False*, *onlyoffline=False*)

Usage:

```
serverlist [-uid] [-all] [-short] [-onlyoffline]
```

Displays a list of virtual servers including their ID, status, number of clients online, etc. If you're using the -all option, the server will list all virtual servers stored in the database. This can be useful when multiple server instances with different machine IDs are using the same database. The machine ID is used to identify the server instance a virtual server is associated with.

The status of a virtual server can be either online, offline, booting up, shutting down and virtual online. While most of them are self-explanatory, virtual online is a bit more complicated. Whenever you select a virtual server which is currently stopped, it will be started in virtual mode which means you are able to change its configuration, create channels or change permissions, but no regular TeamSpeak 3 Client can connect. As soon as the last ServerQuery client deselects the virtual server, its status will be changed back to offline.

Example:

```
serverlist
virtualserver_id=1 virtualserver_port=9987 virtualserver_status=online
virtualserver_clientsonline=6...
error id=0 msg=ok
```

Examples:

```
>>> ts3cmd.serverlist()
...
```

servernotifyregister(*, *event*, *id_=None*)

Usage:

```
servernotifyregister [id={channelID}]
                     event={server|channel|textserver|textchannel|textprivate}
```

Registers for a specified category of events on a virtual server to receive notification messages. Depending on the notifications you've registered for, the server will send you a message on every event in the view of your ServerQuery client (e.g. clients joining your channel, incoming text messages, server configuration changes, etc). The event source is declared by the event parameter while id can be used to limit the notifications to a specific channel.

Example:

```
servernotifyregister event=server
error id=0 msg=ok

servernotifyregister event=channel id=123
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servernotifyregister(event="server")
...
>>> ts3cmd.servernotifyregister(event="channel", id_=123)
...
```

servernotifyunregister()

Usage:

```
servernotifyunregister
```

Unregisters all events previously registered with servernotifyregister so you will no longer receive notification messages.

Example:

```
servernotifyunregister
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servernotifyunregister()
...
```

serverprocessstop()

Usage:

```
serverprocessstop
```

Stops the entire TeamSpeak 3 Server instance by shutting down the process.

Example:

```
serverprocessstop
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serverprocessstop()
...
```

serverrequestconnectioninfo()

Usage:

```
serverrequestconnectioninfo
```

Displays detailed connection information about the selected virtual server including uptime, traffic information, etc.

Example:

```
serverrequestconnectioninfo
connection_filetransfer_bandwidth_sent=0 connection_packets_sent_total=0...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serverrequestconnectioninfo()
...
```

serversnapshotcreate()

Usage:

```
serversnapshotcreate
```

Displays a snapshot of the selected virtual server containing all settings, groups and known client identities. The data from a server snapshot can be used to restore a virtual servers configuration.

Example:

```
serversnapshotcreate
hash=bnTd2E1kNITHjJYRCFjgbKKO5P8=|virtualserver_name=TeamSpeak\sServer...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serversnapshotcreate()
...
```

serversnapshotdeploy(*, *virtualserver_snapshot*)

Usage:

```
serversnapshotdeploy {virtualserver_snapshot}
```

Restores the selected virtual servers configuration using the data from a previously created server snapshot. Please note that the TeamSpeak 3 Server does NOT check for necessary permissions while deploying a snapshot so the command could be abused to gain additional privileges.

Example:

```
serversnapshotdeploy hash=bnTd2E1kNITHjJYRCFjgbKKO5P8=|virtualserver_...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serversnapshotdeploy(
...     "hash=bnTd2E1kNITHjJYRCFjgbKKO5P8=|virtualserver_...")
...
```

serverstart(*, *sid*)

Usage:

```
serverstart sid={serverID}
```

Starts the virtual server specified with sid. Depending on your permissions, you're able to start either your own virtual server only or any virtual server in the server instance.

Example:

```
serverstart sid=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serverstart(sid=1)
...
```

serverstop(*, *sid*)

Usage:

```
serverstop sid={serverID}
```

Stops the virtual server specified with sid. Depending on your permissions, you're able to stop either your own virtual server only or all virtual servers in the server instance.

Example:

```
serverstop sid=1
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.serverstop(sid=1)
...
```

servertemppasswordadd(**, pw, desc, duration, tcid, tcpw*)
Usage:

```
servertemppasswordadd pw={password} desc={description}
                      duration={seconds} tcid={channelID}
                      tcpw={channelPW}
```

Sets a new temporary server password specified with pw. The temporary password will be valid for the number of seconds specified with duration. The client connecting with this password will automatically join the channel specified with tcid. If tcid is set to 0, the client will join the default channel.

Example:

```
servertemppasswordadd pw=secret desc=none duration=3600 tcid=117535 tcpw=123
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servertemppasswordadd(
...     pw="secret", desc="none", duration=3600, tcid=117535,
...     tcpw="123")
...
```

servertemppassworddel(**, pw*)
Usage:

```
servertemppassworddel pw={password}
```

Deletes the temporary server password specified with pw.

Example:

```
servertemppassworddel pw=secret
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servertemppassworddel(pw="secret")
...
```

servertemppasswordlist()
Usage:

```
servertemppasswordlist
```

Returns a list of active temporary server passwords. The output contains the clear-text password, the nickname and unique identifier of the creating client.

Example:

```
servertemppasswordlist
nickname=serveradmin uid=serveradmin desc=none pw_clear=secret
start=1331496494 end=1331500094 tcid=117535|nickname=serveradmin...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.servertemppasswordlist()
...
```

setclientchannelgroup(**, cgid, cid, cldbid*)

Usage:

```
setclientchannelgroup cgid={groupID} cid={channelID}
                      cldbid={clientDBID}
```

Sets the channel group of a client to the ID specified with cgid.

Example:

```
setclientchannelgroup cgid=13 cid=15 cldbid=20
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.setclientchannelgroup(cgid=13, cid=15, cldbid=20)
...
```

tokenadd(**, tokentype, tokenid1, tokenid2, tokendescription=None, tokencustomset=None*)

Usage:

```
tokenadd tokentype={1|0} tokenid1={groupID} tokenid2={channelID}
         [tokendescription={description}]
         [tokencustomset={customFieldSet}]
```

Create a new token. If tokentype is set to 0, the ID specified with tokenid1 will be a server group ID. Otherwise, tokenid1 is used as a channel group ID and you need to provide a valid channel ID using tokenid2.

The tokencustomset parameter allows you to specify a set of custom client properties. This feature can be used when generating tokens to combine a website account database with a TeamSpeak user. The syntax of the value needs to be escaped using the ServerQuery escape patterns and has to follow the general syntax of:

ident=ident1 value=value1|ident=ident2 value=value2|ident=ident3 value=value3

Example:

```
tokenadd tokentype=0 tokenid1=6 tokenid2=0 tokendescription=Test
 tokencustomset=ident=forum_user\svalue=ScP\pident=forum_id\svalue=123
token=eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.tokenadd(
...     tokentype=0, tokenid1=6, tokenid2=0,
...     tokendescription="Test",
```

(continues on next page)

(continued from previous page)

```
...     tokencustomset="ident=forum_user\svalue=ScP\pident=forum_id\svalue=123
↪"
...     )
...
```

tokendelete(*, *token*)

Usage:

```
tokendelete token={tokenKey}
```

Deletes an existing token matching the token key specified with token.

Example:

```
tokendelete token=eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.tokendelete(
...     token="eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW")
...
```

tokenlist()

Usage:

```
tokenlist
```

Displays a list of tokens available including their type and group IDs. Tokens can be used to gain access to specified server or channel groups.

A token is similar to a client with administrator privileges that adds you to a certain permission group, but without the necessity of a such a client with administrator privileges to actually exist. It is a long (random looking) string that can be used as a ticket into a specific server group.

Example:

```
tokenlist
token=88CVUg\/zkujt+y+WfHdko79UcM4R6uyCL6nEfy3B token_type=0 token_id1=9...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.tokenlist()
...
```

tokenuse(*, *token*)

Usage:

```
tokenuse token={tokenKey}
```

Use a token key gain access to a server or channel group. Please note that the server will automatically delete the token after it has been used.

Example:

```
tokenuse token=eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.tokenuse(
...     token="eKnFZQ9EK7G7MhtuQB6+N2B1PNZZ6OZL3ycDp2OW")
...
```

use(**, sid=None, port=None, virtual=False*)

Usage:

```
use [sid={serverID}] [port={serverPort}] [-virtual]
use {serverID}
```

Selects the virtual server specified with sid or port to allow further interaction. The ServerQuery client will appear on the virtual server and acts like a real TeamSpeak 3 Client, except it's unable to send or receive voice data.

If your database contains multiple virtual servers using the same UDP port, use will select a random virtual server using the specified port.

Examples:

```
use sid=1
error id=0 msg=ok

use port=9987
error id=0 msg=ok

use 1
error id=0 msg=ok
```

version()

Usage:

```
version
```

Displays the servers version information including platform and build number.

Example:

```
version
version=3.0.0-beta16 build=9929 platform=Linux
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.version()
...
```

whoami()

Usage:

```
whoami
```

Displays information about your current ServerQuery connection including the ID of the selected virtual server, your loginname, etc.

Example:

```
whoami
virtualserver_status=online virtualserver_id=1 client_channel_id=2 ...
error id=0 msg=ok
```

Example:

```
>>> ts3cmd.whoami()
...
```

1.2.5 definitions

This module contains the definitions described in the TeamSpeak 3 Server Manual, so that the variables can be used instead of the constans to improve the readability of the code.

class ts3.definitions.**HostMessageMode**
Bases: object

LOG = 1
display message in chatlog

MODAL = 2
display message in modal dialog

MODALQUIT = 3
display message in modal dialog and close connection

class ts3.definitions.**HostBannerMode**
Bases: object

NOADJUST = 0
do not adjust

IGNOREASPECT = 1
adjust but ignore aspect ratio (like TeamSpeak 2)

KEEPASPECT = 2
adjust and keep aspect ratio

class ts3.definitions.**Codec**
Bases: object

SPEEX_NARROWBAND = 0
speex narrowband (mono, 16bit, 8kHz)

SPEEX_WIDEBAND = 1
speex wideband (mono, 16bit, 16kHz)

SPEEX_ULTRAWIDEBAND = 2
speex ultra-wideband (mono, 16bit, 32kHz)

CELT_MONO = 3
celt mono (mono, 16bit, 48kHz)

class ts3.definitions.**CodecEncryptionMode**
Bases: object

INDIVIDUAL = 0
configure per channel

DISABLED = 1
globally disabled

ENABLED = 2
globally enabled

class ts3.definitions.**TextMessageTargetMode**
Bases: object

CLIENT = 1
target is a client

CHANNEL = 2
target is a channel

SERVER = 3
target is a virtual server

class ts3.definitions.**LogLevel**
Bases: object

ERROR = 1
everything that is really bad

WARNING = 2
everything that might be bad

DEBUG = 3
output that might help find a problem

INFO = 4
informational output

class ts3.definitions.**ReasonIdentifier**
Bases: object

KICK_CHANNEL = 4
kick client from channel

KICK_SERVER = 5
kick client from server

class ts3.definitions.**PermissionGroupDatabaseTypes**
Bases: object

Template = 0
template group (used for new virtual server)

Regular = 1
regular group (used for regular clients)

Query = 2
global query group (used for ServerQuery clients)

class ts3.definitions.**PermissionGroupTypes**
Bases: object

ServerGroup = 0
server group permission

GlobalClient = 1
client specific permission

Channel = 2
channel specific permission

ChannelGroup = 3
channel group permission

ChannelClient = 4
channel-client specific permission

class ts3.definitions.**TokenType**
Bases: object

ServerGroup = 0
server group token (id1={groupID} id2=0)

ChannelGroup = 1
channel group token (id1={groupID} id2={channelID})

1.2.6 query

This module contains a high-level API for the TeamSpeak 3 Server Query.

exception ts3.query.**TS3QueryError**(*resp*)
Bases: *ts3.common.TS3Error*

Raised, if the error code of the response was not 0.

resp = None
The TS3Response instance with the response data.

exception ts3.query.**TS3TimeoutError**
Bases: *ts3.common.TS3Error*, TimeoutError

Raised, if a response or event could not be received due to a *timeout*.

exception ts3.query.**TS3RecvError**
Bases: *ts3.common.TS3Error*

Raised if receiving data from the server failed, because the connection was closed or for other reasons.

class ts3.query.**TS3BaseConnection**(*host=None*, *port=10011*)
Bases: object

The TS3 query client.

This class provides only the methods to **handle** the connection to a TeamSpeak 3 Server. For a more convenient interface, use the *TS3Connection* class.

Note, that this class supports the with statement:

```
>>> with TS3BaseConnection() as ts3conn:
...     ts3conn.open("localhost")
...     ts3conn.send(...)
>>> # This is equal too:
>>> ts3conn = TS3BaseConnection()
>>> try:
...     ts3conn.open("localhost")
...     ts3conn.send(...)
... finally:
...     ts3conn.close()
```

Warning: This class is **not thread safe**!

telnet_conn

Getter If the client is connected, the used Telnet instance else None.

Type None or `telnetlib.Telnet`.

is_connected()

Returns True, if the client is currently connected.

Return type bool

open(*host*, *port=10011*, *timeout=<object object>*)

Connect to the TS3 Server listening on the address given by the *host* and *port* parmeters. If *timeout* is provided, this is the maximum time in seconds for the connection attempt.

Raises

- `OSError` – If the client is already connected.
- `TimeoutError` – If the connection can not be created.

close()

Sends the `quit` command and closes the telnet connection.

fileno()

Returns The fileno() of the socket object used internally.

Return type int

wait_for_event(*timeout=None*)

Blocks until an event is received or the *timeout* exceeds. The next received event is returned.

A simple event loop looks like this:

```
ts3conn.servernotifyregister(event="server")
while True:
    ts3conn.send_keepalive()
    try:
        event = ts3conn.wait_for_event(timeout=60)
    except TS3TimeoutError:
        pass
    else:
        # Handle the received event here ...
```

Parameters **timeout** (*None or float*) – The maximum number of seconds waited for the next event.

Return type *TS3Event*

Returns The next received ts3 event.

Raises

- *TS3TimeoutError* –
- *TS3RecvError* –

send_keepalive()

Sends an empty query to the server to prevent automatic disconnect. Make sure to call it at least once in 5 minutes (better each minute).

send(*command*, *common_parameters=None*, *unique_parameters=None*, *options=None*, *timeout=None*)
The general structure of a query command is:

```
<command> <options> <common parameters> <unique parameters>|<unique␣
↪parameters>|...
```

Examples are here worth a thousand words:

```
>>> # clientaddperm cldbid=16 permid=17276 permvalue=50␣
↪permskip=1|permid=21415 permvalue=20 permskip=0
>>> ts3conn.send(
...     command = "clientaddperm",
...     common_paramters = {"cldbid": 16},
...     parameterlist = [
...         {"permid": 17276, "permvalue": 50, "permskip": 1},
...         {"permid": 21415, "permvalue": 20, "permskip": 0}
...         ]
...     )
>>> # clientlist -uid -away
>>> ts3conn.send(
...     command = "clientlist",
...     options = ["uid", "away"]
...     )
```

See also:

`recv()`, `wait_for_resp()`

class ts3.query.**TS3Connection**(*host=None*, *port=10011*)
Bases: *ts3.query.TS3BaseConnection*, *ts3.commands.TS3Commands*

TS3 server query client.

This class provides the command wrapper capabilities `TS3Commands` and the ability to handle a connection to a TeamSpeak 3 server of *TS3BaseConnection*.

```
>>> with TS3Connection("localhost") as tsconn:
...     # From the TS3Commands class:
...     ts3conn.login("serveradmin", "FooBar")
...     ts3conn.clientkick(1)
```

quit()
Closes the connection.

1.2.7 filetransfer

This module contains an API for the TS3 file transfer interface.

exception ts3.filetransfer.**TS3FileTransferError**
Bases: *ts3.common.TS3Error*

This is the base class for all exceptions in this module.

exception ts3.filetransfer.**TS3UploadError**(*send_size*, *err=None*)
Bases: *ts3.filetransfer.TS3FileTransferError*

Is raised, when an upload fails.

send_size = None
The number of sent bytes till the error occured.

err = None
If the upload failed due to a thrown exception, this attribute contains it.

exception `ts3.filetransfer.`**TS3DownloadError**(*read_size*, *err=None*)
Bases: *ts3.filetransfer.TS3FileTransferError*

Is raised, when a download fails.

read_size = None
The number of read bytes untill the error occured.

err = None
If the download failed due to a thrown exception, this attribute contains the original exception.

class `ts3.filetransfer.`**TS3FileTransfer**(*ts3conn*)
Bases: `object`

A high-level TS3 file transfer handler.

The recommended methods to download or upload a file are:

- *init_download()*
- *init_upload()*

classmethod get_ftid()

Returns Returns a unique id for a file transfer.

Return type int

init_download(*output_file*, *name*, *cid*, *cpw=”*, *seekpos=0*, *query_resp_hook=None*, *reporthook=None*)
This is the recommended method to download a file from a TS3 server.

name, **cid**, **cpw** and **seekpos** are the parameters for the TS3 query command **ftinitdownload**. The parameter **clientftid** is automatically created and unique for the whole runtime of the programm.

query_resp_hook, if provided, is called, when the response of the ftinitdownload query has been received. Its single parameter is the the response of the querry.

For downloading the file from the server, *download()* is called. So take a look a this method for further information.

See also:

- `ftinitdownload()`
- `urlretrieve()`

classmethod download_by_resp(*output_file*, *ftinitdownload_resp*, *seekpos=0*, *reporthook=None*, *fallbackhost=None*)
This is *almost* a shortcut for:

```
>>> TS3FileTransfer.download(
...     output_file = file,
...     adr = (resp[0]["ip"], int(resp[0]["port"])),
...     ftkey = resp[0]["ftkey"],
...     seekpos = seekpos,
...     total_size = resp[0]["size"],
```

(continues on next page)

(continued from previous page)

```
...     reporthook = reporthook
...     )
```

Note, that the value of `resp[0]["ip"]` is a csv list and needs to be parsed.

classmethod download(*output_file*, *adr*, *ftkey*, *seekpos=0*, *total_size=0*, *reporthook=None*)

Downloads a file from a TS3 server in the file **output_file**. The TS3 file transfer interface is specified with the address tuple **adr** and the download with the file transfer key **ftkey**.

If **seekpos** and the total **size** are provided, the **reporthook** function (lambda read_size, block_size, total_size: None) is called after receiving a new block.

If you provide **seekpos** and **total_size**, this method will check, if the download is complete and raise a `TS3DownloadError` if not.

Note, that if **total_size** is 0 or less, each download will be considered as complete.

If no error is raised, the number of read bytes is returned.

Returns The number of received bytes.

Return type int

Raises ***TS3DownloadError*** – If the download is incomplete or a socket error occured.

init_upload(*input_file*, *name*, *cid*, *cpw=”*, *overwrite=True*, *resume=False*, *query_resp_hook=None*, *reporthook=None*)

This is the recommended method to upload a file to a TS3 server.

name, **cid**, **cpw**, **overwrite** and **resume** are the parameters for the TS3 query command **ftinitdownload**. The parameter **clientftid** is automatically created and unique for the whole runtime of the programm and the value of **size** is retrieved by the size of the **input_file**.

query_resp_hook, if provided, is called, when the response of the ftinitupload query has been received. Its single parameter is the the response of the querry.

For uploading the file to the server `upload()` is called. So take a look at this method for further information.

See also:

- `ftinitupload()`
- `urlretrieve()`

classmethod upload_by_resp(*input_file*, *ftinitupload_resp*, *reporthook=None*, *fallbackhost=None*)

This is *almost* a shortcut for:

```
>>> TS3FileTransfer.upload(
        input_file = file,
        adr = (resp[0]["ip"], int(resp[0]["port"])),
        ftkey = resp[0]["ftkey"],
        seekpos = resp[0]["seekpos"],
        reporthook = reporthook
        )
...
```

Note, that the value of `resp[0]["ip"]` is a csv list and needs to be parsed.

For the final upload, `upload()` is called.

classmethod upload(*input_file*, *adr*, *ftkey*, *seekpos=0*, *reporthook=None*)

Uploads the data in the file **input_file** to the TS3 server listening at the address **adr**. **ftkey** is used to authenticate the file transfer.

When the upload begins, the *get pointer* of the **input_file** is set to seekpos.

If the **reporthook** function (lambda send_size, block_size, total_size) is provided, it is called after sending a block to the server.

1.3 Examples

1.3.1 Endless poke

Download: `endless_poke.py`

```
#!/usr/bin/env python3

# The MIT License (MIT)
#
# Copyright (c) 2013-2018 <see AUTHORS.txt>
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
# this software and associated documentation files (the "Software"), to deal in
# the Software without restriction, including without limitation the rights to
# use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
# the Software, and to permit persons to whom the Software is furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
# COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
# IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


# Modules
# ------------------------------------------------
import time
import ts3


# Data
# ------------------------------------------------
__all__ = ["endless_poke"]


# Functions
# ------------------------------------------------
def endless_poke(ts3conn, nickname, msg=None, num=100, delay=1):
    """
    Pokes all clients where *nickname* matches *num* times with the message
    *msg*. Sleeping *delay* seconds between the single pokes. If *num* is -1,
```

(continues on next page)

(continued from previous page)

```
    the client is poked forever.
    """
    if msg is None:
        msg = "Stop annoying me!"

    # Get the client ids
    clients = ts3conn.clientfind(pattern=nickname)
    clients = [client["clid"] for client in clients]

    # Break, if there's no client.
    if not clients:
        return None

    # Poke them
    i = 0
    while num == -1 or i < num:
        for clid in clients:
            ts3conn.clientpoke(msg=msg, clid=clid)
        time.sleep(delay)
    return None


# Main
# ------------------------------------------------
if __name__ == "__main__":
    # USER, PASS, HOST, ...
    from def_param import *

    with ts3.query.TS3Connection(HOST, PORT) as ts3conn:
        ts3conn.login(client_login_name=USER, client_login_password=PASS)
        ts3conn.use(sid=SID)
        endless_poke(ts3conn, "Ben", delay=0.25)
```

1.3.2 Hello Bot

Download: `hello_bot.py`

```
#!/usr/bin/env python3

# The MIT License (MIT)
#
# Copyright (c) 2013-2018 <see AUTHORS.txt>
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
# this software and associated documentation files (the "Software"), to deal in
# the Software without restriction, including without limitation the rights to
# use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
# the Software, and to permit persons to whom the Software is furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
```

(continues on next page)

(continued from previous page)

```
# FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
# COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
# IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

import time
import ts3


__all__ = ["hello_bot"]


def hello_bot(ts3conn, msg=None):
    """
    Waits for new clients and says hello to them, when they join the server.
    """
    if msg is None:
        msg = "Hello :)"

    # Register for the event.
    ts3conn.servernotifyregister(event="server")

    while True:
        ts3conn.send_keepalive()

        try:
            # This method blocks, but we must sent the keepalive message at
            # least once in 5 minutes to avoid the sever side idle client
            # disconnect. So we set the timeout parameter simply to 1 minute.
            event = ts3conn.wait_for_event(timeout=60)
        except ts3.query.TS3TimeoutError:
            pass
        else:
            # Greet new clients.
            if event[0]["reasonid"] == "0":
                print("Client '{}' connected.".format(event[0]["client_nickname"]))
                ts3conn.clientpoke(clid=event[0]["clid"], msg=msg)
    return None


if __name__ == "__main__":
    # USER, PASS, HOST, ...
    from def_param import *

    with ts3.query.TS3Connection(HOST, PORT) as ts3conn:
        ts3conn.login(client_login_name=USER, client_login_password=PASS)
        ts3conn.use(sid=SID)
        hello_bot(ts3conn)
```

1.3.3 Viewer

Download: `viewer.py`

```
#!/usr/bin/env python3
```

(continues on next page)

(continued from previous page)

```
# The MIT License (MIT)
#
# Copyright (c) 2013-2018 <see AUTHORS.txt>
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
# this software and associated documentation files (the "Software"), to deal in
# the Software without restriction, including without limitation the rights to
# use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
# the Software, and to permit persons to whom the Software is furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
# COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
# IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


# Modules
# ------------------------------------------------
from pprint import pprint
import ts3


# Data
# ------------------------------------------------
__all__ = ["ChannelTreeNode",
           "view"]


# Classes
# ------------------------------------------------
class ChannelTreeNode(object):
    """
    Represents a channel or the virtual server in the channel tree of a virtual
    server. Note, that this is a recursive data structure.

    Common
    ------

    self.childs = List with the child *Channels*.

    self.root = The *Channel* object, that is the root of the whole channel
                tree.

    Channel
    -------

    Represents a real channel.

    self.info =  Dictionary with all informations about the channel obtained by
                 ts3conn.channelinfo
```

(continues on next page)

(continued from previous page)

```
    self.parent = The parent channel, represented by another *Channel* object.

    self.clients = List with dictionaries, that contains informations about the
                   clients in this channel.

    Root Channel
    ------------

    Represents the virtual server itself.

    self.info = Dictionary with all informations about the virtual server
                obtained by ts3conn.serverinfo

    self.parent = None

    self.clients = None

    Usage
    -----

    >>> tree = ChannelTreeNode.build_tree(ts3conn, sid=1)

    Todo
    ----

    * It's not sure, that the tree is always correct sorted.
    """

    def __init__(self, info, parent, root, clients=None):
        """
        Inits a new channel node.

        If root is None, root is set to *self*.
        """
        self.info = info
        self.childs = list()

        # Init a root channel
        if root is None:
            self.parent = None
            self.clients = None
            self.root = self

        # Init a real channel
        else:
            self.parent = parent
            self.root = root
            self.clients = clients if clients is not None else list()
        return None

    @classmethod
    def init_root(cls, info):
        """
        Creates a the root node of a channel tree.
        """
        return cls(info, None, None, None)
```

(continues on next page)

(continued from previous page)

```
    def is_root(self):
        """
        Returns true, if this node is the root of a channel tree (the virtual
        server).
        """
        return self.parent is None

    def is_channel(self):
        """
        Returns true, if this node represents a real channel.
        """
        return self.parent is not None

    @classmethod
    def build_tree(cls, ts3conn, sid):
        """
        Returns the channel tree from the virtual server identified with
        *sid*, using the *TS3Connection* ts3conn.
        """
        ts3conn.use(sid=sid, virtual=True)

        resp = ts3conn.serverinfo()
        serverinfo = resp.parsed[0]

        resp = ts3conn.channellist()
        channellist = resp.parsed

        resp = ts3conn.clientlist()
        clientlist = resp.parsed
        # channel id -> clients
        clientlist = {cid: [client for client in clientlist \
                            if client["cid"] == cid]
                      for cid in map(lambda e: e["cid"], channellist)}

        root = cls.init_root(serverinfo)
        for channel in channellist:
            resp = ts3conn.channelinfo(cid=channel["cid"])
            channelinfo = resp.parsed[0]
            # This makes sure, that *cid* is in the dictionary.
            channelinfo.update(channel)

            channel = cls(
                info=channelinfo, parent=root, root=root,
                clients=clientlist[channel["cid"]])
            root.insert(channel)
        return root

    def insert(self, channel):
        """
        Inserts the channel in the tree.
        """
        self.root._insert(channel)
        return None

    def _insert(self, channel):
        """
        Inserts the channel recursivly in the channel tree.
```

(continues on next page)

(continued from previous page)

```
        Returns true, if the tree has been inserted.
        """
        # We assumed on previous insertions, that a channel is a direct child
        # of the root, if we could not find the parent. Correct this, if ctree
        # is the parent from one of these orpheans.
        if self.is_root():
            i = 0
            while i < len(self.childs):
                child = self.childs[i]
                if channel.info["cid"] == child.info["pid"]:
                    channel.childs.append(child)
                    self.childs.pop(i)
                else:
                    i += 1

        # This is not the root and the channel is a direct child of this one.
        elif channel.info["pid"] == self.info["cid"]:
            self.childs.append(channel)
            return True

        # Try to insert the channel recursive.
        for child in self.childs:
            if child._insert(channel):
                return True

        # If we could not find a parent in the whole tree, assume, that the
        # channel is a child of the root.
        if self.is_root():
            self.childs.append(channel)
        return False

    def print(self, indent=0):
        """
        Prints the channel and it's subchannels recursive. If restore_order is
        true, the child channels will be sorted before printing them.
        """
        if self.is_root():
            print(" "*(indent*3) + "|-", self.info["virtualserver_name"])
        else:
            print(" "*(indent*3) + "|-", self.info["channel_name"])
            for client in self.clients:
                # Ignore query clients
                if client["client_type"] == "1":
                    continue
                print(" "*(indent*3+3) + "->", client["client_nickname"])

        for child in self.childs:
            child.print(indent=indent + 1)
        return None


def view(ts3conn, sid=1):
    """
    Prints the channel tree of the virtual server, including all clients.
    """
    tree = ChannelTreeNode.build_tree(ts3conn, sid)
    tree.print()
```

(continues on next page)

(continued from previous page)

```
    return None


# Main
# --------------------------------------------------
if __name__ == "__main__":
    # USER, PASS, HOST, ...
    from def_param import *

    with ts3.query.TS3Connection(HOST, PORT) as ts3conn:
        ts3conn.login(client_login_name=USER, client_login_password=PASS)
        view(ts3conn, sid=1)
```

1.3.4 Whirlpool

Download: whirlpool.py

```
#!/usr/bin/env python3

# The MIT License (MIT)
#
# Copyright (c) 2013-2018 <see AUTHORS.txt>
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
# this software and associated documentation files (the "Software"), to deal in
# the Software without restriction, including without limitation the rights to
# use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
# the Software, and to permit persons to whom the Software is furnished to do so,
# subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
# COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
# IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
# CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


# Modules
# --------------------------------------------------
import time
import random
import ts3
import ts3.definitions


# Data
# --------------------------------------------------
__all__ = ["whirlpool"]


# Functions
```

(continues on next page)

(continued from previous page)

```
# ------------------------------------------------
def whirlpool(ts3conn, duration=10, relax_time=0.5):
    """
    Moves all clients randomly in other channels for *duration* seconds.
    After the whirpool event, all clients will be in the same channel as
    before. Between the whirlpool cycles, the programm will sleep for
    *relax_time* seconds.
    """
    # Countdown till whirlpool
    for i in range(5, 0, -1):
        ts3conn.sendtextmessage(
            targetmode=ts3.definitions.TextMessageTargetMode.SERVER,
            target=0, msg="Whirpool in {}s".format(i))
        time.sleep(1)

    # Fetch the clientlist and the channellist.
    clientlist = ts3conn.clientlist()
    channellist = ts3conn.channellist()

    # Ignore query clients
    clientlist = [client for client in clientlist \
                  if client["client_type"] != "1"]

    # Whirpool with one channel or no users is boring.
    if len(channellist) == 1 or not clientlist:
        return None

    # We need this try-final construct to make sure, that all
    # clients will be in the same channel at the end of the
    # whirlpool as to the beginning.
    try:
        end_time = time.time() + duration
        while end_time > time.time():
            for client in clientlist:
                clid = client["clid"]
                cid = random.choice(channellist)["cid"]
                try:
                    ts3conn.clientmove(clid=clid, cid=cid)
                except ts3.query.TS3QueryError as err:
                    # Only ignore 'already member of channel error'
                    if err.resp.error["id"] != "770":
                        raise
            time.sleep(relax_time)
    finally:
        # Move all clients back
        for client in clientlist:
            try:
                ts3conn.clientmove(clid=client["clid"], cid=client["cid"])
            except ts3.query.TS3QueryError as err:
                if err.resp.error["id"] != "770":
                    raise
    return None


# Main
# ------------------------------------------------
if __name__ == "__main__":
```

(continues on next page)

(continued from previous page)

```
    # USER, PASS, HOST, ...
    from def_param import *

    with ts3.query.TS3Connection(HOST, PORT) as ts3conn:
        ts3conn.login(client_login_name=USER, client_login_password=PASS)
        ts3conn.use(sid=SID)
        whirlpool(ts3conn)
```

1.4 Changelog

- **1.0.7**

  fix for issue #49: https://github.com/benediktschmitt/py-ts3/issues/49 provided by @LouisChrist

- **1.0.4**

  - **added** fallbackhost parameter to some TS3FileTransfer methods
  - **fixed** UnicodeDecodeError caused by Android clients

    https://github.com/benediktschmitt/py-ts3/issues/34

- **1.0.0**

  All threads have been removed and the event handling has been reworked. Please take a look at the examples and the GitHub README for the new event queue.

  - **removed** *TS3ResponseRecvError*

    Use the *TS3TimeoutError* and *TS3RecvError* exceptions now.

  - **added** *TS3TimeoutError* exception
  - **added** *TS3RecvError* exception
  - **removed** *TS3BaseConnection.keepalive()*

    This method has been removed, because of the bad use of threads. You are now responsible to sent the *keepalive* message by calling *TS3BaseConnection.send_keepalive()* at least once in 10 minutes.

  - **added** *TS3BaseConnection.send_keepalive()*
  - **removed** *TS3BaseConnection.on_event()*

    use the new *TS3BaseConnection.wait_for_event()* now.

  - **removed** *TS3BaseConnection.wait_for_resp()*

    This method is an inplementation detail.

  - **removed** *TS3BaseConnection.stop_recv()*

    This method is no longer needed.

  - **removed** *TS3BaseConnection.recv_in_thread()*

    This method is no longer needed.

  - **removed** *TS3BaseConnection.last_resp*

1.5 Frequently Asked Questions (FAQ)

If you need help or you think you found a bug, please take also a look at the GitHub issue page. If you did not found a solution for your problem, do not hesitate to open a new issue with the corresponding label (e.g. `help wanted`, `bug`, . . .).

Please take also a look at *Contribute*.

1.5.1 Unexpected disconnects

anti-flood

Check the **anti-flood** settings of your TS3 server. Per default, the server limits the number of queries a host can send per minute. Take a look at the TS3 query manual to get to know how you can increase this limit or simply add the host, you are running the Python script from, to the query whitelist of your TS3 server:

```
$ # In your TS3 server folder:
$ echo "192.168.178.42" >> query_ip_whitelist.txt
```

max-idle-time

The ts3 server closes idle connections after 10 minutes automatically. You can use the `send_keepalive()` to sent an empty query to the server and thus avoid automatic disconnect. Make sure to call it at least once in 10 minutes.

1.6 Contribute

This project needs your help! Please help to improve this application and fork the repository on GitHub.

1.6.1 Bug reports

When you found a bug, please create a bug report on GitHub/Issues.

If you know how to fix the bug, you're welcome to send a *pull request*.

1.6.2 Code / Enhancements

If you want to contribute to the code or you have suggestions how we could improve the code, then tell me about it.

1.6.3 Spelling Mistakes

I guess this documentation and the source code contains a lot of spelling mistakes. Please help to reduce them.

1.7 License

Hint: Note, that the docstrings taken from the official TS3 Server Query Manual are the properties of the TeamSpeak System GmbH.

The PyTS3 package is licensed under the **MIT License**:

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

CHAPTER 3

What is PyTS3?

It’s a small package that contains a Python 3.2+ API for:

- TS3 query connections
- TS3 query events
- TS3 file transfers

Python Module Index

Index