

pyTSon

API Documentation

July 18, 2018

Contents

Contents	1
1 Module devtools	2
1.1 Functions	2
1.2 Class PluginInstaller	2
1.2.1 Methods	2
1.2.2 Properties	4
1.2.3 Class Variables	4
2 Module pluginhost	5
2.1 Functions	5
2.2 Variables	5
2.3 Class PluginHost	5
2.3.1 Methods	5
2.3.2 Properties	7
2.3.3 Class Variables	7
3 Module pylupdate	8
3.1 Functions	8
3.2 Variables	8
3.3 Class Message	9
3.3.1 Methods	9
3.3.2 Properties	11
3.4 Class Context	11
3.4.1 Methods	11
3.4.2 Properties	13
3.5 Class Translation	13
3.5.1 Methods	13
3.5.2 Properties	15
3.6 Class ParentVisitor	15
3.6.1 Methods	15
3.6.2 Properties	16
3.7 Class FunctionValidator	16
3.7.1 Methods	16
3.7.2 Properties	17
4 Module pythonqtpytson	18

4.1	Class EventFilterObject	18
4.1.1	Methods	18
5	Module pytson	20
5.1	Functions	20
5.2	Class Translatable	22
5.2.1	Methods	22
5.2.2	Properties	22
6	Package pytsonui	23
6.1	Modules	23
6.2	Functions	23
6.3	Class UiLoader	26
6.3.1	Methods	27
7	Module pytsonui.config	28
7.1	Class ConfigurationDialog	28
7.1.1	Methods	28
7.1.2	Properties	30
7.1.3	Class Variables	30
8	Module pytsonui.console	31
8.1	Functions	31
8.2	Class StdRedirector	31
8.2.1	Methods	31
8.3	Class PythonConsole	31
8.3.1	Methods	31
8.3.2	Properties	33
9	Module pytsonui.dialogs	34
9.1	Class MultiInputDialog	34
9.1.1	Methods	34
10	Module pytsonui.repository	35
10.1	Class RepositoryDialog	35
10.1.1	Methods	35
10.1.2	Properties	36
10.1.3	Class Variables	36
10.2	Class InstallDialog	37
10.2.1	Methods	37
10.2.2	Properties	37
11	Module signalslot	38
11.1	Class Signal	38
11.1.1	Methods	38
11.1.2	Properties	39
12	Module ts3client	40
12.1	Class Config	40
12.1.1	Methods	40
12.1.2	Properties	40
12.1.3	Class Variables	40
12.2	Class IconPack	41

12.2.1	Methods	41
12.2.2	Properties	43
12.3	Class ServerCache	43
12.3.1	Methods	44
12.4	Class CountryFlags	44
12.4.1	Methods	44
13	Module ts3defines	46
13.1	Variables	46
13.2	Class ACLType	57
13.2.1	Methods	57
13.2.2	Properties	57
13.2.3	Class Variables	57
13.3	Class AwayStatus	57
13.3.1	Methods	58
13.3.2	Properties	58
13.3.3	Class Variables	58
13.4	Class BBCodeTags	58
13.4.1	Methods	58
13.4.2	Properties	58
13.4.3	Class Variables	58
13.5	Class ChannelProperties	59
13.5.1	Methods	59
13.5.2	Properties	60
13.5.3	Class Variables	60
13.6	Class ChannelPropertiesRare	61
13.6.1	Methods	61
13.6.2	Properties	61
13.6.3	Class Variables	61
13.7	Class ClientCommand	62
13.7.1	Methods	62
13.7.2	Properties	62
13.7.3	Class Variables	62
13.8	Class ClientProperties	63
13.8.1	Methods	63
13.8.2	Properties	64
13.8.3	Class Variables	64
13.9	Class ClientPropertiesRare	65
13.9.1	Methods	65
13.9.2	Properties	65
13.9.3	Class Variables	65
13.10	Class ClientType	67
13.10.1	Methods	67
13.10.2	Properties	67
13.10.3	Class Variables	67
13.11	Class CodecEncryptionMode	68
13.11.1	Methods	68
13.11.2	Properties	68
13.11.3	Class Variables	68
13.12	Class CodecType	68
13.12.1	Methods	68
13.12.2	Properties	69

13.12.3 Class Variables	69
13.13 Class CommandLinePropertiesRare	69
13.13.1 Methods	69
13.13.2 Properties	69
13.13.3 Class Variables	70
13.14 Class ConnectionProperties	70
13.14.1 Methods	70
13.14.2 Properties	70
13.14.3 Class Variables	70
13.15 Class ConnectionPropertiesRare	74
13.15.1 Methods	74
13.15.2 Properties	74
13.15.3 Class Variables	74
13.16 Class ConnectStatus	75
13.16.1 Methods	75
13.16.2 Properties	75
13.16.3 Class Variables	75
13.17 Class FileListType	76
13.17.1 Methods	76
13.17.2 Properties	76
13.17.3 Class Variables	76
13.18 Class FileTransferState	76
13.18.1 Methods	77
13.18.2 Properties	77
13.18.3 Class Variables	77
13.19 Class FTAction	77
13.19.1 Methods	77
13.19.2 Properties	77
13.19.3 Class Variables	78
13.20 Class GroupShowNameTreeMode	78
13.20.1 Methods	78
13.20.2 Properties	78
13.20.3 Class Variables	78
13.21 Class GroupWhisperTargetMode	79
13.21.1 Methods	79
13.21.2 Properties	79
13.21.3 Class Variables	79
13.22 Class GroupWhisperType	80
13.22.1 Methods	80
13.22.2 Properties	80
13.22.3 Class Variables	80
13.23 Class HardwareInputStatus	81
13.23.1 Methods	81
13.23.2 Properties	81
13.23.3 Class Variables	81
13.24 Class HardwareOutputStatus	82
13.24.1 Methods	82
13.24.2 Properties	82
13.24.3 Class Variables	82
13.25 Class HostBannerMode	82
13.25.1 Methods	82

13.25.2 Properties	83
13.25.3 Class Variables	83
13.26 Class HostMessageMode	83
13.26.1 Methods	83
13.26.2 Properties	83
13.26.3 Class Variables	83
13.27 Class InputDeactivationStatus	84
13.27.1 Methods	84
13.27.2 Properties	84
13.27.3 Class Variables	84
13.28 Class LicenseViolationType	84
13.28.1 Methods	85
13.28.2 Properties	85
13.28.3 Class Variables	85
13.29 Class LocalTestMode	85
13.29.1 Methods	85
13.29.2 Properties	85
13.29.3 Class Variables	85
13.30 Class LogLevel	86
13.30.1 Methods	86
13.30.2 Properties	86
13.30.3 Class Variables	86
13.31 Class LogTypes	87
13.31.1 Methods	87
13.31.2 Properties	87
13.31.3 Class Variables	87
13.32 Class MonoSoundDestination	87
13.32.1 Methods	88
13.32.2 Properties	88
13.32.3 Class Variables	88
13.33 Class MuteInputStatus	88
13.33.1 Methods	88
13.33.2 Properties	88
13.33.3 Class Variables	89
13.34 Class MuteOutputStatus	89
13.34.1 Methods	89
13.34.2 Properties	89
13.34.3 Class Variables	89
13.35 Class PluginConfigureOffer	90
13.35.1 Methods	90
13.35.2 Properties	90
13.35.3 Class Variables	90
13.36 Class PluginConnectTab	90
13.36.1 Methods	90
13.36.2 Properties	91
13.36.3 Class Variables	91
13.37 Class PluginGuiProfile	91
13.37.1 Methods	91
13.37.2 Properties	91
13.37.3 Class Variables	91
13.38 Class PluginItemType	92

13.38.1 Methods	92
13.38.2 Properties	92
13.38.3 Class Variables	92
13.39 Class PluginMenuType	93
13.39.1 Methods	93
13.39.2 Properties	93
13.39.3 Class Variables	93
13.40 Class PluginMessageTarget	93
13.40.1 Methods	93
13.40.2 Properties	94
13.40.3 Class Variables	94
13.41 Class PluginTargetMode	94
13.41.1 Methods	94
13.41.2 Properties	94
13.41.3 Class Variables	94
13.42 Class ReasonIdentifier	95
13.42.1 Methods	95
13.42.2 Properties	95
13.42.3 Class Variables	95
13.43 Class SecuritySaltOptions	96
13.43.1 Methods	96
13.43.2 Properties	96
13.43.3 Class Variables	96
13.44 Class SERVER_BINDING	97
13.44.1 Methods	97
13.44.2 Properties	97
13.44.3 Class Variables	97
13.45 Class ServerInstancePropertiesRare	98
13.45.1 Methods	98
13.45.2 Properties	98
13.45.3 Class Variables	98
13.46 Class TalkStatus	99
13.46.1 Methods	99
13.46.2 Properties	99
13.46.3 Class Variables	100
13.47 Class TextMessageTargetMode	100
13.47.1 Methods	100
13.47.2 Properties	100
13.47.3 Class Variables	100
13.48 Class VirtualServerProperties	101
13.48.1 Methods	101
13.48.2 Properties	101
13.48.3 Class Variables	101
13.49 Class VirtualServerPropertiesRare	102
13.49.1 Methods	102
13.49.2 Properties	102
13.49.3 Class Variables	102
13.50 Class Visibility	107
13.50.1 Methods	107
13.50.2 Properties	107
13.50.3 Class Variables	107

14 Module ts3lib	108
14.1 Functions	108
15 Module ts3plugin	220
15.1 Class PluginMount	220
15.1.1 Methods	220
15.1.2 Properties	220
15.2 Class ts3plugin	221
15.2.1 Methods	221
15.2.2 Properties	271
15.2.3 Class Variables	271
16 Package ts3widgets	274
16.1 Modules	274
17 Module ts3widgets.filetransfer	275
17.1 Functions	275
17.2 Class File	275
17.2.1 Methods	276
17.2.2 Properties	276
17.3 Class FileListModel	276
17.3.1 Methods	277
17.3.2 Properties	278
17.4 Class SmartStatusBar	278
17.4.1 Methods	278
17.4.2 Class Variables	278
17.5 Class FileCollector	279
17.5.1 Methods	279
17.5.2 Properties	280
17.6 Class FileBrowser	280
17.6.1 Methods	281
17.6.2 Properties	283
17.7 Class FileCollisionAction	283
17.7.1 Methods	283
17.7.2 Properties	283
17.7.3 Class Variables	284
17.8 Class FileCollisionDialog	284
17.8.1 Methods	285
17.8.2 Properties	286
17.9 Class FileTransfer	286
17.9.1 Methods	286
17.9.2 Properties	287
17.10 Class Download	287
17.10.1 Methods	287
17.10.2 Properties	288
17.11 Class Upload	288
17.11.1 Methods	288
17.11.2 Properties	289
17.12 Class FileTransferModel	289
17.12.1 Methods	289
17.12.2 Properties	291
17.13 Class FileTransferDelegate	291

17.13.1 Methods	292
17.14 Class FileTransferDialog	292
17.14.1 Methods	292
17.14.2 Properties	294
18 Module ts3widgets.serverview	295
18.1 Class ServerViewRoles	295
18.1.1 Class Variables	295
18.2 Class Channel	295
18.2.1 Methods	295
18.2.2 Properties	297
18.3 Class Server	297
18.3.1 Methods	297
18.3.2 Properties	298
18.4 Class Client	298
18.4.1 Methods	298
18.4.2 Properties	300
18.5 Class ServerviewModel	300
18.5.1 Methods	300
18.6 Class ServerviewDelegate	302
18.6.1 Methods	302
18.7 Class Serverview	303
18.7.1 Methods	303

1 Module devtools

1.1 Functions

installedPackages()

Returns a list of installed packages (installed with pip).

Return Value

a list of dictionaries containing name, version, directory (dir) and dist-info directory (distdir)

(type=list[dict{str: str}])

removePackage(name, version)

Removes a package (installed with pip). Throws an exception if the package could not be found

Parameters

name: the name of the package

(type=str)

version: the version string of the package

(type=str)

1.2 Class PluginInstaller

object —
devtools.PluginInstaller

Class used to install new python plugins and its dependencies.

1.2.1 Methods

__init__(self, stdout=None)

x.__init__(...) initializes x; see help(type(x)) for signature

Parameters

stdout: A callable used as print function (takes str argument); defaults to None; if None stdout is used instead

(type=callable)

Overrides: object.__init__

createPlugin(*name*, *withfile*=True, *content*=None)

Creates the infrastructure for a new plugin.

Parameters

name: the name of the plugin
(*type*=str)

withfile: if True, the file `__init__.py` is created in the plugin directory, defaults to True
(*type*=bool)

content: content of `__init__.py`; defaults to None; if None, an empty plugin skeleton is written to the file (if *withfile* is True)
(*type*=str)

Return Value

the path to the `__init__.py` of the new created plugin
(*type*=str)

removePlugin(*name*)

Uninstall a plugin (delete all data in scripts directory).

Parameters

name: the name of the plugin
(*type*=str)

installPlugin(*self*, *addon*, *data*)

Installs a new plugin into the scripts directory.

Parameters

addon: json dict containing the plugin information
(*type*=dict)

data: either the content of a single python file as string or a file-like-object to a zipfile which will be extracted
(*type*=str or file-like)

installPackages(*self*, *deps*)

Installs packages from pypi.python.org into the include directory.

Parameters

deps: A list of package names
(*type*=list[str])

Return Value

True on success, False otherwise
(*type*=bool)

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,

`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

1.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

1.2.3 Class Variables

Name	Description
<code>PLUGIN_SKELETON</code>	Value: ...

2 Module pluginhost

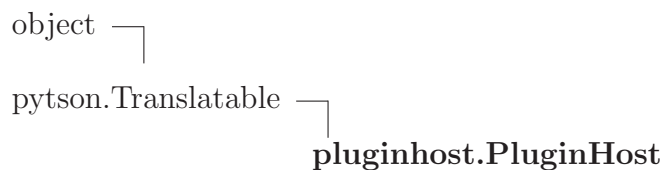
2.1 Functions

logprint(*msg*, *loglevel*, *channel*)

2.2 Variables

Name	Description
REL_URL	Value: <code>QUrl("https://api.github.com/repos/pathmann/pyTSon/releases")</code>

2.3 Class PluginHost



2.3.1 Methods

setupConfig(*cls*)

verboseLog(*cls*, *text*, *channel*)

init(*cls*)

setupTranslator(*cls*)

startPlugin(*cls*, *key*)

start(*cls*)

shutdown(*cls*)

activate(*cls*, *pname*)

```
deactivate(cls, pname)
```

```
reload(cls)
```

```
showScriptingConsole(cls)
```

```
scriptingConsoleDestroyed(cls)
```

```
configure(cls, mainwindow=None)
```

```
callMethod(cls, name, *args)
```

```
invokePlugins(cls, name, *args)
```

```
registerCallbackProxy(cls, obj)
```

```
unregisterCallbackProxy(cls, obj)
```

```
processCommand(cls, schid, command)
```

```
infoData(cls, schid, aid, atype)
```

```
onPluginCommandEvent(cls, schid, pluginName, pluginCommand)
```

```
parseUpdateReply(cls, repstr)
```

```
updateCheckFinished(cls, reply)
```

```
updateCheck(cls)
```

```
initMenus(cls)
```

```
globalMenuID(cls, plugin, localid)
```

```
initHotkeys(cls)
```

```
onMenuItemEvent(cls, schid, atype, menuItemID, selectedItemID)
```

```
globalHotkeyKeyword(cls, plugin, localkeyword)
```

onHotkeyEvent (<i>cls</i> , <i>keyword</i>)
--

showChangelog (<i>cls</i>)

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

2.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

2.3.3 Class Variables

Name	Description
<code>defaultConfig</code>	Value: [("general", [("differentApi", "False"), ("uninstallQuest...

3 Module pylupdate

pylupdate parses python files to search for calls to `pytson.tr` resp `pytson.Translatable._tr` to generate a Qt linguist translation file

These are the two valid usecases from module `pytson`:

```
class Translatable(object):
    @classmethod
    def _tr(cls, sourcetext, *, disambiguation="", n=-1, context=None):
        pass

def tr(context, sourcetext, *, disambiguation="", n=-1):
    pass
```

3.1 Functions

getSourceTexts(*inputfile*)

Extracts all translate function info from a python source file or Qt ui file

Parameters

inputfile: path to the python source file/ui file
(*type=*`str`)

Return Value

a list of tuples, containing the contextname and the message
(*type=*`list[tuple(str, Message)]`)

main(*argv*)

Main function of pylupdate

Parameters

argv: the arguments passed to the scripts
(*type=*`list[str]`)

3.2 Variables

Name	Description
DTD	Value: <code>r"""<...</code>

3.3 Class Message

object —
pylupdate.Message

Wrapper for a translated message

3.3.1 Methods

<code>__init__(self, sourcetext, disambiguation, nused, finished=False)</code>	
Instantiates a new Message object	
Parameters	
sourcetext:	sourcetext (<i>type=</i> str)
disambiguation:	string to distinguish between two equal sourcetexts (<i>type=</i> str or None)
nused:	if True, numerous translations will be used (<i>type=</i> bool)
finished:	defines, whether the translation is finished. defaults to False (<i>type=</i> bool)
Overrides: object.__init__	
<code>sourcetext(self, val)</code>	
<code>disambiguation(self, val)</code>	
<code>isNumerous(self, val)</code>	
<code>isFinished(self, val)</code>	

`__iter__(self)`

Yields the translations of the Message**Return Value**

the translated string(s)

(type=str)

`__repr__(self)`

repr(x)

Overrides: object.__repr__ exitit(inherited documentation)

`setTranslation(self, trans, numeroustrans=None)`

Sets the translation of the Message**Parameters****trans:** the translation*(type=str)***numeroustrans:** if not None, the numerous translation*(type=str or None)*

`update(self, msg)`

Updates the properties with the ones from another message**Parameters****msg:** the other message*(type=Message)*

`fromXml(elem)`

Parses the xml element message to a Message object**Parameters****elem:** the xml element*(type=ElementTree.Element)***Return Value**

a new created message object

(type=Message)

toXml(*self*)

Creates the xml elements of the message

Return Value

the xml element

(*type=ElementTree.Element*)

Inherited from *object*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,
`__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

3.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

3.4 Class Context

object └─ **pylupdate.Context**

Wrapper for a translation context

3.4.1 Methods**`__init__(self)`**

Instantiates a new Context object

Overrides: `object.__init__`

`__iter__(self)`

Yields each Message object in the context

Return Value

the message objects

(*type=Message*)

__repr__(*self*)

repr(x)

Overrides: object.__repr__ extit(inherited documentation)

addMessage(*self*, *msg*)

Adds a message to the context. If already exists, updates it.

Parameters

msg: the message

(*type=Message*)

update(*self*, *ctx*)

Updates all current messages with data from another context (if contained).

Parameters

ctx: the other context

(*type=Context*)

fromXml(*elem*)

Creates a context object from the xml element

Parameters

elem: the xml element

(*type=ElementTree.Element*)

Return Value

a tuple containing the name and the new context object

(*type=tuple(str, Context)*)

toXml(*self*, *name*)

Creates the xml elements of the context

Parameters

name: the name of the context

(*type=str*)

Return Value

the xml element

(*type=ElementTree.Element*)

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,
`__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

3.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

3.5 Class Translation

object └─
pylupdate.Translation

Wrapper for a Qt linguist translation file

3.5.1 Methods

<code>__init__(self, filename, language)</code>
Instantiates a new Translation object
Parameters
filename: path to read from resp. write to <i>(type=str)</i>
language: the target language code <i>(type=str)</i>
Overrides: <code>object.__init__</code>
<code>read(self, filename=None)</code>
Read the translation file given by the filename
Parameters
filename: if given, the path to the file to read. Defaults to None <i>(type=str)</i>

write(*self*, *language*, *dtd*=None, *filename*=None)

Writes the data to a file.

Parameters

language: the target language code

(*type*=str)

dtd: if set, the xml is validated before written, defaults to None, throws Exception if validation failed

(*type*=str)

filename: if given, the path to write to, defaults to None

(*type*=str)

__contains__(*self*, *key*)

Checks, if a context is contained

Parameters

key: the context name

(*type*=str)

Return Value

returns True, if a context is contained by key's name

(*type*=bool)

__getitem__(*self*, *key*)

Returns the context object references by its name

Parameters

key: the name of the context

(*type*=str)

Return Value

the context

(*type*=Context)

__iter__(*self*)

Yields each context name contained in the translation

Return Value

the context names

(*type*=str)

addContext(*self*, *key*)

Creates a new context. A previous context with that name will be overwritten

Parameters

key: the name of the context
(*type=**str*)

removeContext(*self*, *key*)

Remove a context.

Parameters

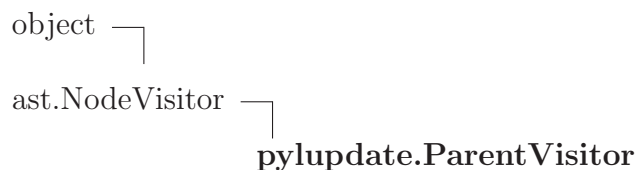
key: the name of the context
(*type=**str*)

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

3.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

3.6 Class ParentVisitor

ast Visitor which sets links to the parent node

3.6.1 Methods**generic_visit**(*self*, *node*)

Called if no explicit visitor function exists for a node.

Overrides: `ast.NodeVisitor.generic_visit` `exit`(inherited documentation)

Inherited from ast.NodeVisitor

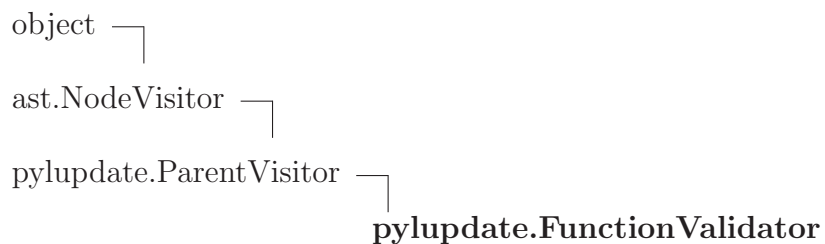
visit()

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __init__(), __new__(), __reduce__(),
 __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

3.6.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

3.7 Class FunctionValidator

ast Visitor to find calls to pyTSon's translation functions

3.7.1 Methods

__init__(self) x.__init__(...) initializes x; see help(type(x)) for signature Overrides: object.__init__ extit(inherited documentation)
visit_Import(self, node) Visits each import of the ast Parameters node: the import node (<i>type=ast.Import</i>)

visit_ImportFrom (<i>self</i> , <i>node</i>)

Visits each import (from) of the ast

Parameters

node: the import node

(<i>type</i> = <i>ast.ImportFrom</i>)

visit_Call (<i>self</i> , <i>node</i>)

Visits each function call of the ast

Parameters

node: the call node

(<i>type</i> = <i>ast.Call</i>)

Inherited from pylupdate.ParentVisitor(Section 3.6)

generic_visit()

Inherited from ast.NodeVisitor

visit()

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

3.7.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

4 Module `pythonqtpytson`

Module `PythonQt.pytson`

4.1 Class `EventFilterObject`

`PythonQt.QtCore.QObject` —
`pythonqtpytson.EventFilterObject`

Class to install as eventfilter on `QObject`-based objects. Connect to the signal `eventFiltered(QObject*, QEvent*)` to receive the filtered event per `installEventFilter`.

4.1.1 Methods

`__init__(self, typelist=list(), parent=None)`

Instantiates a new object.

Parameters

`typelist`: list of Eventtypes to filter (see `QEvent::Type`). Defaults to an empty list.
(type=list(int))

`parent`: `QObject`-parent
(type=QObject)

`setFilterResult(self, val)`

Sets the return value the object should return in the `eventFilter`-method. If not set, `False` will be returned

Parameters

`val`: the value
(type=bool)

`types(self)`

Returns the current list of eventtypes.

Return Value

the current eventtypes
(type=list(int))

addType(*self*, *eventtype*)

Adds an eventtype to the list.

Parameters

eventtype: an eventtype (see `QEvent::Type`)
(*type=int*)

removeType(*self*, *eventtype*)

Removes an eventtype from the list.

Parameters

eventtype: an eventtype (see `QEvent::Type`)
(*type=int*)

5 Module pytsn

5.1 Functions

tr(*context*, *sourcetext*, *disambiguation*="", *n*=-1)

Returns the current translation for a string. This function calls can be extracted by pyTSon's pylupdate.py.

Parameters

context:	context of the string literal, must be a raw string, not the return value of another function, an attribute or such (<i>type=</i> str)
sourcetext:	translatable string, must be a raw string, not the return value of another function, an attribute or such (<i>type=</i> str)
disambiguation:	used to distinguish between two equal sourcetexts in the same context, or as comment, optional, defaults to an empty string, must be a raw string, not the return value of another function, an attribute or such (<i>type=</i> str)
n:	used for strings containing plurals, optional, defaults to -1 (<i>type=</i> int)

locales()

Generator function to return all locale codes available for translation in format language_country (see ISO 639 and ISO 3166)

Return Value

Generator function to return the language codes
(*type=*Generator[str])

getConfigPath(*args)

Returns pyTson's configpath, that is, the subdirectory 'pyTson' in the TeamSpeak 3 config directory.

Parameters

args: path fields joined to the result as list of strings
(*type=list[str]*)

Return Value

The accumulated path
(*type=str*)

getPluginPath(*args)

Returns pyTson's pluginpath, that is, the subdirectory 'pyTson' in the TeamSpeak 3 plugins directory.

Parameters

args: path fields joined to the result as list of strings
(*type=list[str]*)

Return Value

The accumulated path
(*type=str*)

platformstr()

Returns the platform pyTson is currently running on.

Return Value

the platform (and architecture) string
(*type=str*)

getVersion()

Returns the current version of pyTson.

Return Value

the version as string
(*type=str*)

getCurrentApiVersion()

Returns the current apiversion of the ts3 plugin sdk. This should not be used in ts3plugin.apiVersion. Be fair and update your plugin manually!

Return Value

the apiVersion

(*type=*str)

5.2 Class Translatable

object —
 pytsn.Translatable

Baseclass for a class using translatable string literals.

5.2.1 Methods***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

5.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

6 Package pytsnui

6.1 Modules

- **config** (*Section 7, p. 28*)
- **console** (*Section 8, p. 31*)
- **dialogs** (*Section 9, p. 34*)
- **repository** (*Section 10, p. 35*)

6.2 Functions

ts3print(*msg, level, channel, aid*)

setIcon(*obj, iconpack, pluginicons=None*)

Sets the icon of a QWidget (if it has a property Icon) to an icon in the iconpack represented by a variable which is acquired by the property 'pytsnicon' of the object. If the property instead contains a string formatted as "octicons:filename.png", the icon is set to filename.png of the octicons pack. If no such property is available, nothing is done.

Parameters

- obj:** the widget
(*type=QWidget*)
- iconpack:** the iconpack
(*type=ts3client.IconPack*)
- pluginicons:** callable which gets a string and either returns the path to the image file or returns a QPixmap to set the icon property to; defaults to None
(*type=Callable(str) -> str or QIcon*)

connectSignalSlotsByName(*sender, receiver*)

Connects pythonqt signals by name
(receiver.on_<sender.objectname>_<signalname>)

Parameters

- sender:** the sender of signals
(*type=QObject*)
- receiver:** the receiver which has slots as callables defined
(*type=object*)

```
retrieveWidgets(obj, parent, widgets, seticons=True, iconpack=None,
pluginicons=None)
```

Retrieves widgets from a list and adds them as attribute to another object. If defined, signals from widgets are connected by name to methods in obj.

Parameters

- obj:** the object which will get the attributes added
(*type=object*)
- parent:** the toplevel widget
(*type=QWidget*)
- widgets:** a recursive (parent-relation of widgets) list of tuples, defining which widgets should be added as attributes to obj. The elements must be children of parent. First element of tuple must held the widget's objectname. If second element is True, the widget will be added as property (by objectname) to obj. Third element of the tuple are the child widgets, which should be handled by setupui
(*type=list[tuple(str, bool, list(...))]*)
- seticons:** if True, icons will be set according to the widgets 'pytsonicon' attribute
(*type=bool*)
- iconpack:** the iconpack
(*type=ts3client.IconPack*)
- pluginicons:** callable which gets a string and either returns the path to the image file or returns a QPixmap to set the icon property to; defaults to None
(*type=Callable(str) -> str or QIcon*)

```
retrieveAllWidgets(obj, parent, seticons=True, iconpack=None,  
pluginicons=None)
```

Retrieves all child widgets from a parent widget and adds them as attribute to another object. If defined, signals from widgets are connected by name to methods in obj.

Parameters

obj: the object which will get the attributes added
(*type*=object)

parent: the toplevel widget
(*type*=QWidget)

seticons: if True, icons will be set according to the widgets
'pytsonicon' attribute
(*type*=bool)

iconpack: the iconpack
(*type*=ts3client.IconPack)

pluginicons: callable which gets a string and either returns the
path to the image file or returns a QPixmap to set
the icon property to; defaults to None
(*type*=Callable(str) -> str or QIcon)


```
setupUi(obj, uipath, widgets=None, seticons=True, iconpack=None,
pluginicons=None, extraWidgets=None)
```

Loads a Qt designer file (.ui), creates the widgets defined in and adds them as property to a given object. This internally calls retrieveWidgets, so signals from widgets are connected by name to obj.

Parameters

- obj:** The object which will act as parent of the loaded ui (this object will receive a new layout)
(*type=QWidget*)
- uipath:** the path to the Qt designer file
(*type=str*)
- widgets:** optional argument; a recursive (parent-relation of widgets) list of tuples, defining which widgets should be added as attributes to obj. See retrieveWidgets for details. If you omit this or pass None, recursively all child widgets will be stored
(*type=list[tuple(str, bool, list(...))]* or *None*)
- seticons:** if True, widgets containing a string-property called 'pytsonicon' will get the icon of a soundpack (value of property = variable in soundpack)
(*type=bool*)
- iconpack:** if set, the iconpack will be used, if None, the current iconpack is used
(*type=ts3client.IconPack*)
- pluginicons:** callable which gets a string and either returns the path to the image file or returns a QPixmap to set the icon property to; defaults to None
(*type=Callable(str) -> str or QIcon*)
- extraWidgets:** list of extra classes to be created, there must be a constructor which takes only the parent object
(*type=list(QWidget type)*)

6.3 Class UiLoader

PythonQt.QtUiTools.QUiLoader —
pytsonui.UiLoader

QUILoader subclass to omit the parent widget from being recreated.

6.3.1 Methods

```
__init__(self, main, parent=None, extraWidgets=None)
```

Instantiate a new object

Parameters

main: parent class which will be omitted
(*type*=*QWidget*)

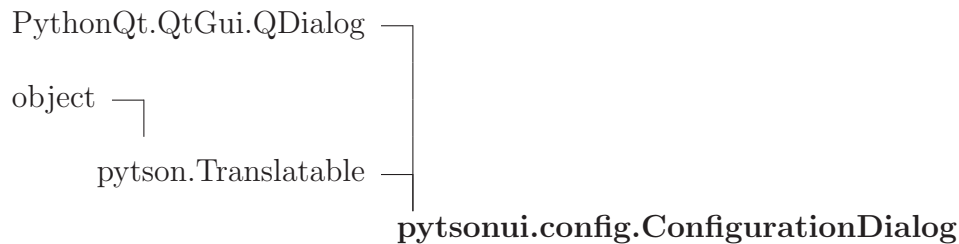
parent: parent class; defaults to None
(*type*=*QObject*)

extraWidgets: list of extra classes to be created, there must be a constructor which takes only the parent object
(*type*=*list(QWidget type)*)

```
createWidget(self, clsname, parent=None, name='')
```

7 Module `pytsonui.config`

7.1 Class `ConfigurationDialog`



7.1.1 Methods

```
__init__(self, cfg, host, parent=None)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

```
setupList(self)
```

```
setupValues(self)
```

```
setupSlots(self)
```

```
onLoadMenusButtonChanged(self, state)
```

```
onDifferentApiButtonChanged(self, state)
```

```
onPluginsTableCurrentItemChanged(self, currow, curcol, prevrow,  
prevcol)
```

```
onPluginsTableItemChanged(self, item)
```

```
onRemoveButtonClicked(self, pluginname)
```

```
onReloadButtonClicked(self)
```

```
onSettingsButtonClicked(self, pluginname)
```

```
onBackgroundColorButtonClicked(self)
```

```
onTextColorButtonClicked(self)
```

```
onFontFamilyComboChanged(self, font)
```

```
onFontSizeSpinChanged(self, size)
```

```
onTabcompleteButtonChanged(self, state)
```

```
onSpacesButtonChanged(self, state)
```

```
onTabwidthSpinChanged(self, width)
```

```
on_scriptButton_clicked(self)
```

```
on_scriptEdit_textEdited(self, text)
```

```
on_silentButton_toggled(self, act)
```

```
on_repositoryButton_clicked(self)
```

```
onLanguageComboCurrentIndexChanged(self, idx)
```

```
onLanguageButtonStateChanged(self, state)
```

```
onVerboseButtonStateChanged(self, state)
```

```
reloadSite(self)
```

```
on_siteTable_itemSelectionChanged(self)
```

```
on_siteaddButton_clicked(self)
```

```
on_siteremoveButton_clicked(self)
```

Inherited from object

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

7.1.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

7.1.3 Class Variables

Name	Description
CONF_WIDGETS	Value: [("tabWidget", False, ["pluginsTab", False, ["different...

8 Module `pytsonui.console`

8.1 Functions

```
defaultFont()
```

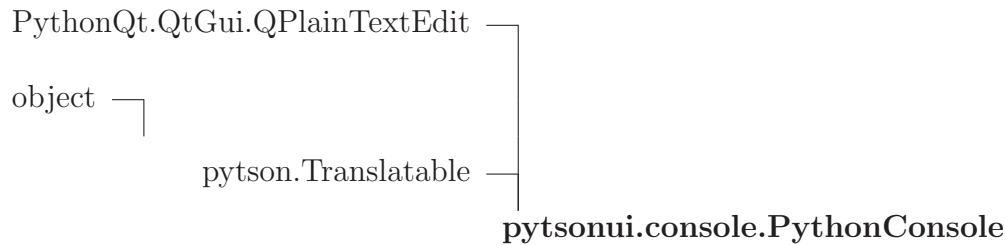
8.2 Class `StdRedirector`

8.2.1 Methods

```
__init__(self, callback)
```

```
write(self, text)
```

8.3 Class `PythonConsole`



8.3.1 Methods

```
__init__(self, tabcomplete=True, spaces=True, tabwidth=2,
font=defaultFont(), bgcolor=Qt.black, textcolor=Qt.white, width=800,
height=600, startup="", silentStartup=False, parent=None, catchstd=False)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
setFont(self, f)
```

```
prompt(self)
```

```
promptLength(self)
```

`writePrompt(self, newline)``promptCursor(self)``keyPressEvent(self, e)``mousePressEvent(self, e)``mouseReleaseEvent(self, e)``doKeyboardInterrupt(self)``doEndFile(self)``currentLine(self)``removeCurrentLine(self)``addHistory(self, cmd)``doHistoryUp(self)``doHistoryDown(self)``doTab(self)``doUntab(self)``appendLine(self, text)``runCommand(self, cmd, silent)``doExecuteCommand(self)`

Inherited from object

`__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()`

8.3.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

9 Module `pytsonui.dialogs`

9.1 Class `MultiInputDialog`

PythonQt.QtGui.QDialog —
`pytsonui.dialogs.MultiInputDialog`

9.1.1 Methods

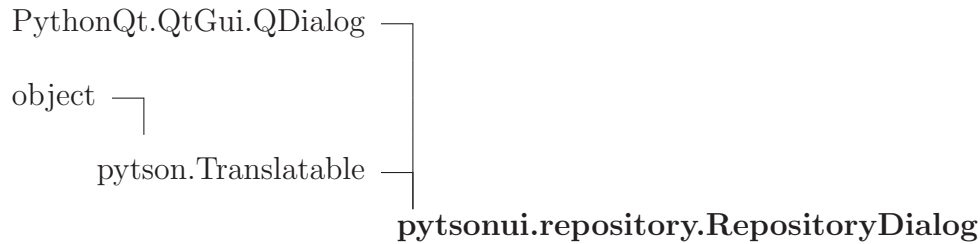
`__init__(self, title, label1, label2, parent=None)`

`cleanup(self)`

`getTexts(title, label1, label2, text1="", text2="", parent=None)`

10 Module *pytsonui.repository*

10.1 Class *RepositoryDialog*



10.1.1 Methods

```
__init__(self, host, parent=None)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ exitit(inherited documentation)
```

```
onClosed(self)
```

```
updatePendingButtons(self)
```

```
updateRepositories(self)
```

```
updateMaster(self)
```

```
handleMasterReply(self, reply)
```

```
updateAddons(self, repo, addons)
```

```
handleRepositoryReply(self, reply)
```

```
onNetworkReply(self, reply)
```

```
addRepository(self, r)
```

```
updateMasterlist(self)
```

<code>updateAddonlist(self)</code>

<code>on_updateButton_clicked(self)</code>
--

<code>on_addButton_clicked(self)</code>

<code>on_deleteButton_clicked(self)</code>
--

<code>on_repositoryList_doubleClicked(self, item)</code>
--

<code>on_repositoryList_currentItemChanged(self, cur, prev)</code>
--

<code>on_repositoryList_itemChanged(self, item)</code>
--

<code>on_pluginsList_currentItemChanged(self, cur, prev)</code>

<code>on_reloadButton_clicked(self)</code>
--

<code>on_installButton_clicked(self)</code>

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

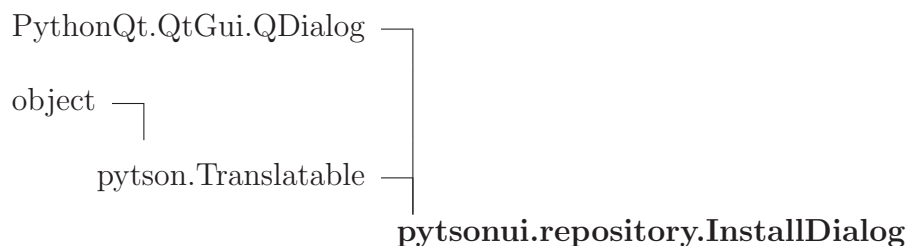
10.1.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

10.1.3 Class Variables

Name	Description
<code>master_url</code>	Value: <code>QUrl("https://raw.githubusercontent.com/pathmann/pyTSon.r..")</code>

10.2 Class *InstallDialog*



10.2.1 Methods

```
__init__(self, host, parent=None)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

```
install(self, addon)
```

```
installPackage(self, pkgstr)
```

```
onNetworkReply(self, reply)
```

```
on_closeButton_clicked(self)
```

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

10.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

11 Module signalslot

11.1 Class Signal



Simple class to emit signals to connected callable receivers.

11.1.1 Methods

`--init--(self)`

Instantiate a new object

Overrides: `object.--init--`

`connect(self, c)`

Connect a callable as receiver for the signal

Parameters

`c`: signal receiver

(*type=Callable*)

`disconnect(self, c)`

Disconnect the callable from receiving the signal

Parameters

`c`: signal receiver

(*type=Callable*)

`disconnectAll(self)`

Disconnects all signal receivers

`emit(self, *args, **kwargs)`

Fires the signal to all connected receivers

Inherited from object

`--delattr--()`, `--format--()`, `--getattrattribute--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

11.1.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

12 Module ts3client

12.1 Class Config



Offers an interface to query the TeamSpeak 3 client's config database (settings.db). You should always del a reference to this object if not needed anymore to assure the database connection is closed.

12.1.1 Methods

```
__init__(self)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
__del__(self)
```

```
__getattr__(self, name)
```

Inherited from object

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

12.1.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

12.1.3 Class Variables

Name	Description
objcount	Value: 0
instance	Value: None

12.2 Class IconPack



Offers an interface to the TeamSpeak 3 Client's iconpack. IconPack is also a context manager.

12.2.1 Methods

current()

Returns the current iconpack used (an Exception is raised if something failed).

Return Value

the iconpack

(*type=IconPack*)

__init__(self, info=None, name=None)

Instantiates a new IconPack object referenced by its name or the internal info string (an Exception is raised if the iconpack could not be located).

Parameters

info: the info string used in the settings.db

(*type=str*)

name: the name of the iconpack

(*type=str*)

Overrides: object.__init__

open(self)

Reads the settings for the iconpack and if it's zip-based, opens the file for reading. Must be called once before any icon can be accessed.

close(self)

If the iconpack is zip-based, the file is closed. After this is called, no icons can be accessed (till open is called again).

__enter__(self)

__exit__(self, type, value, traceback)

defaultName(*var*)

Returns the variable name used in the default iconpack.

Parameters

var: the variable used in an iconpack
(*type=**str*)

Return Value

the variable name
(*type=**str*)

fallback(*self*, *var*)

Returns the fallback icon for a variable according to the iconpack's settings.

Parameters

var: the variable name
(*type=**str*)

Return Value

the resulting pixmap
(*type=**QPixmap*)

icons(*self*)

Returns the list of variables used in the iconpack (excluding fallback mechanisms).

Return Value

a list of variable names
(*type=**list[str]*)

icon(*self*, *var*)

Returns the icon representing a variable used in the iconpack. If the icon cannot be found, the iconpack's fallback mechanisms are used. If everything fails, an empty pixmap is returned.

Parameters

var: the variable name
(*type=**str*)

Return Value

the resulting pixmap
(*type=**QPixmap*)

emoticons(*self*)

Returns the list of emoticon replacements used in the iconpack.

Return Value

a list of emoticon strings

(*type=list[str]*)

emoticon(*self*, *text*)

Returns the icon replacing the emoticon string.

Parameters

text: the emoticon as string

(*type=str*)

Return Value

the resulting pixmap

(*type=QPixmap*)

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

12.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

12.3 Class *ServerCache*

Offers an interface to the cached data of a TeamSpeak 3 server.

12.3.1 Methods

__init__(*self*, *schid*)

Instantiates a new *ServerCache* object referenced by the server connection handler id (an *Exception* is raised if the path in the filesystem could not be located).

Parameters

schid: the ID of the serverconnection
(*type=int*)

icon(*self*, *iconid*)

Returns an icon cached on disk.

Parameters

iconid: ID of the icon
(*type=int*)

Return Value

the icon
(*type=QPixmap*)

12.4 Class *CountryFlags*

Offers an interface to get the client's country flags. *CountryFlags* is also a context manager.

12.4.1 Methods

__init__(*self*)

Instantiates a new object. This will raise an exception, if the Zipfile could not be located.

open(*self*)

Opens the Zipfile for reading. This must be called before any flag is requested with *flag*.

close(*self*)

Closes the Zipfile.

flag(*self*, *code*)

Returns a QPixmap containing the flag of the given country code if exist.

Parameters

code: the country code
(*type=**str*)

Return Value

the flag
(*type=**QPixmap*)

__enter__(*self*)**__exit__**(*self*, *type*, *value*, *traceback*)

13 Module *ts3defines*

13.1 Variables

Name	Description
TS3Vector	Value: namedtuple('TS3Vecor', 'x y')
PLUGIN_MENU_BUFSZ	Value: 128
PLUGIN_HOTKEY_BUF-SZ	Value: 128
TS3_MAX_SIZE_CHAN-NEL_NAME	Value: 40
TS3_MAX_SIZE_VIRTUA-LSERVER_NAME	Value: 64
TS3_MAX_SIZE_CLIENT-NICKNAME	Value: 64
TS3_MIN_SIZE_CLIENT-NICKNAME	Value: 3
TS3_MAX_SIZE.REASO-N_MESSAGE	Value: 80
TS3_MAX_SIZE.TEXTM-ESSA	Value: 1024
TS3_MAX_SIZE.CHANN-EL_TOPIC	Value: 255
TS3_MAX_SIZE.CHANN-EL_DESCRIPTION	Value: 8192
TS3_MAX_SIZE_VIRTUA-LSERVER_WELCOMEM-ESSA	Value: 1024
TS3_MIN_SECONDS_CLI-ENTID_REUSE	Value: 300
MAX_VARIABLES.EXP-ORT_COUNT	Value: 64
BANDWIDTH.LIMIT_U-NLIMITED	Value: 0xFFFFFFFFFFFFFFFF
SPEAKER.FRONT_LEF-T	Value: 0x1
SPEAKER.FRONT_RIG-HT	Value: 0x2
SPEAKER.FRONT_CEN-TER	Value: 0x4
SPEAKER.LOW_FREQU-ENCY	Value: 0x8
SPEAKER.BACK.LEFT	Value: 0x10

continued on next page

Name	Description
SPEAKER_BACK_RIGHT	Value: 0x20
SPEAKER_FRONT_LEFT_OF_CENTER	Value: 0x40
SPEAKER_FRONT_RIGHT_OF_CENTER	Value: 0x80
SPEAKER_BACK_CENTER	Value: 0x100
SPEAKER_SIDE_LEFT	Value: 0x200
SPEAKER_SIDE_RIGHT	Value: 0x400
SPEAKER_TOP_CENTER	Value: 0x800
SPEAKER_TOP_FRONT_LEFT	Value: 0x1000
SPEAKER_TOP_FRONT_CENTER	Value: 0x2000
SPEAKER_TOP_FRONT_RIGHT	Value: 0x4000
SPEAKER_TOP_BACK_LEFT	Value: 0x8000
SPEAKER_TOP_BACK_CENTER	Value: 0x10000
SPEAKER_TOP_BACK_RIGHT	Value: 0x20000
SPEAKER_HEADPHONES_LEFT	Value: 0x10000000
SPEAKER_HEADPHONES_RIGHT	Value: 0x20000000
SPEAKER_MONO	Value: 0x40000000
TS3_MAX_SIZE_CLIENT_NICKNAME_NONSDK	Value: 30
TS3_MIN_SIZE_CLIENT_NICKNAME_NONSDK	Value: 3
TS3_MAX_SIZE_AWAY_MESSAGE	Value: 80
TS3_MAX_SIZE_GROUP_NAME	Value: 30
TS3_MAX_SIZE_TALK_REQUEST_MESSAGE	Value: 50
TS3_MAX_SIZE_COMPLAIN_MESSAGE	Value: 200
TS3_MAX_SIZE_CLIENT_DESCRIPTION	Value: 200

continued on next page

Name	Description
TS3_MAX_SIZE_HOST_MESSAGE	Value: 200
TS3_MAX_SIZE_HOSTBUTTON_TOOLTIP	Value: 50
TS3_MAX_SIZE_POKE_MESSAGE	Value: 100
TS3_MAX_SIZE_OFFLINE_MESSAGE	Value: 4096
TS3_MAX_SIZE_OFFLINE_MESSAGE_SUBJECT	Value: 200
TS3_MAX_SIZE_PLUGIN_COMMAND	Value: 1024* 8
TS3_MAX_SIZE_VIRTUALSERVER_HOSTBANNE-R_GFX_URL	Value: 2000
ERROR_accounting_already_started	Value: 0x0b0c
ERROR_accounting_instance_check_error	Value: 0x0b08
ERROR_accounting_instance_duplicated	Value: 0x0b0b
ERROR_accounting_instance_limit_reached	Value: 0x0b07
ERROR_accounting_license_date_not_ok	Value: 0x0b03
ERROR_accounting_license_file_invalid	Value: 0x0b09
ERROR_accounting_license_file_not_found	Value: 0x0b02
ERROR_accounting_not_started	Value: 0x0b0d
ERROR_accounting_running_elsewhere	Value: 0x0b0a
ERROR_accounting_server_error	Value: 0x0b06
ERROR_accounting_slot_limit_reached	Value: 0x0b01
ERROR_accounting_to_many_starts	Value: 0x0b0e
ERROR_accounting_unable_to_connect_to_server	Value: 0x0b04

continued on next page

Name	Description
ERROR_accounting_unkn-own_error	Value: 0x0b05
ERROR_accounting_virtu-alserver_limit_reached	Value: 0x0b00
ERROR_ban_flooding	Value: 0x0d03
ERROR_ban_invalid_id	Value: 0x0d00
ERROR_channel_already_i-n	Value: 0x0302
ERROR_channel_can_not_-delete_default	Value: 0x0305
ERROR_channel_default_r-require_permanent	Value: 0x0306
ERROR_channel_invalid_fl-ags	Value: 0x0307
ERROR_channel_invalid_i-d	Value: 0x0300
ERROR_channel_invalid_o-rder	Value: 0x030b
ERROR_channel_invalid_p-assword	Value: 0x030d
ERROR_channel_invalid_s-ecurity_hash	Value: 0x030f
ERROR_channel_is_privat-e_channel	Value: 0x030e
ERROR_channel_maxclien-ts_reached	Value: 0x0309
ERROR_channel_maxfami-ly_reached	Value: 0x030a
ERROR_channel_name_in-use	Value: 0x0303
ERROR_channel_no_filetr-ansfer_supported	Value: 0x030c
ERROR_channel_not_emp-ty	Value: 0x0304
ERROR_channel_parent_n-ot_permanent	Value: 0x0308
ERROR_channel_protocol_-limit_reached	Value: 0x0301
ERROR_client_already_su-bscribed	Value: 0x0205
ERROR_client_cannot_ver-ify_now	Value: 0x020e

continued on next page

Name	Description
ERROR_client_could_not_validate_identity	Value: 0x0207
ERROR_client_hacked	Value: 0x020d
ERROR_client_invalid_id	Value: 0x0200
ERROR_client_invalid_password	Value: 0x0208
ERROR_client_invalid_type	Value: 0x0204
ERROR_client_is_flooding	Value: 0x020c
ERROR_client_is_online	Value: 0x020b
ERROR_client_login_not_permitted	Value: 0x020f
ERROR_client_nickname_inuse	Value: 0x0201
ERROR_client_not_logged_in	Value: 0x0206
ERROR_client_not_subscribed	Value: 0x0210
ERROR_client_protocol_limit_reached	Value: 0x0203
ERROR_client_too_many_clones_connected	Value: 0x0209
ERROR_client_version_outdated	Value: 0x020a
ERROR_clientlibrary_not_initialised	Value: 0x0709
ERROR_command_not_found	Value: 0x0100
ERROR_connect_failed_banned	Value: 0x0d01
ERROR_connection_ip_protocol_missing	Value: 0x070d
ERROR_connection_lost	Value: 0x0701
ERROR_could_not_initialise_input_manager	Value: 0x0708
ERROR_could_not_resolve_hostname	Value: 0x0706
ERROR_currently_not_possible	Value: 0x0704
ERROR_database	Value: 0x0500
ERROR_database_constraint	Value: 0x0504

continued on next page

Name	Description
ERROR_database_duplicate_entry	Value: 0x0502
ERROR_database_empty_result	Value: 0x0501
ERROR_database_no_modifications	Value: 0x0503
ERROR_database_reinvoke	Value: 0x0505
ERROR_dont_notify	Value: 0x0004
ERROR_failed_connection_initialisation	Value: 0x0705
ERROR_file_already_exists	Value: 0x0802
ERROR_file_already_in_use	Value: 0x080a
ERROR_file_connection_lost	Value: 0x080f
ERROR_file_could_not_open_connection	Value: 0x080b
ERROR_file_exceeds_file_system_maximum_size	Value: 0x080d
ERROR_file_exceeds_supplied_size	Value: 0x0810
ERROR_file_invalid_name	Value: 0x0800
ERROR_file_invalid_path	Value: 0x0806
ERROR_file_invalid_permissions	Value: 0x0801
ERROR_file_invalid_size	Value: 0x0809
ERROR_file_invalid_transfer_id	Value: 0x0805
ERROR_file_io_error	Value: 0x0804
ERROR_file_no_files_available	Value: 0x0807
ERROR_file_no_space_left_on_device	Value: 0x080c
ERROR_file_not_found	Value: 0x0803
ERROR_file_overwrite_excludes_resume	Value: 0x0808
ERROR_file_transfer_cancelled	Value: 0x0812
ERROR_file_transfer_client_quota_exceeded	Value: 0x0815

continued on next page

Name	Description
ERROR_file_transfer_complete	Value: 0x0811
ERROR_file_transfer_connection_timeout	Value: 0x080e
ERROR_file_transfer_interrupted	Value: 0x0813
ERROR_file_transfer_limit_reached	Value: 0x0817
ERROR_file_transfer_reset	Value: 0x0816
ERROR_file_transfer_server_quota_exceeded	Value: 0x0814
ERROR_invalid_server_connection_handler_id	Value: 0x0707
ERROR_lib_time_limit_reached	Value: 0x0005
ERROR_message_invalid_id	Value: 0x0c00
ERROR_no_cached_connection_info	Value: 0x0703
ERROR_no_network_port_available	Value: 0x0102
ERROR_not_connected	Value: 0x0702
ERROR_not_implemented	Value: 0x0002
ERROR_ok	Value: 0x0000
ERROR_ok_no_update	Value: 0x0003
ERROR_parameter_checksum	Value: 0x0607
ERROR_parameter_convert	Value: 0x0604
ERROR_parameter_invalid	Value: 0x0602
ERROR_parameter_invalid_count	Value: 0x0601
ERROR_parameter_invalid_size	Value: 0x0605
ERROR_parameter_missing	Value: 0x0606
ERROR_parameter_not_found	Value: 0x0603
ERROR_parameter_quote	Value: 0x0600
ERROR_permission_default_group_forbidden	Value: 0x0a04

continued on next page

Name	Description
ERROR_permission_duplicate_entry	Value: 0x0a01
ERROR_permission_empty_result	Value: 0x0a03
ERROR_permission_invalid_group_id	Value: 0x0a00
ERROR_permission_invalid_perm_id	Value: 0x0a02
ERROR_permission_invalid_size	Value: 0x0a05
ERROR_permission_invalid_value	Value: 0x0a06
ERROR_permission_template_group_is_used	Value: 0x0a0b
ERROR_permissions	Value: 0x0a0c
ERROR_permissions_client_insufficient	Value: 0x0a08
ERROR_permissions_group_not_empty	Value: 0x0a07
ERROR_permissions_insufficient_group_power	Value: 0x0a09
ERROR_permissions_insufficient_permission_power	Value: 0x0a0a
ERROR_port_already_in_use	Value: 0x0103
ERROR_privilege_key_invalid	Value: 0x0f00
ERROR_provisioning_already_connected	Value: 0x1115
ERROR_provisioning_already_initialized	Value: 0x1112
ERROR_provisioning_auth_data_too_large	Value: 0x1111
ERROR_provisioning_auth_server_not_connected	Value: 0x1110
ERROR_provisioning_connecting	Value: 0x1114
ERROR_provisioning_could_not_connect	Value: 0x1109
ERROR_provisioning_internal_tries_exceeded	Value: 0x1106
ERROR_provisioning_invalid_password	Value: 0x1100

continued on next page

Name	Description
ERROR_provisioning_invalid_request	Value: 0x1101
ERROR_provisioning_invalid_timeout	Value: 0x1118
ERROR_provisioning_io_error	Value: 0x1117
ERROR_provisioning_no_permission	Value: 0x111A
ERROR_provisioning_no_slots_available	Value: 0x1102
ERROR_provisioning_not_connected	Value: 0x1116
ERROR_provisioning_not_initialized	Value: 0x1113
ERROR_provisioning_pool_missing	Value: 0x1103
ERROR_provisioning_pool_unknown	Value: 0x1104
ERROR_provisioning_too_many_reserved	Value: 0x1108
ERROR_provisioning_too_many_slots_requested	Value: 0x1107
ERROR_provisioning_ts3server_not_found	Value: 0x1119
ERROR_provisioning_unknown_ip_location	Value: 0x1105
ERROR_rename_failed_banned	Value: 0x0d02
ERROR_server_deployement_active	Value: 0x0405
ERROR_server_duplicate_running	Value: 0x040e
ERROR_server_invalid_id	Value: 0x0400
ERROR_server_invalid_password	Value: 0x0404
ERROR_server_is_booting	Value: 0x040a
ERROR_server_is_not_running	Value: 0x0409
ERROR_server_is_shutting_down	Value: 0x0402
ERROR_server_is_virtual	Value: 0x0407

continued on next page

Name	Description
ERROR_server_maxclients_reached	Value: 0x0403
ERROR_server_modal_quit	Value: 0x040c
ERROR_server_running	Value: 0x0401
ERROR_server_status_invalid	Value: 0x040b
ERROR_server_unable_to_stop_own_server	Value: 0x0406
ERROR_server_version_outdated	Value: 0x040d
ERROR_server_wrong_machineid	Value: 0x0408
ERROR_serverlibrary_not_initialised	Value: 0x070a
ERROR_sound_channel_mask_mismatch	Value: 0x0918
ERROR_sound_could_not_open_capture_device	Value: 0x0906
ERROR_sound_could_not_open_playback_device	Value: 0x0907
ERROR_sound_device_already_registered	Value: 0x0910
ERROR_sound_device_busy	Value: 0x0916
ERROR_sound_device_in_use	Value: 0x090f
ERROR_sound_handler_has_device	Value: 0x0908
ERROR_sound_internal_capture	Value: 0x090e
ERROR_sound_internal_encoder	Value: 0x0902
ERROR_sound_internal_playback	Value: 0x0903
ERROR_sound_internal_preprocessor	Value: 0x0901
ERROR_sound_invalid_capture_device	Value: 0x0909
ERROR_sound_invalid_channel_count	Value: 0x0913
ERROR_sound_invalid_playback_device	Value: 0x090a

continued on next page

Name	Description
ERROR_sound_invalid_wave	Value: 0x090b
ERROR_sound_need_more_data	Value: 0x0915
ERROR_sound_no_capture_device_available	Value: 0x0904
ERROR_sound_no_data	Value: 0x0917
ERROR_sound_no_playback_device_available	Value: 0x0905
ERROR_sound_open_wave	Value: 0x090d
ERROR_sound_preprocessor_disabled	Value: 0x0900
ERROR_sound_read_wave	Value: 0x0914
ERROR_sound_unknown_device	Value: 0x0911
ERROR_sound_unsupported_frequency	Value: 0x0912
ERROR_sound_unsupported_wave	Value: 0x090c
ERROR_tts_unable_to_initialize	Value: 0x0e00
ERROR_unable_to_bind_network_port	Value: 0x0101
ERROR_undefined	Value: 0x0001
ERROR_voip_already_initialized	Value: 0x1001
ERROR_voip_cannot_answer_initiated_call	Value: 0x1006
ERROR_voip_internal_error	Value: 0x1004
ERROR_voip_invalid_account	Value: 0x1003
ERROR_voip_invalid_connectionId	Value: 0x1005
ERROR_voip_not_initialized	Value: 0x1007
ERROR_voip_pjsua	Value: 0x1000
ERROR_voip_too_many_accounts	Value: 0x1002
ERROR_vs_critical	Value: 0x0700
ERROR_whisper_no_targets	Value: 0x070c

continued on next page

Name	Description
ERROR_whisper_too_many_targets	Value: 0x070b

13.2 Class ACLType

object —
ts3defines.ACLType

13.2.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.2.3 Class Variables

Name	Description
ACL_NONE	Value: 0
ACL_WHITE_LIST	Value: 1
ACL_BLACK_LIST	Value: 2

13.3 Class AwayStatus

object —
ts3defines.AwayStatus

13.3.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.3.3 Class Variables

Name	Description
<code>AWAY_NONE</code>	Value: 0
<code>AWAY_ZZZ</code>	Value: 1

13.4 Class BBCodeTags



13.4.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.4.3 Class Variables

Name	Description
BBCodeTag_B	Value: 0x00000001
BBCodeTag_I	Value: 0x00000002
BBCodeTag_U	Value: 0x00000004
BBCodeTag_S	Value: 0x00000008
BBCodeTag_SUP	Value: 0x00000010
BBCodeTag_SUB	Value: 0x00000020
BBCodeTag_COLOR	Value: 0x00000040
BBCodeTag_SIZE	Value: 0x00000080
BBCodeTag_group_text	Value: 0x000000FF
BBCodeTag_LEFT	Value: 0x00001000
BBCodeTag_RIGHT	Value: 0x00002000
BBCodeTag_CENTER	Value: 0x00004000
BBCodeTag_group_align	Value: 0x00007000
BBCodeTag_URL	Value: 0x00010000
BBCodeTag_IMAGE	Value: 0x00020000
BBCodeTag_HR	Value: 0x00040000
BBCodeTag_LIST	Value: 0x00100000
BBCodeTag_LISTITEM	Value: 0x00200000
BBCodeTag_group_list	Value: 0x00300000
BBCodeTag_TABLE	Value: 0x00400000
BBCodeTag_TR	Value: 0x00800000
BBCodeTag_TH	Value: 0x01000000
BBCodeTag_TD	Value: 0x02000000
BBCodeTag_group_table	Value: 0x03C00000
BBCodeTag_def_simple	Value: BBCodeTag_B BBCodeTag_I BBCodeTag_U BBCodeTag_S B...
BBCodeTag_def_simple.I- mg	Value: BBCodeTag_def_simple BBCodeTag_IMAGE
BBCodeTag_def_extended	Value: BBCodeTag_group_text BBCodeTag_group_align BBCodeTag_...

13.5 Class ChannelProperties

object —
ts3defines.ChannelProperties

13.5.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.5.3 Class Variables

Name	Description
CHANNEL_NAME	Value: 0
CHANNEL_TOPIC	Value: 1
CHANNEL_DESCRIPTION	Value: 2
CHANNEL_PASSWORD	Value: 3
CHANNEL_CODEC	Value: 4
CHANNEL_CODEC_QUALITY	Value: 5
CHANNEL_MAXCLIENTS	Value: 6
CHANNEL_MAXFAMILYCLIENTS	Value: 7
CHANNEL_ORDER	Value: 8
CHANNEL_FLAG_PERMANENT	Value: 9
CHANNEL_FLAG_SEMI_PERMANENT	Value: 10
CHANNEL_FLAG_DEFAULT	Value: 11
CHANNEL_FLAG_PASSWORD	Value: 12
CHANNEL_CODEC_LATENCY_FACTOR	Value: 13
CHANNEL_CODEC_IS_UNENCRYPTED	Value: 14
CHANNEL_SECURITY_SALT	Value: 15
CHANNEL_DELETE_DELAY	Value: 16

continued on next page

Name	Description
CHANNEL_ENDMARKE- R	Value: 17

13.6 Class ChannelPropertiesRare

object └─
 ts3defines.ChannelPropertiesRare

13.6.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.6.3 Class Variables

Name	Description
CHANNEL_DUMMY_2	Value: 17
CHANNEL_DUMMY_3	Value: 18
CHANNEL_DUMMY_4	Value: 19
CHANNEL_DUMMY_5	Value: 20
CHANNEL_DUMMY_6	Value: 21
CHANNEL_DUMMY_7	Value: 22
CHANNEL_FLAG_MAX- CLIENTS_UNLIMITED	Value: 23
CHANNEL_FLAG_MAXF- AMILYCLIENTS_UNLIM- ITED	Value: 24
CHANNEL_FLAG_MAXF- AMILYCLIENTS_INHER- ITED	Value: 25

continued on next page

Name	Description
CHANNEL_FLAG_ARE_SUBSCRIBED	Value: 26
CHANNEL_FILEPATH	Value: 27
CHANNEL_NEEDED_TALK_POWER	Value: 28
CHANNEL_FORCED_SILENCE	Value: 29
CHANNEL_NAME_PHONETIC	Value: 30
CHANNEL_ICON_ID	Value: 31
CHANNEL_FLAG_PRIVATE	Value: 32
CHANNEL_ENDMARKER_RARE	Value: 33

13.7 Class *ClientCommand*



13.7.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.7.3 Class Variables

Name	Description
<code>CLIENT_COMMAND_requestConnectionInfo</code>	Value: 0

continued on next page

Name	Description
CLIENT_COMMAND_requestClientMove	Value: 1
CLIENT_COMMAND_requestXXMuteClients	Value: 2
CLIENT_COMMAND_requestClientKickFromXXX	Value: 3
CLIENT_COMMAND_flushChannelCreation	Value: 4
CLIENT_COMMAND_flushChannelUpdates	Value: 5
CLIENT_COMMAND_requestChannelMove	Value: 6
CLIENT_COMMAND_requestChannelDelete	Value: 7
CLIENT_COMMAND_requestChannelDescription	Value: 8
CLIENT_COMMAND_requestChannelXXSubscribe-XXX	Value: 9
CLIENT_COMMAND_requestServerConnectionInfo	Value: 10
CLIENT_COMMAND_requestSendXXXTextMsg	Value: 11
CLIENT_COMMAND_filetransfers	Value: 12
CLIENT_COMMAND_ENDMARKER	Value: 13

13.8 Class *ClientProperties*

object —
 ts3defines.ClientProperties

13.8.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.8.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

13.8.3 Class Variables

Name	Description
CLIENT_UNIQUE_IDENTIFIER	Value: 0
CLIENT_NICKNAME	Value: 1
CLIENT_VERSION	Value: 2
CLIENT_PLATFORM	Value: 3
CLIENT_FLAG_TALKING	Value: 4
CLIENT_INPUT_MUTED	Value: 5
CLIENT_OUTPUT_MUTED	Value: 6
CLIENT_OUTPUTONLY_MUTED	Value: 7
CLIENT_INPUT_HARDWARE	Value: 8
CLIENT_OUTPUT_HARDWARE	Value: 9
CLIENT_INPUT_DEACTIVATED	Value: 10
CLIENT_IDLE_TIME	Value: 11
CLIENT_DEFAULT_CHANNEL	Value: 12
CLIENT_DEFAULT_CHANNEL_PASSWORD	Value: 13
CLIENT_SERVER_PASSWORD	Value: 14
CLIENT_META_DATA	Value: 15
CLIENT_IS_MUTED	Value: 16
CLIENT_IS_RECORDING	Value: 17
CLIENT_VOLUME_MODIFICATOR	Value: 18
CLIENT_VERSION_SIGN	Value: 19

continued on next page

Name	Description
CLIENT_SECURITY_HASH	Value: 20
CLIENT_ENDMARKER	Value: 21

13.9 Class *ClientPropertiesRare*

object —
 ts3defines.ClientPropertiesRare

13.9.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.9.3 Class Variables

Name	Description
CLIENT_DUMMY_3	Value: 21
CLIENT_DUMMY_4	Value: 22
CLIENT_DUMMY_5	Value: 23
CLIENT_DUMMY_6	Value: 24
CLIENT_DUMMY_7	Value: 25
CLIENT_DUMMY_8	Value: 26
CLIENT_DUMMY_9	Value: 27
CLIENT_KEY_OFFSET	Value: 28
CLIENT_LAST_VAR_REQUEST	Value: 29
CLIENT_LOGIN_NAME	Value: 30
CLIENT_LOGIN_PASSWORD	Value: 31

continued on next page

Name	Description
CLIENT_DATABASE_ID	Value: 32
CLIENT_CHANNEL_GROUP_ID	Value: 33
CLIENT_SERVERGROUPS	Value: 34
CLIENT_CREATED	Value: 35
CLIENT_LASTCONNECTED	Value: 36
CLIENT_TOTALCONNECTIONS	Value: 37
CLIENT_AWAY	Value: 38
CLIENT_AWAY_MESSAGE	Value: 39
CLIENT_TYPE	Value: 40
CLIENT_FLAG_AVATAR	Value: 41
CLIENT_TALK_POWER	Value: 42
CLIENT_TALK_REQUEST	Value: 43
CLIENT_TALK_REQUEST_MSG	Value: 44
CLIENT_DESCRIPTION	Value: 45
CLIENT_IS_TALKER	Value: 46
CLIENT_MONTH_BYTES_UPLOADED	Value: 47
CLIENT_MONTH_BYTES_DOWNLOADED	Value: 48
CLIENT_TOTAL_BYTES_UPLOADED	Value: 49
CLIENT_TOTAL_BYTES_DOWNLOADED	Value: 50
CLIENT_IS_PRIORITY_SPEAKER	Value: 51
CLIENT_UNREAD_MESSAGES	Value: 52
CLIENT_NICKNAME_PHONETIC	Value: 53
CLIENT_NEEDED_SERVERQUERY_VIEW_POWER	Value: 54
CLIENT_DEFAULT_TOKEN	Value: 55

continued on next page

Name	Description
CLIENT_ICON_ID	Value: 56
CLIENT_IS_CHANNEL_COMMANDER	Value: 57
CLIENT_COUNTRY	Value: 58
CLIENT_CHANNEL_GROUP_INHERITED_CHANNEL_ID	Value: 59
CLIENT_BADGES	Value: 60
CLIENT_ENDMARKER_RARE	Value: 61

13.10 Class *ClientType*

object —
 ts3defines.ClientType

13.10.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.10.2 Properties

Name	Description
<i>Inherited from object</i> <code>--class--</code>	

13.10.3 Class Variables

Name	Description
<code>ClientType_NORMAL</code>	Value: 0
<code>ClientType_SERVERQUERY</code>	Value: 1

13.11 Class CodecEncryptionMode



13.11.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.11.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.11.3 Class Variables

Name	Description
<code>CODEC_ENCRYPTION_PER_CHANNEL</code>	Value: 0
<code>CODEC_ENCRYPTION_FORCED_OFF</code>	Value: 1
<code>CODEC_ENCRYPTION_FORCED_ON</code>	Value: 2

13.12 Class CodecType



13.12.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,

`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.12.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.12.3 Class Variables

Name	Description
<code>CODEC_SPEEX_NARROWBAND</code>	Value: 0
<code>CODEC_SPEEX_WIDEBAND</code>	Value: 1
<code>CODEC_SPEEX_ULTRA_WIDEBAND</code>	Value: 2
<code>CODEC_CELT_MONO</code>	Value: 3
<code>CODEC_OPUS_VOICE</code>	Value: 4
<code>CODEC_OPUS_MUSIC</code>	Value: 5

13.13 Class `CommandLinePropertiesRare`

object —
 `ts3defines.CommandLinePropertiesRare`

13.13.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.13.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.13.3 Class Variables

Name	Description
COMMANDLINE_NOTHING	Value: 0
COMMANDLINE_ENDMARKER_RARE	Value: 1

13.14 Class *ConnectionProperties*

object └─
 ts3defines.ConnectionProperties

13.14.1 Methods***Inherited from object***

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.14.2 Properties

Name	Description
<i>Inherited from object</i> <code>--class--</code>	

13.14.3 Class Variables

Name	Description
CONNECTION_PING	Value: 0
CONNECTION_PING_DEVIATION	Value: 1
CONNECTION_CONNECTED_TIME	Value: 2
CONNECTION_IDLE_TIME	Value: 3
CONNECTION_CLIENT_IP	Value: 4

continued on next page

Name	Description
CONNECTION_CLIENT_PORT	Value: 5
CONNECTION_SERVER_IP	Value: 6
CONNECTION_SERVER_PORT	Value: 7
CONNECTION_PACKETS_SENT_SPEECH	Value: 8
CONNECTION_PACKETS_SENT_KEEPALIVE	Value: 9
CONNECTION_PACKETS_SENT_CONTROL	Value: 10
CONNECTION_PACKETS_SENT_TOTAL	Value: 11
CONNECTION_BYTES_SENT_SPEECH	Value: 12
CONNECTION_BYTES_SENT_KEEPALIVE	Value: 13
CONNECTION_BYTES_SENT_CONTROL	Value: 14
CONNECTION_BYTES_SENT_TOTAL	Value: 15
CONNECTION_PACKETS_RECEIVED_SPEECH	Value: 16
CONNECTION_PACKETS_RECEIVED_KEEPALIVE	Value: 17
CONNECTION_PACKETS_RECEIVED_CONTROL	Value: 18
CONNECTION_PACKETS_RECEIVED_TOTAL	Value: 19
CONNECTION_BYTES_RECEIVED_SPEECH	Value: 20
CONNECTION_BYTES_RECEIVED_KEEPALIVE	Value: 21
CONNECTION_BYTES_RECEIVED_CONTROL	Value: 22
CONNECTION_BYTES_RECEIVED_TOTAL	Value: 23
CONNECTION_PACKETLOSS_SPEECH	Value: 24

continued on next page

Name	Description
CONNECTION_PACKETLOSS_KEEPALIVE	Value: 25
CONNECTION_PACKETLOSS_CONTROL	Value: 26
CONNECTION_PACKETLOSS_TOTAL	Value: 27
CONNECTION_SERVER2CLIENT_PACKETLOSS_SPEECH	Value: 28
CONNECTION_SERVER2CLIENT_PACKETLOSS_KEEPALIVE	Value: 29
CONNECTION_SERVER2CLIENT_PACKETLOSS_CONTROL	Value: 30
CONNECTION_SERVER2CLIENT_PACKETLOSS_TOTAL	Value: 31
CONNECTION_CLIENT2SERVER_PACKETLOSS_SPEECH	Value: 32
CONNECTION_CLIENT2SERVER_PACKETLOSS_KEEPALIVE	Value: 33
CONNECTION_CLIENT2SERVER_PACKETLOSS_CONTROL	Value: 34
CONNECTION_CLIENT2SERVER_PACKETLOSS_TOTAL	Value: 35
CONNECTION_BANDWIDTH_SENT_LAST_SECOND_SPEECH	Value: 36
CONNECTION_BANDWIDTH_SENT_LAST_SECOND_KEEPALIVE	Value: 37
CONNECTION_BANDWIDTH_SENT_LAST_SECOND_CONTROL	Value: 38
CONNECTION_BANDWIDTH_SENT_LAST_SECOND_TOTAL	Value: 39

continued on next page

Name	Description
CONNECTION_BANDWIDTH_SENT_LAST_MINUTE_SPEECH	Value: 40
CONNECTION_BANDWIDTH_SENT_LAST_MINUTE_KEEPALIVE	Value: 41
CONNECTION_BANDWIDTH_SENT_LAST_MINUTE_CONTROL	Value: 42
CONNECTION_BANDWIDTH_SENT_LAST_MINUTE_TOTAL	Value: 43
CONNECTION_BANDWIDTH_RECEIVED_LAST_SECOND_SPEECH	Value: 44
CONNECTION_BANDWIDTH_RECEIVED_LAST_SECOND_KEEPALIVE	Value: 45
CONNECTION_BANDWIDTH_RECEIVED_LAST_SECOND_CONTROL	Value: 46
CONNECTION_BANDWIDTH_RECEIVED_LAST_SECOND_TOTAL	Value: 47
CONNECTION_BANDWIDTH_RECEIVED_LAST_MINUTE_SPEECH	Value: 48
CONNECTION_BANDWIDTH_RECEIVED_LAST_MINUTE_KEEPALIVE	Value: 49
CONNECTION_BANDWIDTH_RECEIVED_LAST_MINUTE_CONTROL	Value: 50
CONNECTION_BANDWIDTH_RECEIVED_LAST_MINUTE_TOTAL	Value: 51
CONNECTION_ENDMARKER	Value: 52

13.15 Class *ConnectionPropertiesRare*

object └─
 ts3defines.ConnectionPropertiesRare

13.15.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.15.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.15.3 Class Variables

Name	Description
<code>CONNECTION_DUMMY-_0</code>	Value: 52
<code>CONNECTION_DUMMY-_1</code>	Value: 53
<code>CONNECTION_DUMMY-_2</code>	Value: 54
<code>CONNECTION_DUMMY-_3</code>	Value: 55
<code>CONNECTION_DUMMY-_4</code>	Value: 56
<code>CONNECTION_DUMMY-_5</code>	Value: 57
<code>CONNECTION_DUMMY-_6</code>	Value: 58
<code>CONNECTION_DUMMY-_7</code>	Value: 59
<code>CONNECTION_DUMMY-_8</code>	Value: 60
<code>CONNECTION_DUMMY-_9</code>	Value: 61

continued on next page

Name	Description
CONNECTION_FILETRANSFER_BANDWIDTH_SENT	Value: 62
CONNECTION_FILETRANSFER_BANDWIDTH_RECEIVED	Value: 63
CONNECTION_FILETRANSFER_BYTES_RECEIVED_TOTAL	Value: 64
CONNECTION_FILETRANSFER_BYTES_SENT_TOTAL	Value: 65
CONNECTION_ENDMARKER_RARE	Value: 66

13.16 Class `ConnectStatus`

object —
 ts3defines.ConnectStatus

13.16.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.16.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

13.16.3 Class Variables

Name	Description
STATUS_DISCONNECTED	Value: 0

continued on next page

Name	Description
STATUS_CONNECTING	Value: 1
STATUS_CONNECTED	Value: 2
STATUS_CONNECTION- _ESTABLISHING	Value: 3
STATUS_CONNECTION- _ESTABLISHED	Value: 4

13.17 Class FileListType

object └─
 ts3defines.FileListType

13.17.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr__()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.17.2 Properties

Name	Description
<i>Inherited from object</i> <code>--class--</code>	

13.17.3 Class Variables

Name	Description
FileListType_Directory	Value: 0
FileListType_File	Value: 1

13.18 Class FileTransferState

object └─
 ts3defines.FileTransferState

13.18.1 Methods***Inherited from object***

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.18.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.18.3 Class Variables

Name	Description
FILETRANSFER_INITIATING	Value: 0
FILETRANSFER_ACTIVE	Value: 1
FILETRANSFER_FINISHED	Value: 2

13.19 Class FTAction**13.19.1 Methods*****Inherited from object***

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.19.2 Properties

Name	Description
<i>Inherited from object</i>	

continued on next page

Name	Description
__class__	

13.19.3 Class Variables

Name	Description
FT_INIT_SERVER	Value: 0
FT_INIT_CHANNEL	Value: 1
FT_UPLOAD	Value: 2
FT_DOWNLOAD	Value: 3
FT_DELETE	Value: 4
FT_CREATEDIR	Value: 5
FT_RENAME	Value: 6
FT_FILELIST	Value: 7
FT_FILEINFO	Value: 8

13.20 Class GroupShowNameTreeMode

object └─
 ts3defines.GroupShowNameTreeMode

13.20.1 Methods

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __init__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

13.20.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

13.20.3 Class Variables

continued on next page

Name	Description
GroupShowNameTreeMode_NONE	Value: 0
GroupShowNameTreeMode_BEFORE	Value: 1
GroupShowNameTreeMode_BEHIND	Value: 2

13.21 Class *GroupWhisperTargetMode*

object └─ **ts3defines.GroupWhisperTargetMode**

13.21.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.21.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.21.3 Class Variables

Name	Description
GROUPWHISPERTARGETMODE_ALL	Value: 0
GROUPWHISPERTARGETMODE_CURRENTCHANNEL	Value: 1
GROUPWHISPERTARGETMODE_PARENTCHANNEL	Value: 2

continued on next page

Name	Description
GROUPWHISPERTARGETMODE_ALLPARENTCHANNELS	Value: 3
GROUPWHISPERTARGETMODE_CHANNELFAMILY	Value: 4
GROUPWHISPERTARGETMODE_ANCESTORCHANNELFAMILY	Value: 5
GROUPWHISPERTARGETMODE_SUBCHANNELS	Value: 6
GROUPWHISPERTARGETMODE_ENDMARKER	Value: 7

13.22 Class *GroupWhisperType*

object —
 ts3defines.GroupWhisperType

13.22.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.22.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

13.22.3 Class Variables

Name	Description
GROUPWHISPERTYPE- _SERVERGROUP	Value: 0
GROUPWHISPERTYPE- _CHANNELGROUP	Value: 1
GROUPWHISPERTYPE- _CHANNELCOMMANDE- R	Value: 2
GROUPWHISPERTYPE- _ALLCLIENTS	Value: 3
GROUPWHISPERTYPE- _ENDMARKER	Value: 4

13.23 Class *HardwareInputStatus*

object —
 ts3defines.HardwareInputStatus

13.23.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.23.2 Properties

Name	Description
<i>Inherited from object</i> <code>--class--</code>	

13.23.3 Class Variables

Name	Description
HARDWAREINPUT_DIS- ABLED	Value: 0
HARDWAREINPUT_EN- ABLED	Value: 1

13.24 Class `HardwareOutputStatus`



13.24.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.24.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.24.3 Class Variables

Name	Description
<code>HARDWAREOUTPUT_DISABLED</code>	Value: 0
<code>HARDWAREOUTPUT_ENABLED</code>	Value: 1

13.25 Class `HostBannerMode`



13.25.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.25.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

13.25.3 Class Variables

Name	Description
HostBannerMode_NO_ADJUST	Value: 0
HostBannerMode_ADJUST_IGNORE_ASPECT	Value: 1
HostBannerMode_ADJUST_KEEP_ASPECT	Value: 2

13.26 Class HostMessageMode**13.26.1 Methods*****Inherited from object***

__delattr__(), __format__(), __getattr__(), __hash__(), __init__(), __new__(), __reduce__(),
 __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

13.26.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

13.26.3 Class Variables

Name	Description
HostMessageMode_NONE	Value: 0

continued on next page

Name	Description
HostMessageMode_LOG	Value: 1
HostMessageMode_MODAL	Value: 2
HostMessageMode_MODALQUIT	Value: 3

13.27 Class InputDeactivationStatus

object └─
 ts3defines.InputDeactivationStatus

13.27.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.27.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.27.3 Class Variables

Name	Description
INPUT_ACTIVE	Value: 0
INPUT_DEACTIVATED	Value: 1

13.28 Class LicenseViolationType

object └─
 ts3defines.LicenseViolationType

13.28.1 Methods***Inherited from object***

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.28.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.28.3 Class Variables

Name	Description
<code>NO_VIOLATION</code>	Value: 0
<code>SLOT_VIOLATION</code>	Value: 1
<code>SLOT_SUSPICION</code>	Value: 2

13.29 Class LocalTestMode**13.29.1 Methods*****Inherited from object***

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.29.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.29.3 Class Variables

Name	Description
TEST_MODE_OFF	Value: 0
TEST_MODE_VOICE_LOCAL_ONLY	Value: 1
TEST_MODE_VOICE_LOCAL_AND_REMOTE	Value: 2

13.30 Class **LogLevel**

object └─
 ts3defines.LogLevel

13.30.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.30.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

13.30.3 Class Variables

Name	Description
<code>LogLevel.CRITICAL</code>	Value: 0
<code>LogLevel.ERROR</code>	Value: 1
<code>LogLevel.WARNING</code>	Value: 2
<code>LogLevel.DEBUG</code>	Value: 3
<code>LogLevel.INFO</code>	Value: 4
<code>LogLevel.DEVEL</code>	Value: 5

13.31 Class LogTypes



13.31.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.31.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.31.3 Class Variables

Name	Description
<code>LogType_NONE</code>	Value: 0x0000
<code>LogType_FILE</code>	Value: 0x0001
<code>LogType_CONSOLE</code>	Value: 0x0002
<code>LogType_USERLOGGING</code>	Value: 0x0004
<code>LogType_NO_NETLOGGING</code>	Value: 0x0008
<code>LogType_DATABASE</code>	Value: 0x0010
<code>LogType_SYSLOG</code>	Value: 0x0020

13.32 Class MonoSoundDestination



13.32.1 Methods***Inherited from object***

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.32.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.32.3 Class Variables

Name	Description
MONO_SOUND_DESTINATION_ALL	Value: 0
MONO_SOUND_DESTINATION_FRONT_CENTER	Value: 1
MONO_SOUND_DESTINATION_FRONT_LEFT_AND_RIGHT	Value: 2

13.33 Class MuteInputStatus**13.33.1 Methods*****Inherited from object***

`--delattr--()`, `--format--()`, `--getattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.33.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

13.33.3 Class Variables

Name	Description
MUTEINPUT_NONE	Value: 0
MUTEINPUT_MUTED	Value: 1

13.34 Class MuteOutputStatus



13.34.1 Methods

Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

13.34.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

13.34.3 Class Variables

Name	Description
MUTEOUTPUT_NONE	Value: 0
MUTEOUTPUT_MUTED	Value: 1

13.35 Class PluginConfigureOffer



13.35.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattrattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.35.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.35.3 Class Variables

Name	Description
<code>PLUGIN_OFFERS_NO_CONFIGURE</code>	Value: 0
<code>PLUGIN_OFFERS_CONFIGURE_NEW_THREAD</code>	Value: 1
<code>PLUGIN_OFFERS_CONFIGURE_QT_THREAD</code>	Value: 2

13.36 Class PluginConnectTab



13.36.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattrattribute--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,

`--reduce_ex__()`, `--repr__()`, `--setattr__()`, `--sizeof__()`, `--str__()`, `--subclasshook__()`

13.36.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class__</code>	

13.36.3 Class Variables

Name	Description
PLUGIN_CONNECT_TAB_NEW	Value: 0
PLUGIN_CONNECT_TAB_CURRENT	Value: 1
PLUGIN_CONNECT_TAB_NEW_IF_CURRENT_CONNECTED	Value: 2

13.37 Class PluginGuiProfile

object └─
 ts3defines.PluginGuiProfile

13.37.1 Methods

Inherited from object

`--delattr__()`, `--format__()`, `--getattr__()`, `--hash__()`, `--init__()`, `--new__()`, `--reduce__()`, `--reduce_ex__()`, `--repr__()`, `--setattr__()`, `--sizeof__()`, `--str__()`, `--subclasshook__()`

13.37.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class__</code>	

13.37.3 Class Variables

Name	Description
PLUGIN_GUISOUND_CAPTURE	Value: 0
PLUGIN_GUISOUND_PLAYBACK	Value: 1
PLUGIN_GUIHOTKEY	Value: 2
PLUGIN_GUISOUNDPACK	Value: 3
PLUGIN_GUIIDENTITY	Value: 4

13.38 Class *PluginItemType*

object └─
 ts3defines.PluginItemType

13.38.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.38.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

13.38.3 Class Variables

Name	Description
PLUGIN_SERVER	Value: 0
PLUGIN_CHANNEL	Value: 1
PLUGIN_CLIENT	Value: 2

13.39 Class `PluginMenuType`



13.39.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.39.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.39.3 Class Variables

Name	Description
<code>PLUGIN_MENU_TYPE_-GLOBAL</code>	Value: 0
<code>PLUGIN_MENU_TYPE_-CHANNEL</code>	Value: 1
<code>PLUGIN_MENU_TYPE_-CLIENT</code>	Value: 2

13.40 Class `PluginMessageTarget`



13.40.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,

`--reduce_ex__()`, `--repr__()`, `--setattr__()`, `--sizeof__()`, `--str__()`, `--subclasshook__()`

13.40.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class__</code>	

13.40.3 Class Variables

Name	Description
<code>PLUGIN_MESSAGE_TARGET_SERVER</code>	Value: 0
<code>PLUGIN_MESSAGE_TARGET_CHANNEL</code>	Value: 1

13.41 Class *PluginTargetMode*



13.41.1 Methods

Inherited from object

`--delattr__()`, `--format__()`, `--getattribute__()`, `--hash__()`, `--init__()`, `--new__()`, `--reduce__()`, `--reduce_ex__()`, `--repr__()`, `--setattr__()`, `--sizeof__()`, `--str__()`, `--subclasshook__()`

13.41.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class__</code>	

13.41.3 Class Variables

Name	Description
PluginCommandTarget_CURRENT_CHANNEL	Value: 0
PluginCommandTarget_SERVER	Value: 1
PluginCommandTarget_CLIENT	Value: 2
PluginCommandTarget_CURRENT_CHANNEL_SUBSCRIBED_CLIENTS	Value: 3
PluginCommandTarget_MAX	Value: 4

13.42 Class ReasonIdentifier



13.42.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.42.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.42.3 Class Variables

Name	Description
REASON_NONE	Value: 0
REASON_MOVED	Value: 1
REASON_SUBSCRIPTION	Value: 2

continued on next page

Name	Description
REASON_LOST_CONNECTION	Value: 3
REASON_KICK_CHANNEL	Value: 4
REASON_KICK_SERVER	Value: 5
REASON_KICK_SERVER_BAN	Value: 6
REASON_SERVERSTOP	Value: 7
REASON_CLIENTDISCONNECT	Value: 8
REASON_CHANNELUPDATE	Value: 9
REASON_CHANNELEDIT	Value: 10
REASON_CLIENTDISCONNECT_SERVER_SHUTDOWN	Value: 11

13.43 Class SecuritySaltOptions

object └─
 ts3defines.SecuritySaltOptions

13.43.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.43.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.43.3 Class Variables

Name	Description
<code>SECURITY_SALT_CHECK_NICKNAME</code>	Value: 1
<code>SECURITY_SALT_CHECK_META_DATA</code>	Value: 2

13.44 Class `SERVER_BINDING`

object └─
 `ts3defines.SERVER_BINDING`

13.44.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.44.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

13.44.3 Class Variables

Name	Description
<code>SERVER_BINDING_VIRTUALSERVER</code>	Value: 0
<code>SERVER_BINDING_SERVERQUERY</code>	Value: 1
<code>SERVER_BINDING_FILETRANSFER</code>	Value: 2

13.45 Class *ServerInstancePropertiesRare*



13.45.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.45.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.45.3 Class Variables

Name	Description
<code>SERVERINSTANCE_DATABASE_VERSION</code>	Value: 0
<code>SERVERINSTANCE_FILTERTRANSFER_PORT</code>	Value: 1
<code>SERVERINSTANCE_SERVER_ENTROPY</code>	Value: 2
<code>SERVERINSTANCE_MONTHLY_TIMESTAMP</code>	Value: 3
<code>SERVERINSTANCE_MAX_DOWNLOAD_TOTAL_BANDWIDTH</code>	Value: 4
<code>SERVERINSTANCE_MAX_UPLOAD_TOTAL_BANDWIDTH</code>	Value: 5
<code>SERVERINSTANCE_GUEST_SERVERQUERY_GROUP</code>	Value: 6
<code>SERVERINSTANCE_SERVERQUERY_FLOOD_COMMANDS</code>	Value: 7

continued on next page

Name	Description
SERVERINSTANCE_SERVERQUERY_FLOOD_TIME	Value: 8
SERVERINSTANCE_SERVERQUERY_BAN_TIME	Value: 9
SERVERINSTANCE_TEMPLATE_SERVERADMIN_GROUP	Value: 10
SERVERINSTANCE_TEMPLATE_SERVERDEFAULT_GROUP	Value: 11
SERVERINSTANCE_TEMPLATE_CHANNELADMIN_GROUP	Value: 12
SERVERINSTANCE_TEMPLATE_CHANNELDEFAULT_GROUP	Value: 13
SERVERINSTANCE_PERMISSIONS_VERSION	Value: 14
SERVERINSTANCE_PENDING_CONNECTIONS_PER_IP	Value: 15
SERVERINSTANCE_ENDMARKER_RARE	Value: 16

13.46 Class *TalkStatus*

object —
 ts3defines.TalkStatus

13.46.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.46.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.46.3 Class Variables

Name	Description
<code>STATUS_NOT_TALKING</code>	Value: 0
<code>STATUS_TALKING</code>	Value: 1
<code>STATUS_TALKING_WHILE_DISABLED</code>	Value: 2

13.47 Class `TextMessageTargetMode`



13.47.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.47.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.47.3 Class Variables

Name	Description
<code>TextMessageTarget_CLIENT</code>	Value: 1
<code>TextMessageTarget_CHANNEL</code>	Value: 2

continued on next page

Name	Description
TextMessageTarget_SERVER	Value: 3
TextMessageTarget_MAX	Value: 4

13.48 Class VirtualServerProperties



13.48.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.48.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.48.3 Class Variables

Name	Description
VIRTUALSERVER_UNIQUE_IDENTIFIER	Value: 0
VIRTUALSERVER_NAME	Value: 1
VIRTUALSERVER_WELCOME_MESSAGE	Value: 2
VIRTUALSERVER_PLATFORM	Value: 3
VIRTUALSERVER_VERSION	Value: 4
VIRTUALSERVER_MAX_CLIENTS	Value: 5

continued on next page

Name	Description
VIRTUALSERVER_PASSWORD	Value: 6
VIRTUALSERVER_CLIENTS_ONLINE	Value: 7
VIRTUALSERVER_CHANNELS_ONLINE	Value: 8
VIRTUALSERVER_CREATED	Value: 9
VIRTUALSERVER_UPTIME	Value: 10
VIRTUALSERVER_CODEC_ENCRYPTION_MODE	Value: 11
VIRTUALSERVER_ENDMARKER	Value: 12

13.49 Class *VirtualServerPropertiesRare*

object —
 ts3defines.VirtualServerPropertiesRare

13.49.1 Methods

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

13.49.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

13.49.3 Class Variables

Name	Description
VIRTUALSERVER_DUM-MY_0	Value: 12
VIRTUALSERVER_DUM-MY_1	Value: 13
VIRTUALSERVER_DUM-MY_2	Value: 14
VIRTUALSERVER_DUM-MY_3	Value: 15
VIRTUALSERVER_DUM-MY_4	Value: 16
VIRTUALSERVER_DUM-MY_5	Value: 17
VIRTUALSERVER_DUM-MY_6	Value: 18
VIRTUALSERVER_DUM-MY_7	Value: 19
VIRTUALSERVER_DUM-MY_8	Value: 20
VIRTUALSERVER_KEY-PAIR	Value: 21
VIRTUALSERVER_HOS-TMESSAGE	Value: 22
VIRTUALSERVER_HOS-TMESSAGE_MODE	Value: 23
VIRTUALSERVER_FILE-BASE	Value: 24
VIRTUALSERVER_DEF-AULT_SERVER_GROUP	Value: 25
VIRTUALSERVER_DEF-AULT_CHANNEL_GROU-P	Value: 26
VIRTUALSERVER_FLA-G_PASSWORD	Value: 27
VIRTUALSERVER_DEF-AULT_CHANNEL_ADMI-N_GROUP	Value: 28
VIRTUALSERVER_MAX-_DOWNLOAD_TOTAL_B-ANDWIDTH	Value: 29
VIRTUALSERVER_MAX-_UPLOAD_TOTAL_BAN-DWIDTH	Value: 30

continued on next page

Name	Description
VIRTUALSERVER_HOS- TBANNER_URL	Value: 31
VIRTUALSERVER_HOS- TBANNER_GFX_URL	Value: 32
VIRTUALSERVER_HOS- TBANNER_GFX.INTER- VAL	Value: 33
VIRTUALSERVER_COM- PLAIN_AUTOBAN_COU- NT	Value: 34
VIRTUALSERVER_COM- PLAIN_AUTOBAN_TIM- E	Value: 35
VIRTUALSERVER_COM- PLAIN_REMOVE_TIME	Value: 36
VIRTUALSERVER_MIN_- CLIENTS_IN_CHANNEL_- BEFORE_FORCED_SILE- NCE	Value: 37
VIRTUALSERVER_PRI- ORITY_SPEAKER_DIM- M.MODIFICATOR	Value: 38
VIRTUALSERVER_ID	Value: 39
VIRTUALSERVER_ANT- IFLOOD_POINTS_TICK_- REDUCE	Value: 40
VIRTUALSERVER_ANT- IFLOOD_POINTS_NEED- ED_COMMAND_BLOCK	Value: 41
VIRTUALSERVER_ANT- IFLOOD_POINTS_NEED- ED_IP_BLOCK	Value: 42
VIRTUALSERVER_CLIE- NT.CONNECTIONS	Value: 43
VIRTUALSERVER_QUE- RY_CLIENT.CONNECTI- ONS	Value: 44
VIRTUALSERVER_HOS- TBUTTON_TOOL TIP	Value: 45
VIRTUALSERVER_HOS- TBUTTON_URL	Value: 46
VIRTUALSERVER_HOS- TBUTTON_GFX_URL	Value: 47

continued on next page

Name	Description
VIRTUALSERVER_QUE- RYCLIENTS_ONLINE	Value: 48
VIRTUALSERVER_DO- WNL0AD_QUOTA	Value: 49
VIRTUALSERVER_UPL- 0AD_QUOTA	Value: 50
VIRTUALSERVER_MON- TH.BYTES_DOWNLOA- DED	Value: 51
VIRTUALSERVER_MON- TH.BYTES_UPLOADED	Value: 52
VIRTUALSERVER_TOT- AL.BYTES_DOWNLOA- DED	Value: 53
VIRTUALSERVER_TOT- AL.BYTES_UPLOADED	Value: 54
VIRTUALSERVER_POR- T	Value: 55
VIRTUALSERVER_AUT- OSTART	Value: 56
VIRTUALSERVER_MAC- HINE.ID	Value: 57
VIRTUALSERVER_NEE- DED_IDENTITY_SECUR- ITY_LEVEL	Value: 58
VIRTUALSERVER_LOG- _CLIENT	Value: 59
VIRTUALSERVER_LOG- _QUERY	Value: 60
VIRTUALSERVER_LOG- _CHANNEL	Value: 61
VIRTUALSERVER_LOG- _PERMISSIONS	Value: 62
VIRTUALSERVER_LOG- _SERVER	Value: 63
VIRTUALSERVER_LOG- _FILETRANSFER	Value: 64
VIRTUALSERVER_MIN_- CLIENT_VERSION	Value: 65
VIRTUALSERVER_NAM- E_PHONETIC	Value: 66

continued on next page

Name	Description
VIRTUALSERVER_ICON_ID	Value: 67
VIRTUALSERVER_RESERVED_SLOTS	Value: 68
VIRTUALSERVER_TOTAL_PACKETLOSS_SPEECH	Value: 69
VIRTUALSERVER_TOTAL_PACKETLOSS_KEEPAIVE	Value: 70
VIRTUALSERVER_TOTAL_PACKETLOSS_CONTROL	Value: 71
VIRTUALSERVER_TOTAL_PACKETLOSS_TOTAL	Value: 72
VIRTUALSERVER_TOTAL_PING	Value: 73
VIRTUALSERVER_IP	Value: 74
VIRTUALSERVER_WEBLIST_ENABLED	Value: 75
VIRTUALSERVER_AUTOGENERATED_PRIVILEGEKEY	Value: 76
VIRTUALSERVER_ASK_FOR_PRIVILEGEKEY	Value: 77
VIRTUALSERVER_HOSTBANNER_MODE	Value: 78
VIRTUALSERVER_CHANNEL_TEMP_DELETE_DELAY_DEFAULT	Value: 79
VIRTUALSERVER_MIN_ANDROID_VERSION	Value: 80
VIRTUALSERVER_MIN_IOS_VERSION	Value: 81
VIRTUALSERVER_MIN_WINPHONE_VERSION	Value: 82
VIRTUALSERVER_ENDMARKER_RARE	Value: 83

13.50 Class Visibility



13.50.1 Methods

Inherited from object

`--delattr--()`, `--format--()`, `--getattr__()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

13.50.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

13.50.3 Class Variables

Name	Description
ENTER_VISIBILITY	Value: 0
RETAIN_VISIBILITY	Value: 1
LEAVE_VISIBILITY	Value: 2

14 Module ts3lib

14.1 Functions

getPluginID()

Returns pyTSON's plugin id

Return Value

the plugin id

(type=string)

acquireCustomPlaybackData(deviceName, samples)

Retrieves playback data from the clientlib

Parameters

deviceName: the name of the playback device previously registered with registerCustomDevice

(type=string)

samples: specifies how long the resultbuffer should be, which is passed to the clientlib

(type=int)

Return Value

a tuple containing the errorcode and the buffer as list of ints

(type=tuple(int, list[int]))

activateCaptureDevice(serverConnectionHandlerID)

Activates the capture device on a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection

(type=int)

Return Value

the errorcode

(type=int)

banadd(*serverConnectionHandlerID*, *ipRegExp*, *nameRegexp*, *uniqueIdentity*, *timeInSeconds*, *banReason*, *returnCode*)

Adds a new ban.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
ipRegExp:	regular expression to match IPs, pass an empty string to ignore IPs (<i>type=string</i>)
nameRegexp:	regular expression to match client nicknames, pass an empty string to ignore nicknames (<i>type=string</i>)
uniqueIdentity:	client UID to ban, pass an empty string to ignore UIDs (<i>type=string</i>)
timeInSeconds:	the time, the client should be banned for, pass 0 to add a permanent ban (<i>type=int</i>)
banReason:	the reason for the ban (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

banclient(*serverConnectionHandlerID*, *clientID*, *timeInSeconds*, *banReason*, *returnCode*)

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the ID of the client (<i>type=int</i>)
timeInSeconds:	the time, the client should be banned for, pass 0 to add a permanent ban (<i>type=int</i>)
banReason:	the reason for the ban (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

banclientdbid(*serverConnectionHandlerID*, *clientDBID*, *timeInSeconds*, *banReason*, *returnCode*)

Bans a user defined by his database ID.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientDBID:	the database ID of the user (<i>type=int</i>)
timeInSeconds:	the time, the client should be banned for, pass 0 to add a permanent ban (<i>type=int</i>)
banReason:	the reason for the ban (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

bandel(*serverConnectionHandlerID*, *banID*, *returnCode*)

Deletes a ban.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

banID: the ID of the ban
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

bandelall(*serverConnectionHandlerID*, *returnCode*)

Deletes all bans on a server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

channelPropertyStringToFlag(*channelPropertyString*)

Converts a channel property name used in strings (eg the serverquery) to the corresponding flag.

Parameters

channelPropertyString: (*type=string*)

Return Value

a tuple, containing the errorcode and the flag (see
ts3defines.ChannelProperties and ts3defines.ChannelPropertiesRare)

(*type=tuple (int, int)*)

channelset3DAttributes(*serverConnectionHandlerID, clientID, position*)

Adjusts a clients position and velocity in 3D space.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client to adjust
(*type=int*)

position: a tuple defining the position of the
clientID
(*type=tuple (float, float, float)*)

Return Value

the errorcode
(*type=int*)

cleanUpConnectionInfo(*serverConnectionHandlerID, clientID*)

//FIXME:

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

Return Value

the errorcode
(*type=int*)

clientChatClosed(*serverConnectionHandlerID*, *clientUniqueIdentifier*, *clientID*, *returnCode*)

Sends the client chat closed command to a client the own client is currently chatting with.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientUniqueIdentifier:	the uid of the own chatting client (<i>type=string</i>)
clientID:	the ID of the client, the own client is chatting with (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

clientChatComposing(*serverConnectionHandlerID*, *clientID*, *returnCode*)

Sends the client chat composing command to a client the own client is currently chatting with.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client, the own client is chatting with
(*type=int*)

returnCode: returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

clientPropertyStringToFlag(*clientPropertyString*)

Converts a client property name used in strings (eg the serverquery) to the corresponding flag.

Parameters

clientPropertyString: (*type=string*)

Return Value

a tuple, containing the errorcode and the flag (see ts3defines.ClientProperties and ts3defines.ClientPropertiesRare)
(*type=tuple (int, int)*)

closeCaptureDevice(*serverConnectionHandlerID*)

Closes a capture device on a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

the errorcode
(*type=int*)

closePlaybackDevice(*serverConnectionHandlerID*)

Closes a playback device on a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

the errorcode
(*type=int*)

closeWaveFileHandle(*serverConnectionHandlerID*, *waveHandle*)

Closes a wavefile sound handle previously returned by playWaveFileHandle.

Parameters

serverConnectionHandlerID: the ID of the serverConnection the
sound was played on
(*type=int*)

waveHandle: the handle returned by
playWaveFileHandle
(*type=int*)

Return Value

the errorcode
(*type=int*)

createBookmark(bookmarkuuid, serverLabel, serverAddress, serverPassword, nickname, channel, channelPassword, captureProfile, playbackProfile, hotkeyProfile, soundProfile, uniqueUserId, oneTimeKey, phoneticName)

Creates a new bookmark.

Parameters

serverLabel:	the label of the connection (type=string)
serverAddress:	host or ip address (type=string)
serverPassword:	password to the server, pass an empty string if the server is not password protected (type=string)
nickname:	the user's nickname (type=string)
channel:	complete path to the channel to connect to (type=string)
channelPassword:	password to the channel, pass an empty string if the channel is not password protected (type=string)
captureProfile:	the name of the capture profile to use; pass an empty string to always use the default one (type=string)
playbackProfile:	the name of the playback profile to use; pass an empty string to always use the default one (type=string)
hotkeyProfile:	the name of the hotkey profile to use; pass an empty string to always use the default one (type=string)
soundProfile:	the name of the sound profile to use; pass an empty string to always use the default one (type=string)
uniqueUserId:	identity (name) to use; pass an empty string to always use the default one (type=string)
oneTimeKey:	privilege key to use on connect (type=string)
phoneticName:	phonetic nickname (type=string)

Return Value

the errorcode

createReturnCode(*maxLen=128*)

Creates a returnCode which can be passed to the other functions and will be passed to the event onServerErrorEvent.

Parameters

maxLen: length of the buffer, passed to the clientlib to store the path to, default value is 256

(*type=int*)

Return Value

the created returnCode

(*type=string*)

destroyServerConnectionHandler(*serverConnectionHandlerID*)

Destroys a server connection handler.

Parameters

serverConnectionHandlerID: the ID of the serverconnection

(*type=int*)

Return Value

the errorcode

(*type=int*)

flushChannelCreation(*serverConnectionHandlerID*, *channelParentID*, *returnCode*)

Flushes the channel creation made by the setChannelVariable-functions to the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelParentID: the ID of the parent channel of the new channel
(*type=int*)

returnCode: returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

flushChannelUpdates(*serverConnectionHandlerID*, *channelID*, *returnCode*)

Flushes the changes made by the setChannelVariable-functions to the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channelID
(*type=int*)

returnCode: returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

flushClientSelfUpdates(*serverConnectionHandlerID*, *returnCode*)

Flushes the changes made by the setClientSelfVariable-functions to the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection

(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.

(*type=string*)

Return Value

the errorcode

(*type=int*)

getAppPath(*maxLen=256*)

Returns the ts3 application path.

Parameters

maxLen: length of the buffer, passed to the clientlib to store the
path to, default value is 256

(*type=int*)

Return Value

the application path

(*type=string*)

getAvatar(*serverConnectionHandlerID*, *clientID*, *maxLen=256*)

Returns the path on the system to the avatar image file of a client.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

maxLen: length of the buffer, passed to the clientlib to store the path to, default value is 256
(*type=int*)

Return Value

a tuple, containing the errorcode and the path to the avatar
(*type=tuple (int, string)*)

getAverageTransferSpeed(*transferID*)

Returns the average transfer speed of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the speed
(*type=tuple (int, float)*)

getBookmarkList()

Returns the list of bookmarks.

Return Value

a tuple, containing the errorcode and a list of tuples (name, isFolder, uid, childs)
(*type=tuple (int, [tuple (string, int or bool, string or None, [childs])])*)

getCaptureDeviceList(modeID)

Queries all available capture devices.

Parameters

modeID: Defines the capture mode to use.
(type=string)

Return Value

A tuple, containing the errorcode and the list of capture devices as tuple (devicename, deviceid)
(type=tuple (int, [(string, string)]))

getCaptureModeList()

Queries all available capture modes.

Return Value

A tuple, containing the errorcode and the list of capture modes
(type=tuple (int, [string]))

getChannelClientList(serverConnectionHandlerID, channelID)

Returns all clients in a specified channel.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

channelID: the ID of the channel
(type=int)

Return Value

a tuple, containing the errorcode and a list of client IDs or None if the call failed
(type=tuple (int, [int]) or tuple(int, None))

getChannelConnectInfo(*serverConnectionHandlerID*, *channelID*, *maxLen*)

Returns the channel connect info (path and password) of a channel.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

maxLen: length of the buffer, passed to the clientlib to store the path to, default value is 256
(*type=int*)

Return Value

a tuple, containing the errorcode, the path and the password of a channel
(*type=tuple (int, string, string)*)

getChannelIDFromChannelNames(*serverConnectionHandlerID*, *channelNameArray*)

Returns the ID of a channel defined by its name.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelNameArray: list of strings, defining the position of the channel (['grandparent', 'parent', 'channel'])
(*type=list [string]*)

Return Value

a tuple, containing the errorcode and the ID of the channel
(*type=tuple (int, int)*)

getChannelList(*serverConnectionHandlerID*)

Returns all channels on the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

a tuple, containing the errorcode and a list of channel IDs
(*type=tuple (int, [int])*)

getChannelOfClient(*serverConnectionHandlerID, clientID*)

Returns the channel of a client.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

Return Value

a tuple, containing the errorcode and the channel
(*type=tuple (int, int)*)

getChannelVariableAsInt(*serverConnectionHandlerID, channelID, flag*)

Returns a channel variable as int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

flag: the flag to return
(*type=int*)

Return Value

a tuple, containing the errorcode and the value of the flag
(*type=tuple (int, int)*)

getChannelVariableAsString(*serverConnectionHandlerID*, *channelID*, *flag*)

Returns a channel variable as string value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

flag: the flag to return
(*type=int*)

Return Value

a tuple, containing the errorcode and the value of the flag
(*type=tuple (int, int)*)

getChannelVariableAsUInt64(*serverConnectionHandlerID*, *channelID*, *flag*)

Returns a channel variable as unsigned long long int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

flag: the flag to return
(*type=int*)

Return Value

a tuple, containing the errorcode and the value of the flag
(*type=tuple (int, int)*)

getClientDisplayName(*serverConnectionHandlerID*, *clientID*,
maxLen=128)

Returns the client display name receiving from the client's contacts settings.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

maxLen: length of the buffer, passed to the
clientlib to store the path to, default
value is 128
(*type=int*)

Return Value

a tuple, containing the errorcode and the display name
(*type=tuple (int, string)*)

getClientID(*serverConnectionHandlerID*)

Returns the own client ID on a given serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

a tuple, containing the errorcode and the client ID
(*type=tuple (int, int)*)

getClientLibVersion()

Returns the clientlib's version as string.

Return Value

A tuple, containing the errorcode and the result
(*type=tuple (int, string)*)

getClientLibVersionNumber()

Returns the clientlib's version number

Return Value

A tuple, containing the errorcode and the result
(*type=tuple (int, int)*)

getClientList(serverConnectionHandlerID)

Returns all clients in view on the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

a tuple, containing the errorcode and the list of client IDs
(*type=tuple (int, [int])*)

getClientNeededPermission(serverConnectionHandlerID, permissionName)

Returns the value of the client's needed permission.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

permissionName: name of the permission
(*type=string*)

Return Value

a tuple, containing the errorcode and the value of the permission
(*type=tuple (int, int)*)

getClientSelfVariableAsInt(*serverConnectionHandlerID*, *flag*)

Returns the value of a given flag of the own client as int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

flag: the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the queried flag
(type=tuple (int, int))

getClientSelfVariableAsString(*serverConnectionHandlerID*, *flag*)

Returns the value of a given flag of the own client as string.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

flag: the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the queried flag
(type=tuple (int, string))

getClientVariableAsInt(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns the value of a given flag of a client as int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

clientID: the ID of the client
(type=int)

flag: the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

getClientVariableAsString(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns the value of a given flag of a client as string value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

clientID: the ID of the client
(type=int)

flag: the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, string))

getClientVariableAsUInt64(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns the value of a given flag of a client as unsigned long long int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

clientID: the ID of the client
(type=int)

flag: the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

getConfigPath(*maxLen=256*)

Returns the ts3 config path.

Parameters

maxLen: length of the buffer, passed to the clientlib to store the path to, default value is 256
(type=int)

Return Value

the config path
(type=string)

getConnectionStatus(*serverConnectionHandlerID*)

Returns the current connection status of a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

a tuple, containing the errorcode and the connection status
(*type=tuple (int, int)*)

getConnectionVariableAsDouble(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns a client's connection variable as python floating point variable.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

Return Value

a tuple, containing the errorcode and the value of the flag
(*type=tuple (int, double)*)

getConnectionVariableAsString(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns a client's connection variable as string variable.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

Return Value

a tuple, containing the errorcode and the value of the flag
(*type=tuple (int, string)*)

getConnectionVariableAsUInt64 (<i>serverConnectionHandlerID</i> , <i>clientID</i> , <i>flag</i>)	
Returns a client's connection variable as unsigned long long int variable.	
Parameters	
serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the ID of the client (<i>type=int</i>)
Return Value	
a tuple, containing the errorcode and the value of the flag (<i>type=tuple (int, int)</i>)	

getCurrentCaptureDeviceName (<i>serverConnectionHandlerID</i>)	
Queries the current playback device's name on a serverconnection.	
Parameters	
serverConnectionHandlerID:	ID of the serverconnection (<i>type=int</i>)
Return Value	
A tuple, containing the errorcode, the capture device's name and the status, if it's default (<i>type=tuple (int, string, int)</i>)	

getCurrentCaptureMode (<i>serverConnectionHandlerID</i>)	
Queries the current capture mode on a serverconnection.	
Parameters	
serverConnectionHandlerID:	ID of the serverconnection (<i>type=int</i>)
Return Value	
A tuple, containing the errorcode and the current capture mode (<i>type=tuple (int, string)</i>)	

getCurrentPlaybackDeviceName(*serverConnectionHandlerID*)

Queries the current playback device's name on a serverconnection.

Parameters

serverConnectionHandlerID: ID of the serverconnection
(*type=int*)

Return Value

A tuple, containing the errorcode, the playback device's name and the status, if it's default
(*type=tuple (int, string, int)*)

getCurrentPlayBackMode(*serverConnectionHandlerID*)

Queries the current playback mode on a serverconnection.

Parameters

serverConnectionHandlerID: ID of the serverconnection
(*type=int*)

Return Value

A tuple, containing the errorcode and the current playback mode
(*type=tuple (int, string)*)

getCurrentServerConnectionHandlerID()

Returns the current serverconnection handler.

Return Value

the ID of the current serverconnection handler
(*type=int*)

getCurrentTransferSpeed(*transferID*)

Returns the current transfer speed of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the speed
(*type=tuple (int, float)*)

getDefaultCaptureDevice(modeID)

Queries the default capture device.

Parameters

modeID: Defines the capture mode to use
(*type=string*)

Return Value

A tuple, containing the errorcode and the default capture device as tuple (devicename, deviceid)
(*type=tuple (int, (string, string))*)

getDefaultCaptureMode()

Queries the default capture mode.

Return Value

A tuple, containing the errorcode and the default capture mode
(*type=tuple (int, string)*)

getDefaultPlaybackDevice(modeID)

Queries the default playback device.

Parameters

modeID: Defines the playback mode to use
(*type=string*)

Return Value

A tuple, containing the errorcode and the default playback device as tuple (devicename, deviceid)
(*type=tuple (int, (string, string))*)

getDefaultPlayBackMode()

Queries the default playback mode.

Return Value

A tuple, containing the errorcode and the default playback mode
(*type=tuple (int, string)*)

getDirectories(*path*, *maxLen*=256)

Returns a list of subdirectories of a path as space-separated string.

Parameters

path: the parent path
(*type=string*)

maxLen: length of the buffer, passed to the clientlib to store the path to, default value is 256
(*type=int*)

Return Value

the resulting path
(*type=string*)

getEncodeConfigValue(*serverConnectionHandlerID*, *ident*)

Queries a speex encoder option.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

ident: the encoder option to be queried
(*type=string*)

Return Value

A tuple, containing the errorcode and the flag's value
(*type=tuple (int, string)*)

getErrorMessage(*errorCode*)

Queries a printable error string for a specific error code.

Parameters

errorCode: The error code returned from all Client Lib functions
(*type=int*)

Return Value

A tuple, containing the errorcode and the resulting string
(*type=tuple (int, string)*)

getHotkeyFromKeyword(*keywords*)

Returns a list of hotkeys by its keywords.

Parameters

keywords: a list of keywords
(*type=list[str]*)

Return Value

a tuple containing the errorcode and the list of hotkeys
(*type=tuple(int, list[str])*)

getParentChannelOfChannel(*serverConnectionHandlerID, channelID*)

Returns the parent channel of another channel.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)
channelID: the ID of the channel
(*type=int*)

Return Value

a tuple, containing the errorcode and the ID of the parent channel
(*type=tuple (int, int)*)

getPermissionIDByName(*serverConnectionHandlerID, permissionName*)

Returns the ID of a permission defined by its name.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)
permissionName: name of the permission
(*type=string*)

Return Value

a tuple, containing the errorcode and the ID of the permission
(*type=tuple (int, int)*)

getPlaybackConfigValueAsFloat(*serverConnectionHandlerID*, *ident*)

Queries a playback option.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

ident: the playback option to be queried
(*type=string*)

Return Value

A tuple, containing the errorcode and the flag's value
(*type=tuple (int, float)*)

getPlaybackDeviceList(*modeID*)

Queries all available playback devices.

Parameters

modeID: Defines the playback mode to use.
(*type=string*)

Return Value

A tuple, containing the errorcode and the list of playback devices as tuple (devicename, deviceid)
(*type=tuple (int, [(string, string)])*)

getPlaybackModeList()

Queries all available playback modes.

Return Value

A tuple, containing the errorcode and the list of modes
(*type=tuple (int, [string])*)

getPluginPath(*maxLen*)

Returns the TeamSpeak 3 client's pluginpath.

Parameters

maxLen: the size of the buffer passed to the clientlib. Optional,
defaults to 256
(*type=int*)

Return Value

the pluginpath
(*type=str*)

getPreProcessorInfoValue(*serverConnectionHandlerID*, *ident*)

Query a sound preprocessor flag and returns it as string.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)
ident: the flag to be queried
(*type=string*)

Return Value

A tuple, containing the errorcode and the value of the queried flag
(*type=tuple (int, string)*)

getPreProcessorInfoValueFloat(*serverConnectionHandlerID*, *ident*)

Queries a sound preprocessor flag and returns it as float.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)
ident: the flag to be queried
(*type=string*)

Return Value

A tuple, containing the errorcode and the value of the queried flag
(*type=tuple (int, float)*)

getProfileList(*profile*)

Returns a list of existing profiles and the default profile's index in list.

Parameters

profile: the profile type, see ts3defines.PluginGuiProfile
(*type=int*)

Return Value

a tuple, containing the errorcode, the default profile's index and the profile list
(*type=tuple (int, int, [string])*)

getResourcesPath(*maxLen=256*)

Returns the ts3 resources path.

Parameters

maxLen: length of the buffer, passed to the clientlib to store the path to. Optional, defaults to 256
(*type=int*)

Return Value

the resources path
(*type=string*)

getServerConnectInfo(*serverConnectionHandlerID, maxLen=256*)

Returns the connect info (host, port and password) of a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

maxLen: length of the buffer, passed to the clientlib to store the path to, default value is 256
(*type=int*)

Return Value

a tuple, containing the errorcode, the host, the port and the password of the serverconnection
(*type=tuple (int, string, int, string)*)

getServerConnectionHandlerList()

Returns a list of serverconnection handlers.

Return Value

a tuple, containing the errorcode and the list of serverconnection handler IDs

(type=tuple (int, [int]))

getServerVariableAsInt(serverConnectionHandlerID, flag)

Returns a server variable as int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

flag: the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag

(type=tuple (int, int))

getServerVariableAsString(serverConnectionHandlerID, flag)

Returns a server variable as string value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

flag: the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag

(type=tuple (int, string))

getServerVariableAsUInt64(*serverConnectionHandlerID*, *flag*)

Returns a server variable as unsigned long long int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

flag: the flag to return
(*type=int*)

Return Value

a tuple, containing the errorcode and the value of the flag
(*type=tuple (int, int)*)

getServerVersion(*serverConnectionHandlerID*)

Returns the server version.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

the server version
(*type=int*)

getTransferFileName(*transferID*)

Returns the filename of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the filename
(*type=tuple (int, string)*)

getTransferFilePath(*transferID*)

Returns the filepath of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the filepath
(*type=tuple (int, string)*)

getTransferFileSize(*transferID*)

Returns the total filesize (in Bytes) of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the filesize
(*type=tuple (int, int)*)

getTransferFileSizeDone(*transferID*)

Returns the already downloaded size (in Bytes) of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the size
(*type=tuple (int, int)*)

getTransferRunTime(*transferID*)

Returns the runtime of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the runtime in seconds
(*type=tuple (int, int)*)

getTransferStatus(*transferID*)

Returns the status of a filetransfer, whether if it is initialising, active or finished see `ts3defines.FileTransferState`

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and the status
(*type=tuple (int, int)*)

guiConnect(*connectTab, serverLabel, serverAddress, serverPassword, nickname, channel, channelPassword, captureProfile, playbackProfile, hotkeyProfile, userIdentity, oneTimeKey, phoneticName*)

Connects to a server and displays it as tab in the client.

Parameters

connectTab:	defines, which tab will be used, see ts3defines.PluginConnectTab (<i>type=int</i>)
serverLabel:	the label of the connection (<i>type=string</i>)
serverAddress:	host or ip address (<i>type=string</i>)
serverPassword:	password to the server, pass an empty string if the server is not password protected (<i>type=string</i>)
nickname:	the user's nickname (<i>type=string</i>)
channel:	complete path to the channel to connect to (<i>type=string</i>)
channelPassword:	password to the channel, pass an empty string if the channel is not password protected (<i>type=string</i>)
captureProfile:	the name of the capture profile to use (<i>type=string</i>)
playbackProfile:	the name of the playback profile to use (<i>type=string</i>)
hotkeyProfile:	the name of the hotkey profile to use (<i>type=string</i>)
userIdentity:	identity to use (<i>type=string</i>)
oneTimeKey:	privilege key to use on connect (<i>type=string</i>)
phoneticName:	phonetic nickname (<i>type=string</i>)

Return Value

a tuple, containing the errorcode and the ID of the created
serverconnection handler
(*type=tuple (int, int)*)

guiConnectBookmark(*connectTab*, *bookmarkuid*)

Connects to a server from a bookmark and displays it as tab in the client.

Parameters

connectTab: defines, which tab will be used, see
ts3defines.PluginConnectTab
(*type=int*)

bookmarkuid: UID of the bookmark
(*type=string*)

Return Value

a tuple, containing the errorcode and the ID of the created
serverconnection handler
(*type=tuple (int, int)*)

haltTransfer(*serverConnectionHandlerID*, *transferID*, *deleteUnfinishedFile*, *returnCode*)

Halts a currently running filetransfer.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

transferID: the ID of the filetransfer
(*type=int*)

deleteUnfinishedFile: if set to 1 (or True) and the file is
not yet finished, it will be deleted; to
prevent, pass 0 (or False)
(*type=int or bool*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

initiateGracefulPlaybackShutdown(*serverConnectionHandlerID*)

Graceful shutdown the playback device on a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

the errorcode
(*type=int*)

isReceivingWhisper(*serverConnectionHandlerID*, *clientID*)

//FIXME:

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

Return Value

a tuple, containing the errorcode and the status
(*type=tuple (int, int or bool)*)

isTransferSender(*transferID*)

Returns the upload/download direction of a filetransfer.

Parameters

transferID: the ID of the filetransfer
(*type=int*)

Return Value

a tuple, containing the errorcode and 1 if it's an upload or 0 if it's a download
(*type=tuple (int, int or bool)*)

isWhispering(*serverConnectionHandlerID*, *clientID*)

Returns the status of a client whether he is currently whispering to the own client.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

a tuple, containing the errorcode and the status
(*type=tuple (int, int or bool)*)

logMessage(*logMessage*, *severity*, *channel*, *logID*)

Logs a string.

Parameters

logMessage: Text which should be logged
(*type=string*)

severity: The level of the message, warning or error. Defined by the class LogLevel
(*type=int*)

channel: Custom text to categorize the message channel
(*type=string*)

logID: ID of the serverconnection to identify the current server connection when using multiple connections, 0 if unused
(*type=int*)

Return Value

The errorcode
(*type=int*)

openCaptureDevice(*serverConnectionHandlerID*, *modeID*, *captureDevice*)

Opens a playback device on a serverconnection.

Parameters

serverConnectionHandlerID:	ID of the serverconnection on which the capture device should be initialized on <i>(type=int)</i>
modeID:	the playback mode to use <i>(type=string)</i>
captureDevice:	the id of the capture device <i>(type=string)</i>

Return Value

the errorcode
(type=int)

openPlaybackDevice(*serverConnectionHandlerID*, *modeID*, *playbackDevice*)

Opens a playback device on a serverconnection.

Parameters

serverConnectionHandlerID:	ID of the serverconnection on which the playback device should be initialized on <i>(type=int)</i>
modeID:	the playback mode to use <i>(type=string)</i>
playbackDevice:	the id of the playback device <i>(type=string)</i>

Return Value

the errorcode
(type=int)

pauseWaveFileHandle(*serverConnectionHandlerID*, *waveHandle*, *pause*)

Pauses a wavefile sound previously started with playWaveFileHandle.

Parameters

serverConnectionHandlerID: the ID of the serverConnection the sound is played on
(*type=int*)

waveHandle: the handle returned by playWaveFileHandle
(*type=int*)

pause: if set to 1 (or True), the sound will pause, 0 (or False) will unpause the sound
(*type=int or bool*)

Return Value

the errorcode
(*type=int*)

playWaveFile(*serverConnectionHandlerID*, *path*)

Plays a wavefile sound on a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

path: the path to the wavefile on the system
(*type=string*)

Return Value

the errorcode
(*type=int*)

playWaveFileHandle(*serverConnectionHandlerID*, *path*, *loop*)

Plays a wavefile sound on a serverconnection and returns a handle to it.

Parameters

serverConnectionHandlerID: the ID of the serverconnection on which the sound will be played on
(*type=int*)

path: the path to the wavefile on the system
(*type=string*)

loop: if set to 1 (or True), the sound will loop
(*type=int or bool*)

Return Value

A tuple, containing the errorcode and the handle, with which the sound can be paused and unpaused
(*type=tuple (int, int)*)

printMessage(*serverConnectionHandlerID*, *message*, *messageTarget*)

Prints a message to a specific client chat tab.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

message: the message to print
(*type=string*)

messageTarget: the target to send the message, see `ts3defines.PluginMessageTarget`
(*type=int*)

printMessageToCurrentTab(*message*)

Prints a message to the currently visible tab.

Parameters

message: the message to send
(*type=string*)

privilegeKeyUse(*serverConnectionHandlerID*, *tokenKey*, *returnCode*)

Uses a privilege key as the current client of the serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection

(*type=int*)

tokenKey: the token

(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.

(*type=string*)

Return Value

the errorcode

(*type=int*)

processCustomCaptureData(*deviceName*, *buffer*)

Sends captured data from a custom device to the client libg

Parameters

deviceName: the name of the device capturing the data, previously
registered with registerCustomDevice

(*type=string*)

buffer: a list containing the buffered data

(*type=list [int]*)

Return Value

the errorcode

(*type=int*)

registerCustomDevice(*deviceID*, *deviceDisplayName*, *capFrequency*,
capChannels, *playFrequency*, *playChannels*)

Registers a custom device, announcing the device ID and name to the Client Lib.

Parameters

deviceID:	ID string of the custom device, under which the device can be later accessed (<i>type=string</i>)
deviceDisplayName:	Displayed name of the custom device. Freely choose a name which identifies your device (<i>type=string</i>)
capFrequency:	Frequency of the capture device (<i>type=int</i>)
capChannels:	Number of channels of the capture device. This value depends on if the used codec is a mono or stereo CodecEncryptionMode (<i>type=int</i>)
playFrequency:	Frequency of the playback deviceDisplayName (<i>type=int</i>)
playChannels:	Number of channels of the playback device (<i>type=int</i>)

Return Value

the errorcode
(*type=int*)

requestBanList(*serverConnectionHandlerID*, *returnCode*)

Requests the banlist on a server. The event onBanListEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelAddPerm(*serverConnectionHandlerID*, *channelID*,
permissionIDArray, *permissionValueArray*, *returnCode*)

Adds a list of permissions to a channel.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

permissionIDArray: a list of permission IDs
(*type=list [int]*)

permissionValueArray: list of permission values, in order of
the permissions in
permissionIDArray
(*type=list [int]*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelClientAddPerm(*serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *permissionIDArray*, *permissionValueArray*, *returnCode*)

Adds a list of permissions on a channel to a user.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
clientDatabaseID:	the database ID of the user (<i>type=int</i>)
permissionIDArray:	a list of permission IDs (<i>type=list [int]</i>)
permissionValueArray:	list of permission values, in order of the permissions in permissionIDArray (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelClientDelPerm(*serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions of a user in a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
clientDatabaseID:	the database ID of the user (<i>type=int</i>)
permissionIDArray:	a list of permission IDs (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelClientPermList(*serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *returnCode*)

Requests the list of permissions of a user in a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
clientDatabaseID:	the database ID of the user (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelDelete(*serverConnectionHandlerID*, *channelID*, *force*, *returnCode*)

Deletes a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel to delete (<i>type=int</i>)
force:	if set to 1 (or True), the channel will be deleted even when it is not empty (<i>type=int or bool</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelDelPerm(*serverConnectionHandlerID*, *channelID*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions from a channel.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

permissionIDArray: a list of permission IDs
(*type=list [int]*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelDescription(*serverConnectionHandlerID*, *channelID*, *returnCode*)

Requests the channel description of a channel. Afterwards, `getChannelVariableAsString` can return it.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelGroupAdd(*serverConnectionHandlerID*, *groupName*,
groupType, *returnCode*)

Adds a channelgroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
groupName:	the name of the channelgroup to create (<i>type=string</i>)
groupType:	type of the channelgroup, see ts3defines.GroupType (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelGroupAddPerm(*serverConnectionHandlerID*,
channelGroupID, *continueonerror*, *permissionIDArray*, *permissionValueArray*,
returnCode)

Adds a list of permissions to a channelgroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelGroupID:	the ID of the channelgroup (<i>type=int</i>)
continueonerror:	if set to True, if an error with a permission occurs, the other permissions will even though be handled (<i>type=bool</i>)
permissionIDArray:	a list of permission IDs (<i>type=list [int]</i>)
permissionValueArray:	list of permission values, in order of the permissions in permissionIDArray (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelGroupDel(*serverConnectionHandlerID*, *channelGroupID*, *force*, *returnCode*)

Deletes a channelgroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelGroupID:	the ID of the channelgroup (<i>type=int</i>)
force:	if set to 1 (or True), even if there are users assigned to this channelgroup, it will be deleted (<i>type=int or bool</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelGroupDelPerm(*serverConnectionHandlerID*,
channelGroupID, *continueOnError*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions from a channelgroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelGroupID:	the ID of the channelgroup (<i>type=int</i>)
continueOnError:	if set to 1 (or True), if an error with a permission occurs, the other permissions will even though be handled (<i>type=int or bool</i>)
permissionIDArray:	a list of permission IDs (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelGroupList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of channelgroups. The events onChannelGroupListEvent and onChannelGroupListEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelGroupPermList(*serverConnectionHandlerID*,
channelGroupID, *returnCode*)

Requests the list of permissions assigned to a channelgroup. The events onChannelGroupPermListEvent and onChannelGroupPermListFinishedEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelGroupID: the ID of the channelGroupID
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

```
requestChannelMove(serverConnectionHandlerID, channelID,  
newChannelParentID, newChannelOrder, returnCode)
```

Moves a channel to a new parent channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel to move (<i>type=int</i>)
newChannelParentID:	the ID of the new parent channel (<i>type=int</i>)
newChannelOrder:	Channel order defining where the channel should be sorted under the new parent. Pass 0 to sort the channel right after the parent (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestChannelPermList(*serverConnectionHandlerID*, *channelID*, *returnCode*)

Requests the list of permissions assigned to a channel. The events `onChannelPermListEvent` and `onChannelPermListFinishedEvent` will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

returnCode: returnCode passed to `onServerErrorEvent` or `onServerPermissionErrorEvent`. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelSubscribe(*serverConnectionHandlerID*, *channelIDArray*, *returnCode*)

Subscribes to a list of channels to get notifications of the clients in them.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelIDArray: a list of channel IDs
(*type=list [int]*)

returnCode: returnCode passed to `onServerErrorEvent` or `onServerPermissionErrorEvent`. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelSubscribeAll(*serverConnectionHandlerID*, *returnCode*)

Subscribes to all channels on the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelUnsubscribe(*serverConnectionHandlerID*, *channelIDArray*, *returnCode*)

Unsubscribes from a list channels.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelIDArray: a list of channel IDs
(*type=list [int]*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestChannelUnsubscribeAll(*serverConnectionHandlerID*, *returnCode*)

Unsubscribes from all channels on the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientAddPerm(*serverConnectionHandlerID*, *clientDatabaseID*,
permissionIDArray, *permissionValueArray*, *permissionSkipArray*, *returnCode*)

Adds a list of permissions to a user.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientDatabaseID: the database ID of the user
(*type=int*)

permissionIDArray: a list of permission IDs
(*type=list [int]*)

permissionValueArray: list of permission values, in order of
the permissions in
permissionIDArray
(*type=list [int]*)

permissionSkipArray: list of permission skip values, in
order of the permissions in
permissionIDArray
(*type=list [int]*)

Return Value

the errorcode
(*type=int*)

requestClientDBIDfromUID(*serverConnectionHandlerID*,
clientUniqueIdentifier, *returnCode*)

Requests the database ID of a client defined by the UID. The event onClientDBIDfromUIDEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientUniqueIdentifier: the UID of the client
(*type=string*)

returnCode: returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientDelPerm(*serverConnectionHandlerID*, *clientDatabaseID*,
permissionIDArray, *returnCode*)

Deletes a list of permissions from a user.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientDatabaseID: the database ID of the user
(*type=int*)

permissionIDArray: a list of permission IDs
(*type=list [int]*)

returnCode: returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientEditDescription(*serverConnectionHandlerID*, *clientID*, *clientDescription*, *returnCode*)

Sets the description of a client.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

clientDescription: the description to set
(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientIDs(*serverConnectionHandlerID*, *clientUniqueIdentifier*, *returnCode*)

Requests the client IDs for a given UID. Will trigger the event
onClientIDsEvent.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientUniqueIdentifier: the UID of the client
(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientKickFromChannel(*serverConnectionHandlerID*, *clientID*, *kickReason*, *returnCode*)

Kicks a client from its current channel to the default one.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client to kick
(*type=int*)

kickReason: the reason for the kick
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientKickFromServer(*serverConnectionHandlerID*, *clientID*, *kickReason*, *returnCode*)

Kicks a client from the server.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client to kick
(*type=int*)

kickReason: the reason for the kick
(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientMove(*serverConnectionHandlerID*, *clientID*, *newChannelID*, *password*, *returnCode*)

Moves a client to a different channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the ID of the client to be moved (<i>type=int</i>)
newChannelID:	the ID of the channel moving the client to (<i>type=int</i>)
password:	password of the channel, leave empty if channel is not password protected (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestClientNamefromDBID(*serverConnectionHandlerID*,
clientDatabaseID, *returnCode*)

Requests the name of a client defined by the database ID. The event onClientNamefromDBIDEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientDatabaseID: the database ID of the client
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientNamefromUID(*serverConnectionHandlerID*,
clientUniqueIdentifier, *returnCode*)

Requests the name of a client defined by the UID. The event onClientNamefromUIDEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientUniqueIdentifier: the UID of the client
(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientPermList(*serverConnectionHandlerID*, *clientDatabaseID*, *returnCode*)

Requests the list of permissions assigned to a user. The events

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientDatabaseID: the database ID of the user
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientPoke(*serverConnectionHandlerID*, *clientID*, *message*, *returnCode*)

Pokes a client with a given message.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

message: the message
(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestClientSetIsTalker(*serverConnectionHandlerID*, *clientID*, *isTalker*, *returnCode*)

Grants or revokes the talker flag of a client.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the ID of the client (<i>type=int</i>)
isTalker:	if set to 1 (or True) grants talker flag, if 0 (or False) revokes talker flag (<i>type=int or bool</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestClientSetWhisperList(*serverConnectionHandlerID*, *clientID*, *targetChannelIDArray*, *targetClientIDArray*, *returnCode*)

Modifies the whisper list of a client.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the ID of the client whose whisperlist is modified. If set to 0, the own whisper list is modified (<i>type=int</i>)
targetChannelIDArray:	a list of channel IDs the client will whisper to. Pass None to clear it (<i>type=list [int]</i>)
targetClientIDArray:	a list of client IDs the client will whisper to. Pass None to clear it (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestClientVariables(*serverConnectionHandlerID*, *clientID*, *returnCode*)

Requests latest data for a given client. The event onUpdateClientEvent will be triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the ID of the client (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestComplainAdd(*serverConnectionHandlerID*, *targetClientDatabaseID*, *complainReason*, *returnCode*)

Adds a complain to a user defined by his database ID.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
targetClientDatabaseID:	the database ID of the user (<i>type=int</i>)
complainReason:	the reason for the complain (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestComplainDel(*serverConnectionHandlerID*, *targetClientDatabaseID*, *fromClientDatabaseID*, *returnCode*)

Deletes a complain to a user by a different user.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
targetClientDatabaseID:	the database ID of the complained user (<i>type=int</i>)
fromClientDatabaseID:	the database ID of the complaining user (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestComplainDelAll(*serverConnectionHandlerID*,
targetClientDatabaseID, *returnCode*)

Deletes all complains to a user.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

targetClientDatabaseID: the database ID of the user
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestComplainList(*serverConnectionHandlerID*, *targetClientDatabaseID*,
returnCode)

Requests the list of complains to a user. The event onComplainListEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

targetClientDatabaseID: the database ID of the user
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestConnectionInfo(*serverConnectionHandlerID*, *clientID*, *returnCode*)

Requests the connection info of a client. The event onConnectionInfoEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the ID of the client
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestCreateDirectory(*serverConnectionHandlerID*, *channelID*, *channelPW*, *directoryPath*, *returnCode*)

Creates a directory in a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
channelPW:	the password of the channel, pass an empty string if channel is not password protected (<i>type=string</i>)
directoryPath:	the complete path of the to be created directory (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestDeleteFile(*serverConnectionHandlerID*, *channelID*, *channelPW*, *files*, *returnCode*)

Deletes a list of files in a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
channelPW:	the password of the channel, pass an empty string if channel is not password protected (<i>type=string</i>)
files:	a list of complete pathes of the file to delete (<i>type=list [string]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestFile(*serverConnectionHandlerID*, *channelID*, *channelPW*, *file*, *overwrite*, *resume*, *destinationDirectory*, *returnCode*)

Starts a filedownload from the server.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel in which the file is placed in (<i>type=int</i>)
channelPW:	the password of the channel, pass an empty string if the channel is not password protected (<i>type=string</i>)
file:	the complete path to the file in the channel (<i>type=string</i>)
overwrite:	if set to 1 (or True) and a file with that name already exists will be overwritten (<i>type=int or bool</i>)
resume:	if set to 1 (or True), a previously started filetransfer can be resumed (<i>type=int or bool</i>)
destinationDirectory:	the path to the directory, where the downloaded fill will be placed in (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

a tuple, containing the errorcode and the ID of the filetransfer
(*type=tuple (int, int)*)

requestFileInfo(*serverConnectionHandlerID*, *channelID*, *channelPW*, *file*, *returnCode*)

Requests the info to a file in a channel. The event `onFileInfoEvent` will be triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
channelPW:	the password of the channel, pass an empty string if the channel is not password protected (<i>type=string</i>)
file:	the complete path to the file (<i>type=string</i>)
returnCode:	returnCode passed to <code>onServerErrorEvent</code> or <code>onServerPermissionErrorEvent</code> . Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestFileList(*serverConnectionHandlerID*, *channelID*, *channelPW*, *path*, *returnCode*)

Requests the filelist of a channel. The events `onFileListEvent` and `onFileListFinishedEvent` will be triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
channelPW:	the password of the channel, pass an empty string if the channel is not password protected (<i>type=string</i>)
path:	the path of the directory to be listed, pass '/' for the root path (<i>type=string</i>)
returnCode:	returnCode passed to <code>onServerErrorEvent</code> or <code>onServerPermissionErrorEvent</code> . Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestHotkeyInputDialog(*keyword, isDown, qParentWindow*)

Shows the hotkeyinputdialog to set the hotkey for a hotkey keyword. When finished ts3plugin.onHotkeyRecordedEvent will be called with the recorded hotkey.

Parameters

keyword:	the global keyword (see PluginHost.globalHotkeyKeyword) (<i>type=str</i>)
isDown:	if True, the hotkey will be triggered on keypress, on keyrelease otherwise (<i>type=bool</i>)
qParentWindow:	the window on which the dialog is shown modal to, optional (<i>type=QWidget</i>)

requestInfoUpdate(*serverConnectionHandlerID, itemType, itemID*)

Requests to update the info data.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
itemType:	specifies, which info data update is requested, see ts3defines.PluginItemType (<i>type=int</i>)
itemID:	the ID of the item to update (only usefull if itemType != ts3defines.PluginItemType.PLUGIN_MENU_TYPE_GLOBAL) (<i>type=int</i>)

Return Value

the errorcode
(*type=int*)

requestIsTalker(*serverConnectionHandlerID*, *isTalkerRequest*,
isTalkerRequestMessage, *returnCode*)

Requests talk power or revokes the talk power request.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
isTalkerRequest:	if set to 1 (or True) requests talk power, if 0 (or False) revokes the talk power request (<i>type=int or bool</i>)
isTalkerRequestMessage:	the message of the request (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestMessageAdd(*serverConnectionHandlerID*, *toClientUID*, *subject*, *message*, *returnCode*)

Sends an offline message to another user.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
toClientUID:	the UID of the user (<i>type=string</i>)
subject:	the subject of the message (<i>type=string</i>)
message:	the message (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestMessageDel(*serverConnectionHandlerID*, *messageID*, *returnCode*)

Deletes an offline message.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
messageID:	the ID of the message (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestMessageGet(*serverConnectionHandlerID*, *messageID*, *returnCode*)

Requests an offline message defined by its ID. The event onMessageGetEvent will be triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
messageID:	the ID of the message (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestMessageList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of offline messages. The event onMessageListEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestMessageUpdateFlag(*serverConnectionHandlerID*, *messageID*, *flag*, *returnCode*)

Sets the message read/unread flag of an offline message

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

messageID: the ID of the message
(*type=int*)

flag: set to 0 to set message as unread,
set to 1 to set message as read
(*type=*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestMuteClients(*serverConnectionHandlerID*, *clientIDArray*, *returnCode*)

Mutes a list of clients.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientIDArray: a list of client IDs
(*type=list [int]*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestPermissionList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of permissions available on the server. The events onPermissionListEvent and onPermissionListFinishedEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestPermissionOverview(*serverConnectionHandlerID*, *clientDBID*, *channelID*, *returnCode*)

Requests the permission overview of a user in a channel. The events `onPermissionOverviewEvent` and `onPermissionOverviewFinishedEvent` will be triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientDBID:	the database ID of the user (<i>type=int</i>)
channelID:	the ID of the channel (<i>type=int</i>)
returnCode:	returnCode passed to <code>onServerErrorEvent</code> or <code>onServerPermissionErrorEvent</code> . Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestRenameFile(*serverConnectionHandlerID, fromChannelID, channelPW, toChannelID, toChannelPW, oldFile, newFile, returnCode*)

Renames a file or moves it to another channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
fromChannelID:	the ID of the channel, the file is currently placed in (<i>type=int</i>)
channelPW:	the password of the channel, the file is currently placed in, pass an empty string if channel is not password protected (<i>type=string</i>)
toChannelID:	the ID of the channel, the file should be placed in after, pass 0, if just renaming, not moving (<i>type=int</i>)
toChannelPW:	the password of the channel, to which the file should move to, pass an empty string if channel is not password protected; this is ignored, if just renaming, not moving (<i>type=string</i>)
oldFile:	the complete path to the file (<i>type=string</i>)
newFile:	the complete path to the new filename (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestSendChannelTextMsg(*serverConnectionHandlerID*, *message*, *targetChannelID*, *returnCode*)

Sends a text message to all clients in a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
message:	the message to send (<i>type=string</i>)
targetChannelID:	the ID of the channel (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestSendClientQueryCommand(*serverConnectionHandlerID*, *command*, *returnCode*)

Requests to execute a clientquery command.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
command:	the command to execute (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestSendPrivateTextMsg(*serverConnectionHandlerID*, *message*, *targetClientID*, *returnCode*)

Sends a private text message to a client.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
message:	the message to send (<i>type=string</i>)
targetClientID:	the ID of the client to send the message to (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestSendServerTextMsg(*serverConnectionHandlerID*, *message*, *returnCode*)

Sends a text message to all clients on the server.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
message:	the message to send (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestServerGroupAdd(*serverConnectionHandlerID*, *groupName*,
groupType, *returnCode*)

Adds a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
groupName:	the name of the group to create (<i>type=string</i>)
groupType:	type of the servergroup, see ts3defines.GroupType (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

```
requestServerGroupAddClient(serverConnectionHandlerID,  
serverGroupID, clientDatabaseID, returnCode)
```

Adds a user defined by his database ID to a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the ID of the servergroup (<i>type=int</i>)
clientDatabaseID:	the database ID of the user (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestServerGroupAddPerm(*serverConnectionHandlerID*,
serverGroupID, *continueonerror*, *permissionIDArray*, *permissionValueArray*,
permissionNegatedArray, *permissionSkipArray*, *returnCode*)

Adds a list of permissions to a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the ID of the servergroup (<i>type=int</i>)
continueonerror:	if set to 1 (or True), if an error with a permission occurs, the other permissions will even though be handled (<i>type=int or bool</i>)
permissionIDArray:	list of permission IDs (<i>type=list [int]</i>)
permissionValueArray:	list of permission values, in order of the permissions in permissionIDArray (<i>type=list [int]</i>)
permissionNegatedArray:	list of permission negated values, in order of the permissions in permissionIDArray (<i>type=list [int]</i>)
permissionSkipArray:	list of permission skip values, in order of the permissions in permissionIDArray (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestServerGroupClientList(*serverConnectionHandlerID*,
serverGroupID, *withNames*, *returnCode*)

Requests the list of clients assigned to a servergroup. The event onServerGroupClientListEvent will be triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the ID of the servergroup (<i>type=int</i>)
withNames:	if set to 1 (or True), the event will contain the nick and uid of the user instead of empty strings (<i>type=int or bool</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestServerGroupDel(*serverConnectionHandlerID*, *serverGroupID*, *force*, *returnCode*)

Deletes a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the ID of the servergroup (<i>type=int</i>)
force:	if set to 1 (or True), even if there are users assigned to this servergroup, it will be deleted (<i>type=int or bool</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

```
requestServerGroupDelClient(serverConnectionHandlerID,  
serverGroupID, clientDatabaseID, returnCode)
```

Deletes a user defined by his database ID from a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the ID of the servergroup (<i>type=int</i>)
clientDatabaseID:	the database ID of the user (<i>type=int</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestServerGroupDelPerm(*serverConnectionHandlerID*, *serverGroupID*, *continueOnError*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions from a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the ID of the servergroup (<i>type=int</i>)
continueOnError:	if set to 1 (or True), if an error with a permission occurs, the other permissions will even though be handled (<i>type=int or bool</i>)
permissionIDArray:	list of permission IDs (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestServerGroupList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of servergroups. The events onServerGroupListEvent and onServerGroupListFinishedEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestServerGroupPermList(*serverConnectionHandlerID*,
serverGroupID, *returnCode*)

Requests the list of permissions assigned to a servergroup. The events onServerGroupPermListEvent and onServerGroupPermListFinishedEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

serverGroupID: the ID of the servergroup
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)


```
requestServerGroupsByClientID(serverConnectionHandlerID,  
clientDatabaseID, returnCode)
```

Requests all servergroups of a user defined by his database ID. The event onServerGroupByClientIDEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection

(*type=int*)

clientDatabaseID: the database ID of the user

(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.

(*type=string*)

Return Value

the errorcode

(*type=int*)

requestServerTemporaryPasswordAdd(*serverConnectionHandlerID*,
password, *description*, *duration*, *targetChannelID*, *targetChannelPW*,
returnCode)

Adds a temporary password to the server.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
password:	the temporary password (<i>type=string</i>)
description:	the description of the temporary password (<i>type=string</i>)
duration:	the duration in seconds (<i>type=int</i>)
targetChannelID:	the ID of the channel to which the accessing clients will join by default (<i>type=int</i>)
targetChannelPW:	the password of the targetChannel, pass an empty string, if the channel is not password protected (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

requestServerTemporaryPasswordDel(*serverConnectionHandlerID*,
password, *returnCode*)

Deletes an existing temporary password.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

password: the password to delete
(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestServerTemporaryPasswordList(*serverConnectionHandlerID*,
returnCode)

Requests a list of existing temporary passwords. The event
onServerTemporaryPasswordListEvent will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestServerVariables(*serverConnectionHandlerID*)

Requests all server variables of a serverconnection. The event `onServerUpdatedEvent` will be triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

the errorcode
(*type=int*)

requestSetClientChannelGroup(*serverConnectionHandlerID*, *channelGroupIDArray*, *channelIDArray*, *clientDatabaseIDArray*, *returnCode*)

Adds a list of users to a list of channelgroups.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelGroupIDArray: a list of channelgroup IDs
(*type=list [int]*)

channelIDArray: a list of channel IDs
(*type=list [int]*)

clientDatabaseIDArray: a list of client database IDs
(*type=list [int]*)

returnCode: returnCode passed to `onServerErrorEvent` or `onServerPermissionErrorEvent`. Optional.
(*type=string*)

Return Value

the errorcode
(*type=int*)

requestUnmuteClients(*serverConnectionHandlerID*, *clientIDArray*,
returnCode)

Unmutes a list of clients.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientIDArray:	a list of client IDs (<i>type=list [int]</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

sendFile(*serverConnectionHandlerID*, *channelID*, *channelPW*, *file*, *overwrite*, *resume*, *sourceDirectory*, *returnCode*)

Starts a fileupload to the server.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel in which the file will be placed in (<i>type=int</i>)
channelPW:	the password of the channel, pass an empty string if the channel is not password protected (<i>type=string</i>)
file:	the complete path to the file in the channel (<i>type=string</i>)
overwrite:	if set to 1 (or True) and a file with that name already exists will be overwritten (<i>type=int or bool</i>)
resume:	if set to 1 (or True), a previously started filetransfer can be resumed (<i>type=int or bool</i>)
sourceDirectory:	the directory on the system, where the original file is placed in (<i>type=string</i>)
returnCode:	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. (<i>type=string</i>)

Return Value

a tuple, containing the errorcode and the ID of the filetransfer
(*type=tuple (int, int)*)

sendPluginCommand(*serverConnectionHandlerID*, *command*, *targetMode*, *targetIDs*, *returnCode*)

Sends a plugin command to other users.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

command: the command string
(*type=string*)

targetMode: specifies, to whom the command will be send, see
ts3defines.PluginTargetMode
(*type=int*)

targetIDs: a list of client IDs, only needed if
targetMode ==
ts3defines.PluginTargetMode.PluginCommandTarget_CLIENT
(*type=list [int]*)

Return Value

the errorcode (as this is wrapped in python, there could be errors)
(*type=int*)

serverPropertyStringToFlag(*serverPropertyString*)

Converts a server property name used in strings (eg the serverquery) to the corresponding flag.

Parameters

serverPropertyString: the lowercase string representation
(*type=str*)

Return Value

a tuple, containing the errorcode and the flag (see
ts3defines.ClientProperties and ts3defines.ClientPropertiesRare)
(*type=tuple (int, int)*)

set3DWaveAttributes(*serverConnectionHandlerID, waveHandle, position*)

Positions a wave file that was opened previously with playWaveFileHandle in 3D space.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

waveHandle: handle of the played wavefile sound
returned by playWaveFileHandle
(*type=int*)

position: A tuple defining the 3D position of
the sound
(*type=tuple (float, float, float)*)

Return Value

the errorcode
(*type=int*)

setChannelVariableAsInt(*serverConnectionHandlerID, channelID, flag, value*)

Sets a channel variable to a new int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel, pass 0 to set a
new channel's variables
(*type=int*)

flag: the flag to set
(*type=int*)

value: the new value
(*type=int*)

Return Value

the errorcode
(*type=int*)

setChannelVariableAsString(*serverConnectionHandlerID*, *channelID*, *flag*, *value*)

Sets a channel variable to a new string value.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel, pass 0 to set a new channel's variables (<i>type=int</i>)
flag:	the flag to set (<i>type=int</i>)
value:	the new value (<i>type=int</i>)

Return Value

the errorcode
(*type=int*)

setChannelVariableAsUInt64(*serverConnectionHandlerID*, *channelID*, *flag*, *value*)

Sets a channel variable to a new unsigned long long int value.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the ID of the channel, pass 0 to set a new channel's variables (<i>type=int</i>)
flag:	the flag to set (<i>type=int</i>)
value:	the new value (<i>type=int</i>)

Return Value

the errorcode
(*type=int*)

setClientSelfVariableAsInt(*serverConnectionHandlerID, flag, value*)

Sets a variable of the own client to a new int value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

flag: the flag to set
(*type=int*)

value: the new value
(*type=int*)

Return Value

the errorcode
(*type=int*)

setClientSelfVariableAsString(*serverConnectionHandlerID, flag, value*)

Sets a variable of the own client to a new string value.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

flag: the flag to set
(*type=int*)

value: the new value
(*type=string*)

Return Value

the errorcode
(*type=int*)

setClientVolumeModifier(*serverConnectionHandlerID*, *clientID*, *value*)

Sets the volume modifier of a client.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

clientID: the client's ID
(type=int)

value: the value to set
(type=float)

Return Value

the errorcode
(type=int)

setPlaybackConfigValue(*serverConnectionHandlerID*, *ident*, *value*)

Sets a playback option.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

ident: the playback option to reset
(type=string)

value: the value to set
(type=string)

Return Value

the errorcode
(type=int)

setPluginMenuEnabled(*menuID*, *enabled*)

Enables or disables a menuitem. The menuID must be the global id, not the local id plugin developers set in menuItems. Retrieve it with PluginHost.globalMenuID.

Parameters

menuID: global id of the menuitem
(type=int)

enabled: set to True to enable it, False otherwise
(type=bool)

setPreProcessorConfigValue(*serverConnectionHandlerID, ident, value*)

Sets a sound preprocessor flag.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

ident: the flag to be set
(*type=string*)

value: the value to set the flag to
(*type=string*)

Return Value

the errorcode
(*type=int*)

showHotkeySetup()

Opens the hotkey settings in the TeamSpeak 3 client's settings dialog.

spawnNewServerConnectionHandler(*port*)

Creates a new server connection handler and receive its ID.

Parameters

port: Port the client should bind on. Specify zero to let the operating system chose any free port
(*type=int*)

Return Value

A tuple, containig the errorcode and the resulting ID
(*type=tuple (int, int)*)

startConnection(*serverConnectionHandlerID*, *identity*, *ip*, *port*, *nickname*, *defaultChannelArray*, *defaultChannelPassword*, *serverPassword*)

Starts a connection to the given server.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
identity:	the client's identity (<i>type=string</i>)
ip:	hostname or ip of the server (<i>type=string</i>)
port:	port of the server (<i>type=int</i>)
nickname:	the client's nickname (<i>type=string</i>)
defaultChannelArray:	list of strings defining the path to a channel on the server, pass empty list to join in server's default channel (<i>type=list [string]</i>)
defaultChannelPassword:	password of the default channel, pass an empty string if not using defaultChannelArray or channel is not password protected (<i>type=string</i>)
serverPassword:	password of the server, pass an empty string if the server is not password protected (<i>type=string</i>)

Return Value

the errorcode
(*type=int*)

startVoiceRecording(*serverConnectionHandlerID*)

Sets the voice recording flag of the own client on a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

the errorcode
(*type=int*)

stopConnection(*serverConnectionHandlerID*, *quitMessage*)

Stops the connection of a serverconnection.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

quitMessage: a message displayed when leaving
the server encoded in UTF-8
(*type=string*)

Return Value

the errorcode
(*type=int*)

stopVoiceRecording(*serverConnectionHandlerID*)

Removes the voice recording flag of the own client on a serverconnection

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

Return Value

the errorcode
(*type=int*)

systemset3DListenerAttributes(*serverConnectionHandlerID*, *position*, *forward*, *up*)

Sets the position, velocity and orientation of the own client in 3D space

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

position: A tuple defining the 3D position,
pass None to ignore
(*type=tuple (float, float, float)*)

forward: A tuple defining the forward
orientation of the listener. The
vector must be of unit length and
perpendicular to the up vector. Pass
None to ignore.
(*type=tuple (float, float, float)*)

up: A tuple defining the upward
orientation of the listener. The
vector must be of unit length and
perpendicular to the forward vector.
Pass None to ignore.
(*type=tuple (float, float, float)*)

Return Value

the errorcode
(*type=int*)

systemset3DSettings(*serverConnectionHandlerID*, *distanceFactor*, *rolloffScale*)

Adjust 3D sound system settings.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

distanceFactor: relative distance factor. Default is 1.0 = 1 meter
(*type=float*)

rolloffScale: Scaling factor for 3D sound rolloff. Defines how fast sound volume will attenuate. As higher the value, as faster the sound is toned with increasing distance.
(*type=float*)

Return Value

the errorcode
(*type=int*)

unregisterCustomDevice(*deviceID*)

Unregisters a custom device, previously registered with registerCustomDevice.

Parameters

deviceID: the ID of the device, used in registerCustomDevice
(*type=string*)

Return Value

the errorcode
(*type=int*)

urlsToBB(*text*, *maxLen*=256)

Converts an url to the BB-code representation.

Parameters

text: the url

(*type=string*)

maxLen: length of the buffer, passed to the clientlib to store the path to, default value is 256

(*type=int*)

Return Value

the BB-code representation

(*type=string*)

verifyChannelPassword(*serverConnectionHandlerID*, *channelID*, *channelPassword*, *returnCode*)

Verifies the password to a channel. The result can be checked in onServerErrorEvent.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the ID of the channel
(*type=int*)

channelPassword: the password to be verified
(*type=string*)

returnCode: returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(*type=string*)

Return Value

the errorcode

(*type=int*)

verifyServerPassword(*serverConnectionHandlerID*, *serverPassword*,
returnCode)

Verifies the password to a server. The result can be checked in
onServerErrorEvent.

Parameters

serverConnectionHandlerID: the ID of the serverconnection

(*type=int*)

serverPassword: the password to be verified

(*type=string*)

returnCode: returnCode passed to
onServerErrorEvent or
onServerPermissionErrorEvent.
Optional.

(*type=string*)

Return Value

the errorcode

(*type=int*)

15 Module ts3plugin

15.1 Class PluginMount



Mountpoint for ts3plugins. This class is used as metaclass for ts3plugin-subclasses to autodetect classes and add them to the PluginHost.

15.1.1 Methods

__init__(*cls, name, bases, attrs*)

x.__init__(...) initializes *x*; see `help(type(x))` for signature

Return Value
the object's type

Overrides: `object.__init__` `exitit`(inherited documentation)

Inherited from type

`__call__()`, `__delattr__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__gt__()`, `__hash__()`, `__instancecheck__()`, `__le__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setattr__()`, `__subclasscheck__()`, `__subclasses__()`, `mro()`

Inherited from object

`__format__()`, `__reduce__()`, `__reduce_ex__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

15.1.2 Properties

Name	Description
<i>Inherited from type</i>	
<code>__abstractmethods__</code> , <code>__base__</code> , <code>__bases__</code> , <code>__basicsize__</code> , <code>__dictoffset__</code> , <code>__flags__</code> , <code>__itemsizes__</code> , <code>__mro__</code> , <code>__name__</code> , <code>__weakrefoffset__</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

15.2 Class *ts3plugin*



15.2.1 Methods

`__init__(self)`

Initializes the plugin. This is called if the plugin is started. After this, the plugin's event-methods will be invoked.

Overrides: `object.__init__`

`stop(self)`

This is called, when the plugin is stopped. After this, event-methods won't be invoked any longer.

`menuCreated(self)`

This is called after the plugin's menuitems are created or the plugin was reactivated. Plugin developers can assume, that when this is called, all menuitems are enabled, disable them with `ts3lib.setPluginMenuEnabled` if necessary.

`configure(self, qParentWidget)`

This is called to show the plugin's configuration ui.

Parameters

`qParentWidget`: the reference to pyTSon's configdialog
(type=`pytsonui.ConfigurationDialog`)

infoData(*self*, *schid*, *aid*, *atype*)

If the classvariable `infoTitle` is not `None`, this is called to show information on a treeitem of the TS3 Client.

Parameters

schid: the ID of the serverconnection

(*type=int*)

aid: the id (channel or client id) of the object represented by the treeitem

(*type=int*)

atype: type of the treeitem (see `ts3defines.PluginItemType`)

(*type=int*)

Return Value

list of strings shown in the client (will be joined by a newline)

(*type=list[str]*)

processCommand(*self*, *schid*, *command*)

If the classvariable `commandKeyword` is set to a string (non-empty), this is called if the user requests a command by typing `/py commandKeyword [args]`.

Parameters

schid: the ID of the serverconnection

(*type=int*)

command: the additional arguments passed by the user

(*type=str*)

Return Value

True, if the plugin handled the command, otherwise the user will receive an error

(*type=bool*)

onServerErrorEvent(*self, schid, errorMessage, error, returnCode, extraMessage*)

This is the global error event. Independent from the return value, all pyTSon plugins will receive this event.

Parameters

schid: the ID of the serverconnection
(*type=int*)

errorMessage: the message
(*type=str*)

error: the errorcode (see ts3defines.ERROR_*)
(*type=int*)

returnCode: the returnCode of the error passed to the causal method or an empty string, if no returnCode was passed
(*type=str*)

extraMessage: additional error information
(*type=str*)

Return Value

True, if the plugin handled the command, so the client will ignore it.
If no returnCode was passed, this return value will be ignored
(*type=bool*)

onTextMessageEvent(*self, schid, targetMode, toID, fromID, fromName, fromUniqueIdentifier, message, ffIgnored*)

This is called when the client receives a textmessage from another client. Independent from the return value, all pyTson plugins will receive this event.

Parameters

schid:	the ID of the serverconnection (<i>type=int</i>)
targetMode:	the target of the message (see <code>ts3defines.TextMessageTargetMode</code>) (<i>type=int</i>)
toID:	the id of the receiver (client or channel) (<i>type=int</i>)
fromID:	the client id of the sending client (<i>type=int</i>)
fromName:	the current nick of the sending client (<i>type=str</i>)
fromUniqueIdentifier:	the uid of the sending client (<i>type=str</i>)
message:	the message (<i>type=str</i>)
ffIgnored:	if set to a value <code>!= 0</code> , the client will ignore this message independent from the return value (eg. the friend/foe manager kicked in) (<i>type=int</i>)

Return Value

True, if the plugin handled the message, so the client will ignore the message
(*type=bool*)

onClientPokeEvent(*self, schid, fromClientID, pokerName, pokerUniqueIdentity, message, ffIgnored*)

This is called when the client is poked by another client. Independent from the return value, all pyTson plugins will receive this event.

Parameters

schid:	the ID of the serverconnection (<i>type=int</i>)
fromClientID:	the id of the poking client (<i>type=int</i>)
pokerName:	the current nick of the poking client (<i>type=str</i>)
pokerUniqueIdentity:	the uid of the poking client (<i>type=str</i>)
message:	the poke message (<i>type=str</i>)
ffIgnored:	if set to a value != 0, the client will ignore this message independent from the return value (eg. the friend/foe manager kicked in) (<i>type=int</i>)

Return Value

True, if the plugin handled the poke, so the client will ignore it
(*type=bool*)

onServerPermissionErrorEvent(*self, schid, errorMessage, error, returnCode, failedPermissionID*)

This is the global error event for permission errors. Independent from the return value, all pyTSon plugins will receive this event.

Parameters

schid:	the ID of the serverconnection (<i>type=int</i>)
errorMessage:	the message (<i>type=str</i>)
error:	the errorcode (see ts3defines.ERROR_*) (<i>type=int</i>)
returnCode:	the returnCode of the error passed to the causal method or an empty string, if no returnCode was passed (<i>type=str</i>)
failedPermissionID:	id of the permission (<i>type=int</i>)

Return Value

True, if the plugin handled the error, so the client will ignore it. If no returnCode was passed, this return value will be ignored
(*type=bool*)

onUserLoggingMessageEvent(*self, logMessage, logLevel, logChannel, logID, logTime, completeLogString*)

This is called whenever a message is added to the clientlog. You should not call `ts3lib.logMessage` in this event to prevent infinite loops. This event can be called asynchronous if called from another thread than the mainthread.

Parameters

logMessage:	the message that has been logged (<i>type=</i> str)
logLevel:	the level of the message (see ts3defines.LogLevel) (<i>type=</i> int)
logChannel:	the logchannel of the message (<i>type=</i> str)
logID:	the id of the server connection handler it the message is connected to one, otherwise set to 0 (<i>type=</i> int)
logTime:	the time of the message as unix timestamp (<i>type=</i> int)
completeLogString:	all infos concatenated as string (<i>type=</i> str)

onFileTransferStatusEvent(*self*, *transferID*, *status*, *statusMessage*, *remotefileSize*, *schid*)

This is called whenever a filetransfer's status changed. This event is called asynchronous.

Parameters

transferID:	the id of the filetransfer (<i>type=int</i>)
status:	the new status (<i>type=int</i>)
statusMessage:	a statusmessage (<i>type=str</i>)
remotefileSize:	size of the file on the remote site (if uploading, this is the incompletefilesize) (<i>type=int</i>)
schid:	the ID of the serverconnection (<i>type=int</i>)

currentServerConnectionChanged(*self*, *serverConnectionHandlerID*)

This is called when the current serverconnection changed (the user switched between tabs)

Parameters

serverConnectionHandlerID:	id of the new serverconnectionhandler (<i>type=int</i>)
-----------------------------------	--

onAvatarUpdated(*self*, *serverConnectionHandlerID*, *clientID*, *avatarPath*)

This is called when a client's avatar changed.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the client's id (<i>type=int</i>)
avatarPath:	path to the avatar (<i>type=str</i>)

onBanListEvent(*self, serverConnectionHandlerID, banid, ip, name, uid, creationTime, durationTime, invokerName, invokercldbid, invokeruid, reason, numberOfEnforcements, lastNickName*)

This is called for each entry in the server's banlist after it was requested with ts3lib.requestBanList.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
banid:	id of the ban (<i>type=int</i>)
ip:	the banned ip pattern or an empty string (<i>type=str</i>)
name:	the banned name pattern or an empty string (<i>type=str</i>)
uid:	the banned uid or an empty string (<i>type=str</i>)
creationTime:	time the ban was created as unix timestamp (<i>type=int</i>)
durationTime:	duration of the ban in seconds (<i>type=int</i>)
invokerName:	nick of the creator (at time the ban was created) (<i>type=str</i>)
invokercldbid:	database id of the creator (<i>type=int</i>)
invokeruid:	uid of the creator (<i>type=str</i>)
reason:	reason for ban (<i>type=str</i>)
numberOfEnforcements:	number of times, the ban has been enforced since (<i>type=int</i>)
lastNickName:	last nickname of the last enforced client (<i>type=str</i>)

onChannelClientPermListEvent(*self*, *serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

This is called for each granted permission of a client in a specific channel requested with `ts3lib.requestChannelClientPermList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	id of the channel (<i>type=int</i>)
clientDatabaseID:	the client's database id (<i>type=int</i>)
permissionID:	id of the permission (<i>type=int</i>)
permissionValue:	value of the permission (<i>type=int</i>)
permissionNegated:	the negated flag (<i>type=int</i>)
permissionSkip:	the skip flag (<i>type=int</i>)

onChannelClientPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelID*, *clientDatabaseID*)

This is called after each permission yielded by `onChannelClientPermListEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	id of the channel (<i>type=int</i>)
clientDatabaseID:	the client's database id (<i>type=int</i>)

onChannelDescriptionUpdateEvent(*self*, *serverConnectionHandlerID*, *channelID*)

This is called whenever a channel's description is updated.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: id of the channel
(*type=int*)

onChannelGroupListEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*, *name*, *atype*, *iconID*, *saveDB*)

This is called for each channelgroup on the server requested with `ts3lib.requestChannelGroupList`.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelGroupID: id of the channelgroup
(*type=int*)

name: name of the channelgroup
(*type=str*)

atype: defines if the channelgroup is a templategroup (value==0) or a regular one (value==1)
(*type=int*)

iconID: id of the icon displayed for members or 0 if no icon is displayed
(*type=int*)

saveDB: set to 1 if memberships are saved to the server's database, otherwise set to 0
(*type=int*)

onChannelGroupListFinishedEvent(*self*, *serverConnectionHandlerID*)

This is called after each channelgroup yielded by onChannelGroupListEvent was triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

onChannelGroupPermListEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

This is called for each granted permission assigned to a channelgroup requested with ts3lib.requestChannelGroupPermList.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelGroupID: id of the channelgroup
(*type=int*)

permissionID: id of the permission
(*type=int*)

permissionValue: value of the permission
(*type=int*)

permissionNegated: negated flag of the permission
(*type=int*)

permissionSkip: skip flag of the permission
(*type=int*)

onChannelGroupPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*)

This is called after each permission yielded by onChannelGroupPermListEvent was triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelGroupID: id of the channelgroup
(*type=int*)

onChannelMoveEvent(*self, serverConnectionHandlerID, channelID, newChannelParentID, invokerID, invokerName, invokerUniqueIdentifier*)

This is called whenever a channel is moved to a new parent. If a channel is moved without changing the parent, onUpdateChannelEditedEvent is called instead.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	id of the moved channel (<i>type=int</i>)
newChannelParentID:	id of the new parent channel (<i>type=int</i>)
invokerID:	id of the moving client (<i>type=int</i>)
invokerName:	nick of the moving client (<i>type=str</i>)
invokerUniqueIdentifier:	uid of the moving client (<i>type=str</i>)

onChannelPasswordChangedEvent(*self, serverConnectionHandlerID, channelID*)

This is called whenever a channelpassword is changed.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	if of the channel (<i>type=int</i>)

onChannelPermListEvent(*self*, *serverConnectionHandlerID*, *channelID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

This is called for each granted permission of a channel requested by `ts3lib.requestChannelPermList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)
permissionID:	the id of the permission (<i>type=int</i>)
permissionValue:	the value of the permission (<i>type=int</i>)
permissionNegated:	negated flag of the permission (<i>type=int</i>)
permissionSkip:	skip flag of the permission (<i>type=int</i>)

onChannelPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelID*)

This is called after each permission yielded by `onChannelPermListEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)

onChannelSubscribeEvent(*self*, *serverConnectionHandlerID*, *channelID*)

This is called whenever a channel was subscribed.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)

onChannelSubscribeFinishedEvent(*self*, *serverConnectionHandlerID*)

This is called after the subscription on a server has finished (either after subscribing one channel, after all subscriptions of a channel family has been yielded by `onChannelSubscribeEvent` or after all subscriptions had been reset after connecting).

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

onChannelUnsubscribeEvent(*self*, *serverConnectionHandlerID*, *channelID*)

This is called whenever a channel was unsubscribed.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

channelID: the id of the channel
(*type=int*)

onChannelUnsubscribeFinishedEvent(*self*, *serverConnectionHandlerID*)

This is called after the subscription on a server has finished (either after unsubscribing one channel or after all unsubscriptions of a channel family has been yielded by `onChannelUnsubscribeEvent`).

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

```
onClientBanFromServerEvent(self, serverConnectionHandlerID, clientID, oldChannelID, newChannelID, visibility, kickerID, kickerName, kickerUniqueIdentifier, time, kickMessage)
```

This is called after a client was banned from the server.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the banned client (<i>type=int</i>)
oldChannelID:	the id of the last channel, the banned client was in (<i>type=int</i>)
newChannelID:	always set to 0 (<i>type=int</i>)
visibility:	always set to ts3defines.Visibility.LEAVE_VISIBILITY (<i>type=int</i>)
kickerID:	id of the banning client (<i>type=int</i>)
kickerName:	nick of the banning client (<i>type=str</i>)
kickerUniqueIdentifier:	uid of the banning client (<i>type=str</i>)
time:	duration of the ban in seconds (<i>type=int</i>)
kickMessage:	the kick and ban reason (<i>type=str</i>)

onClientChannelGroupChangedEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*, *channelID*, *clientID*, *invokerClientID*, *invokerName*, *invokerUniqueIdentity*)

This is called whenever a client is added to a channelgroup in a specific channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelGroupID:	the id of the channelgroup (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)
clientID:	the id of the client (<i>type=int</i>)
invokerClientID:	the id of the client who added the channelgroup or 0 if the server did (<i>type=int</i>)
invokerName:	the nick of the client who added the channelgroup or "Server" if the server did (<i>type=str</i>)
invokerUniqueIdentity:	uid of the client who added the channelgroup or an empty string if the server did (<i>type=str</i>)

onClientChatClosedEvent(*self*, *serverConnectionHandlerID*, *clientID*, *clientUniqueIdentity*)

This is called after a client closed the chat to this client (but only after the other client has sent at least one message). This is either invoked by the sdk with `ts3lib.clientChatClosed` or the user has closed the conversation tab).

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the id of the other client
(*type=int*)

clientUniqueIdentity: the uid of the other client
(*type=str*)

onClientChatComposingEvent(*self*, *serverConnectionHandlerID*, *clientID*, *clientUniqueIdentity*)

This is called whenever another client sends the chat composing command (either invoked by the sdk with `ts3lib.clientChatComposing` or when the user is really writing in the chat).

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: (*type=*)

clientUniqueIdentity: (*type=*)

onClientDBIDfromUIDEvent(*self*, *serverConnectionHandlerID*, *uniqueClientIdentifier*, *clientDatabaseID*)

This is called whenever a database id was requested with `ts3lib.requestClientDBIDfromUID`.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

uniqueClientIdentifier: the uid of the requested client
(*type=str*)

clientDatabaseID: the resulting id in the database
(*type=int*)

onClientDisplayNameChanged(*self*, *serverConnectionHandlerID*, *clientID*, *displayName*, *uniqueClientIdentifier*)

This is called whenever a client's displayname changed (nickname or friend/foe manager).

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the client (<i>type=int</i>)
displayName:	the new displayname (<i>type=str</i>)
uniqueClientIdentifier:	the uid of the client (<i>type=str</i>)

onClientIDsEvent(*self*, *serverConnectionHandlerID*, *uniqueClientIdentifier*, *clientID*, *clientName*)

This is called for each client matching a specific uid requested by `ts3lib.requestClientIDs`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
uniqueClientIdentifier:	the uid of the client (<i>type=str</i>)
clientID:	the id of a client (<i>type=int</i>)
clientName:	the nick of the client (<i>type=str</i>)

onClientIDsFinishedEvent(*self*, *serverConnectionHandlerID*)

This is called after each client yielded by `onClientIDsEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
-----------------------------------	---

```
onClientKickFromChannelEvent(self, serverConnectionHandlerID,  
clientID, oldChannelID, newChannelID, visibility, kickerID, kickerName,  
kickerUniqueIdentifier, kickMessage)
```

This is called whenever a client is kicked from a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the kicked client (<i>type=int</i>)
oldChannelID:	the id of the channel the client was kicked from (<i>type=int</i>)
newChannelID:	the id of the channel the client was kicked to (the default channel) (<i>type=int</i>)
visibility:	defines the new state of the client in the view (see <i>ts3defines.Visibility</i>) (<i>type=int</i>)
kickerID:	the id of the kicking client (<i>type=int</i>)
kickerName:	the nick of the kicking client (<i>type=str</i>)
kickerUniqueIdentifier:	the uid of the kicking client (<i>type=str</i>)
kickMessage:	the kick reason (<i>type=str</i>)

```
onClientKickFromServerEvent(self, serverConnectionHandlerID, clientID,  
oldChannelID, newChannelID, visibility, kickerID, kickerName,  
kickerUniqueIdentifier, kickMessage)
```

This is called whenever a client is kicked from the server.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the kicked client (<i>type=int</i>)
oldChannelID:	the id of the channel the client was in (<i>type=int</i>)
newChannelID:	always set to 0 (<i>type=int</i>)
visibility:	always set to ts3defines.Visibility.LEAVE_VISIBILITY (<i>type=int</i>)
kickerID:	the id of the kicking client (<i>type=int</i>)
kickerName:	nick of the kicking client (<i>type=str</i>)
kickerUniqueIdentifier:	uid of the kicking client (<i>type=str</i>)
kickMessage:	the kick reason (<i>type=str</i>)


```
onClientMoveEvent(self, serverConnectionHandlerID, clientID,  
oldChannelID, newChannelID, visibility, moveMessage)
```

This is called whenever a client enters a another channel (moving, joining or leaving the server).

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the client (<i>type=int</i>)
oldChannelID:	the id of the former channel or 0 if the client joined the server (<i>type=int</i>)
newChannelID:	the id of the new channel or 0 if the client disconnected (<i>type=int</i>)
visibility:	defines the new state of the client in the view (see <code>ts3defines.Visibility</code>) (<i>type=int</i>)
moveMessage:	the disconnect message if the client left the server or an empty string (<i>type=str</i>)

```
onClientMoveMovedEvent(self, serverConnectionHandlerID, clientID,
oldChannelID, newChannelID, visibility, moverID, moverName,
moverUniqueIdentifier, moveMessage)
```

This is called whenever a client is moved to another channel by another client.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the moved client (<i>type=int</i>)
oldChannelID:	the id of the former channel (<i>type=int</i>)
newChannelID:	the id of the new channel (<i>type=int</i>)
visibility:	defines the new state of the client in the view (see <code>ts3defines.Visibility</code>) (<i>type=int</i>)
moverID:	the id of the moving client (<i>type=int</i>)
moverName:	nick of the moving client (<i>type=str</i>)
moverUniqueIdentifier:	uid of the moving client (<i>type=str</i>)
moveMessage:	always set to an empty string (<i>type=str</i>)

onClientMoveSubscriptionEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*)

This is called whenever a new client enters the view when subscribing a channel.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the client (<i>type=int</i>)
oldChannelID:	always set to 0 (<i>type=int</i>)
newChannelID:	the id of the subscribed channel (<i>type=int</i>)
visibility:	always set to ts3defines.Visibility.ENTER_VISIBILITY (<i>type=int</i>)

onClientMoveTimeoutEvent(*self, serverConnectionHandlerID, clientID, oldChannelID, newChannelID, visibility, timeoutMessage*)

This is called when a client timed out.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the client (<i>type=int</i>)
oldChannelID:	the id of the channel the client was in (<i>type=int</i>)
newChannelID:	always set to 0 (<i>type=int</i>)
visibility:	always set to ts3defines.Visibility.LEAVE_VISIBILITY (<i>type=int</i>)
timeoutMessage:	the timeout message (<i>type=str</i>)

onClientNamefromDBIDEvent(*self, serverConnectionHandlerID, uniqueClientIdentifier, clientDatabaseID, clientNickName*)

This is called to return the last nickname of a client referenced by the database id after it was requested with ts3lib.requestClientNamefromDBID.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
uniqueClientIdentifier:	the uid of the client (<i>type=str</i>)
clientDatabaseID:	the database id of the client (<i>type=int</i>)
clientNickName:	the last nickname of the client (<i>type=str</i>)

onClientNamefromUIDEvent(*self*, *serverConnectionHandlerID*,
uniqueClientIdentifier, *clientDatabaseID*, *clientNickName*)

This is called to return the last nickname of a client referenced by the uid after it was requested with `ts3lib.requestClientNamefromUID`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
uniqueClientIdentifier:	the uid of the client (<i>type=str</i>)
clientDatabaseID:	the database id of the client (<i>type=int</i>)
clientNickName:	the last nickname of the client (<i>type=str</i>)

onClientNeededPermissionsEvent(*self*, *serverConnectionHandlerID*,
permissionID, *permissionValue*)

This is called whenever a permission the TS3 client needed changes.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
permissionID:	the id of the permission (<i>type=int</i>)
permissionValue:	the value of the permission (<i>type=int</i>)

onClientNeededPermissionsFinishedEvent(*self*,
serverConnectionHandlerID)

This is called after each permission yielded by `onClientNeededPermissionsEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
-----------------------------------	---

onClientPermListEvent(*self*, *serverConnectionHandlerID*,
clientDatabaseID, *permissionID*, *permissionValue*, *permissionNegated*,
permissionSkip)

This is called for each granted permission to a specific client requested with `ts3lib.requestClientPermList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientDatabaseID:	the database id of the client (<i>type=int</i>)
permissionID:	the id of the permission (<i>type=int</i>)
permissionValue:	the value of the permission (<i>type=int</i>)
permissionNegated:	negated flag of the permission (<i>type=int</i>)
permissionSkip:	skip flag of the permission (<i>type=int</i>)

onClientPermListFinishedEvent(*self*, *serverConnectionHandlerID*,
clientDatabaseID)

This is called after each permission yielded by `onClientPermListEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientDatabaseID:	the database id of the client (<i>type=int</i>)

onClientSelfVariableUpdateEvent(*self*, *serverConnectionHandlerID*, *flag*, *oldValue*, *newValue*)

This is called whenever a variable of the own client is changed.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
flag:	the changed variable (see ts3defines.ClientProperties and ts3defines.ClientPropertiesRare) (<i>type=int</i>)
oldValue:	the former value (<i>type=str</i>)
newValue:	the new value (<i>type=str</i>)

onClientServerQueryLoginPasswordEvent(*self*,
serverConnectionHandlerID, *loginPassword*)

This is called when a new query login was requested.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
loginPassword:	the new password (<i>type=str</i>)

onComplainListEvent(*self*, *serverConnectionHandlerID*,
targetClientDatabaseID, *targetClientNickName*, *fromClientDatabaseID*,
fromClientNickName, *complainReason*, *timestamp*)

This is called for each entry in the complaintslist.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
targetClientDatabaseID:	the database id of the complained client (<i>type=int</i>)
targetClientNickName:	the last nickname of the complained client (<i>type=str</i>)
fromClientDatabaseID:	the database id of the complaining client (<i>type=int</i>)
fromClientNickName:	the last nickname of the complaining client (<i>type=str</i>)
complainReason:	the reason (<i>type=str</i>)
timestamp:	the time of the complain as unix timestamp (<i>type=int</i>)

onConnectionInfoEvent(*self*, *serverConnectionHandlerID*, *clientID*)

This is called when the connection info of a client has been updated requested with `ts3lib.requestConnectionInfo`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the client (<i>type=int</i>)

onConnectStatusChangeEvent(*self*, *serverConnectionHandlerID*,
newStatus, *errorNumber*)

This is called whenever the status of a serverconnection changed.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
newStatus:	the new status (see ts3defines.ConnectStatus) (<i>type=int</i>)
errorNumber:	the error (see ts3defines.ERROR_*) (<i>type=int</i>)

onDelChannelEvent(*self*, *serverConnectionHandlerID*, *channelID*,
invokerID, *invokerName*, *invokerUniqueIdentifier*)

This is called whenever a channel was deleted.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)
invokerID:	the id of the client who deleted the channel (<i>type=int</i>)
invokerName:	the nick of the deleting client (<i>type=str</i>)
invokerUniqueIdentifier:	the uid of the deleting client (<i>type=str</i>)

onFileInfoEvent(*self*, *serverConnectionHandlerID*, *channelID*, *name*, *size*, *datetime*)

This is called with the fileinfo of a remote file requested with `ts3lib.requestFileInfo`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel the file is in (<i>type=int</i>)
name:	the full path of the file (<i>type=str</i>)
size:	the filesize in bytes (<i>type=int</i>)
datetime:	time the file was last changed as unix timestamp (<i>type=int</i>)

onFileListEvent(*self, serverConnectionHandlerID, channelID, path, name, size, datetime, atype, incompletesize, returnCode*)

This is called for each file and directory in path requested with `ts3lib.requestFileList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)
path:	the path (<i>type=str</i>)
name:	the filename (<i>type=str</i>)
size:	the filesize in bytes (<i>type=int</i>)
datetime:	time the file or directory was last changed as unix timestamp (<i>type=</i>)
atype:	set to 1 if it's a directory, otherwise set to 0 (<i>type=int</i>)
incompletesize:	the complete filesize in bytes or 0 if the file is already complete (<i>type=int</i>)
returnCode:	the returncode passed to the request or an empty string (<i>type=str</i>)

onFileListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelID*, *path*)

This is called after each file and directory yielded by onFileListEvent was triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

channelID: *(type=int)*

path: *(type=)*

onHotkeyEvent(*self*, *keyword*)

This is called when a plugin's hotkey is triggered.

Parameters

keyword: the local keyword set in cls.hotkeys
(type=str)

onHotkeyRecordedEvent(*self*, *keyword*, *key*)

This is called when a hotkey was recorded requested by ts3lib.requestHotkeyInputDialog.

Parameters

keyword: the keyword
(type=str)

key: the hotkey to trigger the keyword
(type=str)

onIncomingClientQueryEvent(*self*, *serverConnectionHandlerID*, *commandText*)

This callback was designed for the clientquery plugin. It combines many callbacks and is called with a representing string.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

commandText: the text of the client query
(type=str)

onMenuItemEvent(*self*, *serverConnectionHandlerID*, *atype*, *menuItemID*, *selectedItemID*)

This is called when a plugin's menuitem defined in `cls.menuItems` is clicked.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

atype: type of the menuitem (see
`ts3defines.PluginMenuType`)
(*type=int*)

menuItemID: the local id of the menuitem defines
in `cls.menuItems`
(*type=int*)

selectedItemID: set to the id of the selected channel
if `atype` is
`ts3defines.PluginMenuType.PLUGIN_MENU_TYPE_CHANNEL`
set to the id of the selected client if
`atype` is
`ts3defines.PluginMenuType.PLUGIN_MENU_TYPE_CLIENT`,
otherwise always set to 0
(*type=int*)

onMessageGetEvent(*self, serverConnectionHandlerID, messageID, fromClientUniqueIdentity, subject, message, timestamp*)

This is called with the information about an offline message requested with `ts3lib.requestMessageGet`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
messageID:	the id of the message (<i>type=int</i>)
fromClientUniqueIdentity:	the uid of the message's sender (<i>type=str</i>)
subject:	the subject of the message (<i>type=str</i>)
message:	the content of the message (<i>type=str</i>)
timestamp:	time the message was sent as unix timestamp (<i>type=int</i>)

onMessageListEvent(*self, serverConnectionHandlerID, messageID, fromClientUniqueIdentity, subject, timestamp, flagRead*)

This is called for each offline message available on the server requested with `ts3lib.requestMessageList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
messageID:	the id of the message (<i>type=int</i>)
fromClientUniqueIdentity:	the uid of the message's sender (<i>type=str</i>)
subject:	the subject of the message (<i>type=str</i>)
timestamp:	time the message was sent as unix timestamp (<i>type=int</i>)
flagRead:	defines the read status of the message (<i>type=int</i>)

onNewChannelCreatedEvent(*self*, *serverConnectionHandlerID*, *channelID*, *channelParentID*, *invokerID*, *invokerName*, *invokerUniqueIdentifier*)

This is called whenever a new channel was created.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the new channel (<i>type=int</i>)
channelParentID:	the id of the parent channel (<i>type=int</i>)
invokerID:	the id of the creating client (<i>type=int</i>)
invokerName:	nick of the creating client (<i>type=str</i>)
invokerUniqueIdentifier:	the uid of the creating client (<i>type=str</i>)

onNewChannelEvent(*self*, *serverConnectionHandlerID*, *channelID*, *channelParentID*)

This is called whenever a new channel enters the view (at connect).

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)
channelParentID:	the id of the parent channel (<i>type=int</i>)

onPermissionListEvent(*self, serverConnectionHandlerID, permissionID, permissionName, permissionDescription*)

This is called for each permission on the server requested with `ts3lib.requestPermissionList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
permissionID:	id of the permission (<i>type=int</i>)
permissionName:	name of the permission (<i>type=str</i>)
permissionDescription:	description of the permission (<i>type=str</i>)

onPermissionListFinishedEvent(*self, serverConnectionHandlerID*)

This is called after each permission yielded by `onPermissionListEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
-----------------------------------	---

onPermissionListGroupEndIDEvent(*self, serverConnectionHandlerID, groupEndID*)

This is called for each last permission in the groups of permissions after requesting the permissionlist with `ts3lib.requestPermissionList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
groupEndID:	id of the last permission in group (<i>type=int</i>)

```
onPermissionOverviewEvent(self, serverConnectionHandlerID,  
clientDatabaseID, channelID, overviewType, overviewID1, overviewID2,  
permissionID, permissionValue, permissionNegated, permissionSkip)
```

This is called for each permission of a pair of client and channel requested with `ts3lib.requestPermissionOverview`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientDatabaseID:	the database id of the client (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)
overviewType:	defines the type of entry in the overview (0 for servergroup, 1 for client permissions, 2 for needed channel permissions, 3 for channelgroup) (<i>type=int</i>)
overviewID1:	depending on the overviewType, set to the id of the servergroup, to the client's database id or the id of the channel (<i>type=int</i>)
overviewID2:	only used with overviewType=3, then set to the id of the channelgroup; otherwise set to 0 (<i>type=int</i>)
permissionID:	the id of the permission (<i>type=int</i>)
permissionValue:	the value of the permission (<i>type=int</i>)
permissionNegated:	negated flag of the permission (<i>type=int</i>)
permissionSkip:	skip flag of the permission (<i>type=int</i>)

onPermissionOverviewFinishedEvent(*self*, *serverConnectionHandlerID*)

This is called after each permission yielded by `onPermissionOverviewEvent` was triggered.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

onPlaybackShutdownCompleteEvent(*self*, *serverConnectionHandlerID*)

This is called when a playback device can be shutdown with `ts3lib.closePlaybackDevice` after the process was initiated with `ts3lib.initiateGracefulPlaybackShutdown`.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

onPluginCommandEvent(*self*, *serverConnectionHandlerID*, *sender*, *pluginCommand*)

This is called whenever pyTSon receives a plugincommand from another client. All pyTSon plugins will receive this callback. pyTSon recommends to prefix plugincommands with the pluginname.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

sender: the id of the sending client
(*type=int*)

pluginCommand: the command
(*type=str*)

onServerConnectionInfoEvent(*self*, *serverConnectionHandlerID*)

This is called whenever the server's connectioninfo was updated.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

onServerEditedEvent(*self*, *serverConnectionHandlerID*, *editorID*, *editorName*, *editorUniqueIdentifier*)

This is called whenever the server was edited by a client.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
editorID:	the id of the client (<i>type=int</i>)
editorName:	nick of the client (<i>type=int</i>)
editorUniqueIdentifier:	uid of the client (<i>type=str</i>)

onServerGroupByClientIDEvent(*self*, *serverConnectionHandlerID*, *name*, *serverGroupList*, *clientDatabaseID*)

This is called for each servergroup of a client requested with `ts3lib.requestServerGroupsByClientID`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
name:	name of the servergroup (<i>type=str</i>)
serverGroupList:	id of the servergroup (<i>type=int</i>)
clientDatabaseID:	the database id of the client (<i>type=int</i>)

```
onServerGroupClientAddedEvent(self, serverConnectionHandlerID,  
clientID, clientName, clientUniqueIdentity, serverGroupID, invokerClientID,  
invokerName, invokerUniqueIdentity)
```

This is called whenever a client is added to a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the added client (<i>type=int</i>)
clientName:	nick of the added client (<i>type=str</i>)
clientUniqueIdentity:	uid of the added client (<i>type=str</i>)
serverGroupID:	the id of the servergroup (<i>type=int</i>)
invokerClientID:	the id of the adding client (<i>type=int</i>)
invokerName:	nick of the adding client (<i>type=str</i>)
invokerUniqueIdentity:	uid of the adding client (<i>type=str</i>)

onServerGroupClientDeletedEvent(*self*, *serverConnectionHandlerID*, *clientID*, *clientName*, *clientUniqueIdentity*, *serverGroupID*, *invokerClientID*, *invokerName*, *invokerUniqueIdentity*)

This is called whenever a client was removed from a servergroup.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientID:	the id of the removed client (<i>type=int</i>)
clientName:	nick of the removed client (<i>type=str</i>)
clientUniqueIdentity:	uid of the removed client (<i>type=str</i>)
serverGroupID:	id the servergroup (<i>type=int</i>)
invokerClientID:	the id of the removing client (<i>type=int</i>)
invokerName:	nick of the removing client (<i>type=str</i>)
invokerUniqueIdentity:	uid of the removing client (<i>type=str</i>)

onServerGroupClientListEvent(*self*, *serverConnectionHandlerID*,
serverGroupID, *clientDatabaseID*, *clientNameIdentifier*, *clientUniqueID*)

This is called for each member of a servergroup requested with
ts3lib.requestServerGroupClientList.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the id of the servergroup (<i>type=int</i>)
clientDatabaseID:	the database id of the member (<i>type=int</i>)
clientNameIdentifier:	the last nick of the member or an empty string if <i>withNames</i> was set to <i>False</i> in the request (<i>type=str</i>)
clientUniqueID:	the uid of the member or an empty string if <i>withNames</i> was set to <i>False</i> in the request (<i>type=str</i>)

onServerGroupListEvent(*self*, *serverConnectionHandlerID*, *serverGroupID*, *name*, *atype*, *iconID*, *saveDB*)

This is called for each servergroup on the server requested with `ts3lib.requestServerGroupList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the id of the servergroup (<i>type=int</i>)
name:	name of the servergroup (<i>type=str</i>)
atype:	type of the servergroup (0=template, 1=regular, 2=serverquery) (<i>type=int</i>)
iconID:	icon id of the servergroup or 0 if no icon in this group (<i>type=int</i>)
saveDB:	set to 1 if memberships are saved to the database, set to 0 otherwise (<i>type=int</i>)

onServerGroupListFinishedEvent(*self*, *serverConnectionHandlerID*)

This is called after each servergroup yielded by `onServerGroupListEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
-----------------------------------	---

onServerGroupPermListEvent(*self*, *serverConnectionHandlerID*, *serverGroupID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

This is called for each granted permission of a servergroup requested with `ts3lib.requestServerGroupPermList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	the id of the servergroup (<i>type=int</i>)
permissionID:	the id of the permission (<i>type=int</i>)
permissionValue:	value of the permission (<i>type=int</i>)
permissionNegated:	negated flag (<i>type=int</i>)
permissionSkip:	skip flag (<i>type=int</i>)

onServerGroupPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *serverGroupID*)

This is called after each permission yielded by `onServerGroupPermListEvent` was triggered.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
serverGroupID:	id of the servergroup (<i>type=int</i>)

onServerLogEvent(*self*, *serverConnectionHandlerID*, *logMsg*)

This is called for each line of the serverlog requested by the TS3 Client.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

logMsg: the message
(*type=str*)

onServerLogFinishedEvent(*self*, *serverConnectionHandlerID*, *lastPos*, *fileSize*)

This is called after the requested number of loglines were yielded by onServerLogEvent.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

lastPos: (*type=*)

fileSize: (*type=*)

onServerStopEvent(*self*, *serverConnectionHandlerID*, *shutdownMessage*)

This is called when the server was stopped.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

shutdownMessage: if given, the shutdownmessage
(*type=str*)

onServerTemporaryPasswordListEvent(*self*, *serverConnectionHandlerID*, *clientNickname*, *uniqueClientIdentifier*, *description*, *password*, *timestampStart*, *timestampEnd*, *targetChannelID*, *targetChannelPW*)

This is called for each temporary password on the server requested with `ts3lib.requestServerTemporaryPasswordList`.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
clientNickname:	nick of the creator (<i>type=str</i>)
uniqueClientIdentifier:	uid of the creator (<i>type=str</i>)
description:	description of the password (<i>type=str</i>)
password:	the password (<i>type=str</i>)
timestampStart:	time the password was created as unix timestamp (<i>type=int</i>)
timestampEnd:	time the password expires as unix timestamp (<i>type=int</i>)
targetChannelID:	the id of the channel clients join in (<i>type=int</i>)
targetChannelPW:	password to the targetChannel (<i>type=str</i>)

onServerUpdatedEvent(*self*, *serverConnectionHandlerID*)

This is called whenever the server variables were updated.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
-----------------------------------	---

onSoundDeviceListChangedEvent(*self, modeID, playOrCap*)

This is called when the list of sounddevices changed.

Parameters

modeID: defines the playback/capture mode
(type=int)

playOrCap: defines whether the playback- or capturelist changed
(type=int)

onTalkStatusChangeEvent(*self, serverConnectionHandlerID, status, isReceivedWhisper, clientID*)

This is called whenever a client starts or stops talking.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(type=int)

status: defines whether the client starts or stops talking (see `ts3defines.TalkStatus`)
(type=int)

isReceivedWhisper: set to 1 if the client whispered, set to 0 otherwise
(type=int)

clientID: the id of the client
(type=int)

onUpdateChannelEditedEvent(*self, serverConnectionHandlerID, channelID, invokerID, invokerName, invokerUniqueIdentifier*)

This is called whenever a channel was edited by a client.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)
invokerID:	the id of the client (<i>type=int</i>)
invokerName:	nick of the client (<i>type=str</i>)
invokerUniqueIdentifier:	uid of the client (<i>type=str</i>)

onUpdateChannelEvent(*self, serverConnectionHandlerID, channelID*)

This is called whenever the channel variables of a specific channel are updated.

Parameters

serverConnectionHandlerID:	the ID of the serverconnection (<i>type=int</i>)
channelID:	the id of the channel (<i>type=int</i>)

onUpdateClientEvent(*self*, *serverConnectionHandlerID*, *clientID*, *invokerID*, *invokerName*, *invokerUniqueIdentifier*)

This is called whenever the client variables of a specific client are updated.

Parameters

serverConnectionHandlerID: the ID of the serverconnection
(*type=int*)

clientID: the id of the client
(*type=int*)

invokerID: id of the client invoking the change
or 0 if it was a selfupdate
(*type=int*)

invokerName: nick of the invoking client
(*type=str*)

invokerUniqueIdentifier: uid of the invoking client
(*type=str*)

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

15.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

15.2.3 Class Variables

Name	Description
<code>requestAutoload</code>	If set to True, the plugin is automatically loaded on startup. This check is only done once per new plugin, after that users can enable/disable the plugin. Value: False
<code>name</code>	The name of the plugin. Use meaningful names. It has to be unique in the list of plugins. Value: <code>"__ts3plugin__"</code>

continued on next page

Name	Description
version	Version string of the plugin. pyTSon will use this string to determine, if a new version is available in an online repository. Value: "1.0"
apiVersion	apiVersion the plugin was developed for. Value: 21
author	Let the world know who made the plugin. Value: "Thomas \ "PLuS\ " Pathmann"
description	Explain, what the plugin does. Value: "This is the baseclass for all ts3 python plugins"
offersConfigure	Set this to True, if the plugin offers a configuration ui. In this case the method configure is called. Value: False
commandKeyword	Set this to a keyword (non-empty) your plugin can be called with. Users may type /py <thecommand> [moreargs]. The method processCommand will be called with any additional args. Value: "py"
infoTitle	If set to a string, this title is shown in the info frame of the client on top of the infoData. If set to None, nothing is shown and infoData won't be called. Value: "pyTSon"
menuItems	List of tuple(int, int, str, str) containing the menuitems. The tuple has to contain the type (see ts3defines.PluginMenuType), an int identifier (unique in this list), the title and the name of the icon. The icon has to be a path relative to pytson.getPluginPath(). Pass an empty string to omit using an icon. The method onMenuItemEvent with menuItemID=identifier is called. Value: [(ts3defines.PluginMenuType.PLUGIN_MENU_TYPE_CLIENT, 0, "...

continued on next page

Name	Description
hotkeys	List of tuple(str, str) containing the hotkeys. The tuple has to contain a string identifier (unique in this list) and a description shown in the TS3 Client's hotkey dialog. The method onHotkeyEvent with keyword=identifier is called. Value: [("keyword", "description")]

16 Package *ts3widgets*

16.1 Modules

- **filetransfer** (*Section 17, p. 275*)
- **serverview** (*Section 18, p. 295*)

17 Module `ts3widgets.filetransfer`

17.1 Functions

splitpath(*path*)

Splits a TS3 filepath into its sections.

Parameters

path: the path to split
(*type=**str*)

Return Value

the list of sections
(*type=**list[str]*)

joinpath(**args*)

Joins multiple sections into a TS3 filepath.

Parameters

args: sections to join
(*type=**tuple(str)*)

Return Value

the resulting path
(*type=**str*)

bytesToStr(*size*)

Creates a human readable string of a number of bytes.

Parameters

size: number of bytes
(*type=**int*)

Return Value

the converted size and most fitting unit
(*type=**str*)

17.2 Class File

object —
 ts3widgets.filetransfer.File

Container class to hold all information on a remote TS3 file.

17.2.1 Methods

```
__init__(self, path, name, size, date, atype, incompletesize)
```

x.__init__(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

```
isDirectory(self)
```

```
icon(self)
```

Returns the most fitting icon for the file

Return Value

the icon

(*type=QIcon*)

```
fullpath(self)
```

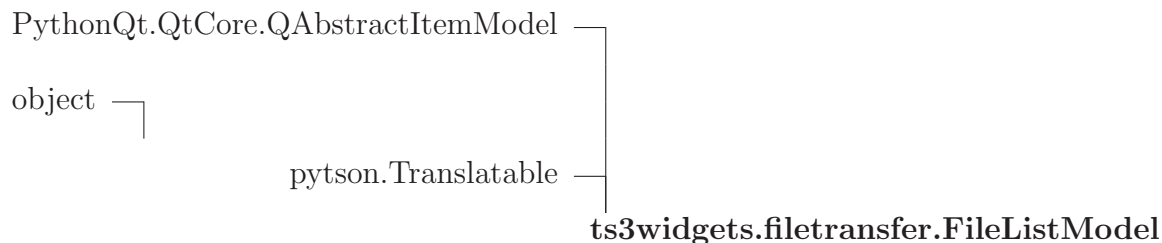
Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

17.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

17.3 Class *FileListModel*



Itemmodel to abstract the files contained on a TS3 filepath.

17.3.1 Methods

`__init__(self, schid, cid, password, parent=None, readonly=False)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

`__del__(self)`

`path(self, val)`

`currentFiles(self)`

`onFileListEvent(self, schid, channelID, path, name, size, date, atype, incompletesize, returnCode)`

`onFileListFinishedEvent(self, schid, channelID, path)`

`onServerErrorEvent(self, schid, errorMessage, error, returnCode, extraMessage)`

`onServerPermissionErrorEvent(self, schid, errorMessage, error, returnCode, failedPermissionID)`

`headerData(self, section, orientation, role=Qt.DisplayRole)`

`flags(self, idx)`

`index(self, row, column, parent=QModelIndex())`

`parent(self, idx)`

`rowCount(self, parent=QModelIndex())`

`columnCount(self, parent=QModelIndex())`

`data(self, idx, role=Qt.DisplayRole)`

```
setData(self, idx, value, role=Qt.EditRole)
```

```
fileByIndex(self, idx)
```

Inherited from object

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

17.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

17.4 Class SmartStatusBar

PythonQt.QtGui.QStatusBar —
 ts3widgets.filetransfer.SmartStatusBar

StatusBar which automatically hides itself, when the message is cleared.

17.4.1 Methods

```
__init__(self, parent=None)
```

```
showMessage(self, message, timeout=0)
```

Displays a message for a specified duration.

Parameters

message: the message to display

(*type*=*str*)

timeout: duration in ms; optional; if set to 0,
 SmartStatusBar.defaultTimeout is used

(*type*=*int*)

17.4.2 Class Variables

Name	Description
defaultTimeout	Value: 5000

17.5 Class FileCollector



Collects all files recursively from TS3 filetransfer directories with their corresponding download path. Emits a signal collectionFinished with a list of tuples(str, list[File]) containing the download dir and a list of files. The signal collectionError(str, int) is emitted on error with the errorstring and the errorcode.

17.5.1 Methods

__init__(self, schid, cid, password, rootdir)

Instantiates a new object.

Parameters

schid: the id of the serverconnection handler
(type=int)

cid: the id of the channel
(type=int)

password: the password of the channel
(type=str)

rootdir: the root download directory
(type=str)

Overrides: object.__init__

__del__(self)

addFiles(*self*, *files*)

Manually adds a list of files to the collection (emitted with the rootdir)

Parameters

files: list of files to emit
(*type=list(File)*)

collect(*self*, *dirs*)

Starts collecting files from a list of directories

Parameters

dirs: list of directories
(*type=list(File)*)

onServerErrorEvent(*self*, *schid*, *errorMessage*, *error*, *returnCode*, *extraMessage*)

onFileListEvent(*self*, *schid*, *channelID*, *path*, *name*, *size*, *datetime*, *atype*, *incompletesize*, *returnCode*)

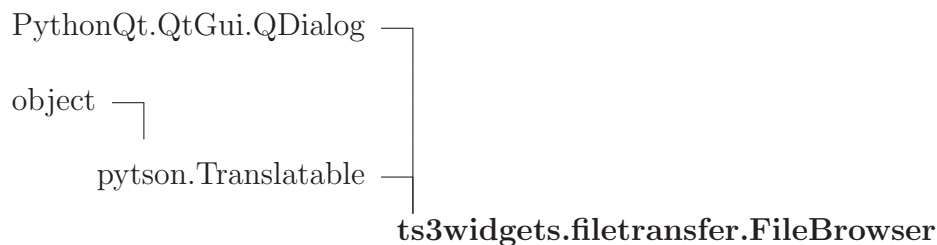
Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

17.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

17.6 Class FileBrowser



Dialog to display files contained on a TS3 filepath.

17.6.1 Methods

```
__init__(self, schid, cid, password='', path='/', parent=None,
staticpath=False, readonly=False, downloaddir=None, iconpack=None)
```

Instantiates a new object.

Parameters

schid:	the id of the serverconnection handler (<i>type=int</i>)
cid:	the id of the channel (<i>type=int</i>)
password:	password to the channel, defaults to an empty string (<i>type=str</i>)
path:	path to display, defaults to the root path (<i>type=str</i>)
parent:	parent of the dialog; optional keyword arg; defaults to None (<i>type=QWidget</i>)
staticpath:	if set to True, the initial path can't be changed by the user; optional keyword arg; defaults to False (<i>type=bool</i>)
readonly:	if set to True, the user can't download, upload or delete files, or create new directories; optional keyword arg; defaults to False (<i>type=bool</i>)
downloaddir:	directory to download files to; optional keyword arg; defaults to None; if set to None, the TS3 client's download directory is used (<i>type=str</i>)
iconpack:	iconpack to load icons from; optional keyword arg; defaults to None; if set to None, the current iconpack is used (<i>type=ts3client.IconPack</i>)

Overrides: object.__init__

```
__del__(self)
```



```
onPathChanged(self, newpath)
```

```
on_pathEdit_returnPressed(self)
```

```
on_iconButton_toggled(self, act)
```

```
on_detailedButton_toggled(self, act)
```

```
on_filterButton_clicked(self)
```

```
on_clearButton_clicked(self)
```

```
on_filterEdit_textChanged(self, newtext)
```

```
on_upButton_clicked(self)
```

```
on_homeButton_clicked(self)
```

```
refresh(self)
```

```
on_downloadaddirButton_clicked(self)
```

```
showError(self, prefix, errcode, msg=None)
```

```
uploadFiles(self)
```

```
onServerErrorEvent(self, schid, errorMessage, error, returnCode,  
extraMessage)
```

```
selectedFiles(self)
```

```
currentItem(self, source=True)
```

```
downloadFiles(self, files=None)
```

```
createFolder(self)
```

```
deleteFiles(self, files=None)
```

<code>on_table_customContextMenuRequested(<i>self</i>, <i>pos</i>)</code>

<code>on_list_customContextMenuRequested(<i>self</i>, <i>pos</i>)</code>
--

<code>viewDoubleClicked(<i>self</i>, <i>idx</i>)</code>

<code>on_openAction_triggered(<i>self</i>)</code>

<code>on_renameAction_triggered(<i>self</i>)</code>

<code>on_copyAction_triggered(<i>self</i>)</code>

Inherited from object

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

17.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

17.7 Class FileCollisionAction

object └─ **ts3widgets.filetransfer.FileCollisionAction**

17.7.1 Methods**Inherited from object**

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

17.7.2 Properties

Name	Description
<i>Inherited from object</i>	

continued on next page

Name	Description
__class__	

17.7.3 Class Variables

Name	Description
overwrite	Value: 1
resume	Value: 2
skip	Value: 4
toall	Value: 8

17.8 Class *FileCollisionDialog*



Dialog to inform about a filecollision and requests input how to handle it.

17.8.1 Methods

getAction(*cls*, *localfile*, *remotefile*, *isdownload*, *multi*, *parent=None*)

Convenience function to execute (blocks) the dialog.

Parameters

localfile: the path to the local file
(type=str)

remotefile: the remote file
(type=File)

isdownload: set to True if remotefile should be downloaded
(type=bool)

multi: set to True, if there are multiple files which could collide
(type=bool)

parent: parent widget of the dialog; optional; defaults to None
(type=QWidget)

__init__(*self*, *localfile*, *remotefile*, *isdownload*, *multi*, *parent=None*)

Instantiates a new dialog.

Parameters

localfile: the path to the local file
(type=str)

remotefile: the remote file
(type=File)

isdownload: set to True if remotefile should be downloaded
(type=bool)

multi: set to True, if there are multiple files which could collide
(type=bool)

parent: parent widget of the dialog; optional; defaults to None
(type=QWidget)

Overrides: *object.__init__*

on_overwriteButton_clicked(*self*)

<code>on_resumeButton_clicked(<i>self</i>)</code>

<code>on_skipButton_clicked(<i>self</i>)</code>

<code>on_skipallButton_clicked(<i>self</i>)</code>
--

<code>on_cancelButton_clicked(<i>self</i>)</code>

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

17.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

17.9 Class FileTransfer

Abstract container class to hold information on a filetransfer

17.9.1 Methods

<code>__init__(<i>self</i>, <i>err</i>, <i>retcode</i>)</code>
--

<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)

<code>updateSize(<i>self</i>, <i>val</i>)</code>
--

<code>updateError(<i>self</i>, <i>err</i>, <i>msg</i>=None)</code>
--

progress(*self*)

hasError(*self*)

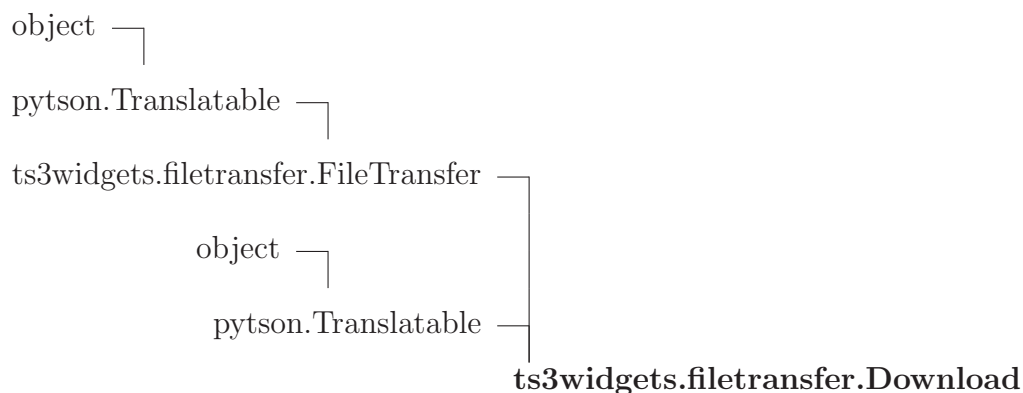
Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

17.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

17.10 Class Download



Container class to hold information on a download

17.10.1 Methods

--init--(*self*, *err*, *retcode*, *thefile*, *todir*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

progress(*self*)

Overrides: `ts3widgets.filetransfer.FileTransfer.progress`

description(*self*)

localpath(*self*)

Inherited from ts3widgets.filetransfer.FileTransfer(Section 17.9)

hasError(), updateError(), updateSize()

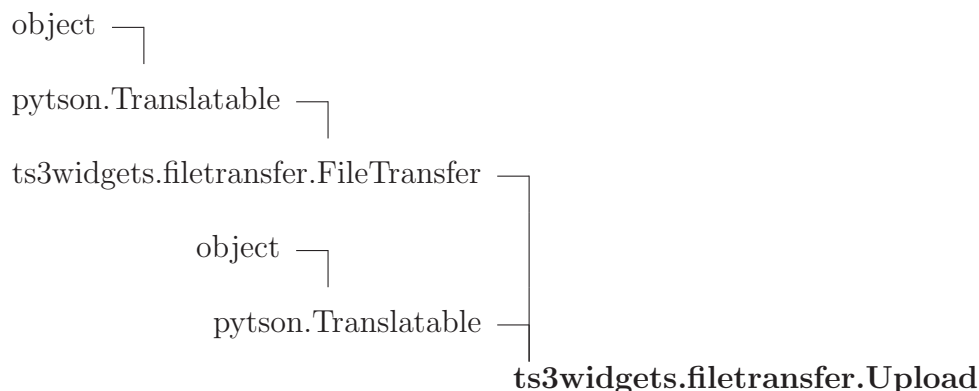
Inherited from object

__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

17.10.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

17.11 Class Upload



Container class to hold information on an upload

17.11.1 Methods

__init__(*self*, *err*, *retcode*, *localfile*)

x.__init__(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

progress(*self*)

Overrides: `ts3widgets.filetransfer.FileTransfer.progress`

description(*self*)

Inherited from `ts3widgets.filetransfer.FileTransfer` (Section 17.9)

`hasError()`, `updateError()`, `updateSize()`

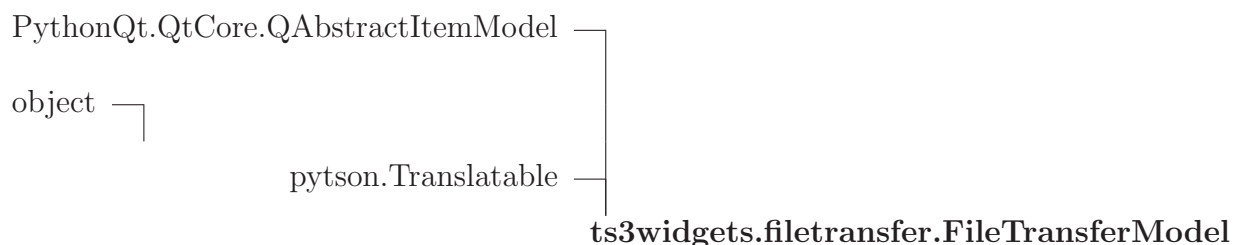
Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

17.11.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

17.12 Class `FileTransferModel`



Itemmodel to abstract multiple filetransfers.

17.12.1 Methods

__init__(*self*, *schid*, *cid*, *password*, *parent*=None)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit` (inherited documentation)

__del__(*self*)

timerEvent(*self*, *event*)

addDownload(*self*, *thefile*, *downloaddir*, *overwrite*, *resume*)

Requests a download from the server and monitors its progress

Parameters

thefile: remote file to download
(*type=File*)

downloaddir: path to the download directory
(*type=str*)

overwrite: set to True to overwrite an existing file
(*type=bool*)

resume: set to True to resume a previous download
(*type=bool*)

Return Value

the filetransfer id
(*type=int*)

addUpload(*self*, *path*, *localfile*, *overwrite*, *resume*)

Requests an upload to the server.

Parameters

path: path to upload the file to
(*type=str*)

localfile: path to the file to upload
(*type=str*)

overwrite: set to True to overwrite an existing file
(*type=bool*)

resume: set to True to resume a previous upload
(*type=bool*)

Return Value

the filetransfer id
(*type=int*)

cleanup(*self*)

Cleanup finished and broken downloads

```
onFileTransferStatusEvent(self, transferID, status, statusMessage,  
remotefileSize, schid)
```

```
onServerErrorEvent(self, schid, errorMessage, error, returnCode,  
extraMessage)
```

```
onServerPermissionErrorEvent(self, *args)
```

```
headerData(self, section, orientation, role=Qt.DisplayRole)
```

```
index(self, row, column, parent=QModelIndex())
```

```
parent(self, idx)
```

```
rowCount(self, parent=QModelIndex())
```

```
columnCount(self, parent=QModelIndex())
```

```
data(self, idx, role=Qt.DisplayRole)
```

Inherited from object

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

17.12.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

17.13 Class *FileTransferDelegate*

PythonQt.QtGui.QStyledItemDelegate

ts3widgets.filetransfer.FileTransferDelegate

Delegate which displays a progress bar in the second column of an itemview

17.13.1 Methods

paint(*self*, *painter*, *option*, *idx*)

17.14 Class FileTransferDialog

Dialog to display filetransfers from/to a ts3 channel.

17.14.1 Methods

__init__(*self*, *schid*, *cid*, *password*, *parent*=None)
x.__init__(...) initializes *x*; see `help(type(x))` for signature
 Overrides: `object.__init__` `exitit`(inherited documentation)

on_closeButton_clicked(*self*)

on_cleanupButton_clicked(*self*)

addUpload(*self, path, localfile, overwrite, resume*)

Adds an upload.

Parameters

path: path to upload the file to
(type=str)

localfile: path to the file to upload
(type=str)

overwrite: set to True to overwrite an existing file
(type=bool)

resume: set to True to resume a previous upload
(type=bool)

Return Value

the filetransfer id
(type=int)

addDownload(*self, thefile, downloaddir, overwrite, resume*)

Adds a download.

Parameters

thefile: remote file to download
(type=File)

downloaddir: path to the download directory
(type=str)

overwrite: set to True to overwrite an existing file
(type=bool)

resume: set to True to resume a previous download
(type=bool)

Return Value

the filetransfer id
(type=int)

Inherited from object

`--delattr--()`, `--format--()`, `--getattrattribute--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce-ex--()`,
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

17.14.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

18 Module ts3widgets.serverview

18.1 Class ServerViewRoles

Additional roles used in ServerviewModel to deliver icons and spacer properties.

18.1.1 Class Variables

Name	Description
itemtype	Value: Qt.UserRole
statusicons	Value: Qt.UserRole+ 1
isspacer	Value: Qt.UserRole+ 2
spacertype	Value: Qt.UserRole+ 3
spaceralignment	Value: Qt.UserRole+ 4
spacercustomtext	Value: Qt.UserRole+ 5

18.2 Class Channel

object  **ts3widgets.serverview.Channel**

Object wrapper for a channel on a TS3 server.

18.2.1 Methods

```
__init__(self, schid, cid)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
append(self, obj, sort=True)
```

```
rowOf(self, obj=None, pretend=False)
```

```
remove(self, obj)
```

```
update(self)
```

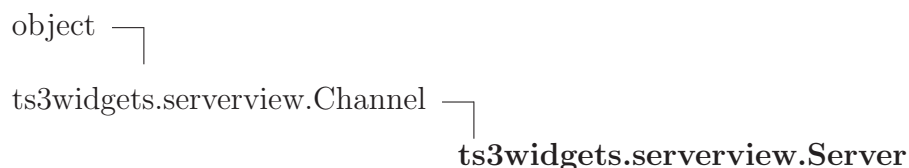
`name(self)``sortOrder(self)``isPermanent(self)``isSpacer(self)``spacerAlignment(self)``spacerType(self)``spacerCustomtext(self)``isPasswordProtected(self)``isSubscribed(self)``neededTalkPower(self)``isDefault(self)``iconID(self)``maxClients(self)``codec(self)``isFull(self)``iconVariable(self)``count(self)``child(self, row)``sort(self)``__iter__(self)`

Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

18.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

18.3 Class Server

Object wrapper for a TS3 server connection.

18.3.1 Methods

`__init__(self, schid)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

`update(self)`

Overrides: `ts3widgets.serverview.Channel.update`

`name(self)`

Overrides: `ts3widgets.serverview.Channel.name`

`iconID(self)`

Overrides: `ts3widgets.serverview.Channel.iconID`

`rowOf(self, obj=None)`

Overrides: `ts3widgets.serverview.Channel.rowOf`

iconVariable(*self*)

Overrides: *ts3widgets.serverview.Channel.iconVariable*

Inherited from ts3widgets.serverview.Channel(Section 18.2)

__iter__(), *append*(), *child*(), *codec*(), *count*(), *isDefault*(), *isFull*(), *isPasswordProtected*(), *isPermanent*(), *isSpacer*(), *isSubscribed*(), *maxClients*(), *neededTalkPower*(), *remove*(), *sort*(), *sortOrder*(), *spacerAlignment*(), *spacerCustomtext*(), *spacerType*()

Inherited from object

__delattr__(), *__format__*(), *__getattr__*(), *__hash__*(), *__new__*(), *__reduce__*(), *__reduce_ex__*(), *__repr__*(), *__setattr__*(), *__sizeof__*(), *__str__*(), *__subclasshook__*()

18.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

18.4 Class Client



Object wrapper for a connected client on a TS3 server.

18.4.1 Methods

__init__(*self*, *schid*, *clid*, *isme*)

x.__init__(...) initializes x; see *help(type(x))* for signature

Overrides: *object.__init__* *extit*(inherited documentation)

update(*self*)

count(*self*)

rowOf(*self*)

`--lt--(self, other)``--gt--(self, other)``name(self)``displayName(self)``talkPower(self)``isRecording(self)``isChannelCommander(self)``isTalking(self, val)``iconID(self)``isPrioritySpeaker(self)``isAway(self)``country(self)``isRequestingTalkPower(self)``isTalker(self)``outputMuted(self)``inputMuted(self)``hardwareInputMuted(self)``hardwareOutputMuted(self)``inputDeactivated(self)``channelGroup(self)`

serverGroups(*self*)

iconVariable(*self*)

Inherited from object

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

18.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

18.5 Class *ServerviewModel*

PythonQt.QtCore.QAbstractItemModel — **ts3widgets.serverview.ServerviewModel**

ItemModel to deliver data of a serverview to ItemWidgets. The data is delivered in one column. Limitations: no badges, no friend/foe status

18.5.1 Methods

--init--(*self*, *schid*, *iconpack*=None, *parent*=None)

Instantiates a new ServerviewModel object. This raises an exception if the iconpack could not be opened. The object registers itself as callbackproxy to the PythonHost.

Parameters

schid: the ID of the serverconnection
(type=int)

iconpack: the iconpack to use icons from. defaults to None to use the TS3 client's current IconPack
(type=ts3client.IconPack)

parent: the QObject-parent. defaults to None
(type=QObject)

`--del--`(*self*)

`setServerConnectionHandlerId`(*self*, *schid*)

`onServerEditedEvent`(*self*, *schid*, *editorID*, *editorName*, *editorUID*)

`onNewChannelEvent`(*self*, *schid*, *cid*, *pcid*)

`onNewChannelCreatedEvent`(*self*, *schid*, *cid*, *parent*, *invokerID*,
invokerName, *invokerUniqueIdentifier*)

`onUpdateChannelEditedEvent`(*self*, *schid*, *cid*, *invokerID*, *invokerName*,
invokerUniqueIdentifier)

`onChannelMoveEvent`(*self*, *schid*, *cid*, *newpcid*, *invokerID*, *invokerName*,
invokerUniqueIdentifier)

`onDelChannelEvent`(*self*, *schid*, *cid*, *invokerID*, *invokerName*, *invokerUID*)

`onClientMoveEvent`(*self*, *schid*, *clientID*, *oldChannelID*, *newChannelID*,
visibility, *moveMessage*)

`onChannelUnsubscribeEvent`(*self*, *schid*, *channelID*)

`onChannelSubscribeEvent`(*self*, *schid*, *channelID*)

`onClientMoveMovedEvent`(*self*, *schid*, *clientID*, *oldChannelID*,
newChannelID, *visibility*, *moverID*, *moverName*, *moverUniqueIdentifier*,
moveMessage)

`onUpdateClientEvent`(*self*, *schid*, *clientID*, *invokerID*, *invokerName*,
invokerUniqueIdentifier)

`onClientSelfVariableUpdateEvent`(*self*, *schid*, *flag*, *oldValue*, *newValue*)

`onClientMoveSubscriptionEvent`(*self*, *schid*, *clientID*, *oldChannelID*,
newChannelID, *visibility*)

`onClientMoveTimeoutEvent`(*self*, *schid*, *clientID*, *oldChannelID*,
newChannelID, *visibility*, *timeoutMessage*)

```
onClientKickFromServerEvent(self, schid, clientID, oldChannelID,  
newChannelID, visibility, kickerID, kickerName, kickerUniqueIdentifier,  
kickMessage)
```

```
onClientDisplayNameChanged(self, schid, clientID, displayName, uid)
```

```
onTalkStatusChangeEvent(self, schid, status, isReceivedWhisper, clid)
```

```
onServerGroupListEvent(self, schid, serverGroupID, name, atype, iconID,  
saveDB)
```

```
onChannelGroupListEvent(self, schid, channelGroupID, name, atype,  
iconID, saveDB)
```

```
index(self, row, column, parent)
```

```
parent(self, index)
```

```
rowCount(self, parent)
```

```
columnCount(self, parent)
```

```
data(self, index, role)
```

18.6 Class `ServerviewDelegate`

PythonQt.QtGui.QStyledItemDelegate

`ts3widgets.serverview.ServerviewDelegate`

Delegate to display `Serverview` items and query the properties and icons from the model to display and show them in one column.

18.6.1 Methods

```
paint(self, painter, option, index)
```

18.7 Class Serverview

PythonQt.QtGui.QTreeView —
 ts3widgets.serverview.Serverview

A QTreeView widget to display the complete view on a TS3 Server.

18.7.1 Methods

__init__(*self*, *parent*=None, *schid*=None)

Instantiates a new Serverview widget (including model and delegate).

Parameters

schid: the ID of the serverconnection
 (*type*=int)
parent: parent widget
 (*type*=QWidget)

setServerConnectionHandlerId(*self*, *schid*)

indexToObject(*self*, *index*)

Returns the underlying object of a QModelIndex.

Parameters

index: the index of the model
 (*type*=QModelIndex)

Return Value

the wrapped viewitem
 (*type*=Server or Channel or Client)

Index

- devtools (*module*), 2–4
 - devtools.installedPackages (*function*), 2
 - devtools.PluginInstaller (*class*), 2–4
 - devtools.PluginInstaller.createPlugin (*static method*), 2
 - devtools.PluginInstaller.installPackages (*method*), 3
 - devtools.PluginInstaller.installPlugin (*method*), 3
 - devtools.PluginInstaller.removePlugin (*static method*), 3
 - devtools.removePackage (*function*), 2
- pluginhost (*module*), 5–7
 - pluginhost.logprint (*function*), 5
 - pluginhost.PluginHost (*class*), 5–7
 - pluginhost.PluginHost.activate (*class method*), 5
 - pluginhost.PluginHost.callMethod (*class method*), 6
 - pluginhost.PluginHost.configure (*class method*), 6
 - pluginhost.PluginHost.deactivate (*class method*), 5
 - pluginhost.PluginHost.globalHotkeyKeyword (*class method*), 6
 - pluginhost.PluginHost.globalMenuID (*class method*), 6
 - pluginhost.PluginHost.infoData (*class method*), 6
 - pluginhost.PluginHost.init (*class method*), 5
 - pluginhost.PluginHost.initHotkeys (*class method*), 6
 - pluginhost.PluginHost.initMenus (*class method*), 6
 - pluginhost.PluginHost.invokePlugins (*class method*), 6
 - pluginhost.PluginHost.onHotkeyEvent (*class method*), 6
 - pluginhost.PluginHost.onMenuItemEvent (*class method*), 6
 - pluginhost.PluginHost.onPluginCommandEvent (*class method*), 6
 - pluginhost.PluginHost.parseUpdateReply (*class method*), 6
 - pluginhost.PluginHost.processCommand (*class method*), 6
 - pluginhost.PluginHost.registerCallbackProxy (*class method*), 6
 - pluginhost.PluginHost.reload (*class method*), 6
 - pluginhost.PluginHost.scriptingConsoleDestroyed (*class method*), 6
 - pluginhost.PluginHost.setupConfig (*class method*), 5
 - pluginhost.PluginHost.setupTranslator (*class method*), 5
 - pluginhost.PluginHost.showChangelog (*class method*), 7
 - pluginhost.PluginHost.showScriptingConsole (*class method*), 6
 - pluginhost.PluginHost.shutdown (*class method*), 5
 - pluginhost.PluginHost.start (*class method*), 5
 - pluginhost.PluginHost.startPlugin (*class method*), 5
 - pluginhost.PluginHost.unregisterCallbackProxy (*class method*), 6
 - pluginhost.PluginHost.updateCheck (*class method*), 6
 - pluginhost.PluginHost.updateCheckFinished (*class method*), 6
 - pluginhost.PluginHost.verboseLog (*class method*), 5
- pylupdate (*module*), 8–17
 - pylupdate.Context (*class*), 11–13
 - pylupdate.Context.__iter__ (*method*), 11
 - pylupdate.Context.addMessage (*method*), 12
 - pylupdate.Context.fromXml (*static method*), 12
 - pylupdate.Context.toXml (*method*), 12

- pylupdate.Context.update (*method*), 12
- pylupdate.FunctionValidator (*class*), 16–17
- pylupdate.FunctionValidator.visit_Call (*method*), (*method*), 17
- pylupdate.FunctionValidator.visit_Import (*method*), 16
- pylupdate.FunctionValidator.visit_ImportFrom (*method*), 16
- pylupdate.getSourceTexts (*function*), 8
- pylupdate.main (*function*), 8
- pylupdate.Message (*class*), 8–11
 - pylupdate.Message.__iter__ (*method*), 9
 - pylupdate.Message.disambiguation (*method*), 9
 - pylupdate.Message.fromXml (*static method*), 10
 - pylupdate.Message.isFinished (*method*), 9
 - pylupdate.Message.isNumerous (*method*), 9
 - pylupdate.Message.setTranslation (*method*), 10
 - pylupdate.Message.sourcetext (*method*), 9
 - pylupdate.Message.toXml (*method*), 10
 - pylupdate.Message.update (*method*), 10
- pylupdate.ParentVisitor (*class*), 15–16
- pylupdate.Translation (*class*), 13–15
 - pylupdate.Translation.__contains__ (*method*), 14
 - pylupdate.Translation.__getitem__ (*method*), 14
 - pylupdate.Translation.__iter__ (*method*), 14
 - pylupdate.Translation.addContext (*method*), 14
 - pylupdate.Translation.read (*method*), 13
 - pylupdate.Translation.removeContext (*method*), 15
 - pylupdate.Translation.write (*method*), 13
- pythonqtpytsn (*module*), 18–19
 - pythonqtpytsn.EventFilterObject (*class*), 18–19
 - pythonqtpytsn.EventFilterObject.__init__ (*method*), 18
 - pythonqtpytsn.EventFilterObject.addType
 - pythonqtpytsn.EventFilterObject.removeType (*method*), 19
 - pythonqtpytsn.EventFilterObject.setFilterResult (*method*), 18
 - pythonqtpytsn.EventFilterObject.types (*method*), 18
- pytsn (*module*), 20–22
 - pytsn.getConfigPath (*function*), 20
 - pytsn.getCurrentApiVersion (*function*), 21
 - pytsn.getPluginPath (*function*), 21
 - pytsn.getVersion (*function*), 21
 - pytsn.locales (*function*), 20
 - pytsn.platformstr (*function*), 21
 - pytsn.tr (*function*), 20
 - pytsn.Translatable (*class*), 22
- pytsnui (*package*), 23–27
 - pytsnui.config (*module*), 28–30
 - pytsnui.config.ConfigurationDialog (*class*), 28–30
 - pytsnui.connectSignalSlotsByName (*function*), 23
 - pytsnui.console (*module*), 31–33
 - pytsnui.console.defaultFont (*function*), 31
 - pytsnui.console.PythonConsole (*class*), 31–33
 - pytsnui.console.StdRedirector (*class*), 31
 - pytsnui.dialogs (*module*), 34
 - pytsnui.dialogs.MultiInputDialog (*class*), 34
 - pytsnui.repository (*module*), 35–37
 - pytsnui.repository.InstallDialog (*class*), 36–37
 - pytsnui.repository.RepositoryDialog (*class*), 35–36
 - pytsnui.retrieveAllWidgets (*function*), 24
 - pytsnui.retrieveWidgets (*function*), 23
 - pytsnui.setIcon (*function*), 23

- pytsonui.setupUi (*function*), 25
- pytsonui.ts3print (*function*), 23
- pytsonui.UiLoader (*class*), 26–27
 - pytsonui.UiLoader.__init__ (*method*), 27
 - pytsonui.UiLoader.createWidget (*method*), 27
- signalslot (*module*), 38–39
 - signalslot.Signal (*class*), 38–39
 - signalslot.Signal.connect (*method*), 38
 - signalslot.Signal.disconnect (*method*), 38
 - signalslot.Signal.disconnectAll (*method*), 38
 - signalslot.Signal.emit (*method*), 38
- ts3client (*module*), 40–45
 - ts3client.Config (*class*), 40
 - ts3client.Config.__del__ (*method*), 40
 - ts3client.Config.__getattr__ (*method*), 40
 - ts3client.CountryFlags (*class*), 44–45
 - ts3client.CountryFlags.__enter__ (*method*), 45
 - ts3client.CountryFlags.__exit__ (*method*), 45
 - ts3client.CountryFlags.__init__ (*method*), 44
 - ts3client.CountryFlags.close (*method*), 44
 - ts3client.CountryFlags.flag (*method*), 44
 - ts3client.CountryFlags.open (*method*), 44
 - ts3client.IconPack (*class*), 40–43
 - ts3client.IconPack.__enter__ (*method*), 41
 - ts3client.IconPack.__exit__ (*method*), 41
 - ts3client.IconPack.close (*method*), 41
 - ts3client.IconPack.current (*static method*), 41
 - ts3client.IconPack.defaultName (*static method*), 41
 - ts3client.IconPack.emoticon (*method*), 43
 - ts3client.IconPack.emoticons (*method*), 42
 - ts3client.IconPack.fallback (*method*), 42
 - ts3client.IconPack.icon (*method*), 42
 - ts3client.IconPack.icons (*method*), 42
 - ts3client.IconPack.open (*method*), 41
 - ts3client.ServerCache (*class*), 43–44
 - ts3client.ServerCache.__init__ (*method*), 44
 - ts3client.ServerCache.icon (*method*), 44
- ts3defines (*module*), 46–107
 - ts3defines.ACLType (*class*), 57
 - ts3defines.AwayStatus (*class*), 57–58
 - ts3defines.BBCodeTags (*class*), 58–59
 - ts3defines.ChannelProperties (*class*), 59–61
 - ts3defines.ChannelPropertiesRare (*class*), 61–62
 - ts3defines.ClientCommand (*class*), 62–63
 - ts3defines.ClientProperties (*class*), 63–65
 - ts3defines.ClientPropertiesRare (*class*), 65–67
 - ts3defines.ClientType (*class*), 67
 - ts3defines.CodecEncryptionMode (*class*), 67–68
 - ts3defines.CodecType (*class*), 68–69
 - ts3defines.CommandLinePropertiesRare (*class*), 69–70
 - ts3defines.ConnectionProperties (*class*), 70–73
 - ts3defines.ConnectionPropertiesRare (*class*), 73–75
 - ts3defines.ConnectStatus (*class*), 75–76
 - ts3defines.FileListType (*class*), 76
 - ts3defines.FileTransferState (*class*), 76–77
 - ts3defines.FTAction (*class*), 77–78
 - ts3defines.GroupShowNameTreeMode (*class*), 78–79
 - ts3defines.GroupWhisperTargetMode (*class*), 79–80
 - ts3defines.GroupWhisperType (*class*), 80–81
 - ts3defines.HardwareInputStatus (*class*), 81
 - ts3defines.HardwareOutputStatus (*class*), 81–82
 - ts3defines.HostBannerMode (*class*), 82–83
 - ts3defines.HostMessageMode (*class*), 83–84
 - ts3defines.InputDeactivationStatus (*class*), 84

- ts3defines.LicenseViolationType (*class*), 84–85
- ts3defines.LocalTestMode (*class*), 85–86
- ts3defines.LogLevel (*class*), 86
- ts3defines.LogTypes (*class*), 86–87
- ts3defines.MonoSoundDestination (*class*), 87–88
- ts3defines.MuteInputStatus (*class*), 88–89
- ts3defines.MuteOutputStatus (*class*), 89
- ts3defines.PluginConfigureOffer (*class*), 89–90
- ts3defines.PluginConnectTab (*class*), 90–91
- ts3defines.PluginGuiProfile (*class*), 91–92
- ts3defines.PluginItemType (*class*), 92
- ts3defines.PluginMenuType (*class*), 92–93
- ts3defines.PluginMessageTarget (*class*), 93–94
- ts3defines.PluginTargetMode (*class*), 94–95
- ts3defines.ReasonIdentifier (*class*), 95–96
- ts3defines.SecuritySaltOptions (*class*), 96–97
- ts3defines.SERVER_BINDING (*class*), 97
- ts3defines.ServerInstancePropertiesRare (*class*), 97–99
- ts3defines.TalkStatus (*class*), 99–100
- ts3defines.TextMessageTargetMode (*class*), 100–101
- ts3defines.VirtualServerProperties (*class*), 101–102
- ts3defines.VirtualServerPropertiesRare (*class*), 102–106
- ts3defines.Visibility (*class*), 106–107
- ts3lib (*module*), 108–219
 - ts3lib.acquireCustomPlaybackData (*function*), 108
 - ts3lib.activateCaptureDevice (*function*), 108
 - ts3lib.banadd (*function*), 108
 - ts3lib.banclient (*function*), 109
 - ts3lib.banclientdbid (*function*), 110
 - ts3lib.bandel (*function*), 111
 - ts3lib.bandelall (*function*), 112
 - ts3lib.channelPropertyStringToFlag (*function*), 112
 - ts3lib.channelset3DAttributes (*function*), 113
 - ts3lib.cleanUpConnectionInfo (*function*), 113
 - ts3lib.clientChatClosed (*function*), 113
 - ts3lib.clientChatComposing (*function*), 114
 - ts3lib.clientPropertyStringToFlag (*function*), 115
 - ts3lib.closeCaptureDevice (*function*), 115
 - ts3lib.closePlaybackDevice (*function*), 115
 - ts3lib.closeWaveFileHandle (*function*), 116
 - ts3lib.createBookmark (*function*), 116
 - ts3lib.createReturnCode (*function*), 117
 - ts3lib.destroyServerConnectionHandler (*function*), 118
 - ts3lib.flushChannelCreation (*function*), 118
 - ts3lib.flushChannelUpdates (*function*), 119
 - ts3lib.flushClientSelfUpdates (*function*), 119
 - ts3lib.getAppPath (*function*), 120
 - ts3lib.getAvatar (*function*), 120
 - ts3lib.getAverageTransferSpeed (*function*), 121
 - ts3lib.getBookmarkList (*function*), 121
 - ts3lib.getCaptureDeviceList (*function*), 121
 - ts3lib.getCaptureModeList (*function*), 122
 - ts3lib.getChannelClientList (*function*), 122
 - ts3lib.getChannelConnectInfo (*function*), 122
 - ts3lib.getChannelIDFromChannelNames (*function*), 123
 - ts3lib.getChannelList (*function*), 123
 - ts3lib.getChannelOfClient (*function*), 124
 - ts3lib.getChannelVariableAsInt (*function*), 124
 - ts3lib.getChannelVariableAsString (*function*), 124
 - ts3lib.getChannelVariableAsUInt64 (*function*), 125
 - ts3lib.getClientDisplayName (*function*), 125
 - ts3lib.getClientID (*function*), 126
 - ts3lib.getClientLibVersion (*function*), 126
 - ts3lib.getClientLibVersionNumber (*function*), 126

- ts3lib.getClientList (*function*), 127
- ts3lib.getClientNeededPermission (*function*), 127
- ts3lib.getClientSelfVariableAsInt (*function*), 127
- ts3lib.getClientSelfVariableAsString (*function*), 128
- ts3lib.getClientVariableAsInt (*function*), 128
- ts3lib.getClientVariableAsString (*function*), 128
- ts3lib.getClientVariableAsUInt64 (*function*), 129
- ts3lib.getConfigPath (*function*), 129
- ts3lib.getConnectionStatus (*function*), 129
- ts3lib.getConnectionVariableAsDouble (*function*), 130
- ts3lib.getConnectionVariableAsString (*function*), 130
- ts3lib.getConnectionVariableAsUInt64 (*function*), 130
- ts3lib.getCurrentCaptureDeviceName (*function*), 131
- ts3lib.getCurrentCaptureMode (*function*), 131
- ts3lib.getCurrentPlaybackDeviceName (*function*), 131
- ts3lib.getCurrentPlayBackMode (*function*), 132
- ts3lib.getCurrentServerConnectionHandlerID (*function*), 132
- ts3lib.getCurrentTransferSpeed (*function*), 132
- ts3lib.getDefaultCaptureDevice (*function*), 132
- ts3lib.getDefaultCaptureMode (*function*), 133
- ts3lib.getDefaultPlaybackDevice (*function*), 133
- ts3lib.getDefaultPlayBackMode (*function*), 133
- ts3lib.getDirectories (*function*), 133
- ts3lib.getEncodeConfigValue (*function*), 134
- ts3lib.getErrorMessage (*function*), 134
- ts3lib.getHotkeyFromKeyword (*function*), 134
- ts3lib.getParentChannelOfChannel (*function*), 135
- ts3lib.getPermissionIDByName (*function*), 135
- ts3lib.getPlaybackConfigValueAsFloat (*function*), 135
- ts3lib.getPlaybackDeviceList (*function*), 136
- ts3lib.getPlaybackModeList (*function*), 136
- ts3lib.getPluginID (*function*), 108
- ts3lib.getPluginPath (*function*), 136
- ts3lib.getPreProcessorInfoValue (*function*), 137
- ts3lib.getPreProcessorInfoValueFloat (*function*), 137
- ts3lib.getProfileList (*function*), 137
- ts3lib.getResourcesPath (*function*), 138
- ts3lib.getServerConnectInfo (*function*), 138
- ts3lib.getServerConnectionHandlerList (*function*), 138
- ts3lib.getServerVariableAsInt (*function*), 139
- ts3lib.getServerVariableAsString (*function*), 139
- ts3lib.getServerVariableAsUInt64 (*function*), 139
- ts3lib.getServerVersion (*function*), 140
- ts3lib.getTransferFileName (*function*), 140
- ts3lib.getTransferFilePath (*function*), 140
- ts3lib.getTransferFileSize (*function*), 141
- ts3lib.getTransferFileSizeDone (*function*), 141
- ts3lib.getTransferRunTime (*function*), 141
- ts3lib.getTransferStatus (*function*), 141
- ts3lib.guiConnect (*function*), 142
- ts3lib.guiConnectBookmark (*function*), 143
- ts3lib.haltTransfer (*function*), 144
- ts3lib.initiateGracefulPlaybackShutdown (*function*), 144
- ts3lib.isReceivingWhisper (*function*), 145
- ts3lib.isTransferSender (*function*), 145
- ts3lib.isWhispering (*function*), 145
- ts3lib.logMessage (*function*), 146

- ts3lib.openCaptureDevice (*function*), 146
- ts3lib.openPlaybackDevice (*function*), 147
- ts3lib.pauseWaveFileHandle (*function*), 147
- ts3lib.playWaveFile (*function*), 148
- ts3lib.playWaveFileHandle (*function*), 148
- ts3lib.printMessage (*function*), 149
- ts3lib.printMessageToCurrentTab (*function*), 149
- ts3lib.privilegeKeyUse (*function*), 149
- ts3lib.processCustomCaptureData (*function*), 150
- ts3lib.registerCustomDevice (*function*), 150
- ts3lib.requestBanList (*function*), 151
- ts3lib.requestChannelAddPerm (*function*), 152
- ts3lib.requestChannelClientAddPerm (*function*), 152
- ts3lib.requestChannelClientDelPerm (*function*), 153
- ts3lib.requestChannelClientPermList (*function*), 154
- ts3lib.requestChannelDelete (*function*), 155
- ts3lib.requestChannelDelPerm (*function*), 156
- ts3lib.requestChannelDescription (*function*), 157
- ts3lib.requestChannelGroupAdd (*function*), 157
- ts3lib.requestChannelGroupAddPerm (*function*), 158
- ts3lib.requestChannelGroupDel (*function*), 159
- ts3lib.requestChannelGroupDelPerm (*function*), 160
- ts3lib.requestChannelGroupList (*function*), 161
- ts3lib.requestChannelGroupPermList (*function*), 162
- ts3lib.requestChannelMove (*function*), 162
- ts3lib.requestChannelPermList (*function*), 163
- ts3lib.requestChannelSubscribe (*function*), 164
- ts3lib.requestChannelSubscribeAll (*function*), 164
- ts3lib.requestChannelUnsubscribe (*function*), 165
- ts3lib.requestChannelUnsubscribeAll (*function*), 165
- ts3lib.requestClientAddPerm (*function*), 166
- ts3lib.requestClientDBIDfromUID (*function*), 166
- ts3lib.requestClientDelPerm (*function*), 167
- ts3lib.requestClientEditDescription (*function*), 167
- ts3lib.requestClientIDs (*function*), 168
- ts3lib.requestClientKickFromChannel (*function*), 168
- ts3lib.requestClientKickFromServer (*function*), 169
- ts3lib.requestClientMove (*function*), 169
- ts3lib.requestClientNamefromDBID (*function*), 170
- ts3lib.requestClientNamefromUID (*function*), 171
- ts3lib.requestClientPermList (*function*), 171
- ts3lib.requestClientPoke (*function*), 172
- ts3lib.requestClientSetIsTalker (*function*), 172
- ts3lib.requestClientSetWhisperList (*function*), 173
- ts3lib.requestClientVariables (*function*), 174
- ts3lib.requestComplainAdd (*function*), 175
- ts3lib.requestComplainDel (*function*), 175
- ts3lib.requestComplainDelAll (*function*), 176
- ts3lib.requestComplainList (*function*), 177
- ts3lib.requestConnectionInfo (*function*), 177
- ts3lib.requestCreateDirectory (*function*), 178
- ts3lib.requestDeleteFile (*function*), 179
- ts3lib.requestFile (*function*), 180
- ts3lib.requestFileInfo (*function*), 181
- ts3lib.requestFileList (*function*), 182
- ts3lib.requestHotkeyInputDialog (*function*), 183
- ts3lib.requestInfoUpdate (*function*), 184
- ts3lib.requestIsTalker (*function*), 184

- ts3lib.requestMessageAdd (*function*), 185
- ts3lib.requestMessageDel (*function*), 186
- ts3lib.requestMessageGet (*function*), 187
- ts3lib.requestMessageList (*function*), 187
- ts3lib.requestMessageUpdateFlag (*function*), 188
- ts3lib.requestMuteClients (*function*), 188
- ts3lib.requestPermissionList (*function*), 189
- ts3lib.requestPermissionOverview (*function*), 189
- ts3lib.requestRenameFile (*function*), 190
- ts3lib.requestSendChannelTextMsg (*function*), 191
- ts3lib.requestSendClientQueryCommand (*function*), 192
- ts3lib.requestSendPrivateTextMsg (*function*), 192
- ts3lib.requestSendServerTextMsg (*function*), 193
- ts3lib.requestServerGroupAdd (*function*), 193
- ts3lib.requestServerGroupAddClient (*function*), 194
- ts3lib.requestServerGroupAddPerm (*function*), 195
- ts3lib.requestServerGroupClientList (*function*), 196
- ts3lib.requestServerGroupDel (*function*), 197
- ts3lib.requestServerGroupDelClient (*function*), 198
- ts3lib.requestServerGroupDelPerm (*function*), 199
- ts3lib.requestServerGroupList (*function*), 200
- ts3lib.requestServerGroupPermList (*function*), 201
- ts3lib.requestServerGroupsByClientID (*function*), 201
- ts3lib.requestServerTemporaryPasswordAdd (*function*), 202
- ts3lib.requestServerTemporaryPasswordDel (*function*), 203
- ts3lib.requestServerTemporaryPasswordList (*function*), 204
- ts3lib.requestServerVariables (*function*), 204
- ts3lib.requestSetClientChannelGroup (*function*), 205
- ts3lib.requestUnmuteClients (*function*), 205
- ts3lib.sendFile (*function*), 206
- ts3lib.sendPluginCommand (*function*), 207
- ts3lib.serverPropertyStringToFlag (*function*), 208
- ts3lib.set3DWaveAttributes (*function*), 208
- ts3lib.setChannelVariableAsInt (*function*), 209
- ts3lib.setChannelVariableAsString (*function*), 209
- ts3lib.setChannelVariableAsUInt64 (*function*), 210
- ts3lib.setClientSelfVariableAsInt (*function*), 210
- ts3lib.setClientSelfVariableAsString (*function*), 211
- ts3lib.setClientVolumeModifier (*function*), 211
- ts3lib.setPlaybackConfigValue (*function*), 212
- ts3lib.setPluginMenuEnabled (*function*), 212
- ts3lib.setPreProcessorConfigValue (*function*), 212
- ts3lib.showHotkeySetup (*function*), 213
- ts3lib.spawnNewServerConnectionHandler (*function*), 213
- ts3lib.startConnection (*function*), 213
- ts3lib.startVoiceRecording (*function*), 214
- ts3lib.stopConnection (*function*), 215
- ts3lib.stopVoiceRecording (*function*), 215
- ts3lib.systemset3DListenerAttributes (*function*), 215
- ts3lib.systemset3DSettings (*function*), 216
- ts3lib.unregisterCustomDevice (*function*), 217
- ts3lib.urlsToBB (*function*), 217
- ts3lib.verifyChannelPassword (*function*), 218
- ts3lib.verifyServerPassword (*function*), 218

- ts3plugin (*module*), 220–273
 - ts3plugin.PluginMount (*class*), 220
 - ts3plugin.ts3plugin (*class*), 220–273
 - ts3plugin.ts3plugin.configure (*method*), 221
 - ts3plugin.ts3plugin.currentServerConnectionClass (*method*), 228
 - ts3plugin.ts3plugin.infoData (*method*), 221
 - ts3plugin.ts3plugin.menuCreated (*method*), 221
 - ts3plugin.ts3plugin.onAvatarUpdated (*method*), 228
 - ts3plugin.ts3plugin.onBanListEvent (*method*), 228
 - ts3plugin.ts3plugin.onChannelClientPermListEvent (*method*), 229
 - ts3plugin.ts3plugin.onChannelClientPermListFinishedEvent (*method*), 230
 - ts3plugin.ts3plugin.onChannelDescriptionUpdateEvent (*method*), 230
 - ts3plugin.ts3plugin.onChannelGroupListEvent (*method*), 231
 - ts3plugin.ts3plugin.onChannelGroupListFinishedEvent (*method*), 231
 - ts3plugin.ts3plugin.onChannelGroupPermListEvent (*method*), 232
 - ts3plugin.ts3plugin.onChannelGroupPermListFinishedEvent (*method*), 232
 - ts3plugin.ts3plugin.onChannelMoveEvent (*method*), 232
 - ts3plugin.ts3plugin.onChannelPasswordChangeEvent (*method*), 233
 - ts3plugin.ts3plugin.onChannelPermListEvent (*method*), 233
 - ts3plugin.ts3plugin.onChannelPermListFinishedEvent (*method*), 234
 - ts3plugin.ts3plugin.onChannelSubscribeEvent (*method*), 234
 - ts3plugin.ts3plugin.onChannelSubscribeFinishedEvent (*method*), 234
 - ts3plugin.ts3plugin.onChannelUnsubscribeEvent (*method*), 235
 - ts3plugin.ts3plugin.onChannelUnsubscribeFinishedEvent (*method*), 235
 - (*method*), 235
 - ts3plugin.ts3plugin.onClientBanFromServerEvent (*method*), 235
 - ts3plugin.ts3plugin.onClientChannelGroupChangedEvent (*method*), 236
 - ts3plugin.ts3plugin.onClientChatClosedEvent (*method*), 237
 - ts3plugin.ts3plugin.onClientChatComposingEvent (*method*), 238
 - ts3plugin.ts3plugin.onClientDBIDfromUIDEvent (*method*), 238
 - ts3plugin.ts3plugin.onClientDisplayNameChangedEvent (*method*), 238
 - ts3plugin.ts3plugin.onClientIDsEvent (*method*), 239
 - ts3plugin.ts3plugin.onClientIDsFinishedEvent (*method*), 239
 - ts3plugin.ts3plugin.onClientKickFromChannelEvent (*method*), 239
 - ts3plugin.ts3plugin.onClientKickFromServerEvent (*method*), 240
 - ts3plugin.ts3plugin.onClientMoveEvent (*method*), 241
 - ts3plugin.ts3plugin.onClientMoveMovedEvent (*method*), 242
 - ts3plugin.ts3plugin.onClientMoveSubscriptionEvent (*method*), 243
 - ts3plugin.ts3plugin.onClientMoveTimeoutEvent (*method*), 244
 - ts3plugin.ts3plugin.onClientNamefromDBIDEvent (*method*), 245
 - ts3plugin.ts3plugin.onClientNamefromUIDEvent (*method*), 245
 - ts3plugin.ts3plugin.onClientNeededPermissionsEvent (*method*), 246
 - ts3plugin.ts3plugin.onClientNeededPermissionsFinishedEvent (*method*), 246
 - ts3plugin.ts3plugin.onClientPermListEvent (*method*), 246
 - ts3plugin.ts3plugin.onClientPermListFinishedEvent (*method*), 247
 - ts3plugin.ts3plugin.onClientPokeEvent (*method*), 224
 - ts3plugin.ts3plugin.onClientSelfVariableUpdateEvent (*method*), 235

- (method), 247
 ts3plugin.ts3plugin.onClientServerQueryLoginPasswordEvent
 (method), 248
 ts3plugin.ts3plugin.onComplainListEvent
 (method), 248
 ts3plugin.ts3plugin.onConnectionInfoEvent
 (method), 249
 ts3plugin.ts3plugin.onConnectStatusChangeEvent
 (method), 249
 ts3plugin.ts3plugin.onDelChannelEvent
 (method), 250
 ts3plugin.ts3plugin.onFileInfoEvent (method),
 250
 ts3plugin.ts3plugin.onFileListEvent (method),
 251
 ts3plugin.ts3plugin.onFileListFinishedEvent
 (method), 252
 ts3plugin.ts3plugin.onFileTransferStatusEvent
 (method), 227
 ts3plugin.ts3plugin.onHotkeyEvent (method),
 253
 ts3plugin.ts3plugin.onHotkeyRecordedEvent
 (method), 253
 ts3plugin.ts3plugin.onIncomingClientQueryEvent
 (method), 253
 ts3plugin.ts3plugin.onMenuItemEvent (method),
 253
 ts3plugin.ts3plugin.onMessageGetEvent
 (method), 254
 ts3plugin.ts3plugin.onMessageListEvent
 (method), 255
 ts3plugin.ts3plugin.onNewChannelCreatedEvent
 (method), 256
 ts3plugin.ts3plugin.onNewChannelEvent
 (method), 257
 ts3plugin.ts3plugin.onPermissionListEvent
 (method), 257
 ts3plugin.ts3plugin.onPermissionListFinishedEvent
 (method), 258
 ts3plugin.ts3plugin.onPermissionListGroupEnd
 (method), 258
 ts3plugin.ts3plugin.onPermissionOverviewEvent
 (method), 258
 ts3plugin.ts3plugin.onPermissionOverviewFinishedEvent
 (method), 259
 ts3plugin.ts3plugin.onPlaybackShutdownCompleteEvent
 (method), 260
 ts3plugin.ts3plugin.onPluginCommandEvent
 (method), 260
 ts3plugin.ts3plugin.onServerConnectionInfoEvent
 (method), 260
 ts3plugin.ts3plugin.onServerEditedEvent
 (method), 260
 ts3plugin.ts3plugin.onServerErrorEvent
 (method), 222
 ts3plugin.ts3plugin.onServerGroupByClientIDEvent
 (method), 261
 ts3plugin.ts3plugin.onServerGroupClientAddedEvent
 (method), 261
 ts3plugin.ts3plugin.onServerGroupClientDeletedEvent
 (method), 262
 ts3plugin.ts3plugin.onServerGroupClientListEvent
 (method), 263
 ts3plugin.ts3plugin.onServerGroupListEvent
 (method), 264
 ts3plugin.ts3plugin.onServerGroupListFinishedEvent
 (method), 265
 ts3plugin.ts3plugin.onServerGroupPermListEvent
 (method), 265
 ts3plugin.ts3plugin.onServerGroupPermListFinishedEvent
 (method), 266
 ts3plugin.ts3plugin.onServerLogEvent (method),
 266
 ts3plugin.ts3plugin.onServerLogFinishedEvent
 (method), 267
 ts3plugin.ts3plugin.onServerPermissionErrorEvent
 (method), 225
 ts3plugin.ts3plugin.onServerStopEvent (method),
 267
 ts3plugin.ts3plugin.onServerTemporaryPasswordListEvent
 (method), 267
 ts3plugin.ts3plugin.onServerUpdatedEvent
 (method), 268
 ts3plugin.ts3plugin.onSoundDeviceListChangedEvent
 (method), 268
 ts3plugin.ts3plugin.onTalkStatusChangeEvent
 (method), 269
 ts3plugin.ts3plugin.onTextMessageEvent

- (method), 223
- ts3plugin.ts3plugin.onUpdateChannelEditedEvent
 - (method), 269
- ts3plugin.ts3plugin.onUpdateChannelEvent
 - (method), 270
- ts3plugin.ts3plugin.onUpdateClientEvent
 - (method), 270
- ts3plugin.ts3plugin.onUserLoggingMessageEvent
 - (method), 226
- ts3plugin.ts3plugin.processCommand (method), 222
- ts3plugin.ts3plugin.stop (method), 221
- ts3widgets (package), 274
 - ts3widgets.filetransfer (module), 275–294
 - ts3widgets.filetransfer.bytesToStr (function), 275
 - ts3widgets.filetransfer.Download (class), 287–288
 - ts3widgets.filetransfer.File (class), 275–276
 - ts3widgets.filetransfer.FileBrowser (class), 280–283
 - ts3widgets.filetransfer.FileCollector (class), 279–280
 - ts3widgets.filetransfer.FileCollisionAction (class), 283–284
 - ts3widgets.filetransfer.FileCollisionDialog (class), 284–286
 - ts3widgets.filetransfer.FileListModel (class), 276–278
 - ts3widgets.filetransfer.FileTransfer (class), 286–287
 - ts3widgets.filetransfer.FileTransferDelegate (class), 291–292
 - ts3widgets.filetransfer.FileTransferDialog (class), 292–294
 - ts3widgets.filetransfer.FileTransferModel (class), 289–291
 - ts3widgets.filetransfer.joinpath (function), 275
 - ts3widgets.filetransfer.SmartStatusBar (class), 278–279
 - ts3widgets.filetransfer.splitpath (function), 275
 - ts3widgets.filetransfer.Upload (class), 288–289
 - ts3widgets.serverview (module), 295–303
 - ts3widgets.serverview.Channel (class), 295–297
 - ts3widgets.serverview.Client (class), 298–300
 - ts3widgets.serverview.Server (class), 297–298
 - ts3widgets.serverview.Serverview (class), 302–303
 - ts3widgets.serverview.ServerviewDelegate (class), 302
 - ts3widgets.serverview.ServerviewModel (class), 300–302
 - ts3widgets.serverview.ServerViewRoles (class), 295