

# PSEUDO-IRC-SERVER

## 1.0

Generated by Doxygen 1.8.3.1

Sat Nov 16 2013 18:42:47



# Contents

<b>1</b>	<b>IRC</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Namespace Documentation</b>	<b>11</b>
6.1	CMD Namespace Reference . . . . .	11
6.1.1	Detailed Description . . . . .	11
6.1.2	Enumeration Type Documentation . . . . .	11
6.1.2.1	anonymous enum . . . . .	11
6.2	ERROR Namespace Reference . . . . .	12
6.2.1	Detailed Description . . . . .	12
6.2.2	Enumeration Type Documentation . . . . .	12
6.2.2.1	anonymous enum . . . . .	12
<b>7</b>	<b>Class Documentation</b>	<b>13</b>
7.1	ban Class Reference . . . . .	13
7.1.1	Constructor & Destructor Documentation . . . . .	13
7.1.1.1	ban . . . . .	13
7.1.2	Member Function Documentation . . . . .	13
7.1.2.1	verify . . . . .	13
7.2	banlist Class Reference . . . . .	14
7.2.1	Constructor & Destructor Documentation . . . . .	14
7.2.1.1	banlist . . . . .	14
7.2.2	Member Function Documentation . . . . .	14

7.2.2.1	verify	14
7.3	Channel Class Reference	15
7.3.1	Detailed Description	15
7.3.2	Constructor & Destructor Documentation	16
7.3.2.1	Channel	16
7.3.3	Member Function Documentation	16
7.3.3.1	addClient	16
7.3.3.2	getChannelName	16
7.3.3.3	getClientList	16
7.3.3.4	getTopic	16
7.3.3.5	isEmpty	17
7.3.3.6	isStatus	17
7.3.3.7	removeClient	17
7.3.3.8	setOperator	17
7.3.3.9	setTopic	17
7.3.3.10	unbanClient	18
7.3.3.11	unsetOperator	18
7.4	Client Class Reference	18
7.4.1	Detailed Description	19
7.4.2	Constructor & Destructor Documentation	19
7.4.2.1	Client	19
7.4.3	Member Function Documentation	19
7.4.3.1	getMsg	19
7.4.3.2	getNickname	19
7.4.3.3	getSocket	19
7.4.3.4	getState	20
7.4.3.5	onDataReady	20
7.4.3.6	setMsg	20
7.4.3.7	setNickname	20
7.4.3.8	setSocket	20
7.4.3.9	setState	20
7.5	Command Class Reference	21
7.5.1	Detailed Description	22
7.5.2	Member Function Documentation	22
7.5.2.1	getCommand	22
7.6	cwho Class Reference	22
7.6.1	Constructor & Destructor Documentation	22
7.6.1.1	cwho	22
7.6.2	Member Function Documentation	23
7.6.2.1	verify	23

7.7	deop Class Reference	23
7.7.1	Constructor & Destructor Documentation	23
7.7.1.1	deop	23
7.7.2	Member Function Documentation	24
7.7.2.1	verify	24
7.8	Frame Class Reference	24
7.8.1	Detailed Description	24
7.8.2	Constructor & Destructor Documentation	24
7.8.2.1	Frame	24
7.8.3	Member Function Documentation	25
7.8.3.1	getArgList	25
7.8.3.2	getCode	25
7.8.3.3	getId	25
7.8.3.4	getNbArg	25
7.8.3.5	getReadyToSendFrame	25
7.8.3.6	getSize	25
7.9	gwho Class Reference	26
7.9.1	Constructor & Destructor Documentation	26
7.9.1.1	gwho	26
7.9.2	Member Function Documentation	26
7.9.2.1	verify	26
7.10	join Class Reference	26
7.10.1	Constructor & Destructor Documentation	27
7.10.1.1	join	27
7.10.2	Member Function Documentation	27
7.10.2.1	verify	27
7.11	kick Class Reference	27
7.11.1	Constructor & Destructor Documentation	28
7.11.1.1	kick	28
7.11.2	Member Function Documentation	28
7.11.2.1	verify	28
7.12	leave Class Reference	28
7.12.1	Constructor & Destructor Documentation	29
7.12.1.1	leave	29
7.12.2	Member Function Documentation	29
7.12.2.1	verify	29
7.13	list Class Reference	29
7.13.1	Constructor & Destructor Documentation	30
7.13.1.1	list	30
7.13.2	Member Function Documentation	30

7.13.2.1	verify	30
7.14	nick Class Reference	30
7.14.1	Constructor & Destructor Documentation	31
7.14.1.1	nick	31
7.14.2	Member Function Documentation	31
7.14.2.1	verify	31
7.15	op Class Reference	31
7.15.1	Constructor & Destructor Documentation	32
7.15.1.1	op	32
7.15.2	Member Function Documentation	32
7.15.2.1	verify	32
7.16	privmsg Class Reference	32
7.16.1	Constructor & Destructor Documentation	33
7.16.1.1	privmsg	33
7.16.2	Member Function Documentation	33
7.16.2.1	verify	33
7.17	pubmsg Class Reference	33
7.17.1	Constructor & Destructor Documentation	34
7.17.1.1	pubmsg	34
7.17.2	Member Function Documentation	34
7.17.2.1	verify	34
7.18	Server Class Reference	34
7.18.1	Detailed Description	35
7.18.2	Constructor & Destructor Documentation	36
7.18.2.1	Server	36
7.18.3	Member Function Documentation	36
7.18.3.1	broadCast	36
7.18.3.2	getChannelFromName	36
7.18.3.3	getClientFromName	36
7.18.3.4	init	36
7.19	topic Class Reference	36
7.19.1	Constructor & Destructor Documentation	37
7.19.1.1	topic	37
7.19.2	Member Function Documentation	37
7.19.2.1	verify	37
7.20	unban Class Reference	37
7.20.1	Constructor & Destructor Documentation	38
7.20.1.1	unban	38
7.20.2	Member Function Documentation	38
7.20.2.1	verify	38

<b>8 File Documentation</b>	<b>39</b>
8.1 channel.h File Reference . . . . .	39
8.1.1 Detailed Description . . . . .	39
8.1.2 Enumeration Type Documentation . . . . .	39
8.1.2.1 status . . . . .	39
8.2 client.h File Reference . . . . .	40
8.2.1 Detailed Description . . . . .	40
8.3 command.h File Reference . . . . .	40
8.3.1 Detailed Description . . . . .	41
8.4 frame.h File Reference . . . . .	41
8.4.1 Detailed Description . . . . .	41
8.5 server.h File Reference . . . . .	41
8.5.1 Detailed Description . . . . .	41
 <b>Index</b>	 <b>41</b>





# Chapter 1

## IRC

IRC [Server](#)

### I - How to run

- Download the repository
- In the directory use the command 'qmake' and then 'make'
- Start the server with './irc\_server'
- If you get the error "undefined reference to vtable for ..." try this : make clean; qmake; make



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">CMD</a>	.....	<a href="#">11</a>
<a href="#">ERROR</a>	.....	<a href="#">12</a>



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Channel . . . . .	15
Command . . . . .	21
ban . . . . .	13
banlist . . . . .	14
cwho . . . . .	22
deop . . . . .	23
gwho . . . . .	26
join . . . . .	26
kick . . . . .	27
leave . . . . .	28
list . . . . .	29
nick . . . . .	30
op . . . . .	31
privmsg . . . . .	32
pubmsg . . . . .	33
topic . . . . .	36
unban . . . . .	37
Frame . . . . .	24
QObject	
Client . . . . .	18
Server . . . . .	34



## Chapter 4

# Class Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">ban</a>	13
<a href="#">banlist</a>	14
<a href="#">Channel</a>	
Class representing a channel	15
<a href="#">Client</a>	
Class representing a <a href="#">Client</a> connected to the server. This class inherits from QObject to use slots and signals	18
<a href="#">Command</a>	
Class representing an abstract command	21
<a href="#">cwho</a>	22
<a href="#">deop</a>	23
<a href="#">Frame</a>	
This class provides tools for frame analysis	24
<a href="#">gwho</a>	26
<a href="#">join</a>	26
<a href="#">kick</a>	27
<a href="#">leave</a>	28
<a href="#">list</a>	29
<a href="#">nick</a>	30
<a href="#">op</a>	31
<a href="#">privmsg</a>	32
<a href="#">pubmsg</a>	33
<a href="#">Server</a>	
Class defining the <a href="#">Server</a> . This class inherits from QObject to use slots and signals. This class uses the pattern Singleton	34
<a href="#">topic</a>	36
<a href="#">unban</a>	37





## Chapter 5

# File Index

### 5.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">channel.h</a>	This file gathers tools to manage clients inside channels . . . . .	39
<a href="#">client.h</a>	This file gathers informations on a client connected to the server using a TCP connection . . .	40
<a href="#">command.h</a>	File containing the declaration of the commands . . . . .	40
<a href="#">frame.h</a>	<a href="#">Frame</a> analyser based on a Pseudo-Irc protocol . . . . .	41
<a href="#">server.h</a>	IRC <a href="#">Server</a> Near-IRC server : A simplified IRC server not in accordance with RFC 1459 . . . .	41



## Chapter 6

# Namespace Documentation

### 6.1 CMD Namespace Reference

#### Enumerations

- enum {  
    C\_PRIVMSG = 1, C\_PUBMSG, C\_GWHO, C\_CWHO,  
    C\_LIST, C\_TOPIC, C\_KICK, C\_BAN,  
    C\_OP, C\_DEOP = 20, C\_JOIN, C\_NICK,  
    C\_LEAVE, C\_UNBAN, C\_BANLIST }

#### 6.1.1 Detailed Description

Namespace gathering command codes referring to the commands

#### 6.1.2 Enumeration Type Documentation

##### 6.1.2.1 anonymous enum

#### Enumerator

**C\_PRIVMSG** [Command](#) private message  
**C\_PUBMSG** [Command](#) public message  
**C\_GWHO** [Command](#) general who  
**C\_CWHO** [Command](#) who on a channel  
**C\_LIST** [Command](#) list  
**C\_TOPIC** [Command](#) topic  
**C\_KICK** [Command](#) kick  
**C\_BAN** [Command](#) ban  
**C\_OP** [Command](#) op  
**C\_DEOP** [Command](#) deop  
**C\_JOIN** [Command](#) join  
**C\_NICK** [Command](#) nick  
**C\_LEAVE** [Command](#) leave  
**C\_UNBAN** [Command](#) unban  
**C\_BANLIST** [Command](#) banlist

## 6.2 ERROR Namespace Reference

### Enumerations

- enum {  
    [esuccess](#) = 0, [eBadArg](#) = 250, [eNickCollision](#), [eNotAuthorised](#),  
    [eMissingArg](#), [eNotExist](#), [error](#) }

#### 6.2.1 Detailed Description

Namespace gathering error codes binded to the commands

#### 6.2.2 Enumeration Type Documentation

##### 6.2.2.1 anonymous enum

###### Enumerator

***esuccess*** [Command](#) has been executed successfully

***eBadArg*** The parameter is non-compliant

***eNickCollision*** The nickname is already in use by another client

***eNotAuthorised*** Not enough rights to use this command

***eMissingArg*** A parameter is missing to use this command correctly

***eNotExist*** The argument refers to a client/channel that doesn't exist

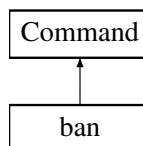
***error*** Other errors

## Chapter 7

# Class Documentation

### 7.1 ban Class Reference

Inheritance diagram for ban:



#### Public Member Functions

- **ban** ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 **verify** ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 **execute** ()

#### Additional Inherited Members

##### 7.1.1 Constructor & Destructor Documentation

###### 7.1.1.1 ban::ban ( [Client](#) \* sender, [Frame](#) & frame )

Constructor.

###### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

##### 7.1.2 Member Function Documentation

###### 7.1.2.1 quint8 ban::verify ( ) [virtual]

Check the validity of the command according to the parameters contained in the frame given to the construcor.

The method check wether one argument is missing or not and then check wether the regex given is valid or not.

**Returns**

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

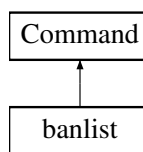
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.2 banlist Class Reference

Inheritance diagram for banlist:

**Public Member Functions**

- [banlist](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 [execute](#) ()

**Additional Inherited Members**

### 7.2.1 Constructor & Destructor Documentation

#### 7.2.1.1 banlist::banlist ( Client \* sender, Frame & frame )

Constructor.

**Parameters**

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.2.2 Member Function Documentation

#### 7.2.2.1 quint8 banlist::verify ( ) [virtual]

Check the validity of the command according to the parameters contained in the frame given to the construcor.

The method check wether one argument is missing or not.

**Returns**

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.3 Channel Class Reference

Class representing a channel.

**Public Member Functions**

- [Channel](#) (const QString &name)  
*Constructor.*
- void [setTopic](#) (const QString &topic)  
*Set the topic of the channel.*
- QString & [getTopic](#) (void)  
*Getter of the topic of the channel.*
- QString & [getChannelName](#) (void)  
*Getter of the name of the channel.*
- std::list< [Client](#) \* > & [getClientList](#) (status s=[REGULAR](#))  
*Getter of the client lists of the channel.*
- void [addClient](#) ([Client](#) \*c, status s=[REGULAR](#))  
*Allows to add and/or set the status of a client on the channel.*
- void [removeClient](#) ([Client](#) \*c)  
*Remove a client from a channel.*
- void [unbanClient](#) ([Client](#) \*c)  
*Unban a client.*
- bool [isStatus](#) ([Client](#) \*c, status s=[REGULAR](#))  
*Check the status of a client on a channel.*
- void [setOperator](#) ([Client](#) \*c)  
*Set a client operator.*
- void [unsetOperator](#) ([Client](#) \*c)  
*Unset operator a client.*
- bool [isEmpty](#) (void)

### 7.3.1 Detailed Description

Class representing a channel.

This class manage 3 lists of clients : banned, regular (connected to the channel including operators) and operator clients.

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 Channel::Channel ( const QString & name )

Constructor.

Parameters

<i>name</i>	: Name of the channel.
-------------	------------------------

### 7.3.3 Member Function Documentation

#### 7.3.3.1 void Channel::addClient ( Client \* c, status s = REGULAR )

Allows to add and/or set the status of a client on the channel.

Parameters

<i>c</i>	: The address of the client you want to add and or change the status on the channel.
<i>s</i>	: The status you want to give to the client you are adding.

Returns

void.

#### 7.3.3.2 QString & Channel::getChannelName ( void )

Getter of the name of the channel.

Returns

A string containing the name of the channel.

#### 7.3.3.3 std::list< Client \* > & Channel::getClientList ( status s = REGULAR )

Getter of the client lists of the channel.

For example : sampleChannel.getClientList(OPERATOR) returns the operator clients list of the channel.

If no status is given, the getter returns the connected clients list.

Parameters

<i>s</i>	: The client list you want to get
----------	-----------------------------------

Returns

A list (from STL) containing the addresses of clients depending on the status given in parameter.

#### 7.3.3.4 QString & Channel::getTopic ( void )

Getter of the topic of the channel.

Returns

A string containing the topic of the channel.



### 7.3.3.5 bool Channel::isEmpty ( void )

#### Returns

True if the channel is empty, false either.

### 7.3.3.6 bool Channel::isStatus ( Client \* c, status s = REGULAR )

Check the status of a client on a channel.

#### Parameters

<i>c</i>	: The address of the client you want to check the status.
<i>s</i>	: The status you want to check if the client has.

#### Returns

True if the client *c* is in the list corresponding to status given, false either.

### 7.3.3.7 void Channel::removeClient ( Client \* c )

Remove a client from a channel.

Remove a client from the regular and operator lists if the client is operator on the channel. This function doesn't intend to remove a client from the ban list, if you want to do so, you should use [unbanClient\(\)](#).

#### Parameters

<i>c</i>	: The address of the client you want to remove from the channel.
----------	--

#### Returns

void.

### 7.3.3.8 void Channel::setOperator ( Client \* c )

Set a client operator.

#### Parameters

<i>c</i>	: The address of the client you want to set operator.
----------	---

#### Returns

void.

### 7.3.3.9 void Channel::setTopic ( const QString & topic )

Set the topic of the channel.

#### Parameters

<i>topic</i>	: A string containing the topic of the channel.
--------------	---

**Returns**

void.

**7.3.3.10 void Channel::unbanClient ( Client \* c )**

Unban a client.

**Parameters**

<b>c</b>	: The address of the client you want to unban.
----------	--

**Returns**

void.

**7.3.3.11 void Channel::unsetOperator ( Client \* c )**

Unset operator a client.

**Parameters**

<b>c</b>	: The address of the client you want to unset operator.
----------	---

**Returns**

void.

The documentation for this class was generated from the following files:

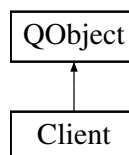
- [channel.h](#)
- channel.cpp

## 7.4 Client Class Reference

Class representing a [Client](#) connected to the server. This class inherits from QObject to use slots and signals.

```
#include <client.h>
```

Inheritance diagram for Client:

**Public Slots**

- void [onDisconnection](#) (void)  
*Remove the client from the server list of clients when it has left the server. SLOT connected to SIGNAL disconnected() of the QTcpSocket attribute.*
- void [onDataReady](#) ()

## Public Member Functions

- [Client](#) (QTcpSocket \*socket, QObject \*parent)  
*Constructor.*
- [~Client](#) ()  
*Destructeur Frees up the memory allocated for the attribute m\_socket.*
- void [setNickname](#) (const QString &nickname)
- void [setSocket](#) (QTcpSocket \*socket)
- void [setState](#) (bool state)
- QTcpSocket \* [getSocket](#) (void) const
- QString [getNickname](#) (void) const
- QString [getMsg](#) (void) const
- void [setMsg](#) (const QString &msg)
- bool [getState](#) (void) const

### 7.4.1 Detailed Description

Class representing a [Client](#) connected to the server. This class inherits from QObject to use slots and signals.

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 Client::Client ( QTcpSocket \* socket, QObject \* parent )

Constructor.

##### Parameters

<i>socket</i>	: the socket used to communicate with the server. Given by the server.
<i>parent</i>	: The adress of the <a href="#">Server</a> instance. The client is binded to the server, when the server shuts down, the client is destroyed regularly by the application. It prevents problems like memory leak.

### 7.4.3 Member Function Documentation

#### 7.4.3.1 QString Client::getMsg ( void ) const

Get the current message held by the client.

##### Returns

A QString containing the current message held by the client.

#### 7.4.3.2 QString Client::getNickname ( void ) const

Get the nickname of the client

##### Returns

A QString containing the nickname of the client.

#### 7.4.3.3 QTcpSocket \* Client::getSocket ( void ) const

Get the socket used by the client to communicate with the server.

**Returns**

The address of the socket used by the client to communicate with the server.

**7.4.3.4 bool Client::getState ( void ) const**

Get the current state of the client

**Returns**

A boolean containing true if the client has a nickname, false either.

**7.4.3.5 void Client::onDataReady ( ) [slot]**

data sent by the client program corresponding.

Read and execute a command sent by the client program corresponding to the instance of the client. SLOT connected to SIGNAL readyRead() of the QTcpSocket attribute.

**7.4.3.6 void Client::setMsg ( const QString & msg )**

Set the message of the client.

**Parameters**

<i>msg</i>	: A QString depicting the result of the last command sent by the client.
------------	--

**7.4.3.7 void Client::setNickname ( const QString & nickname )**

Set the nickname of the client.

**Parameters**

<i>nickname</i>	: A QString containing the new nickname for the client.
-----------------	---

**7.4.3.8 void Client::setSocket ( QTcpSocket \* socket )**

Set the socket of the client.

**Parameters**

<i>socket</i>	: A QTcpSocket pointer containing the address of the QTcpSocket for the client.
---------------	---

**7.4.3.9 void Client::setState ( bool state )**

Set the state of the client.

**Parameters**

<i>state</i>	: A boolean containing the new state of the client depending on whether it has set its nickname(true) or not(false).
--------------	--

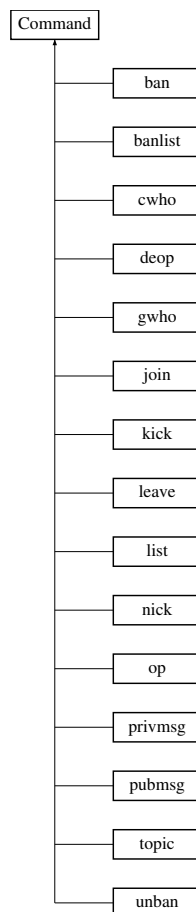
The documentation for this class was generated from the following files:

- [client.h](#)
- [client.cpp](#)

## 7.5 Command Class Reference

Class representing an abstract command.

Inheritance diagram for Command:



### Public Member Functions

- virtual quint8 **execute** ()=0
- virtual quint8 **verify** ()=0

### Static Public Member Functions

- static [Command](#) \* [getCommand](#) ([Client](#) \*c, [Frame](#) &frame)  
*Get a parameterized command object from a raw [Frame](#) object.*

### Protected Member Functions

- [Command](#) ()  
*Constructor.*

### 7.5.1 Detailed Description

Class representing an abstract command.

This class encapsulate a request as an object. It uses the pattern [Command](#).

### 7.5.2 Member Function Documentation

#### 7.5.2.1 `Command * Command::getCommand ( Client * c, Frame & frame )` [static]

Get a parameterized command object from a raw [Frame](#) object.

This method uses the parameters and the command id code contained in the [Frame](#) object to return a specialized command (inherited from [Command](#)) and parameterized correctly, ready to be self verified and executed.

##### Parameters

<code>c</code>	: The client who requests the <a href="#">Command</a> object.
<code>frame</code>	: The frame providing all the informations to instantiate the right <a href="#">Command</a> object well parameterized.

##### Returns

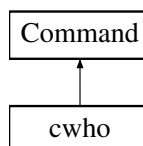
A pointer on the [Command](#) object parameterized according to the [Frame](#) object.

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.6 cwho Class Reference

Inheritance diagram for cwho:



### Public Member Functions

- `cwho (Client *sender, Frame &frame)`  
*Constructor.*
- `virtual quint8 verify ()`  
*Check the validity of the command according to the parameters contained in the frame given to the constructor.*
- `virtual quint8 execute ()`

### Additional Inherited Members

#### 7.6.1 Constructor & Destructor Documentation

##### 7.6.1.1 `cwho::cwho ( Client * sender, Frame & frame )`

Constructor.

## Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

## 7.6.2 Member Function Documentation

## 7.6.2.1 quint8 cwho::verify ( ) [virtual]

Check the validity of the command according to the parameters contained in the frame given to the construcor.

The method check wether one argument is missing or not.

## Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

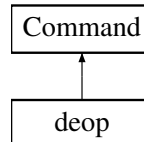
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.7 deop Class Reference

Inheritance diagram for deop:



## Public Member Functions

- [deop](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 [execute](#) ()

## Additional Inherited Members

## 7.7.1 Constructor &amp; Destructor Documentation

## 7.7.1.1 deop::deop ( Client \* sender, Frame &amp; frame )

Constructor.

## Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

## 7.7.2 Member Function Documentation

### 7.7.2.1 `quint8 deop::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the constructor.

The method check whether one argument is missing or not.

#### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.8 Frame Class Reference

This class provides tools for frame analysis.

### Public Member Functions

- [Frame](#) (QByteArray &data)  
*Constructor.*
- quint16 [getSize](#) (void) const
- quint16 [getId](#) (void) const
- quint8 [getCode](#) (void) const
- QStringList [getArgList](#) (void) const
- quint16 [getNbArg](#) (void) const

### Static Public Member Functions

- static QByteArray [getReadyToSendFrame](#) (QString &data, quint16 id, quint8 code)  
*Allows to get a ready to send formatted frame.*

### 7.8.1 Detailed Description

This class provides tools for frame analysis.

[Frame](#) allows to send formatted frames according to a Pseudo-Irc protocol from a simple string plus tools for a complete analysis of the frames received on the Pseudo-Irc protocol compatible server.

## 7.8.2 Constructor & Destructor Documentation

### 7.8.2.1 `Frame::Frame ( QByteArray & frame )`

Constructor.

#### Parameters

<i>frame</i>	: A QByteArray containing the formatted frame received.
--------------	---



### 7.8.3 Member Function Documentation

#### 7.8.3.1 QStringList Frame::getArgList ( void ) const

##### Returns

A QStringList containing the arguments of the command.

#### 7.8.3.2 quint8 Frame::getCode ( void ) const

##### Returns

The code of the command contained in the frame.

#### 7.8.3.3 quint16 Frame::getId ( void ) const

##### Returns

The Id of the command contained in the frame.

#### 7.8.3.4 quint16 Frame::getNbArg ( void ) const

##### Returns

The number of arguments contained in the command.

#### 7.8.3.5 QByteArray Frame::getReadyToSendFrame ( QString & data, quint16 id, quint8 code ) [static]

Allows to get a ready to send formatted frame.

Allows to get a ready to send formatted frame according to the pseudo-irc protocol. depending on the parameters given.

##### Parameters

<i>data</i>	: The arguments of the command separated by a ' ' character.
<i>id</i>	: The id of the command.
<i>code</i>	: The code of the command (see Command.h for an exhaustive list).

##### Returns

The formatted frame according to the parameters given.

#### 7.8.3.6 quint16 Frame::getSize ( void ) const

##### Returns

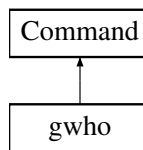
The size of the arguments of the command.

The documentation for this class was generated from the following files:

- [frame.h](#)
- [frame.cpp](#)

## 7.9 gwho Class Reference

Inheritance diagram for gwho:



### Public Member Functions

- [gwho](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the constructor.*
- virtual quint8 **execute** ()

### Additional Inherited Members

#### 7.9.1 Constructor & Destructor Documentation

##### 7.9.1.1 gwho::gwho ( [Client](#) \* sender, [Frame](#) & frame )

Constructor.

##### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

#### 7.9.2 Member Function Documentation

##### 7.9.2.1 quint8 gwho::verify ( ) [[virtual](#)]

Check the validity of the command according to the parameters contained in the frame given to the constructor.

The method begin to check if one argument is missing and then check whether the regex given is valid or not.

##### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::eBadArg](#) if the regex given isn't valid, [ERROR::esuccess](#) if the command is valid.

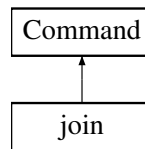
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.10 join Class Reference

Inheritance diagram for join:



## Public Member Functions

- [join](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the constructor.*
- virtual quint8 **execute** ()

## Additional Inherited Members

### 7.10.1 Constructor & Destructor Documentation

#### 7.10.1.1 join::join ( [Client](#) \* sender, [Frame](#) & frame )

Constructor.

##### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.10.2 Member Function Documentation

#### 7.10.2.1 quint8 join::verify ( ) [[virtual](#)]

Check the validity of the command according to the parameters contained in the frame given to the constructor.

The method begin to check if one argument is missing and then check whether the channel name given is valid or not.

##### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::eBadArg](#) if the channel name given isn't valid, [ERROR::esuccess](#) if the command is valid.

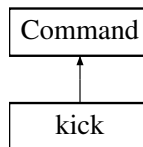
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.11 kick Class Reference

Inheritance diagram for kick:



## Public Member Functions

- [kick](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 [execute](#) ()

## Additional Inherited Members

### 7.11.1 Constructor & Destructor Documentation

#### 7.11.1.1 `kick::kick ( Client * sender, Frame & frame )`

Constructor.

##### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.11.2 Member Function Documentation

#### 7.11.2.1 `quint8 kick::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the construcor.  
 The method check wether one argument is missing or not and then check wether the regex given is valid or not.

##### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

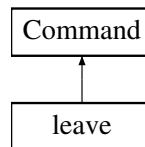
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.12 leave Class Reference

Inheritance diagram for leave:



## Public Member Functions

- [leave](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 **execute** ()

## Additional Inherited Members

### 7.12.1 Constructor & Destructor Documentation

#### 7.12.1.1 `leave::leave ( Client * sender, Frame & frame )`

Constructor.

#### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.12.2 Member Function Documentation

#### 7.12.2.1 `quint8 leave::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the construcor.

The method check wether one argument is missing or not.

#### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

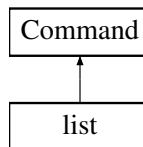
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.13 list Class Reference

Inheritance diagram for list:



## Public Member Functions

- [list](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the constructor.*
- virtual quint8 **execute** ()

## Additional Inherited Members

### 7.13.1 Constructor & Destructor Documentation

#### 7.13.1.1 list::list ( [Client](#) \* sender, [Frame](#) & frame )

Constructor.

#### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.13.2 Member Function Documentation

#### 7.13.2.1 quint8 list::verify ( ) [virtual]

Check the validity of the command according to the parameters contained in the frame given to the constructor.

The method begin to check if one argument is missing and then check whether the regex given is valid or not.

#### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::eBadArg](#) if the regex given isn't valid, [ERROR::esuccess](#) if the command is valid.

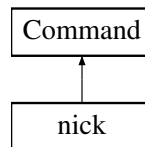
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.14 nick Class Reference

Inheritance diagram for nick:



## Public Member Functions

- `nick (Client *sender, Frame &frame)`  
*Constructor.*
- `virtual quint8 verify ()`  
*Check the validity of the command according to the parameters contained in the frame given to the constructor.*
- `virtual quint8 execute ()`

## Additional Inherited Members

### 7.14.1 Constructor & Destructor Documentation

#### 7.14.1.1 `nick::nick ( Client * sender, Frame & frame )`

Constructor.

##### Parameters

<code>sender</code>	: A pointer on the client who requests the <code>Command</code> object.
<code>frame</code>	: The frame providing all the informations to parameterized the command.

### 7.14.2 Member Function Documentation

#### 7.14.2.1 `quint8 nick::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the constructor.

The method begin to check if one argument is missing and then check whether the nickname given is valid or not.

##### Returns

`ERROR::eMissingArg` if an argument is missing, `ERROR::eBadArg` if the nickname given isn't valid, `ERROR::esuccess` if the command is valid.

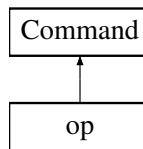
Implements `Command`.

The documentation for this class was generated from the following files:

- `command.h`
- `command.cpp`

## 7.15 op Class Reference

Inheritance diagram for op:



## Public Member Functions

- `op (Client *sender, Frame &frame)`  
*Constructor.*
- `virtual quint8 verify ()`  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- `virtual quint8 execute ()`

## Additional Inherited Members

### 7.15.1 Constructor & Destructor Documentation

#### 7.15.1.1 `op::op ( Client * sender, Frame & frame )`

Constructor.

#### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.15.2 Member Function Documentation

#### 7.15.2.1 `quint8 op::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the construcor.

The method check wether one argument is missing or not.

#### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

Implements [Command](#).

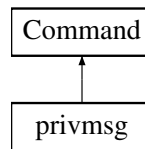
The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.16 privmsg Class Reference

Inheritance diagram for privmsg:





## Public Member Functions

- [privmsg](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 **execute** ()

## Additional Inherited Members

### 7.16.1 Constructor & Destructor Documentation

#### 7.16.1.1 `privmsg::privmsg ( Client * sender, Frame & frame )`

Constructor.

##### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.16.2 Member Function Documentation

#### 7.16.2.1 `quint8 privmsg::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the construcor.

The method check wether one argument is missing or not.

##### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

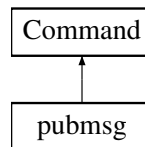
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.17 pubmsg Class Reference

Inheritance diagram for pubmsg:



## Public Member Functions

- [pubmsg](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 [execute](#) ()

## Additional Inherited Members

### 7.17.1 Constructor & Destructor Documentation

#### 7.17.1.1 `pubmsg::pubmsg ( Client * sender, Frame & frame )`

Constructor.

##### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.17.2 Member Function Documentation

#### 7.17.2.1 `quint8 pubmsg::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the construcor.

The method check wether one argument is missing or not.

##### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

Implements [Command](#).

The documentation for this class was generated from the following files:

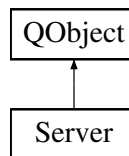
- [command.h](#)
- [command.cpp](#)

## 7.18 Server Class Reference

Class defining the [Server](#). This class inherits from QObjet to use slots and signals. This class uses the pattern Singleton.

```
#include <server.h>
```

Inheritance diagram for Server:



## Public Slots

- void **onNewConnection** (void)  
*Create and add a client to the server.*

## Public Member Functions

- **~Server** ()  
*Destructor.*
- void **delClient** (Client \*c)  
*Remove a client from the server list of clients.*
- Channel \* **getChannelFromName** (QString &name)
- Client \* **getClientFromName** (QString &name)
- quint8 **nick** (Client \*c, QString &nickname)
- quint8 **privmsg** (Client \*c, QString &dest, QString &message)
- quint8 **pubmsg** (Client \*c, QString &dest, QString &message)
- quint8 **join** (Client \*c, QString &dest)
- quint8 **leave** (Client \*c, QString &dest)
- quint8 **list** (Client \*c, QString &filter)
- quint8 **topic** (Client \*c, QString &dest, QString &topic)
- quint8 **gwho** (Client \*c, QString &filter)
- quint8 **cwho** (Client \*c, QString &dest)
- quint8 **kick** (Client \*c, QString &dest\_channel, QString &filter)
- quint8 **ban** (Client \*c, QString &dest\_channel, QString &filter)
- quint8 **unban** (Client \*c, QString &dest\_channel, QString &filter)
- quint8 **banlist** (Client \*c, QString &dest\_channel)
- quint8 **op** (Client \*c, QString &dest\_channel, QString &dest\_client)
- quint8 **deop** (Client \*c, QString &dest\_channel, QString &dest\_client)

## Static Public Member Functions

- static Server \* **Instance** ()

## Protected Member Functions

- **Server** (QObject \*parent=0)  
*Constructor.*
- void **init** ()  
*Initialize the server from the server.conf file.*
- void **broadCast** (QString &message, quint16 id, quint8 code, Channel \*chan=NULL, Client \*sender=NULL)  
*Broadcast a message on several channels/to several clients.*

### 7.18.1 Detailed Description

Class defining the **Server**. This class inherits from QObject to use slots and signals. This class uses the pattern Singleton.

## 7.18.2 Constructor & Destructor Documentation

### 7.18.2.1 `Server::Server ( QObject * parent = 0 )` `[protected]`

Constructor.

The list of the channel of the server

## 7.18.3 Member Function Documentation

### 7.18.3.1 `void Server::broadcast ( QString & message, quint16 id, quint8 code, Channel * chan = NULL, Client * sender = NULL )` `[protected]`

Broadcast a message on several channels/to several clients.

#### Parameters

<i>message</i>	: The message to broadcast.
<i>id</i>	: the id of the message (in our protocol, always 255).
<i>code</i>	: see protocol description document.
<i>chan</i>	: if <i>chan</i> is not specified, message will be sent to all clients connected to the server else, the message will be sent to all the clients connected to the channel.
<i>sender</i>	: if a sender is specified, message will not be sent to him. false sinon

### 7.18.3.2 `Channel * Server::getChannelFromName ( QString & name )`

#### Returns

the channel corresponding to the name given or null if channel doesn't exist.

### 7.18.3.3 `Client * Server::getClientFromName ( QString & name )`

#### Returns

the client corresponding to the name given or null if channel doesn't exist.

### 7.18.3.4 `void Server::init ( void )` `[protected]`

Initialize the server from the server.conf file.

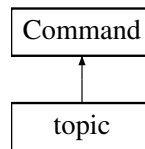
The file server.conf must be placed in the folder of the executable program.

The documentation for this class was generated from the following files:

- [server.h](#)
- [server.cpp](#)

## 7.19 topic Class Reference

Inheritance diagram for topic:



## Public Member Functions

- [topic](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the constructor.*
- virtual quint8 [execute](#) ()

## Additional Inherited Members

### 7.19.1 Constructor & Destructor Documentation

#### 7.19.1.1 `topic::topic ( Client * sender, Frame & frame )`

Constructor.

#### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.19.2 Member Function Documentation

#### 7.19.2.1 `quint8 topic::verify ( ) [virtual]`

Check the validity of the command according to the parameters contained in the frame given to the constructor.

The method check whether one argument is missing or not.

#### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

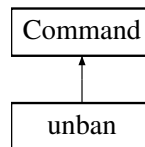
Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## 7.20 unban Class Reference

Inheritance diagram for unban:



## Public Member Functions

- [unban](#) ([Client](#) \*sender, [Frame](#) &frame)  
*Constructor.*
- virtual quint8 [verify](#) ()  
*Check the validity of the command according to the parameters contained in the frame given to the construcor.*
- virtual quint8 [execute](#) ()

## Additional Inherited Members

### 7.20.1 Constructor & Destructor Documentation

#### 7.20.1.1 `unban::unban ( Client * sender, Frame & frame )`

Constructor.

##### Parameters

<i>sender</i>	: A pointer on the client who requests the <a href="#">Command</a> object.
<i>frame</i>	: The frame providing all the informations to parameterized the command.

### 7.20.2 Member Function Documentation

#### 7.20.2.1 `quint8 unban::verify ( )` [virtual]

Check the validity of the command according to the parameters contained in the frame given to the construcor.  
The method check wether one argument is missing or not and then check wether the regex given is valid or not.

##### Returns

[ERROR::eMissingArg](#) if an argument is missing, [ERROR::esuccess](#) if the command is valid.

Implements [Command](#).

The documentation for this class was generated from the following files:

- [command.h](#)
- [command.cpp](#)

## Chapter 8

# File Documentation

### 8.1 channel.h File Reference

This file gathers tools to manage clients inside channels.

```
#include <list>
#include <iostream>
#include "client.h"
```

#### Classes

- class [Channel](#)  
*Class representing a channel.*

#### Enumerations

- enum [status](#) { [BANNED](#), [REGULAR](#), [OPERATOR](#) }  
*Status of a client on the channel.*

#### 8.1.1 Detailed Description

This file gathers tools to manage clients inside channels.

#### 8.1.2 Enumeration Type Documentation

##### 8.1.2.1 enum status

Status of a client on the channel.

#### Enumerator

**BANNED** [Client](#) is banned of the channel

**REGULAR** [Client](#) is connected to the channel

**OPERATOR** [Client](#) is operator on the channel note that it means the client is also connected to the channel

## 8.2 client.h File Reference

This file gathers informations on a client connected to the server using a TCP connection.

```
#include <QTcpSocket>
```

### Classes

- class [Client](#)

*Class representing a [Client](#) connected to the server. This class inherits from QObjet to use slots and signals.*

### 8.2.1 Detailed Description

This file gathers informations on a client connected to the server using a TCP connection.

## 8.3 command.h File Reference

File containing the declaration of the commands.

```
#include "server.h"  
#include "frame.h"
```

### Classes

- class [Command](#)

*Class representing an abstract command.*

- class [nick](#)
- class [privmsg](#)
- class [pubmsg](#)
- class [join](#)
- class [leave](#)
- class [list](#)
- class [topic](#)
- class [gwho](#)
- class [cwho](#)
- class [kick](#)
- class [ban](#)
- class [unban](#)
- class [banlist](#)
- class [op](#)
- class [deop](#)

### Namespaces

- namespace [ERROR](#)
- namespace [CMD](#)



## Enumerations

- enum {  
    [ERROR::esuccess](#) = 0, [ERROR::eBadArg](#) = 250, [ERROR::eNickCollision](#), [ERROR::eNotAuthorised](#),  
    [ERROR::eMissingArg](#), [ERROR::eNotExist](#), [ERROR::error](#) }
- enum {  
    [CMD::C\\_PRIVMSG](#) = 1, [CMD::C\\_PUBMSG](#), [CMD::C\\_GWHO](#), [CMD::C\\_CWHO](#),  
    [CMD::C\\_LIST](#), [CMD::C\\_TOPIC](#), [CMD::C\\_KICK](#), [CMD::C\\_BAN](#),  
    [CMD::C\\_OP](#), [CMD::C\\_DEOP](#) = 20, [CMD::C\\_JOIN](#), [CMD::C\\_NICK](#),  
    [CMD::C\\_LEAVE](#), [CMD::C\\_UNBAN](#), [CMD::C\\_BANLIST](#) }

### 8.3.1 Detailed Description

File containing the declaration of the commands.

## 8.4 frame.h File Reference

[Frame](#) analyser based on a Pseudo-Irc protocol.

```
#include <QStringList>
```

## Classes

- class [Frame](#)  
*This class provides tools for frame analysis.*

### 8.4.1 Detailed Description

[Frame](#) analyser based on a Pseudo-Irc protocol.

## 8.5 server.h File Reference

IRC [Server](#) Near-IRC server : A simplified IRC server not in accordance with RFC 1459.

```
#include <QTcpServer>  
#include <list>  
#include "channel.h"
```

## Classes

- class [Server](#)  
*Class defining the [Server](#). This class inherits from QObjet to use slots and signals. This class uses the pattern Singleton.*

### 8.5.1 Detailed Description

IRC [Server](#) Near-IRC server : A simplified IRC server not in accordance with RFC 1459.

# Index

addClient  
    Channel, 16

BANNED  
    channel.h, 39

ban, 13  
    ban, 13  
    verify, 13

banlist, 14  
    banlist, 14  
    verify, 14

broadCast  
    Server, 36

C\_BAN  
    CMD, 11

C\_BANLIST  
    CMD, 11

C\_CWHO  
    CMD, 11

C\_DEOP  
    CMD, 11

C\_GWHO  
    CMD, 11

C\_JOIN  
    CMD, 11

C\_KICK  
    CMD, 11

C\_LEAVE  
    CMD, 11

C\_LIST  
    CMD, 11

C\_NICK  
    CMD, 11

C\_OP  
    CMD, 11

C\_PRIVMSG  
    CMD, 11

C\_PUBMSG  
    CMD, 11

C\_TOPIC  
    CMD, 11

C\_UNBAN  
    CMD, 11

CMD  
    C\_BAN, 11  
    C\_BANLIST, 11  
    C\_CWHO, 11  
    C\_DEOP, 11  
    C\_GWHO, 11

C\_JOIN, 11  
C\_KICK, 11  
C\_LEAVE, 11  
C\_LIST, 11  
C\_NICK, 11  
C\_OP, 11  
C\_PRIVMSG, 11  
C\_PUBMSG, 11  
C\_TOPIC, 11  
C\_UNBAN, 11

CMD, 11

Channel, 15  
    addClient, 16  
    Channel, 16  
    getChannelName, 16  
    getClientList, 16  
    getTopic, 16  
    isEmpty, 16  
    isStatus, 17  
    removeClient, 17  
    setOperator, 17  
    setTopic, 17  
    unbanClient, 18  
    unsetOperator, 18

channel.h  
    BANNED, 39  
    OPERATOR, 39  
    REGULAR, 39

channel.h, 39  
    status, 39

Client, 18  
    Client, 19  
    getMsg, 19  
    getNickname, 19  
    getSocket, 19  
    getState, 20  
    onDataReady, 20  
    setMsg, 20  
    setNickname, 20  
    setSocket, 20  
    setState, 20

client.h, 40

Command, 21  
    getCommand, 22

command.h, 40

cwho, 22  
    cwho, 22  
    verify, 23

deop, 23

- deop, [23](#)
- verify, [24](#)
- eBadArg
  - ERROR, [12](#)
- eMissingArg
  - ERROR, [12](#)
- eNickCollision
  - ERROR, [12](#)
- eNotAuthorised
  - ERROR, [12](#)
- eNotExist
  - ERROR, [12](#)
- ERROR
  - eBadArg, [12](#)
  - eMissingArg, [12](#)
  - eNickCollision, [12](#)
  - eNotAuthorised, [12](#)
  - eNotExist, [12](#)
  - error, [12](#)
  - esuccess, [12](#)
- ERROR, [12](#)
- error
  - ERROR, [12](#)
- esuccess
  - ERROR, [12](#)
- Frame, [24](#)
  - Frame, [24](#)
  - getArgList, [25](#)
  - getCode, [25](#)
  - getId, [25](#)
  - getNbArg, [25](#)
  - getReadyToSendFrame, [25](#)
  - getSize, [25](#)
- frame.h, [41](#)
- getArgList
  - Frame, [25](#)
- getChannelFromName
  - Server, [36](#)
- getChannelName
  - Channel, [16](#)
- getClientFromName
  - Server, [36](#)
- getClientList
  - Channel, [16](#)
- getCode
  - Frame, [25](#)
- getCommand
  - Command, [22](#)
- getId
  - Frame, [25](#)
- getMsg
  - Client, [19](#)
- getNbArg
  - Frame, [25](#)
- getNickname
  - Client, [19](#)
- getReadyToSendFrame
  - Frame, [25](#)
- getSize
  - Frame, [25](#)
- getSocket
  - Client, [19](#)
- getState
  - Client, [20](#)
- getTopic
  - Channel, [16](#)
- gwho, [26](#)
  - gwho, [26](#)
  - verify, [26](#)
- init
  - Server, [36](#)
- isEmpty
  - Channel, [16](#)
- isStatus
  - Channel, [17](#)
- join, [26](#)
  - join, [27](#)
  - verify, [27](#)
- kick, [27](#)
  - kick, [28](#)
  - verify, [28](#)
- leave, [28](#)
  - leave, [29](#)
  - verify, [29](#)
- list, [29](#)
  - list, [30](#)
  - verify, [30](#)
- nick, [30](#)
  - nick, [31](#)
  - verify, [31](#)
- OPERATOR
  - channel.h, [39](#)
- onDataReady
  - Client, [20](#)
- op, [31](#)
  - op, [32](#)
  - verify, [32](#)
- privmsg, [32](#)
  - privmsg, [33](#)
  - verify, [33](#)
- pubmsg, [33](#)
  - pubmsg, [34](#)
  - verify, [34](#)
- REGULAR
  - channel.h, [39](#)
- removeClient
  - Channel, [17](#)

- Server, [34](#)
  - broadCast, [36](#)
  - getChannelFromName, [36](#)
  - getClientFromName, [36](#)
  - init, [36](#)
  - Server, [36](#)
- server.h, [41](#)
- setMsg
  - Client, [20](#)
- setNickname
  - Client, [20](#)
- setOperator
  - Channel, [17](#)
- setSocket
  - Client, [20](#)
- setState
  - Client, [20](#)
- setTopic
  - Channel, [17](#)
- status
  - channel.h, [39](#)
- topic, [36](#)
  - topic, [37](#)
  - verify, [37](#)
- unban, [37](#)
  - unban, [38](#)
  - verify, [38](#)
- unbanClient
  - Channel, [18](#)
- unsetOperator
  - Channel, [18](#)
- verify
  - ban, [13](#)
  - banlist, [14](#)
  - cwho, [23](#)
  - deop, [24](#)
  - gwho, [26](#)
  - join, [27](#)
  - kick, [28](#)
  - leave, [29](#)
  - list, [30](#)
  - nick, [31](#)
  - op, [32](#)
  - privmsg, [33](#)
  - pubmsg, [34](#)
  - topic, [37](#)
  - unban, [38](#)