# My Project

Generated by Doxygen 1.8.4

Mon Jun 17 2013 14:44:45

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

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

# Chapter 2

# Class Index

## 2.1 Class List

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

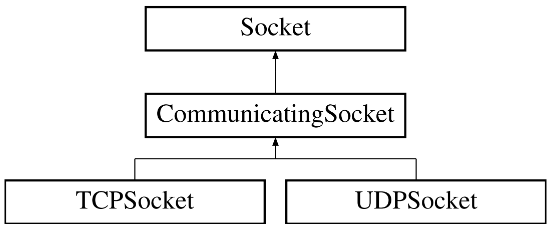
# Chapter 3

# Class Documentation

## 3.1 CommunicatingSocket Class Reference

```
#include <PracticalSocket.h>
```

Inheritance diagram for CommunicatingSocket:



### Public Member Functions

- void connect (const string &foreignAddress, unsigned short foreignPort) throw (SocketException)
- void send (const void ∗buffer, int bufferLen) throw (SocketException)
- int recv (void ∗buffer, int bufferLen) throw (SocketException)
- string getForeignAddress () throw (SocketException)
- unsigned short getForeignPort () throw (SocketException)

### Protected Member Functions

- **CommunicatingSocket** (int type, int protocol) throw (SocketException)
- **CommunicatingSocket** (int newConnSD)

### Additional Inherited Members

### 3.1.1 Detailed Description

Socket which is able to connect, send, and receive

### 3.1.2 Member Function Documentation

#### 3.1.2.1 void CommunicatingSocket::connect ( const string & *foreignAddress,* unsigned short *foreignPort* ) throw SocketException)

Establish a socket connection with the given foreign address and port

**Parameters**

| | |
|---|---|
| *foreignAddress* | foreign address (IP address or name) |
| *foreignPort* | foreign port |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to establish connection |

#### 3.1.2.2 string CommunicatingSocket::getForeignAddress ( ) throw **SocketException**)

Get the foreign address. Call connect() before calling recv()

**Returns**

foreign address

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to fetch foreign address |

#### 3.1.2.3 unsigned short CommunicatingSocket::getForeignPort ( ) throw **SocketException**)

Get the foreign port. Call connect() before calling recv()

**Returns**

foreign port

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to fetch foreign port |

#### 3.1.2.4 int CommunicatingSocket::recv ( void ∗ *buffer,* int *bufferLen* ) throw **SocketException**)

Read into the given buffer up to bufferLen bytes data from this socket. Call connect() before calling recv()

**Parameters**

| | |
|---|---|
| *buffer* | buffer to receive the data |
| *bufferLen* | maximum number of bytes to read into buffer |

**Returns**

number of bytes read, 0 for EOF, and -1 for error

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to receive data |

#### 3.1.2.5 void CommunicatingSocket::send ( const void ∗ *buffer,* int *bufferLen* ) throw **SocketException**)

Write the given buffer to this socket. Call connect() before calling send()

**Parameters**

| | |
|---|---|
| *buffer* | buffer to be written |
| *bufferLen* | number of bytes from buffer to be written |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to send data |

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

- PracticalSocket.h
- PracticalSocket.cpp

## 3.2 Data Class Reference

```
#include <Data.h>
```

### Public Member Functions

- Data (double value, unsigned int datatype, unsigned int position)
- double getValue ()
- unsigned int getDatatype ()
- unsigned int getPosition ()

### 3.2.1 Detailed Description

Datenstruktur die einen Fahrzeugwert und die dazugehörigen Daten speichert.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 Data::Data ( double *value,* unsigned int *datatype,* unsigned int *position* )

Erzeugt eine Datenstruktur zur Speicherung von Fahrzeugdaten.

**Parameters**

| | |
|---|---|
| *value* | Wert des Datensatzes. |
| *datatype* | Datentyp des Datensatzes. |
| *position* | Position des Datensatzes in den ursprünglichen Daten. |

### 3.2.3 Member Function Documentation

#### 3.2.3.1 unsigned int Data::getDatatype ( )

**Returns**

Gibt den Datentyp des Datensatzes zurück.

#### 3.2.3.2 unsigned int Data::getPosition ( )

**Returns**

Gibt die Position des Datensatzes in den ursprünlichen Daten zurück.

#### 3.2.3.3 double Data::getValue ( )

**Returns**

Gibt den Wert des Datensatzes zurück.

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

- Data.h
- Data.cpp

## 3.3 Decoder Class Reference

### Public Member Functions

- Decoder (char ∗buffer, const int bufferlen)
- Decoder (char ∗buffer, const int bufferlen, char ∗vecLayout, const int vecLayoutlen, char ∗vecDatatypes, const int vecDatatypeslen, char ∗vecComma, const int vecCommalen)
- Data getNextData ()
- unsigned int getPackageNum ()
- unsigned int getPackagePos (char ∗vecLayout, const int vecLayoutlen)

### 3.3.1 Constructor & Destructor Documentation

#### 3.3.1.1 Decoder::Decoder ( char ∗ *buffer*, const int *bufferlen* )

Erzeugt einen Dekoder der zum dekodieren der Paketinformation dient.

**Parameters**

| | |
|---:|---|
| *buffer* | Speicher der die Paketinformationen enthält. [Layout,Datentypen,Kommasetzung] |
| *bufferlen* | Länge von *buffer*. |

#### 3.3.1.2 Decoder::Decoder ( char ∗ *buffer*, const int *bufferlen*, char ∗ *vecLayout*, const int *vecLayoutlen*, char ∗ *vecDatatypes*, const int *vecDatatypeslen*, char ∗ *vecComma*, const int *vecCommalen* )

Erzeugt einen Dekoder der ein Datenpaket anhand der übergebenen Informationen dekodiert.

**Parameters**

| | |
|---:|---|
| *buffer* | Speicher des Datenpakets. |
| *buffernlen* | Länge von *buffer*. |
| *vecLayout* | Aufteilung des ursprünglichen Datenstroms die aus den Paketinformationen dekodiert wurden. Dient zur Ermittlung der konkreten Datensätze. |
| *vecLayoutlen* | Länge von *vecLayout*. |
| *vecDatatypes* | Beinhaltet die Informationen zu den Datentypen der jeweiligen Datensätze. |
| *vecDatatypeslen* | Länge von *vecDatatypes*. |
| *vecComma* | Beinhaltet die Kommasetzung sämtlicher Datensätze. |
| *vecCommalen* | Länge von *vecComma*. |

### 3.3.2 Member Function Documentation

#### 3.3.2.1 Data Decoder::getNextData ( )

Holt den nächsten Datensatz aus den empfangenen Daten.

**Returns**

Gibt ein Datenobjekt Data zurück das sämtlich Informationen über den Datensatz enthält.

#### 3.3.2.2 unsigned int Decoder::getPackageNum ( )

Holt die Paketnummer des akutell bearbeiteten Pakets.

**Returns**

Paketnummer das aktuellen Pakets

#### 3.3.2.3 unsigned int Decoder::getPackagePos ( char ∗ *vecLayout*, const int *vecLayoutlen* )

Holt die Position des aktuellen Pakets im ursprünglichen Datensatz.

**Parameters**

| | |
|---|---|
| *vecLayout* | Aufteilung des ursprünglichen Datenstroms die aus den Paketinformationen dekodiert wurden. |
| *vecLayoutlen* | Länge von *vecLayout*. |

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

- Encoding.h
- Encoding.cpp

## 3.4 Encoder Class Reference

```
#include <Encoding.h>
```

### Public Member Functions

- Encoder (const char ∗buffer, const int bufferlen, const char ∗vecLayout, const int vecLayoutlen, const char ∗vecDatatypes, const int vecDatatypeslen)
- int getPackage (char ∗package, size_t len, unsigned short packageNumber)
- int getNextPackage (char ∗package, size_t len)
- int getPackageSize (unsigned short packageNumber)
- unsigned int getPackageSum ()

### 3.4.1 Detailed Description

Service der aus einem kompletten Satz Fahrzeugdaten mehrere Pakete erzeugt und komprimiert. Die Komprimierung ist noch nicht implementiert.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 Encoder::Encoder ( const char ∗ *buffer*, const int *bufferlen*, const char ∗ *vecLayout*, const int *vecLayoutlen*, const char ∗ *vecDatatypes*, const int *vecDatatypeslen* )

Erzeugt einen Encoder.

**Parameters**

| | |
|---:|---|
| *buffer* | Die zu bearbeitenden Daten. Dabei muss es sich um einen Datenstrom handeln in dem jeweils 2 Byte einen Fahrzeugwert entsprechen. |
| *bufferlen* | Die Länge der zu bearbeitenden Daten. |
| *vecLayout* | Gibt an wie die Daten geteilt werden sollen.<br>[Anfangsbyte Paket 1, Anfangsbyte Paket 2, ..., Anfangsbyte Paket n] |
| *vecLayoutlen* | Die Länge von *vecLayout*. |
| *vecDatatypes* | Gibt an um welchen Datentyp es sich jeweils handelt. |
| *vecDatatypeslen* | Die Länge von *vecDatatypes*. |

### 3.4.3 Member Function Documentation

#### 3.4.3.1 int Encoder::getNextPackage ( char ∗ *package*, size_t *len* )

Holt das jeweils nächste Paket. (1,2,...,n,1,2,...)

**Parameters**

| | |
|---:|---|
| *package* | Speicher in den das Paket geschrieben werden soll. |
| *len* | Länge von *package*. |

**Returns**

Die Länge des Pakets oder -1 falls *len* zu klein.

#### 3.4.3.2 int Encoder::getPackage ( char ∗ *package*, size_t *len*, unsigned short *packageNumber* )

Holt ein Paket mit einer speziellen Paketnummer.

**Parameters**

| | |
|---:|---|
| *package* | Speicher in den das Paket geschrieben werden soll. |
| *len* | Länge von *package*. |
| *packageNumber* | Paketnummer des gewünschten Pakets. |

**Returns**

Die Länge des Pakets oder -1 falls Paket mit *packageNumber* nicht vorhanden oder *len* zu klein.

#### 3.4.3.3 unsigned int Encoder::getPackageSize ( unsigned short *packageNumber* )

Gibt die Paketgröße eines speziellen Pakets zurück.

**Parameters**

| | |
|---:|---|
| *packageNumber* | Paketnummer dessen Größe gesucht ist. |

**Returns**

Größe des Pakets oder -1 falls Paket mit *packageNumber* nicht vorhanden.

#### 3.4.3.4 unsigned int Encoder::getPackageSum ( )

Gibt die Anzahl der Pakete zurück.

**Returns**

Anzahl der Pakete.

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

- Encoding.h
- Encoding.cpp

## 3.5 Location Class Reference

```
#include <Location.h>
```

### Public Member Functions

- Location (std::string address, short port)
- std::string getAddress ()
- int getPort ()

### 3.5.1 Detailed Description

Datenstruktur die Netzwerkdaten bestimmter Teilnehmer speichert.

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 Location::Location ( std::string *address*, short *port* )

Erzeugt einen Teilnehmer.

**Parameters**

| | |
|---|---|
| *address* | IP-Adresse des Teilnehmers. |
| *port* | Port-Nummer des Teilnehmers. |

### 3.5.3 Member Function Documentation

#### 3.5.3.1 std::string Location::getAddress ( )

**Returns**

Gibt die Adresse zurück.

#### 3.5.3.2 int Location::getPort ( )

**Returns**

Gibt die Portnummer zurück.

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

- Location.h
- Location.cpp

## 3.6 Receiver Class Reference

### Public Member Functions

- **Receiver** (Source)
- void **setSource** (Source)
- Header **recvHeader** ()
- Data **recvData** ()

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

- Communication.h

## 3.7 Sender Class Reference

### Public Member Functions

- **Sender** (Header, Data, Destination)
- void **addHeader** (Header)
- void **removeHeader** ()
- void **addData** (Data)
- void **removeData** ()
- void **setDestination** (Destination)
- void **resetDestination** ()
- bool **sendPackage** ()

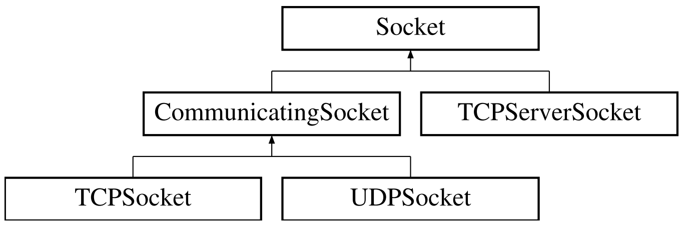
The documentation for this class was generated from the following files:

- Communication.h
- Communication.cpp

## 3.8 Socket Class Reference

```
#include <PracticalSocket.h>
```

Inheritance diagram for Socket:



### Public Member Functions

- **Socket** (unsigned short localPort)
- **Socket** (unsigned short remoteAddr, unsigned short remotePort)
- void **setRemoteAddr** (unsigned short remoteAddr)
- void **setRemotePort** (unsigned short remotePort)

- void **setLocalPort** (unsigned short localPort)
- unsigned short **getSocketDescriptor** ()
- unsigned int ∗ **getRemoteAddr** ()
- ∼Socket ()
- string getLocalAddress () throw (SocketException)
- unsigned short getLocalPort () throw (SocketException)
- void setLocalPort (unsigned short localPort) throw (SocketException)
- void setLocalAddressAndPort (const string &localAddress, unsigned short localPort=0) throw (Socket-Exception)

### Static Public Member Functions

- static void cleanUp () throw (SocketException)
- static unsigned short resolveService (const string &service, const string &protocol="tcp")

### Protected Member Functions

- **Socket** (int type, int protocol) throw (SocketException)
- **Socket** (int sockDesc)

### Protected Attributes

- int **sockDesc**

## 3.8.1 Detailed Description

Base class representing basic communication endpoint

## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 Socket::∼Socket ( )

Close and deallocate this socket

## 3.8.3 Member Function Documentation

### 3.8.3.1 void Socket::cleanUp ( ) throw SocketException) [static]

If WinSock, unload the WinSock DLLs; otherwise do nothing. We ignore this in our sample client code but include it in the library for completeness. If you are running on Windows and you are concerned about DLL resource consumption, call this after you are done with all Socket instances. If you execute this on Windows while some instance of Socket exists, you are toast. For portability of client code, this is an empty function on non-Windows platforms so you can always include it.

**Parameters**

| | |
|---|---|
| *buffer* | buffer to receive the data |
| *bufferLen* | maximum number of bytes to read into buffer |

**Returns**

number of bytes read, 0 for EOF, and -1 for error

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown WinSock clean up fails |

#### 3.8.3.2 string Socket::getLocalAddress ( ) throw SocketException)

Get the local address

**Returns**

local address of socket

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if fetch fails |

#### 3.8.3.3 unsigned short Socket::getLocalPort ( ) throw SocketException)

Get the local port

**Returns**

local port of socket

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if fetch fails |

#### 3.8.3.4 unsigned short Socket::resolveService ( const string & *service*, const string & *protocol* = `"tcp"` ) `[static]`

Resolve the specified service for the specified protocol to the corresponding port number in host byte order

**Parameters**

| | |
|---|---|
| *service* | service to resolve (e.g., "http") |
| *protocol* | protocol of service to resolve. Default is "tcp". |

#### 3.8.3.5 void Socket::setLocalAddressAndPort ( const string & *localAddress*, unsigned short *localPort* = `0` ) throw SocketException)

Set the local port to the specified port and the local address to the specified address. If you omit the port, a random port will be selected.

**Parameters**

| | |
|---|---|
| *localAddress* | local address |
| *localPort* | local port |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if setting local port or address fails |

#### 3.8.3.6 void Socket::setLocalPort ( unsigned short *localPort* ) throw SocketException)

Set the local port to the specified port and the local address to any interface

**Parameters**

| | |
|---|---|
| *localPort* | local port |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if setting local port fails |

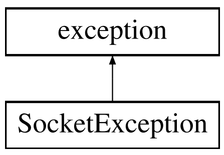
The documentation for this class was generated from the following files:

- Communication.h
- PracticalSocket.h
- Communication.cpp
- PracticalSocket.cpp

## 3.9 SocketException Class Reference

```
#include <PracticalSocket.h>
```

Inheritance diagram for SocketException:



### Public Member Functions

- SocketException (const string &message, bool inclSysMsg=false) throw ()
- ∼SocketException () throw ()
- const char ∗ what () const throw ()

### 3.9.1 Detailed Description

Signals a problem with the execution of a socket call.

### 3.9.2 Constructor & Destructor Documentation

#### 3.9.2.1 SocketException::SocketException ( const string & *message*, bool *inclSysMsg* = `false` ) throw )

Construct a SocketException with a explanatory message.

**Parameters**

| | |
|---|---|
| *message* | explanatory message |
| *incSysMsg* | true if system message (from strerror(errno)) should be postfixed to the user provided message |

#### 3.9.2.2 SocketException::∼SocketException ( ) throw )

Provided just to guarantee that no exceptions are thrown.

### 3.9.3 Member Function Documentation

#### 3.9.3.1 const char ∗ SocketException::what ( ) const throw )

Get the exception message

**Returns**

exception message

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

- PracticalSocket.h
- PracticalSocket.cpp

## 3.10 T_nuex Struct Reference

### Public Attributes

- short int **testen** [401]

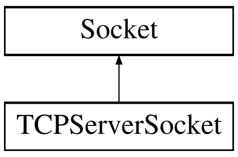
The documentation for this struct was generated from the following file:

- mab.cpp

## 3.11 TCPServerSocket Class Reference

```
#include <PracticalSocket.h>
```

Inheritance diagram for TCPServerSocket:



### Public Member Functions

- TCPServerSocket (unsigned short localPort, int queueLen=5) throw (SocketException)
- TCPServerSocket (const string &localAddress, unsigned short localPort, int queueLen=5) throw (Socket-Exception)
- TCPSocket ∗ accept () throw (SocketException)

### Additional Inherited Members

### 3.11.1 Detailed Description

TCP socket class for servers

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 TCPServerSocket::TCPServerSocket ( unsigned short *localPort*, int *queueLen =* `5` ) throw **SocketException**)

Construct a TCP socket for use with a server, accepting connections on the specified port on any interface

**Parameters**

| | |
|---|---|
| *localPort* | local port of server socket, a value of zero will give a system-assigned unused port |
| *queueLen* | maximum queue length for outstanding connection requests (default 5) |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to create TCP server socket |

#### 3.11.2.2 TCPServerSocket::TCPServerSocket ( const string & *localAddress*, unsigned short *localPort*, int *queueLen* = 5 ) throw SocketException)

Construct a TCP socket for use with a server, accepting connections on the specified port on the interface specified by the given address

**Parameters**

| | |
|---|---|
| *localAddress* | local interface (address) of server socket |
| *localPort* | local port of server socket |
| *queueLen* | maximum queue length for outstanding connection requests (default 5) |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to create TCP server socket |

### 3.11.3 Member Function Documentation

#### 3.11.3.1 TCPSocket ∗ TCPServerSocket::accept ( ) throw SocketException)

Blocks until a new connection is established on this socket or error

**Returns**

new connection socket

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if attempt to accept a new connection fails |

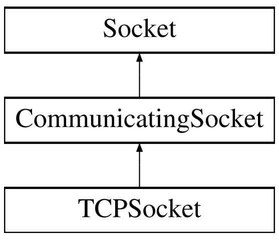
The documentation for this class was generated from the following files:

- PracticalSocket.h
- PracticalSocket.cpp

## 3.12 TCPSocket Class Reference

```
#include <PracticalSocket.h>
```

Inheritance diagram for TCPSocket:



#### Public Member Functions

- TCPSocket () throw (SocketException)
- TCPSocket (const string &foreignAddress, unsigned short foreignPort) throw (SocketException)

#### Friends

- class **TCPServerSocket**

#### Additional Inherited Members

### 3.12.1 Detailed Description

TCP socket for communication with other TCP sockets

### 3.12.2 Constructor & Destructor Documentation

#### 3.12.2.1 TCPSocket::TCPSocket ( ) throw SocketException)

Construct a TCP socket with no connection

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to create TCP socket |

#### 3.12.2.2 TCPSocket::TCPSocket ( const string & *foreignAddress*, unsigned short *foreignPort* ) throw SocketException)

Construct a TCP socket with a connection to the given foreign address and port

**Parameters**

| | |
|---|---|
| *foreignAddress* | foreign address (IP address or name) |
| *foreignPort* | foreign port |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to create TCP socket |

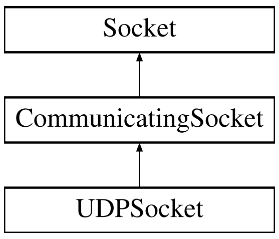
The documentation for this class was generated from the following files:

- PracticalSocket.h
- PracticalSocket.cpp

## 3.13 UDPSocket Class Reference

```
#include <PracticalSocket.h>
```

Inheritance diagram for UDPSocket:



## Public Member Functions

- UDPSocket () throw (SocketException)
- UDPSocket (unsigned short localPort) throw (SocketException)
- UDPSocket (const string &localAddress, unsigned short localPort) throw (SocketException)
- void disconnect () throw (SocketException)
- void sendTo (const void ∗buffer, int bufferLen, const string &foreignAddress, unsigned short foreignPort) throw (SocketException)
- int recvFrom (void ∗buffer, int bufferLen, string &sourceAddress, unsigned short &sourcePort) throw (Socket-Exception)
- void setMulticastTTL (unsigned char multicastTTL) throw (SocketException)
- void joinGroup (const string &multicastGroup) throw (SocketException)
- void leaveGroup (const string &multicastGroup) throw (SocketException)

## Additional Inherited Members

## 3.13.1 Detailed Description

UDP socket class

## 3.13.2 Constructor & Destructor Documentation

### 3.13.2.1 UDPSocket::UDPSocket ( ) throw SocketException)

Construct a UDP socket

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to create UDP socket |

### 3.13.2.2 UDPSocket::UDPSocket ( unsigned short *localPort* ) throw SocketException)

Construct a UDP socket with the given local port

**Parameters**

| | |
|---|---|
| *localPort* | local port |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to create UDP socket |

### 3.13.2.3 UDPSocket::UDPSocket ( const string & *localAddress*, unsigned short *localPort* ) throw SocketException)

Construct a UDP socket with the given local port and address

**Parameters**

| | |
|---|---|
| *localAddress* | local address |
| *localPort* | local port |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to create UDP socket |

### 3.13.3 Member Function Documentation

#### 3.13.3.1 void UDPSocket::disconnect ( ) throw SocketException)

Unset foreign address and port

**Returns**

true if disassociation is successful

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to disconnect UDP socket |

#### 3.13.3.2 void UDPSocket::joinGroup ( const string & *multicastGroup* ) throw SocketException)

Join the specified multicast group

**Parameters**

| | |
|---|---|
| *multicastGroup* | multicast group address to join |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to join group |

#### 3.13.3.3 void UDPSocket::leaveGroup ( const string & *multicastGroup* ) throw SocketException)

Leave the specified multicast group

**Parameters**

| | |
|---|---|
| *multicastGroup* | multicast group address to leave |

**Exceptions**

| | |
|---|---|
| *SocketException* | thrown if unable to leave group |

#### 3.13.3.4 int UDPSocket::recvFrom ( void $*$ *buffer,* int *bufferLen,* string & *sourceAddress,* unsigned short & *sourcePort* ) throw SocketException)

Read read up to bufferLen bytes data from this socket. The given buffer is where the data will be placed

**Parameters**

| | |
|---:|---|
| *buffer* | buffer to receive data |
| *bufferLen* | maximum number of bytes to receive |
| *sourceAddress* | address of datagram source |
| *sourcePort* | port of data source |

**Returns**

number of bytes received and -1 for error

**Exceptions**

| | |
|---:|---|
| *SocketException* | thrown if unable to receive datagram |

#### 3.13.3.5 void UDPSocket::sendTo ( const void ∗ *buffer,* int *bufferLen,* const string & *foreignAddress,* unsigned short *foreignPort* ) throw **SocketException**)

Send the given buffer as a UDP datagram to the specified address/port

**Parameters**

| | |
|---:|---|
| *buffer* | buffer to be written |
| *bufferLen* | number of bytes to write |
| *foreignAddress* | address (IP address or name) to send to |
| *foreignPort* | port number to send to |

**Returns**

true if send is successful

**Exceptions**

| | |
|---:|---|
| *SocketException* | thrown if unable to send datagram |

#### 3.13.3.6 void UDPSocket::setMulticastTTL ( unsigned char *multicastTTL* ) throw **SocketException**)

Set the multicast TTL

**Parameters**

| | |
|---:|---|
| *multicastTTL* | multicast TTL |

**Exceptions**

| | |
|---:|---|
| *SocketException* | thrown if unable to set TTL |

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

- PracticalSocket.h
- PracticalSocket.cpp

# Index