Service-Interface für ein Formula Student Fahrzeug

Erzeugt von Doxygen 1.8.4

Mon Jun 17 2013 23:58:43

Inhaltsverzeichnis

1	Hier	archie-\	/erzeichn	is	1
	1.1	Klasse	nhierarchi	ie	1
2	Klas	sen-Ve	rzeichnis		3
	2.1	Auflistu	ung der Kl	assen	3
3	Klas	sen-Do	kumentat	tion	5
	3.1	Comm	unicatingS	Socket Klassenreferenz	5
		3.1.1	Ausführli	iche Beschreibung	5
		3.1.2	Dokume	entation der Elementfunktionen	5
			3.1.2.1	connect	6
			3.1.2.2	getForeignAddress	7
			3.1.2.3	getForeignPort	7
			3.1.2.4	recv	7
			3.1.2.5	send	7
	3.2	Data K	lassenrefe	erenz	8
		3.2.1	Ausführli	iche Beschreibung	8
		3.2.2	Beschrei	ibung der Konstruktoren und Destruktoren	8
			3.2.2.1	Data	8
		3.2.3	Dokume	entation der Elementfunktionen	8
			3.2.3.1	getDatatype	8
			3.2.3.2	getPosition	8
			3.2.3.3	getValue	9
	3.3	Decod	er Klasser	nreferenz	9
		3.3.1	Beschrei	ibung der Konstruktoren und Destruktoren	9
			3.3.1.1	Decoder	9
			3.3.1.2	Decoder	9
		3.3.2	Dokume	entation der Elementfunktionen	9
			3.3.2.1	getNextData	g
			3.3.2.2	getPackageNum	10
			3.3.2.3	getPackagePos	10
	3.4	Encode	er Klasser		10

iv INHALTSVERZEICHNIS

	3.4.1	Ausführliche Beschreibung	10
	3.4.2	Beschreibung der Konstruktoren und Destruktoren	10
		3.4.2.1 Encoder	10
	3.4.3	Dokumentation der Elementfunktionen	11
		3.4.3.1 getNextPackage	11
		3.4.3.2 getPackage	11
		3.4.3.3 getPackageSize	11
		3.4.3.4 getPackageSum	11
3.5	Locatio	Nassenreferenz	12
	3.5.1	Ausführliche Beschreibung	12
	3.5.2	Beschreibung der Konstruktoren und Destruktoren	12
		3.5.2.1 Location	12
	3.5.3	Dokumentation der Elementfunktionen	12
		3.5.3.1 getAddress	12
		3.5.3.2 getPort	12
3.6	Receive	r Klassenreferenz	13
3.7	Sender	Klassenreferenz	13
3.8	Socket	Klassenreferenz	13
	3.8.1	Ausführliche Beschreibung	14
	3.8.2	Beschreibung der Konstruktoren und Destruktoren	14
		3.8.2.1 ~Socket	14
	3.8.3	Dokumentation der Elementfunktionen	14
		3.8.3.1 cleanUp	14
		3.8.3.2 getLocalAddress	15
		3.8.3.3 getLocalPort	15
		3.8.3.4 resolveService	15
		3.8.3.5 setLocalAddressAndPort	15
		3.8.3.6 setLocalPort	15
3.9	Socket	xception Klassenreferenz	16
	3.9.1	Ausführliche Beschreibung	16
	3.9.2	Beschreibung der Konstruktoren und Destruktoren	16
		3.9.2.1 SocketException	16
		3.9.2.2 ~SocketException	16
	3.9.3	Dokumentation der Elementfunktionen	17
		3.9.3.1 what	17
3.10	T_nuex	Strukturreferenz	17
3.11	TCPSe	verSocket Klassenreferenz	17
	3.11.1	Ausführliche Beschreibung	17
	3.11.2	Beschreibung der Konstruktoren und Destruktoren	18
		3.11.2.1 TCPServerSocket	18

INHALTSVERZEICHNIS

		3.11.2.2	TCPServerSocket	19
	3.11.3	Dokumen	tation der Elementfunktionen	19
		3.11.3.1	accept	19
3.12	TCPSo	cket Klass	enreferenz	19
	3.12.1	Ausführlic	che Beschreibung	20
	3.12.2	Beschreib	bung der Konstruktoren und Destruktoren	20
			TCPSocket	20
			TCPSocket	20
2 12	LIDPS		enreferenz	20
0.10			che Beschreibung	21
			•	
	3.13.2		bung der Konstruktoren und Destruktoren	21
			UDPSocket	21
			UDPSocket	21
		3.13.2.3	UDPSocket	21
	3.13.3	Dokumen	tation der Elementfunktionen	22
		3.13.3.1	disconnect	22
		3.13.3.2	joinGroup	22
		3.13.3.3	leaveGroup	22
		3.13.3.4	recvFrom	22
		3.13.3.5	sendTo	23
		3.13.3.6	setMulticastTTL	23
In dec				0.6
Index				24

Kapitel 1

Hierarchie-Verzeichnis

1.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Data	8
Decoder	
Encoder	10
exception	
SocketException	
Location	
Receiver	
Sender	13
Socket	13
CommunicatingSocket	5
TCPSocket	19
UDPSocket	20
TCPServerSocket	17
T nuex	17

2 Hierarchie-Verzeichnis

Kapitel 2

Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

CommunicatingSocket	
Data	
Decoder	
Encoder	
Location	
Receiver	
Sender	
Socket	
SocketException	
T_nuex	
TCPServerSocket	
TCPSocket	
LIDPSocket	20

Klassen-Verzeichnis

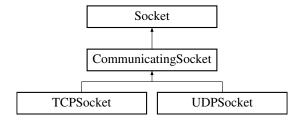
Kapitel 3

Klassen-Dokumentation

3.1 CommunicatingSocket Klassenreferenz

#include <PracticalSocket.h>

Klassendiagramm für CommunicatingSocket:



Öffentliche Methoden

- 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)

Geschützte Methoden

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

Weitere Geerbte Elemente

3.1.1 Ausführliche Beschreibung

Socket which is able to connect, send, and receive

3.1.2 Dokumentation der Elementfunktionen

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

Parameter

foreignAddress	foreign address (IP address or name)
foreignPort	foreign port

Ausnahmebehandlung

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()

Rückgabe

foreign address

Ausnahmebehandlung

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()

Rückgabe

foreign port

Ausnahmebehandlung

SocketException	thrown if unable to fetch foreign port

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

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

Parameter

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

Rückgabe

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

Ausnahmebehandlung

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()

Parameter

buffer	buffer to be written
bufferLen	number of bytes from buffer to be written

Ausnahmebehandlung

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · PracticalSocket.h
- PracticalSocket.cpp

3.2 Data Klassenreferenz

#include <Data.h>

Öffentliche Methoden

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

3.2.1 Ausführliche Beschreibung

Datenstruktur die einen Fahrzeugwert und die dazugehörigen Daten speichert.

3.2.2 Beschreibung der Konstruktoren und Destruktoren

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

Erzeugt eine Datenstruktur zur Speicherung von Fahrzeugdaten.

Parameter

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

3.2.3 Dokumentation der Elementfunktionen

3.2.3.1 unsigned int Data::getDatatype ()

Rückgabe

Gibt den Datentyp des Datensatzes zurück.

3.2.3.2 unsigned int Data::getPosition ()

Rückgabe

Gibt die Position des Datensatzes in den ursprünlichen Daten zurück.

3.2.3.3 double Data::getValue ()

Rückgabe

Gibt den Wert des Datensatzes zurück.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · Data.h
- · Data.cpp

3.3 Decoder Klassenreferenz

Öffentliche Methoden

- 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 Beschreibung der Konstruktoren und Destruktoren

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

Erzeugt einen Dekoder der zum dekodieren der Paketinformation dient.

Parameter

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.

Parameter

buffer	Speicher des Datenpakets.
buffernlen	Länge von buffer.
vecLayout	Aufteilung des ursprünglichen Datenstroms die aus den Paketinformationen dekodiert wur-
	den. 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 Dokumentation der Elementfunktionen

3.3.2.1 Data Decoder::getNextData ()

Holt den nächsten Datensatz aus den empfangenen Daten.

Rückgabe

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.

Rückgabe

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.

Parameter

vecLayout	Aufteilung des ursprünglichen Datenstroms die aus den Paketinformationen dekodiert wur-
	den.
vecLayoutlen	Länge von vecLayout.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · Encoding.h
- · Encoding.cpp

3.4 Encoder Klassenreferenz

#include <Encoding.h>

Öffentliche Methoden

- 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 Ausführliche Beschreibung

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

3.4.2 Beschreibung der Konstruktoren und Destruktoren

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.

Parameter

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.
	[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 Dokumentation der Elementfunktionen

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

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

Parameter

package	Speicher in den das Paket geschrieben werden soll.
len	Länge von package.

Rückgabe

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.

Parameter

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

Rückgabe

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.

Parameter

packageNumber	Paketnummer dessen Größe gesucht ist.
---------------	---------------------------------------

Rückgabe

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.

Rückgabe

Anzahl der Pakete.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · Encoding.h
- · Encoding.cpp

3.5 Location Klassenreferenz

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

Öffentliche Methoden

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

3.5.1 Ausführliche Beschreibung

Datenstruktur die Netzwerkdaten bestimmter Teilnehmer speichert.

3.5.2 Beschreibung der Konstruktoren und Destruktoren

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

Erzeugt einen Teilnehmer.

Parameter

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

3.5.3 Dokumentation der Elementfunktionen

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

Rückgabe

Gibt die Adresse zurück.

3.5.3.2 int Location::getPort ()

Rückgabe

Gibt die Portnummer zurück.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · Location.h
- · Location.cpp

3.6 Receiver Klassenreferenz

Öffentliche Methoden

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

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

· Communication.h

3.7 Sender Klassenreferenz

Öffentliche Methoden

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

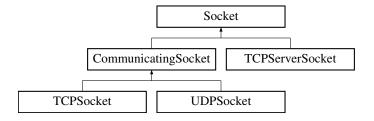
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · Communication.h
- · Communication.cpp

3.8 Socket Klassenreferenz

#include <PracticalSocket.h>

Klassendiagramm für Socket:



Öffentliche Methoden

- 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)

Öffentliche, statische Methoden

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

Geschützte Methoden

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

Geschützte Attribute

int sockDesc

3.8.1 Ausführliche Beschreibung

Base class representing basic communication endpoint

3.8.2 Beschreibung der Konstruktoren und Destruktoren

```
3.8.2.1 Socket::\simSocket ( )
```

Close and deallocate this socket

3.8.3 Dokumentation der Elementfunktionen

```
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.

Parameter

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

Rückgabe

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

Ausnahmebehandlung

SocketException	thrown WinSock clean up fails

3.8.3.2 string Socket::getLocalAddress () throw SocketException)

Get the local address

Rückgabe

local address of socket

Ausnahmebehandlung

SocketException

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

Get the local port

Rückgabe

local port of socket

Ausnahmebehandlung

SocketException thrown if fetch fails	SocketException
---	-----------------

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 Parameter

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.

Parameter

localAddress	local address
localPort	local port

Ausnahmebehandlung

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

Parameter

localPort	local port
-----------	------------

Ausnahmebehandlung

SocketException	thrown if setting local port fails

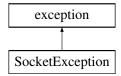
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

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

3.9 SocketException Klassenreferenz

#include <PracticalSocket.h>

Klassendiagramm für SocketException:



Öffentliche Methoden

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

3.9.1 Ausführliche Beschreibung

Signals a problem with the execution of a socket call.

3.9.2 Beschreibung der Konstruktoren und Destruktoren

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

Construct a SocketException with a explanatory message.

Parameter

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

3.9.2.2 SocketException:: ~SocketException () throw)

Provided just to guarantee that no exceptions are thrown.

3.9.3 Dokumentation der Elementfunktionen

3.9.3.1 const char * SocketException::what () const throw)

Get the exception message

Rückgabe

exception message

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · PracticalSocket.h
- · PracticalSocket.cpp

3.10 T_nuex Strukturreferenz

Öffentliche Attribute

• short int testen [401]

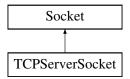
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

· mab.cpp

3.11 TCPServerSocket Klassenreferenz

#include <PracticalSocket.h>

Klassendiagramm für TCPServerSocket:



Öffentliche Methoden

- 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)

Weitere Geerbte Elemente

3.11.1 Ausführliche Beschreibung

TCP socket class for servers

	3.11.2	Beschreibung	der Konstruktoren	und Destruktore
--	--------	--------------	-------------------	-----------------

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

Parameter

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)

Ausnahmebehandlung

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

Parameter

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

Ausnahmebehandlung

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

3.11.3 Dokumentation der Elementfunktionen

3.11.3.1 TCPSocket * TCPServerSocket::accept () throw SocketException)

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

Rückgabe

new connection socket

Ausnahmebehandlung

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

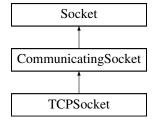
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · PracticalSocket.h
- · PracticalSocket.cpp

3.12 TCPSocket Klassenreferenz

#include <PracticalSocket.h>

Klassendiagramm für TCPSocket:



Öffentliche Methoden

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

Freundbeziehungen

· class TCPServerSocket

Weitere Geerbte Elemente

3.12.1 Ausführliche Beschreibung

TCP socket for communication with other TCP sockets

3.12.2 Beschreibung der Konstruktoren und Destruktoren

3.12.2.1 TCPSocket::TCPSocket () throw SocketException)

Construct a TCP socket with no connection

Ausnahmebehandlung

	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

Parameter

foreignAddress	foreign address (IP address or name)
foreignPort	foreign port

Ausnahmebehandlung

SocketException	thrown if unable to create TCP socket

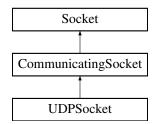
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · PracticalSocket.h
- PracticalSocket.cpp

3.13 UDPSocket Klassenreferenz

#include <PracticalSocket.h>

Klassendiagramm für UDPSocket:



Öffentliche Methoden

- 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)

Weitere Geerbte Elemente

3.13.1 Ausführliche Beschreibung

UDP socket class

3.13.2 Beschreibung der Konstruktoren und Destruktoren

3.13.2.1 UDPSocket::UDPSocket () throw SocketException)

Construct a UDP socket

Ausnahmebehandlung

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

Parameter

localPort

Ausnahmebehandlung

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

Parameter

22

localAddress	local address
localPort	local port

Ausnahmebehandlung

SocketException	thrown if unable to create UDP socket

3.13.3 Dokumentation der Elementfunktionen

3.13.3.1 void UDPSocket::disconnect () throw SocketException)

Unset foreign address and port

Rückgabe

true if disassociation is successful

Ausnahmebehandlung

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

Parameter

multicastGroup	multicast group address to join

Ausnahmebehandlung

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

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

Leave the specified multicast group

Parameter

multicastGroup	multicast group address to leave

Ausnahmebehandlung

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

Parameter

buffer	buffer to receive data
bufferLen	maximum number of bytes to receive
sourceAddress	address of datagram source
sourcePort	port of data source

Rückgabe

number of bytes received and -1 for error

Ausnahmebehandlung

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

Parameter

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

Rückgabe

true if send is successful

Ausnahmebehandlung

SocketException	thrown if unable to send datagram

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

Set the multicast TTL

Parameter

multicastTTL	multicast TTL

Ausnahmebehandlung

SocketException	thrown if unable to set TTL

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- · PracticalSocket.h
- · PracticalSocket.cpp

Index

\sim Socket	getNextData
Socket, 14	Decoder, 9
\sim SocketException	getNextPackage
SocketException, 16	Encoder, 11
	getPackage
accept	Encoder, 11
TCPServerSocket, 19	getPackageNum
cleanUp	Decoder, 10
Socket, 14	getPackagePos
CommunicatingSocket, 5	Decoder, 10
connect, 5	getPackageSize
getForeignAddress, 7	Encoder, 11
getForeignPort, 7	getPackageSum
recv, 7	Encoder, 11
send, 7	getPort
connect	Location, 12
CommunicatingSocket, 5	getPosition
Communicating Cocket, C	Data, 8
Data, 8	getValue
Data, 8	Data, 8
getDatatype, 8	
getPosition, 8	joinGroup
getValue, 8	UDPSocket, 22
Decoder, 9	
Decoder, 9	leaveGroup
getNextData, 9	UDPSocket, 22
getPackageNum, 10	Location, 12
getPackagePos, 10	getAddress, 12
disconnect	getPort, 12
UDPSocket, 22	Location, 12
,	
Encoder, 10	Receiver, 13
Encoder, 10	recv
getNextPackage, 11	CommunicatingSocket, 7
getPackage, 11	recvFrom
getPackageSize, 11	UDPSocket, 22
getPackageSum, 11	resolveService
	Socket, 15
getAddress	
Location, 12	send
getDatatype	CommunicatingSocket, 7
Data, 8	sendTo
getForeignAddress	UDPSocket, 23
CommunicatingSocket, 7	Sender, 13
getForeignPort	setLocalAddressAndPort
CommunicatingSocket, 7	Socket, 15
getLocalAddress	setLocalPort
Socket, 15	Socket, 15
getLocalPort	setMulticastTTL
Socket, 15	UDPSocket, 23

INDEX 25

```
Socket, 13
    \simSocket, 14
    cleanUp, 14
    getLocalAddress, 15
    getLocalPort, 15
    resolveService, 15
    setLocalAddressAndPort, 15
    setLocalPort, 15
SocketException, 16
    \simSocketException, 16
    SocketException, 16
    SocketException, 16
    what, 17
T_nuex, 17
TCPServerSocket, 17
    accept, 19
    TCPServerSocket, 18, 19
    TCPServerSocket, 18, 19
TCPSocket, 19
    TCPSocket, 20
    TCPSocket, 20
UDPSocket, 20
    disconnect, 22
    joinGroup, 22
    leaveGroup, 22
    recvFrom, 22
    sendTo, 23
    setMulticastTTL, 23
    UDPSocket, 21
    UDPSocket, 21
what
    SocketException, 17
```