

S²Sim 1.0.0

SmartGrid **Swarm Simulator**

Alper Sinan Akyurek

System Energy Efficiency Lab

University of California, San Diego

February 21, 2014

Contents

1	S2Sim	1
2	Todo List	3
3	Namespace Index	5
3.1	Namespace List	5
4	Hierarchical Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	11
6.1	File List	11
7	Namespace Documentation	15
7.1	TerraSwarm Namespace Reference	15
7.1.1	Detailed Description	15
7.1.2	Typedef Documentation	16
7.1.2.1	TByteIndex	16
7.1.2.2	TDataSize	16
7.2	TerraSwarm::Asynchronous Namespace Reference	16
7.2.1	Detailed Description	16
7.3	TerraSwarm::Synchronous Namespace Reference	16
7.3.1	Detailed Description	17
8	Class Documentation	19
8.1	TerraSwarm::Asynchronous::ClientConnectionRequest Class Reference	19
8.1.1	Detailed Description	20
8.1.2	Member Typedef Documentation	20

8.1.2.1	TCheckResult	20
8.1.2.2	TClientName	20
8.1.3	Member Enumeration Documentation	20
8.1.3.1	CheckResultValues	20
8.1.3.2	HeaderValues	20
8.1.4	Constructor & Destructor Documentation	21
8.1.4.1	ClientConnectionRequest	21
8.1.4.2	~ClientConnectionRequest	21
8.1.5	Member Function Documentation	21
8.1.5.1	CheckMessage	21
8.1.5.2	GetClientName	21
8.1.5.3	GetNewClientConnectionRequest	22
8.2	TerraSwarm::Synchronous::ClientConnectionRequest Class Reference	22
8.2.1	Detailed Description	23
8.2.2	Member Typedef Documentation	23
8.2.2.1	TCheckResult	23
8.2.2.2	TClientName	23
8.2.3	Member Enumeration Documentation	23
8.2.3.1	CheckResultValues	23
8.2.3.2	HeaderValues	24
8.2.4	Constructor & Destructor Documentation	24
8.2.4.1	ClientConnectionRequest	24
8.2.4.2	~ClientConnectionRequest	24
8.2.5	Member Function Documentation	24
8.2.5.1	CheckMessage	24
8.2.5.2	GetClientName	24
8.2.5.3	GetNewClientConnectionRequest	25
8.3	TerraSwarm::Asynchronous::ClientConnectionResponse Class Reference	25
8.3.1	Detailed Description	27
8.3.2	Member Typedef Documentation	27
8.3.2.1	TCheckResult	27
8.3.2.2	TNumberOfClients	28
8.3.2.3	TNumberOfClientsAccessor	28
8.3.2.4	TRequestResult	28
8.3.2.5	TRequestResultAccessor	28
8.3.2.6	TSystemMode	28
8.3.2.7	TSystemModeAccessor	28

8.3.2.8	TSystemTime	28
8.3.2.9	TSystemTimeAccessor	28
8.3.3	Member Enumeration Documentation	29
8.3.3.1	CheckResultValues	29
8.3.3.2	FieldIndexValues	29
8.3.3.3	FieldSizeValues	29
8.3.3.4	HeaderValues	29
8.3.3.5	RequestResultValues	30
8.3.3.6	SystemModeValues	30
8.3.4	Constructor & Destructor Documentation	30
8.3.4.1	ClientConnectionResponse	30
8.3.4.2	~ClientConnectionResponse	30
8.3.5	Member Function Documentation	30
8.3.5.1	CheckMessage	30
8.3.5.2	GetNewClientConnectionResponse	31
8.3.5.3	GetNumberOfClients	31
8.3.5.4	GetRequestResult	31
8.3.5.5	GetSize	31
8.3.5.6	GetSystemMode	32
8.3.5.7	GetSystemTime	32
8.4	TerraSwarm::Synchronous::ClientConnectionResponse Class Reference	32
8.4.1	Detailed Description	34
8.4.2	Member Typedef Documentation	34
8.4.2.1	TCheckResult	34
8.4.2.2	TNumberOfClients	34
8.4.2.3	TNumberOfClientsAccessor	34
8.4.2.4	TRequestResult	35
8.4.2.5	TRequestResultAccessor	35
8.4.2.6	TSystemMode	35
8.4.2.7	TSystemModeAccessor	35
8.4.2.8	TSystemTime	35
8.4.2.9	TSystemTimeAccessor	35
8.4.3	Member Enumeration Documentation	35
8.4.3.1	CheckResultValues	35
8.4.3.2	FieldIndexValues	36
8.4.3.3	FieldSizeValues	36
8.4.3.4	HeaderValues	36

8.4.3.5	RequestResultValues	36
8.4.3.6	SystemModeValues	37
8.4.4	Constructor & Destructor Documentation	37
8.4.4.1	ClientConnectionResponse	37
8.4.4.2	~ClientConnectionResponse	37
8.4.5	Member Function Documentation	37
8.4.5.1	CheckMessage	37
8.4.5.2	GetNewClientConnectionResponse	37
8.4.5.3	GetNumberOfClients	38
8.4.5.4	GetRequestResult	38
8.4.5.5	GetSize	38
8.4.5.6	GetSystemMode	38
8.4.5.7	GetSystemTime	39
8.5	TerraSwarm::Asynchronous::ClientData Class Reference	39
8.5.1	Detailed Description	41
8.5.2	Member Typedef Documentation	41
8.5.2.1	TCheckResult	41
8.5.2.2	TDataPoint	41
8.5.2.3	TNumberOfDataPoints	41
8.5.2.4	TNumberOfDataPointsAccessor	41
8.5.2.5	TStartTime	41
8.5.2.6	TStartTimeAccessor	41
8.5.2.7	TTimeResolution	41
8.5.2.8	TTimeResolutionAccessor	42
8.5.3	Member Enumeration Documentation	42
8.5.3.1	CheckResultValues	42
8.5.3.2	FieldIndexValues	42
8.5.3.3	FieldSizeValues	42
8.5.3.4	HeaderValues	43
8.5.4	Constructor & Destructor Documentation	43
8.5.4.1	ClientData	43
8.5.4.2	~ClientData	43
8.5.5	Member Function Documentation	43
8.5.5.1	CheckMessage	43
8.5.5.2	GetDataPoints	43
8.5.5.3	GetNewClientData	44
8.5.5.4	GetNumberOfDataPoints	45

8.5.5.5	GetStartTime	45
8.5.5.6	GetTimeResolution	45
8.6	TerraSwarm::Synchronous::ClientData Class Reference	46
8.6.1	Detailed Description	47
8.6.2	Member Typedef Documentation	47
8.6.2.1	TCheckResult	47
8.6.2.2	TDataPoint	47
8.6.2.3	TDataPointAccessor	47
8.6.3	Member Enumeration Documentation	47
8.6.3.1	CheckResultValues	47
8.6.3.2	FieldIndexValues	47
8.6.3.3	FieldSizeValues	48
8.6.3.4	HeaderValues	48
8.6.4	Constructor & Destructor Documentation	48
8.6.4.1	ClientData	48
8.6.4.2	~ClientData	48
8.6.5	Member Function Documentation	48
8.6.5.1	CheckMessage	48
8.6.5.2	GetDataPoint	49
8.6.5.3	GetNewClientData	49
8.7	ClientManager Class Reference	49
8.7.1	Detailed Description	52
8.7.2	Member Typedef Documentation	52
8.7.2.1	TClientType	52
8.7.2.2	TId	52
8.7.2.3	TInterval	52
8.7.2.4	TPrice	52
8.7.3	Member Enumeration Documentation	52
8.7.3.1	ClientTypeValues	52
8.7.4	Constructor & Destructor Documentation	53
8.7.4.1	ClientManager	53
8.7.4.2	ClientManager	53
8.7.4.3	ClientManager	53
8.7.4.4	~ClientManager	53
8.7.5	Member Function Documentation	53
8.7.5.1	ConnectionBroken	53
8.7.5.2	IsAsynchronous	54

8.7.5.3	IsSynchronous	54
8.7.5.4	MessageReceived	54
8.7.5.5	operator=	55
8.7.5.6	PriceProposal	56
8.7.5.7	ProcessClientConnectionRequest	56
8.7.5.8	ProcessClientConnectionRequest	58
8.7.5.9	ProcessClientData	59
8.7.5.10	ProcessClientData	59
8.7.5.11	ProcessDemandNegotiation	60
8.7.5.12	ProcessGetPrice	61
8.7.5.13	SetCurrentPrice	61
8.7.6	Member Data Documentation	62
8.7.6.1	m_client	62
8.7.6.2	m_clientId	62
8.7.6.3	m_clientType	62
8.7.6.4	nextClientId	62
8.8	CompileCheck< expression, Reason > Class Template Reference	63
8.8.1	Detailed Description	63
8.8.2	Member Enumeration Documentation	63
8.8.2.1	anonymous enum	63
8.8.3	Member Function Documentation	63
8.8.3.1	Check	63
8.9	CompileCheck< false, Reason > Class Template Reference	64
8.9.1	Detailed Description	64
8.9.2	Member Enumeration Documentation	64
8.9.2.1	anonymous enum	64
8.9.3	Member Function Documentation	64
8.9.3.1	Check	64
8.10	ConnectionManager Class Reference	65
8.10.1	Detailed Description	66
8.10.2	Member Typedef Documentation	67
8.10.2.1	TClientId	67
8.10.2.2	TClientIdList	67
8.10.2.3	TClientList	67
8.10.2.4	TNumberOfClients	67
8.10.2.5	TReversedClientList	67
8.10.3	Constructor & Destructor Documentation	67

8.10.3.1	ConnectionManager	67
8.10.4	Member Function Documentation	68
8.10.4.1	BrokenConnection	68
8.10.4.2	DeleteClient	69
8.10.4.3	GetNumberOfClients	69
8.10.4.4	IncomingConnection	70
8.10.4.5	IncomingMessage	70
8.10.5	Friends And Related Function Documentation	71
8.10.5.1	GetConnectionManager	71
8.10.6	Member Data Documentation	71
8.10.6.1	m_clientList	71
8.10.6.2	m_reversedClientList	71
8.10.6.3	m_server	71
8.11	ControlManager Class Reference	72
8.11.1	Detailed Description	74
8.11.2	Member Typedef Documentation	74
8.11.2.1	TClientId	74
8.11.2.2	TClientIdMap	75
8.11.2.3	TClientManagerMap	75
8.11.2.4	TClientName	75
8.11.2.5	TDataPoint	75
8.11.2.6	TDataSize	75
8.11.2.7	TMessageType	75
8.11.2.8	TNumberOfClients	75
8.11.2.9	TNumberOfDataPoints	75
8.11.2.10	TPrice	75
8.11.2.11	TVoltage	76
8.11.2.12	TWattage	76
8.11.3	Member Enumeration Documentation	76
8.11.3.1	MessageTypeValues	76
8.11.4	Constructor & Destructor Documentation	76
8.11.4.1	ControlManager	76
8.11.5	Member Function Documentation	77
8.11.5.1	ClientDemandNegotiation	77
8.11.5.2	ClientPriceRequest	77
8.11.5.3	GetClientName	78
8.11.5.4	MakeDecision	79

8.11.5.5	ProcessData	79
8.11.5.6	RegisterClient	80
8.11.5.7	SetClient	81
8.11.5.8	UnRegisterClient	82
8.11.5.9	WaitUntilReady	82
8.11.6	Friends And Related Function Documentation	83
8.11.6.1	GetControlManager	83
8.11.7	Member Data Documentation	83
8.11.7.1	m_client	83
8.11.7.2	m_clientIdMap	83
8.11.7.3	m_clientManagerMap	83
8.11.7.4	m_readySemaphore	83
8.11.7.5	m_server	84
8.12	TerraSwarm::Synchronous::DemandNegotiation Class Reference	84
8.12.1	Detailed Description	85
8.12.2	Member Typedef Documentation	85
8.12.2.1	TCheckResult	85
8.12.2.2	TDataPoint	85
8.12.2.3	TNumberOfDataPoints	85
8.12.2.4	TNumberOfDataPointsAccessor	86
8.12.3	Member Enumeration Documentation	86
8.12.3.1	CheckResultValues	86
8.12.3.2	FieldIndexValues	86
8.12.3.3	FieldSizeValues	86
8.12.3.4	HeaderValues	86
8.12.4	Constructor & Destructor Documentation	87
8.12.4.1	DemandNegotiation	87
8.12.4.2	~DemandNegotiation	87
8.12.5	Member Function Documentation	87
8.12.5.1	CheckMessage	87
8.12.5.2	GetDataPoints	87
8.12.5.3	GetNewDemandNegotiation	88
8.12.5.4	GetNumberOfDataPoints	88
8.13	TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size > Class Template Reference	89
8.13.1	Detailed Description	89
8.13.2	Member Function Documentation	89

8.13.2.1	HostToNetwork	89
8.13.2.2	NetworkToHost	90
8.14	TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 1 > Class Template Reference	90
8.14.1	Detailed Description	91
8.14.2	Member Function Documentation	91
8.14.2.1	HostToNetwork	91
8.14.2.2	NetworkToHost	91
8.15	TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 > Class Template Reference	91
8.15.1	Detailed Description	91
8.15.2	Member Function Documentation	92
8.15.2.1	HostToNetwork	92
8.15.2.2	NetworkToHost	92
8.16	TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 4 > Class Template Reference	92
8.16.1	Detailed Description	92
8.16.2	Member Function Documentation	92
8.16.2.1	HostToNetwork	92
8.16.2.2	NetworkToHost	93
8.17	TerraSwarm::Synchronous::GetPrice Class Reference	93
8.17.1	Detailed Description	94
8.17.2	Member Typedef Documentation	94
8.17.2.1	TCheckResult	94
8.17.3	Member Enumeration Documentation	94
8.17.3.1	CheckResultValues	94
8.17.3.2	FieldSizeValues	94
8.17.3.3	HeaderValues	94
8.17.4	Constructor & Destructor Documentation	95
8.17.4.1	GetPrice	95
8.17.4.2	~GetPrice	95
8.17.5	Member Function Documentation	95
8.17.5.1	CheckMessage	95
8.17.5.2	GetNewGetPrice	95
8.17.5.3	GetSize	95
8.18	ThreadBase::InputStructure Struct Reference	96
8.18.1	Detailed Description	97
8.18.2	Constructor & Destructor Documentation	97

8.18.2.1	InputStructure	97
8.18.2.2	InputStructure	97
8.18.3	Member Data Documentation	97
8.18.3.1	m_classPointer	97
8.18.3.2	m_mainInput	97
8.19	IPAddress Class Reference	98
8.19.1	Detailed Description	99
8.19.2	Member Typedef Documentation	99
8.19.2.1	TAddress	99
8.19.2.2	TAddressStruct	99
8.19.2.3	TPort	99
8.19.2.4	TSocketSize	99
8.19.3	Constructor & Destructor Documentation	99
8.19.3.1	IPAddress	99
8.19.3.2	IPAddress	100
8.19.3.3	~IPAddress	100
8.19.4	Member Function Documentation	100
8.19.4.1	GetAddress	100
8.19.4.2	GetAddress	100
8.19.4.3	GetPort	101
8.19.4.4	GetPort	101
8.19.4.5	GetSocketSize	101
8.19.4.6	operator sockaddr *	102
8.19.4.7	operator=	102
8.19.4.8	operator==	102
8.19.4.9	SetAddress	102
8.19.4.10	SetPort	103
8.19.5	Member Data Documentation	104
8.19.5.1	m_address	104
8.20	MatlabManager Class Reference	104
8.20.1	Detailed Description	107
8.20.2	Member Typedef Documentation	107
8.20.2.1	TClientCheckResult	107
8.20.2.2	TClientName	107
8.20.2.3	TMessageType	107
8.20.2.4	TVoltage	107
8.20.2.5	TWattage	107

8.20.3	Member Enumeration Documentation	108
8.20.3.1	ClientCheckResultValues	108
8.20.3.2	MessageTypeValues	108
8.20.4	Constructor & Destructor Documentation	108
8.20.4.1	MatlabManager	108
8.20.5	Member Function Documentation	109
8.20.5.1	AdvanceTimeStep	109
8.20.5.2	GetVoltage	109
8.20.5.3	GetWattage	110
8.20.5.4	IsClientPresent	111
8.20.5.5	ProcessData	112
8.20.5.6	SetClient	113
8.20.5.7	SetWattage	114
8.20.6	Friends And Related Function Documentation	114
8.20.6.1	GetMatlabManager	114
8.20.7	Member Data Documentation	115
8.20.7.1	m_client	115
8.20.7.2	m_clientPresenceSemaphore	115
8.20.7.3	m_clientPresentInformation	115
8.20.7.4	m_clientVoltageInformation	115
8.20.7.5	m_clientVoltageSemaphore	115
8.20.7.6	m_clientWattageInformation	115
8.20.7.7	m_clientWattageSemaphore	115
8.20.7.8	m_server	115
8.21	TerraSwarm::MessageEnder Class Reference	116
8.21.1	Detailed Description	116
8.21.2	Member Typedef Documentation	116
8.21.2.1	TCheckResult	116
8.21.2.2	TEndOfMessage	117
8.21.2.3	TEndOfMessageAccessor	117
8.21.3	Member Enumeration Documentation	117
8.21.3.1	CheckResultValues	117
8.21.3.2	EndOfMessageValues	117
8.21.3.3	SizeValues	117
8.21.4	Member Function Documentation	117
8.21.4.1	CheckEndOfMessageField	117
8.21.4.2	SetEndOfMessageField	118

8.22 TerraSwarm::MessageHeader Class Reference	118
8.22.1 Detailed Description	121
8.22.2 Member Typedef Documentation	121
8.22.2.1 TDataSizeAccessor	121
8.22.2.2 TId	121
8.22.2.3 TMessageId	121
8.22.2.4 TMessageIdAccessor	121
8.22.2.5 TMessageType	121
8.22.2.6 TMessageTypeAccessor	121
8.22.2.7 TReceiverId	121
8.22.2.8 TReceiverIdAccessor	122
8.22.2.9 TSenderId	122
8.22.2.10 TSenderIdAccessor	122
8.22.2.11 TSequenceNumber	122
8.22.2.12 TSequenceNumberAccessor	122
8.22.2.13 TStartOfMessage	122
8.22.2.14 TStartOfMessageAccessor	122
8.22.3 Member Enumeration Documentation	122
8.22.3.1 HeaderFieldIndexValues	122
8.22.3.2 HeaderFieldSizeValues	123
8.22.3.3 SizeValues	123
8.22.3.4 StartOfMessageValues	123
8.22.4 Constructor & Destructor Documentation	124
8.22.4.1 MessageHeader	124
8.22.5 Member Function Documentation	124
8.22.5.1 Access	124
8.22.5.2 Access	124
8.22.5.3 GetDataSize	124
8.22.5.4 GetMessageId	125
8.22.5.5 GetMessageType	125
8.22.5.6 GetNewMessageHeader	125
8.22.5.7 GetNextSequenceNumber	125
8.22.5.8 GetReceiverId	126
8.22.5.9 GetSenderId	126
8.22.5.10 PrepareOutgoingMessage	126
8.23 TerraSwarm::NetworkByteAccessor< byteIndex, dataSize > Class Template Reference	127
8.23.1 Detailed Description	128

8.23.2	Member Function Documentation	128
8.23.2.1	operator=	128
8.23.2.2	Read	128
8.23.2.3	Write	129
8.24	TerraSwarm::Synchronous::PriceProposal Class Reference	130
8.24.1	Detailed Description	131
8.24.2	Member Typedef Documentation	131
8.24.2.1	TCheckResult	131
8.24.2.2	TInterval	132
8.24.2.3	TIntervalBeginAccessor	132
8.24.2.4	TIntervalEndAccessor	132
8.24.2.5	TPrice	132
8.24.2.6	TPriceAccessor	132
8.24.3	Member Enumeration Documentation	132
8.24.3.1	CheckResultValues	132
8.24.3.2	FieldIndexValues	132
8.24.3.3	FieldSizeValues	133
8.24.3.4	HeaderValues	133
8.24.4	Constructor & Destructor Documentation	133
8.24.4.1	PriceProposal	133
8.24.4.2	~PriceProposal	133
8.24.5	Member Function Documentation	133
8.24.5.1	CheckMessage	133
8.24.5.2	GetIntervalBegin	134
8.24.5.3	GetIntervalEnd	134
8.24.5.4	GetNewPriceProposal	134
8.24.5.5	GetPrice	135
8.24.5.6	GetSize	135
8.25	TerraSwarm::Synchronous::SetCurrentPrice Class Reference	135
8.25.1	Detailed Description	136
8.25.2	Member Typedef Documentation	137
8.25.2.1	TCheckResult	137
8.25.2.2	TInterval	137
8.25.2.3	TIntervalBeginAccessor	137
8.25.2.4	TIntervalEndAccessor	137
8.25.2.5	TPrice	137
8.25.2.6	TPriceAccessor	137

8.25.3	Member Enumeration Documentation	137
8.25.3.1	CheckResultValues	137
8.25.3.2	FieldIndexValues	138
8.25.3.3	FieldSizeValues	138
8.25.3.4	HeaderValues	138
8.25.4	Constructor & Destructor Documentation	138
8.25.4.1	SetCurrentPrice	138
8.25.4.2	~SetCurrentPrice	138
8.25.5	Member Function Documentation	139
8.25.5.1	CheckMessage	139
8.25.5.2	GetIntervalBegin	139
8.25.5.3	GetIntervalEnd	139
8.25.5.4	GetNewSetCurrentPrice	139
8.25.5.5	GetPrice	140
8.25.5.6	GetSize	140
8.26	SizeCheck< checkedType, checkedSize, Reason > Class Template Reference	140
8.26.1	Detailed Description	141
8.26.2	Member Enumeration Documentation	141
8.26.2.1	anonymous enum	141
8.26.3	Member Function Documentation	141
8.26.3.1	Check	141
8.27	SocketBase< SocketType > Class Template Reference	142
8.27.1	Detailed Description	143
8.27.2	Member Typedef Documentation	143
8.27.2.1	SOCKET	143
8.27.2.2	TBuffer	143
8.27.2.3	TNumberOfBytes	143
8.27.2.4	TSocketId	143
8.27.3	Member Enumeration Documentation	144
8.27.3.1	SocketIdValues	144
8.27.4	Constructor & Destructor Documentation	144
8.27.4.1	SocketBase	144
8.27.4.2	SocketBase	144
8.27.4.3	SocketBase	144
8.27.4.4	~SocketBase	145
8.27.5	Member Function Documentation	145
8.27.5.1	CloseSocket	145

8.27.5.2	OpenSocket	145
8.27.6	Member Data Documentation	145
8.27.6.1	m_socketId	145
8.28	SystemManager Class Reference	145
8.28.1	Detailed Description	147
8.28.2	Member Typedef Documentation	147
8.28.2.1	TClientId	147
8.28.2.2	TClientName	147
8.28.2.3	TDataMap	147
8.28.2.4	TDataPoint	147
8.28.2.5	TNumberOfDataPoints	148
8.28.2.6	TSystemMap	148
8.28.2.7	TSystemMode	148
8.28.2.8	TSystemTime	148
8.28.2.9	TVoltage	148
8.28.2.10	TWattage	148
8.28.3	Member Enumeration Documentation	148
8.28.3.1	SystemModeValues	148
8.28.4	Constructor & Destructor Documentation	149
8.28.4.1	SystemManager	149
8.28.5	Member Function Documentation	149
8.28.5.1	AdvanceTimeStep	149
8.28.5.2	GetSystemMode	150
8.28.5.3	GetSystemTime	150
8.28.5.4	RegisterData	151
8.28.5.5	RegisterData	151
8.28.6	Friends And Related Function Documentation	152
8.28.6.1	GetSystemManager	152
8.28.7	Member Data Documentation	152
8.28.7.1	m_systemMap	152
8.28.7.2	m_systemMode	152
8.28.7.3	m_systemTime	152
8.29	TCPClient Class Reference	153
8.29.1	Detailed Description	154
8.29.2	Member Typedef Documentation	155
8.29.2.1	TBaseType	155
8.29.3	Constructor & Destructor Documentation	155

8.29.3.1	TCPClient	155
8.29.3.2	~TCPClient	155
8.29.4	Member Function Documentation	155
8.29.4.1	Connect	155
8.29.4.2	ReceiveData	155
8.29.4.3	SendData	157
8.30	TCPConnectedClient Class Reference	157
8.30.1	Detailed Description	159
8.30.2	Member Typedef Documentation	159
8.30.2.1	TBaseType	159
8.30.3	Constructor & Destructor Documentation	159
8.30.3.1	TCPConnectedClient	159
8.30.3.2	TCPConnectedClient	160
8.30.3.3	~TCPConnectedClient	160
8.30.4	Member Function Documentation	160
8.30.4.1	GetClientAddress	160
8.30.4.2	ReceiveData	160
8.30.4.3	SendData	161
8.30.5	Member Data Documentation	161
8.30.5.1	m_clientAddress	161
8.31	TCPServer Class Reference	162
8.31.1	Detailed Description	163
8.31.2	Member Typedef Documentation	164
8.31.2.1	TBaseType	164
8.31.3	Constructor & Destructor Documentation	164
8.31.3.1	TCPServer	164
8.31.3.2	~TCPServer	164
8.31.4	Member Function Documentation	164
8.31.4.1	Accept	164
8.31.4.2	Listen	165
8.31.4.3	SetAddress	165
8.31.4.4	SetPort	166
8.32	ThreadBase Class Reference	167
8.32.1	Detailed Description	170
8.32.2	Member Typedef Documentation	170
8.32.2.1	TStackSize	170
8.32.2.2	TStackSize	170

8.32.2.3	TStarted	170
8.32.2.4	TStarted	170
8.32.2.5	TThreadAttribute	170
8.32.2.6	TThreadHandle	170
8.32.2.7	TThreadId	170
8.32.2.8	TThreadId	171
8.32.3	Member Enumeration Documentation	171
8.32.3.1	StartedValues	171
8.32.3.2	StartedValues	171
8.32.4	Constructor & Destructor Documentation	171
8.32.4.1	ThreadBase	171
8.32.4.2	ThreadBase	171
8.32.4.3	~ThreadBase	172
8.32.4.4	ThreadBase	172
8.32.4.5	ThreadBase	172
8.32.4.6	~ThreadBase	172
8.32.4.7	ThreadBase	172
8.32.5	Member Function Documentation	173
8.32.5.1	ExecuteThread	173
8.32.5.2	ExecuteThread	173
8.32.5.3	ExecutionBody	174
8.32.5.4	ExecutionBody	174
8.32.5.5	GetStackSize	175
8.32.5.6	GetStackSize	175
8.32.5.7	SetStackSize	175
8.32.5.8	SetStackSize	175
8.32.5.9	StartThread	175
8.32.5.10	StartThread	175
8.32.6	Friends And Related Function Documentation	176
8.32.6.1	PosixThreadCover	176
8.32.6.2	WindowsThreadCover	176
8.32.7	Member Data Documentation	177
8.32.7.1	m_stackSize	177
8.32.7.2	m_started	177
8.32.7.3	m_threadAttribute	177
8.32.7.4	m_threadHandle	177
8.32.7.5	m_threadId	177

8.33 ThreadedTCPClient Class Reference	178
8.33.1 Detailed Description	180
8.33.2 Member Typedef Documentation	180
8.33.2.1 TNotification	180
8.33.3 Constructor & Destructor Documentation	180
8.33.3.1 ThreadedTCPClient	180
8.33.3.2 ThreadedTCPClient	181
8.33.3.3 ~ThreadedTCPClient	181
8.33.4 Member Function Documentation	181
8.33.4.1 ExecutionBody	181
8.33.4.2 SetNotificationCallback	182
8.33.5 Member Data Documentation	182
8.33.5.1 m_allowingSemaphore	182
8.33.5.2 m_notification	182
8.33.5.3 m_started	182
8.34 ThreadedTCPConnectedClient Class Reference	183
8.34.1 Detailed Description	185
8.34.2 Member Typedef Documentation	185
8.34.2.1 TNotification	185
8.34.3 Constructor & Destructor Documentation	185
8.34.3.1 ThreadedTCPConnectedClient	185
8.34.3.2 ThreadedTCPConnectedClient	186
8.34.3.3 ~ThreadedTCPConnectedClient	186
8.34.4 Member Function Documentation	186
8.34.4.1 ExecutionBody	186
8.34.4.2 SetNotificationCallback	187
8.34.5 Member Data Documentation	187
8.34.5.1 m_allowingSemaphore	187
8.34.5.2 m_notification	187
8.34.5.3 m_started	187
8.35 ThreadedTCPServer Class Reference	187
8.35.1 Detailed Description	189
8.35.2 Member Typedef Documentation	189
8.35.2.1 TNotification	189
8.35.3 Constructor & Destructor Documentation	190
8.35.3.1 ThreadedTCPServer	190
8.35.3.2 ~ThreadedTCPServer	190

8.35.4	Member Function Documentation	190
8.35.4.1	ExecutionBody	190
8.35.4.2	SetNotificationCallback	190
8.35.5	Member Data Documentation	191
8.35.5.1	m_notification	191
8.35.5.2	m_started	191
8.36	UDPSocket Class Reference	192
8.36.1	Detailed Description	193
8.36.2	Member Typedef Documentation	193
8.36.2.1	TBaseType	193
8.36.3	Constructor & Destructor Documentation	193
8.36.3.1	UDPSocket	193
8.36.3.2	~UDPSocket	194
8.36.4	Member Function Documentation	194
8.36.4.1	ReceiveData	194
8.36.4.2	SendData	194
8.36.4.3	SetIPAddress	195
8.36.4.4	SetPort	196
9	File Documentation	199
9.1	S2Sim/ClientManager.cpp File Reference	199
9.1.1	Detailed Description	199
9.2	S2Sim/ClientManager.h File Reference	200
9.2.1	Detailed Description	201
9.3	S2Sim/ConnectionManager.cpp File Reference	201
9.3.1	Detailed Description	202
9.3.2	Function Documentation	202
9.3.2.1	ConnectionNotificationHandler	202
9.3.2.2	ConnectionReceiveHandler	202
9.3.2.3	GetConnectionManager	203
9.4	S2Sim/ConnectionManager.h File Reference	204
9.4.1	Detailed Description	205
9.4.2	Function Documentation	205
9.4.2.1	ConnectionNotificationHandler	205
9.4.2.2	ConnectionReceiveHandler	206
9.4.2.3	GetConnectionManager	206
9.5	S2Sim/ControlManager.cpp File Reference	207

9.5.1	Detailed Description	207
9.5.2	Function Documentation	208
9.5.2.1	ControlNotificationHandler	208
9.5.2.2	ControlReceiveHandler	208
9.5.2.3	GetControlManager	209
9.6	S2Sim/ControlManager.h File Reference	210
9.6.1	Detailed Description	211
9.6.2	Function Documentation	211
9.6.2.1	ControlNotificationHandler	211
9.6.2.2	ControlReceiveHandler	211
9.6.2.3	GetControlManager	212
9.7	S2Sim/LogPrint.cpp File Reference	212
9.7.1	Detailed Description	213
9.7.2	Variable Documentation	214
9.7.2.1	functionPrint	214
9.7.2.2	logLevel	214
9.8	S2Sim/LogPrint.h File Reference	214
9.8.1	Detailed Description	217
9.8.2	Macro Definition Documentation	217
9.8.2.1	LOG_FUNCTION_END	217
9.8.2.2	LOG_FUNCTION_START	217
9.8.3	Function Documentation	217
9.8.3.1	ErrorPrint	217
9.8.3.2	ErrorPrint	218
9.8.3.3	ErrorPrint	218
9.8.3.4	ErrorPrint	219
9.8.3.5	ErrorPrint	219
9.8.3.6	ErrorPrint	219
9.8.3.7	LogPrint	220
9.8.3.8	LogPrint	221
9.8.3.9	LogPrint	222
9.8.3.10	LogPrint	222
9.8.3.11	LogPrint	223
9.8.3.12	LogPrint	223
9.8.3.13	LogPrint	223
9.8.3.14	LogPrint	224
9.8.3.15	WarningPrint	224

9.8.3.16	WarningPrint	225
9.8.3.17	WarningPrint	225
9.8.3.18	WarningPrint	225
9.8.3.19	WarningPrint	226
9.8.3.20	WarningPrint	226
9.8.4	Variable Documentation	226
9.8.4.1	functionPrint	227
9.8.4.2	logLevel	227
9.9	S2Sim/main.cpp File Reference	227
9.9.1	Detailed Description	227
9.9.2	Function Documentation	228
9.9.2.1	main	228
9.10	S2Sim/MatlabManager.cpp File Reference	228
9.10.1	Detailed Description	229
9.10.2	Function Documentation	229
9.10.2.1	GetMatlabManager	229
9.10.2.2	MatlabNotificationHandler	230
9.10.2.3	MatlabReceiveHandler	230
9.11	S2Sim/MatlabManager.h File Reference	231
9.11.1	Detailed Description	232
9.11.2	Function Documentation	232
9.11.2.1	GetMatlabManager	232
9.11.2.2	MatlabReceiveHandler	233
9.12	S2Sim/SemaphoreLibrary/Semaphore.cpp File Reference	234
9.12.1	Detailed Description	234
9.13	S2Sim/SemaphoreLibrary/Semaphore.h File Reference	234
9.13.1	Detailed Description	235
9.14	S2Sim/SocketLibrary/IPAddress.cpp File Reference	236
9.14.1	Detailed Description	236
9.15	S2Sim/SocketLibrary/IPAddress.h File Reference	236
9.15.1	Detailed Description	237
9.16	S2Sim/SocketLibrary/SocketBase.h File Reference	238
9.16.1	Detailed Description	239
9.17	S2Sim/SocketLibrary/TCPClient.cpp File Reference	239
9.17.1	Detailed Description	240
9.18	S2Sim/SocketLibrary/TCPClient.h File Reference	240
9.18.1	Detailed Description	242

9.19	S2Sim/SocketLibrary/TCPConnectedClient.cpp File Reference	242
9.19.1	Detailed Description	242
9.20	S2Sim/SocketLibrary/TCPConnectedClient.h File Reference	243
9.20.1	Detailed Description	244
9.21	S2Sim/SocketLibrary/TCPServer.cpp File Reference	244
9.21.1	Detailed Description	245
9.22	S2Sim/SocketLibrary/TCPServer.h File Reference	245
9.22.1	Detailed Description	247
9.23	S2Sim/SocketLibrary/ThreadedTCPClient.cpp File Reference	247
9.23.1	Detailed Description	247
9.24	S2Sim/SocketLibrary/ThreadedTCPClient.h File Reference	248
9.24.1	Detailed Description	249
9.25	S2Sim/SocketLibrary/ThreadedTCPConnectedClient.cpp File Reference	249
9.25.1	Detailed Description	250
9.26	S2Sim/SocketLibrary/ThreadedTCPConnectedClient.h File Reference	250
9.26.1	Detailed Description	252
9.27	S2Sim/SocketLibrary/ThreadedTCPServer.cpp File Reference	252
9.27.1	Detailed Description	253
9.28	S2Sim/SocketLibrary/ThreadedTCPServer.h File Reference	253
9.28.1	Detailed Description	254
9.29	S2Sim/SocketLibrary/UDPSocket.cpp File Reference	254
9.29.1	Detailed Description	255
9.30	S2Sim/SocketLibrary/UDPSocket.h File Reference	255
9.30.1	Detailed Description	256
9.31	S2Sim/SystemManager.cpp File Reference	257
9.31.1	Detailed Description	257
9.31.2	Function Documentation	258
9.31.2.1	GetSystemManager	258
9.32	S2Sim/SystemManager.h File Reference	258
9.32.1	Detailed Description	259
9.32.2	Function Documentation	259
9.32.2.1	GetSystemManager	259
9.33	S2Sim/TerraswarmLibrary/ClientConnectionRequest.cpp File Reference	260
9.33.1	Detailed Description	261
9.34	S2Sim/TerraswarmLibrary/ClientConnectionRequest.h File Reference	262
9.34.1	Detailed Description	263
9.35	S2Sim/TerraswarmLibrary/ClientConnectionResponse.cpp File Reference	264

9.35.1 Detailed Description	264
9.36 S2Sim/TerraswarmLibrary/ClientConnectionResponse.h File Reference	265
9.36.1 Detailed Description	266
9.37 S2Sim/TerraswarmLibrary/ClientData.cpp File Reference	267
9.37.1 Detailed Description	268
9.38 S2Sim/TerraswarmLibrary/ClientData.h File Reference	268
9.38.1 Detailed Description	269
9.39 S2Sim/TerraswarmLibrary/CompileTimeCheckerLibrary.h File Reference	269
9.39.1 Detailed Description	270
9.40 S2Sim/TerraswarmLibrary/DemandNegotiation.cpp File Reference	270
9.40.1 Detailed Description	271
9.41 S2Sim/TerraswarmLibrary/DemandNegotiation.h File Reference	272
9.41.1 Detailed Description	273
9.42 S2Sim/TerraswarmLibrary/GetPrice.cpp File Reference	273
9.42.1 Detailed Description	274
9.43 S2Sim/TerraswarmLibrary/GetPrice.h File Reference	275
9.43.1 Detailed Description	276
9.44 S2Sim/TerraswarmLibrary/MessageEnder.cpp File Reference	276
9.44.1 Detailed Description	277
9.45 S2Sim/TerraswarmLibrary/MessageEnder.h File Reference	278
9.45.1 Detailed Description	279
9.46 S2Sim/TerraswarmLibrary/MessageHeader.cpp File Reference	279
9.46.1 Detailed Description	280
9.47 S2Sim/TerraswarmLibrary/MessageHeader.h File Reference	280
9.47.1 Detailed Description	281
9.48 S2Sim/TerraswarmLibrary/NetworkByteAccessor.h File Reference	281
9.48.1 Detailed Description	283
9.49 S2Sim/TerraswarmLibrary/PriceProposal.cpp File Reference	283
9.49.1 Detailed Description	284
9.50 S2Sim/TerraswarmLibrary/PriceProposal.h File Reference	285
9.50.1 Detailed Description	286
9.51 S2Sim/TerraswarmLibrary/SetCurrentPrice.cpp File Reference	286
9.51.1 Detailed Description	287
9.52 S2Sim/TerraswarmLibrary/SetCurrentPrice.h File Reference	288
9.52.1 Detailed Description	289
9.53 S2Sim/ThreadLibrary/PosixThreadBase.h File Reference	289
9.53.1 Detailed Description	290

9.53.2	Function Documentation	290
9.53.2.1	PosixThreadCover	290
9.54	S2Sim/ThreadLibrary/ThreadBase.h File Reference	291
9.54.1	Detailed Description	291
9.55	S2Sim/ThreadLibrary/ThreadImplementation.cpp File Reference	291
9.55.1	Detailed Description	292
9.56	S2Sim/ThreadLibrary/WindowsThreadBase.h File Reference	292
9.56.1	Detailed Description	293
9.56.2	Function Documentation	293
9.56.2.1	WindowsThreadCover	293

Chapter 1

S2Sim

S2Sim, SmartGrid Swarm Simulator is a Smart Grid simulation tool that allows clients to cosimulate their local controls and test their outputs within the big picture of the grid. S2Sim uses OpenDSS to solve the power flow equations. More on the usage of S2Sim is written in its Interoperability Document.

Chapter 2

Todo List

Member `ClientManager::~~ClientManager` (void)

This should be replaced by a reference counting memory manager.

Member `ControlManager::GetClientName` (const TClientId clientId)

Let's make this not inline for debugging.

Member `ControlManager::ProcessData` (void *data, const size_t size)

Divide the function into multiple functions, processing each message separately.

Member `ControlManager::RegisterClient` (const TClientId clientId, const TClientName &clientName, `ClientManager` *clientManager)

This shouldn't be inline.

Member `ControlManager::WaitUntilReady` (void)

This doesn't need to be inline.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

TerraSwarm	
TerraSwarm related classes are defined under this namespace	15
TerraSwarm::Asynchronous	
Asynchronous client messages are defined under this namespace	16
TerraSwarm::Synchronous	
Synchronous client message are defined under this namespace	16

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

TerraSwarm::Asynchronous::ClientConnectionRequest	19
TerraSwarm::Synchronous::ClientConnectionRequest	22
TerraSwarm::Asynchronous::ClientConnectionResponse	25
TerraSwarm::Synchronous::ClientConnectionResponse	32
TerraSwarm::Asynchronous::ClientData	39
TerraSwarm::Synchronous::ClientData	46
ClientManager	49
CompileCheck< expression, Reason >	63
CompileCheck< false, Reason >	64
ConnectionManager	65
ControlManager	72
TerraSwarm::Synchronous::DemandNegotiation	84
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size >	89
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 1 >	90
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 >	91
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 4 >	92
TerraSwarm::Synchronous::GetPrice	93
ThreadBase::InputStructure	96
IPAddress	98
MatlabManager	104
TerraSwarm::MessageEnder	116
TerraSwarm::MessageHeader	118
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >	127
TerraSwarm::Synchronous::PriceProposal	130
TerraSwarm::Synchronous::SetCurrentPrice	135
SizeCheck< checkedType, checkedSize, Reason >	140
SocketBase< SocketType >	142
SocketBase< SOCK_DGRAM >	142
UDPSocket	192
SocketBase< SOCK_STREAM >	142
TCPClient	153
ThreadedTCPClient	178
TCPConnectedClient	157

ThreadedTCPConnectedClient	183
TCPServer	162
ThreadedTCPServer	187
SystemManager	145
ThreadBase	167
ThreadedTCPClient	178
ThreadedTCPConnectedClient	183
ThreadedTCPServer	187

Chapter 5

Class Index

5.1 Class List

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

TerraSwarm::Asynchronous::ClientConnectionRequest	
This class implements the asynchronous client connection request message	19
TerraSwarm::Synchronous::ClientConnectionRequest	
This class implements the synchronous client connection request message	22
TerraSwarm::Asynchronous::ClientConnectionResponse	
This class implements the Response message of the controller to the clients connection request . .	25
TerraSwarm::Synchronous::ClientConnectionResponse	
This class implements the Response message of the controller to the clients connection request . .	32
TerraSwarm::Asynchronous::ClientData	
Asynchronous Client Data message from the client to indicate its consumption for a time interval . .	39
TerraSwarm::Synchronous::ClientData	
Synchronous Client Data message from the client to indicate its consumption for a time interval . .	46
ClientManager	
Manages the connection with a client	49
CompileCheck< expression, Reason >	
This class has two specializations	63
CompileCheck< false, Reason >	
Second specialization of the class, when the expression is not true	64
ConnectionManager	
Manages connections to all clients	65
ControlManager	
Manages the connection with the External Controller	72
TerraSwarm::Synchronous::DemandNegotiation	
Defines the DemandNegotiation message sent from the client to the Controller as a response to the price proposal	84
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size >	
Template class that uses the correct conversion function according to the size of the data	89
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 1 >	
Template specialization for a type with size 1 (char)	90
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 >	
Template specialization for a type with size 2 (short)	91
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 4 >	
Template specialization for a type with size 4 (int)	92

TerraSwarm::Synchronous::GetPrice	
Defines the GetPrice message sent from the client to the Controller to get the current price value	93
ThreadBase::InputStructure	
Special Input structure sent to the wrapper function: PosixThreadCover()	96
IPAddress	
This class is an abstraction of the OS IP Address structure	98
MatlabManager	
Manages the connection to the OpenDSS-Matlab controller	104
TerraSwarm::MessageEnder	
Class to send the end of message field	116
TerraSwarm::MessageHeader	
This class defines the common message header for all messages	118
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >	
Template class to automatically convert byte order and help access ordered bytes in the memory	127
TerraSwarm::Synchronous::PriceProposal	
Price Proposal message sent by the controller to the clients to propose a price	130
TerraSwarm::Synchronous::SetCurrentPrice	
Set Current Price message sent for the Controller to the clients to set the current price and advance the time frame	135
SizeCheck< checkedType, checkedSize, Reason >	
This class checks the size of a type with the given size and gives a compile error with the reason parameter is the check fails	140
SocketBase< SocketType >	
This class is an abstraction over the OS socket and defines a base class for socket opening and closing utilities	142
SystemManager	
Manages the various components of the system and timing	145
TCPClient	
This class defines a TCP client that can connect to a TCP server and communicate	153
TCPConnectedClient	
Manages the connection to a connected client on the server side	157
TCPServer	
Defines a TCP server that can listen to connection attempts and can accept them for communication	162
ThreadBase	
Provides a base class that can be inherited from to gain threading capabilities	167
ThreadedTCPClient	
This is a TCP client class that receives data in a separate thread in the background	178
ThreadedTCPConnectedClient	
Manages the connection to an accepted client on the server side and receives data in a separate thread in the background	183
ThreadedTCPServer	
Defines the threaded version of TCPServer that accepts clients in another thread in the background	187
UDPSocket	
Manages a UDP connection	192

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

S2Sim/ ClientManager.cpp	
Source file for the ClientManager class	199
S2Sim/ ClientManager.h	
Header file for the ClientManager class	200
S2Sim/ ConnectionManager.cpp	
This file implements the ConnectionManager class	201
S2Sim/ ConnectionManager.h	
This file defines the ConnectionManager class	204
S2Sim/ ControlManager.cpp	
This file implements the ControlManager class	207
S2Sim/ ControlManager.h	
This file defines the ControlManager class	210
S2Sim/ LogPrint.cpp	
Defines logging related parameters	212
S2Sim/ LogPrint.h	
Defines logging related functions	214
S2Sim/ main.cpp	
Main file that starts an infinite running loop on the SystemManager::AdvanceTimeStep method	227
S2Sim/ MatlabManager.cpp	
Implements the MatlabManager class	228
S2Sim/ MatlabManager.h	
Defines the MatlabManager class	231
S2Sim/ SystemManager.cpp	
Implements the SystemManager class	257
S2Sim/ SystemManager.h	
Defines the SystemManager class	258
S2Sim/SemaphoreLibrary/ Semaphore.cpp	
Implements the Semaphore class under Windows 32/64bit and POSIX enables OSes	234
S2Sim/SemaphoreLibrary/ Semaphore.h	
Defines the Semaphore class under Windows 32/64bit and POSIX enables OSes	234
S2Sim/SocketLibrary/ IPAddress.cpp	
Implements the IPAddress class	236
S2Sim/SocketLibrary/ IPAddress.h	
Defines the IPAddress class	236

S2Sim/SocketLibrary/SocketBase.h	Defines the SocketBase template class	238
S2Sim/SocketLibrary/TCPClient.cpp	Implements the TCPClient class	239
S2Sim/SocketLibrary/TCPClient.h	Defines the TCPClient class	240
S2Sim/SocketLibrary/TCPConnectedClient.cpp	Implements the TCPConnectedClient class	242
S2Sim/SocketLibrary/TCPConnectedClient.h	Defines the TCPConnectedClient class	243
S2Sim/SocketLibrary/TCPServer.cpp	Implements the TCPServer class	244
S2Sim/SocketLibrary/TCPServer.h	Defines the TCPServer class	245
S2Sim/SocketLibrary/ThreadedTCPClient.cpp	Implements the ThreadedTCPClient class	247
S2Sim/SocketLibrary/ThreadedTCPClient.h	Defines the ThreadedTCPClient class	248
S2Sim/SocketLibrary/ThreadedTCPConnectedClient.cpp	Implements the ThreadedTCPConnectedClient class	249
S2Sim/SocketLibrary/ThreadedTCPConnectedClient.h	Defines the ThreadedTCPConnectedClient class	250
S2Sim/SocketLibrary/ThreadedTCPServer.cpp	Implements the ThreadedTCPServer class	252
S2Sim/SocketLibrary/ThreadedTCPServer.h	Defines the ThreadedTCPServer class	253
S2Sim/SocketLibrary/UDPSocket.cpp	Implements the UDPSocket class	254
S2Sim/SocketLibrary/UDPSocket.h	Defines the UDPSocket class	255
S2Sim/TerraswarmLibrary/ClientConnectionRequest.cpp	Implements the ClientConnectionRequest class	260
S2Sim/TerraswarmLibrary/ClientConnectionRequest.h	Defines the ClientConnectionRequest classes	262
S2Sim/TerraswarmLibrary/ClientConnectionResponse.cpp	Implements the Async and Synchronous ClientConnectionResponse message and classes	264
S2Sim/TerraswarmLibrary/ClientConnectionResponse.h	Defines the Async and Synchronous ClientConnectionResponse messages and classes	265
S2Sim/TerraswarmLibrary/ClientData.cpp	Implements the Async and Synchronous ClientData classes and messages	267
S2Sim/TerraswarmLibrary/ClientData.h	Defines the Aysnchronous and Synchronous ClientData class and messages	268
S2Sim/TerraswarmLibrary/CompileTimeCheckerLibrary.h	This file contains template classes to do compile time checking	269
S2Sim/TerraswarmLibrary/DemandNegotiation.cpp	Implements the DemandNegotiation class	270
S2Sim/TerraswarmLibrary/DemandNegotiation.h	Defines the DemandNegotiation class and message	272
S2Sim/TerraswarmLibrary/GetPrice.cpp	Implements the GetPrice class	273
S2Sim/TerraswarmLibrary/GetPrice.h	Defines the GetPrice class and message	275
S2Sim/TerraswarmLibrary/MessageEnder.cpp	Implements a MessageEnder class	276

S2Sim/TerraswarmLibrary/ MessageEnder.h	
Defines a MessageEnder class	278
S2Sim/TerraswarmLibrary/ MessageHeader.cpp	
Implements the MessageHeader class	279
S2Sim/TerraswarmLibrary/ MessageHeader.h	
Defines the MessageHeader class	280
S2Sim/TerraswarmLibrary/ NetworkByteAccessor.h	
Defines the NetworkByteAccessor class for byte order conversion and easier data access	281
S2Sim/TerraswarmLibrary/ PriceProposal.cpp	
Implements the PriceProposal class	283
S2Sim/TerraswarmLibrary/ PriceProposal.h	
Defines the PriceProposal class	285
S2Sim/TerraswarmLibrary/ SetCurrentPrice.cpp	
Implements the SetCurrentPrice class	286
S2Sim/TerraswarmLibrary/ SetCurrentPrice.h	
Defines the SetCurrentPrice class	288
S2Sim/ThreadLibrary/ PosixThreadBase.h	
Defines the ThreadBase class	289
S2Sim/ThreadLibrary/ ThreadBase.h	
This file selects the right implementation according to the current OS	291
S2Sim/ThreadLibrary/ ThreadImplementation.cpp	
Implements the Thread Wrapper functions	291
S2Sim/ThreadLibrary/ WindowsThreadBase.h	
Definition of the ThreadBase class under Windows implementation	292

Chapter 7

Namespace Documentation

7.1 TerraSwarm Namespace Reference

[TerraSwarm](#) related classes are defined under this namespace.

Namespaces

- [Asynchronous](#)
[Asynchronous](#) client messages are defined under this namespace.
- [Synchronous](#)
[Synchronous](#) client message are defined under this namespace.

Classes

- class [MessageEnder](#)
Class to send the end of message field.
- class [MessageHeader](#)
This class defines the common message header for all messages.
- class [NetworkByteAccessor](#)
Template class to automatically convert byte order and help access ordered bytes in the memory.

Typedefs

- typedef unsigned int [TByteIndex](#)
Index of a byte in memory.
- typedef unsigned int [TDataSize](#)
Size of a data in memory.

7.1.1 Detailed Description

[TerraSwarm](#) related classes are defined under this namespace.

7.1.2 Typedef Documentation

7.1.2.1 typedef unsigned int TerraSwarm::TByteIndex

Index of a byte in memory.

Definition at line 25 of file NetworkByteAccessor.h.

7.1.2.2 typedef unsigned int TerraSwarm::TDataSize

Size of a data in memory.

Definition at line 30 of file NetworkByteAccessor.h.

7.2 TerraSwarm::Asynchronous Namespace Reference

[Asynchronous](#) client messages are defined under this namespace.

Classes

- class [ClientConnectionRequest](#)
This class implements the asynchronous client connection request message.
- class [ClientConnectionResponse](#)
This class implements the Response message of the controller to the clients connection request.
- class [ClientData](#)
[Asynchronous](#) Client Data message from the client to indicate its consumption for a time interval.

7.2.1 Detailed Description

[Asynchronous](#) client messages are defined under this namespace.

7.3 TerraSwarm::Synchronous Namespace Reference

[Synchronous](#) client message are defined under this namespace.

Classes

- class [ClientConnectionRequest](#)
This class implements the synchronous client connection request message.
- class [ClientConnectionResponse](#)
This class implements the Response message of the controller to the clients connection request.
- class [ClientData](#)
[Synchronous](#) Client Data message from the client to indicate its consumption for a time interval.
- class [DemandNegotiation](#)
Defines the [DemandNegotiation](#) message sent from the client to the Controller as a response to the price proposal.
- class [GetPrice](#)

Defines the [GetPrice](#) message sent from the client to the Controller to get the current price value.

- class [PriceProposal](#)

Price Proposal message sent by the controller to the clients to propose a price.

- class [SetCurrentPrice](#)

Set Current Price message sent for the Controller to the clients to set the current price and advance the time frame.

7.3.1 Detailed Description

[Synchronous](#) client message are defined under this namespace.

Chapter 8

Class Documentation

8.1 TerraSwarm::Asynchronous::ClientConnectionRequest Class Reference

This class implements the asynchronous client connection request message.

```
#include <ClientConnectionRequest.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef std::string [TClientName](#)
Defines the object name type.
- typedef bool [TCheckResult](#)
Defines the message check result type.

Public Member Functions

- [~ClientConnectionRequest](#) (void)
Deletes the current memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory address contains a Client Connection Request message.
- [TClientName GetClientName](#) (void) const
Returns the object name.

Static Public Member Functions

- static [ClientConnectionRequest * GetNewClientConnectionRequest](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::TReceiverId](#) receiverId, const [TClientName](#) &clientName)
This function creates a new Client Connection Request message.

Private Types

- enum `HeaderValues` { `MessageType` = 0x0001, `MessageId` = 0x0001 }

Defines the header values for message recognition.

Private Member Functions

- `ClientConnectionRequest` (void)

Not used.

8.1.1 Detailed Description

This class implements the asynchronous client connection request message.

Definition at line 29 of file `ClientConnectionRequest.h`.

8.1.2 Member Typedef Documentation

8.1.2.1 typedef bool TerraSwarm::Asynchronous::ClientConnectionRequest::TCheckResult

Defines the message check result type.

Definition at line 50 of file `ClientConnectionRequest.h`.

8.1.2.2 typedef std::string TerraSwarm::Asynchronous::ClientConnectionRequest::TClientName

Defines the object name type.

Definition at line 45 of file `ClientConnectionRequest.h`.

8.1.3 Member Enumeration Documentation

8.1.3.1 enum TerraSwarm::Asynchronous::ClientConnectionRequest::CheckResultValues

Defines the values for `TCheckResult`.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 55 of file `ClientConnectionRequest.h`.

8.1.3.2 enum TerraSwarm::Asynchronous::ClientConnectionRequest::HeaderValues [private]

Defines the header values for message recognition.

Enumerator

MessageType

MessageId

Definition at line 35 of file ClientConnectionRequest.h.

8.1.4 Constructor & Destructor Documentation**8.1.4.1 TerraSwarm::Asynchronous::ClientConnectionRequest::ClientConnectionRequest (void) [private]**

Not used.

Made private to force the usage of the static method for creation.

Definition at line 15 of file ClientConnectionRequest.cpp.

8.1.4.2 TerraSwarm::Asynchronous::ClientConnectionRequest::~~ClientConnectionRequest (void)

Deletes the current memory.

Definition at line 19 of file ClientConnectionRequest.cpp.

8.1.5 Member Function Documentation**8.1.5.1 ClientConnectionRequest::TCheckResult TerraSwarm::Asynchronous::ClientConnectionRequest::CheckMessage (void) const**

Checks whether the current memory address contains a Client Connection Request message.

Cast the received memory to the pointer of this class and call this method for checking.

Returns

Check result of the memory.

Definition at line 41 of file ClientConnectionRequest.cpp.

8.1.5.2 ClientConnectionRequest::TClientName TerraSwarm::Asynchronous::ClientConnectionRequest::GetClientName (void) const

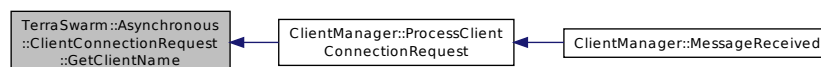
Returns the object name.

Returns

Name of the object within the message.

Definition at line 52 of file ClientConnectionRequest.cpp.

Here is the caller graph for this function:



8.1.5.3 **ClientConnectionRequest** * TerraSwarm::Asynchronous::ClientConnectionRequest::GetNewClientConnectionRequest
 (const MessageHeader::TSenderId *senderId*, const MessageHeader::TReceiverId *receiverId*, const
 TClientName & *clientName*) [static]

This function creates a new Client Connection Request message.

Parameters

<i>senderId</i>	Id of the client.
<i>receiverId</i>	Id of S2Sim.
<i>clientName</i>	Name of the object, the client is representing.

Returns

A new allocated message.

Warning

The deallocation is user's duty.

Definition at line 25 of file ClientConnectionRequest.cpp.

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

- S2Sim/TerraswarmLibrary/ClientConnectionRequest.h
- S2Sim/TerraswarmLibrary/ClientConnectionRequest.cpp

8.2 TerraSwarm::Synchronous::ClientConnectionRequest Class Reference

This class implements the synchronous client connection request message.

```
#include <ClientConnectionRequest.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef std::string [TClientName](#)
Defines the object name type.
- typedef bool [TCheckResult](#)
Defines the message check result type.

Public Member Functions

- [~ClientConnectionRequest](#) (void)
Deletes the current memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory address contains a Client Connection Request message.
- [TClientName GetClientName](#) (void) const
Returns the object name.

Static Public Member Functions

- static `ClientConnectionRequest * GetNewClientConnectionRequest` (const `MessageHeader::TSenderId` senderId, const `MessageHeader::TReceiverId` receiverId, const `TClientName` &clientName)

This function creates a new Client Connection Request message.

Private Types

- enum `HeaderValues` { `MessageType` = 0x0001, `MessageId` = 0x0004 }

Defines the header values for message recognition.

Private Member Functions

- `ClientConnectionRequest` (void)

Not used.

8.2.1 Detailed Description

This class implements the synchronous client connection request message.

Definition at line 113 of file `ClientConnectionRequest.h`.

8.2.2 Member Typedef Documentation

8.2.2.1 `typedef bool TerraSwarm::Synchronous::ClientConnectionRequest::TCheckResult`

Defines the message check result type.

Definition at line 134 of file `ClientConnectionRequest.h`.

8.2.2.2 `typedef std::string TerraSwarm::Synchronous::ClientConnectionRequest::TClientName`

Defines the object name type.

Definition at line 129 of file `ClientConnectionRequest.h`.

8.2.3 Member Enumeration Documentation

8.2.3.1 `enum TerraSwarm::Synchronous::ClientConnectionRequest::CheckResultValues`

Defines the values for `TCheckResult`.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 139 of file `ClientConnectionRequest.h`.

8.2.3.2 enum TerraSwarm::Synchronous::ClientConnectionRequest::HeaderValues [private]

Defines the header values for message recognition.

Enumerator

MessageType

MessageId

Definition at line 119 of file ClientConnectionRequest.h.

8.2.4 Constructor & Destructor Documentation

8.2.4.1 TerraSwarm::Synchronous::ClientConnectionRequest::ClientConnectionRequest (void) [private]

Not used.

Made private to force the usage of the static method for creation.

Definition at line 62 of file ClientConnectionRequest.cpp.

8.2.4.2 TerraSwarm::Synchronous::ClientConnectionRequest::~~ClientConnectionRequest (void)

Deletes the current memory.

Definition at line 66 of file ClientConnectionRequest.cpp.

8.2.5 Member Function Documentation

8.2.5.1 ClientConnectionRequest::TCheckResult TerraSwarm::Synchronous::ClientConnectionRequest::CheckMessage (void) const

Checks whether the current memory address contains a Client Connection Request message.

Cast the received memory to the pointer of this class and call this method for checking.

Returns

Check result of the memory.

Definition at line 88 of file ClientConnectionRequest.cpp.

8.2.5.2 ClientConnectionRequest::TClientName TerraSwarm::Synchronous::ClientConnectionRequest::GetClientName (void) const

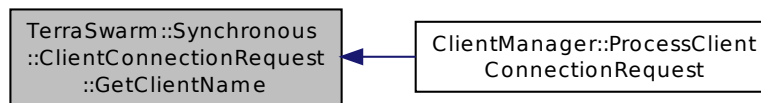
Returns the object name.

Returns

Name of the object within the message.

Definition at line 99 of file ClientConnectionRequest.cpp.

Here is the caller graph for this function:



8.2.5.3 ClientConnectionRequest * TerraSwarm::Synchronous::ClientConnectionRequest::GetNewClientConnectionRequest (const MessageHeader::TSenderId *senderId*, const MessageHeader::TReceiverId *receiverId*, const TClientName & *clientName*) [static]

This function creates a new Client Connection Request message.

Parameters

<i>senderId</i>	Id of the client.
<i>receiverId</i>	Id of S2Sim.
<i>clientName</i>	Name of the object, the client is representing.

Returns

A new allocated message.

Warning

The deallocation is user's duty.

Definition at line 72 of file ClientConnectionRequest.cpp.

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

- S2Sim/TerraswarmLibrary/[ClientConnectionRequest.h](#)
- S2Sim/TerraswarmLibrary/[ClientConnectionRequest.cpp](#)

8.3 TerraSwarm::Asynchronous::ClientConnectionResponse Class Reference

This class implements the Response message of the controller to the clients connection request.

```
#include <ClientConnectionResponse.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- enum [RequestResultValues](#) { [RequestAccepted](#) = (TRequestResult)0x00000000, [RequestObjectIdNotFound](#) = (TRequestResult)0x00000001 }
Defines values for the TRequestResult type.
- enum [SystemModeValues](#) { [SimulationMode](#) = (TSystemMode)0x0001, [RealTimeMode](#) = (TSystemMode)0x0002 }
Defines the values for the TSystemMode type.
- typedef bool [TCheckResult](#)
Defines the message check result type.
- typedef unsigned int [TRequestResult](#)
Defines the result of the request type.
- typedef unsigned int [TSystemTime](#)
Defines the time of the system in epoch format.
- typedef unsigned short [TNumberOfClients](#)
Defines the number of clients type.
- typedef unsigned short [TSystemMode](#)
Defines the current system working mode.

Public Member Functions

- [~ClientConnectionResponse](#) (void)
Deallocates the memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory is a [ClientConnectionResponse](#) message.
- [TRequestResult GetRequestResult](#) (void) const
Reads the request result field.
- [TSystemTime GetSystemTime](#) (void) const
Reads the system time field.
- [TNumberOfClients GetNumberOfClients](#) (void) const
Reads the number of clients field.
- [TSystemMode GetSystemMode](#) (void) const
Reads the system mode field.

Static Public Member Functions

- static [ClientConnectionResponse](#) * [GetNewClientConnectionResponse](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::TReceiverId](#) receiverId, const [TRequestResult](#) requestResult, const [TSystemTime](#) systemTime, const [TNumberOfClients](#) numberOfClients, const [TSystemMode](#) systemMode)
Creates a new [ClientConnectionResponse](#) message and allocates memory for it.
- static [TDataSize GetSize](#) (void)
Returns the size of a [ClientConnectionResponse](#) message.

Private Types

- enum [HeaderValues](#) { [MessageType](#) = 0x0001, [Messageld](#) = 0x0002 }
Message Header Values.
- enum [FieldSizeValues](#) {
 [RequestResultSize](#) = sizeof(TRequestResult), [SystemTimeSize](#) = sizeof(TSystemTime), [NumberOfClientsSize](#)
 = sizeof(TNumberOfClients), [SystemModeSize](#) = sizeof(TSystemMode),
 [TotalSize](#) = (RequestResultSize + SystemTimeSize + NumberOfClientsSize + SystemModeSize) }
Size of the fields within the message.
- enum [FieldIndexValues](#) { [RequestResultIndex](#) = MessageHeader::MessageHeaderSize, [SystemTimeIndex](#) =
 RequestResultIndex + RequestResultSize, [NumberOfClientsIndex](#) = SystemTimeIndex + SystemTimeSize,
 [SystemModelIndex](#) = NumberOfClientsIndex + NumberOfClientsSize }
Index of the fields within the message.
- typedef [NetworkByteAccessor](#)
 < [RequestResultIndex](#),
 [RequestResultSize](#) > [TRequestResultAccessor](#)
Accessor helper for the RequestResult field.
- typedef [NetworkByteAccessor](#)
 < [SystemTimeIndex](#),
 [SystemTimeSize](#) > [TSystemTimeAccessor](#)
Accessor helper for the SystemTime field.
- typedef [NetworkByteAccessor](#)
 < [NumberOfClientsIndex](#),
 [NumberOfClientsSize](#) > [TNumberOfClientsAccessor](#)
Accessor helper for the NumberOfClients field.
- typedef [NetworkByteAccessor](#)
 < [SystemModelIndex](#),
 [SystemModeSize](#) > [TSystemModeAccessor](#)
Accessor helper for the SystemMode field.

Private Member Functions

- [ClientConnectionResponse](#) (void)
Private constructor to force the usage of the static creation method.

8.3.1 Detailed Description

This class implements the Response message of the controller to the clients connection request.

Definition at line 22 of file ClientConnectionResponse.h.

8.3.2 Member Typedef Documentation

8.3.2.1 typedef bool TerraSwarm::Asynchronous::ClientConnectionResponse::TCheckResult

Defines the message check result type.

Definition at line 38 of file ClientConnectionResponse.h.

8.3.2.2 `typedef unsigned short TerraSwarm::Asynchronous::ClientConnectionResponse::TNumberOfClients`

Defines the number of clients type.

Definition at line 71 of file ClientConnectionResponse.h.

8.3.2.3 `typedef NetworkByteAccessor<NumberOfClientsIndex, NumberOfClientsSize> TerraSwarm::Asynchronous::ClientConnectionResponse::TNumberOfClientsAccessor [private]`

Accessor helper for the NumberOfClients field.

Definition at line 124 of file ClientConnectionResponse.h.

8.3.2.4 `typedef unsigned int TerraSwarm::Asynchronous::ClientConnectionResponse::TRequestResult`

Defines the result of the request type.

Definition at line 52 of file ClientConnectionResponse.h.

8.3.2.5 `typedef NetworkByteAccessor<RequestResultIndex, RequestResultSize> TerraSwarm::Asynchronous::ClientConnectionResponse::TRequestResultAccessor [private]`

Accessor helper for the RequestResult field.

Definition at line 114 of file ClientConnectionResponse.h.

8.3.2.6 `typedef unsigned short TerraSwarm::Asynchronous::ClientConnectionResponse::TSystemMode`

Defines the current system working mode.

Definition at line 76 of file ClientConnectionResponse.h.

8.3.2.7 `typedef NetworkByteAccessor<SystemModelIndex, SystemModeSize> TerraSwarm::Asynchronous::ClientConnectionResponse::TSystemModeAccessor [private]`

Accessor helper for the SystemMode field.

Definition at line 129 of file ClientConnectionResponse.h.

8.3.2.8 `typedef unsigned int TerraSwarm::Asynchronous::ClientConnectionResponse::TSystemTime`

Defines the time of the system in epoch format.

Definition at line 66 of file ClientConnectionResponse.h.

8.3.2.9 `typedef NetworkByteAccessor<SystemTimeIndex, SystemTimeSize> TerraSwarm::Asynchronous::ClientConnectionResponse::TSystemTimeAccessor [private]`

Accessor helper for the SystemTime field.

Definition at line 119 of file ClientConnectionResponse.h.

8.3.3 Member Enumeration Documentation

8.3.3.1 enum TerraSwarm::Asynchronous::ClientConnectionResponse::CheckResultValues

Defines the values for TCheckResult.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 43 of file ClientConnectionResponse.h.

8.3.3.2 enum TerraSwarm::Asynchronous::ClientConnectionResponse::FieldIndexValues [private]

Index of the fields within the message.

Enumerator

RequestResultIndex

SystemTimeIndex

NumberOfClientsIndex

SystemModelIndex

Definition at line 103 of file ClientConnectionResponse.h.

8.3.3.3 enum TerraSwarm::Asynchronous::ClientConnectionResponse::FieldSizeValues [private]

Size of the fields within the message.

Enumerator

RequestResultSize

SystemTimeSize

NumberOfClientsSize

SystemModeSize

TotalSize

Definition at line 91 of file ClientConnectionResponse.h.

8.3.3.4 enum TerraSwarm::Asynchronous::ClientConnectionResponse::HeaderValues [private]

Message Header Values.

Enumerator

MessageType

MessageId

Definition at line 28 of file ClientConnectionResponse.h.

8.3.3.5 enum TerraSwarm::Asynchronous::ClientConnectionResponse::RequestResultValues

Defines values for the TRequestResult type.

Enumerator

RequestAccepted The request is accepted.

RequestObjectIdNotFound The requested object name is not found and rejected.

Definition at line 57 of file ClientConnectionResponse.h.

8.3.3.6 enum TerraSwarm::Asynchronous::ClientConnectionResponse::SystemModeValues

Defines the values for the TSystemMode type.

Enumerator

SimulationMode Simulation working mode where the system is started artificially.

RealTimeMode Real time working mode where the system is working in real time. Not implemented yet.

Definition at line 81 of file ClientConnectionResponse.h.

8.3.4 Constructor & Destructor Documentation

8.3.4.1 TerraSwarm::Asynchronous::ClientConnectionResponse::ClientConnectionResponse (void) [private]

Private constructor to force the usage of the static creation method.

Definition at line 15 of file ClientConnectionResponse.cpp.

8.3.4.2 TerraSwarm::Asynchronous::ClientConnectionResponse::~~ClientConnectionResponse (void)

Deallocates the memory.

Definition at line 19 of file ClientConnectionResponse.cpp.

8.3.5 Member Function Documentation

8.3.5.1 ClientConnectionResponse::TCheckResult TerraSwarm::Asynchronous::ClientConnectionResponse::CheckMessage (void) const

Checks whether the current memory is a [ClientConnectionResponse](#) message.

Returns

Result of the check.

Definition at line 44 of file ClientConnectionResponse.cpp.

8.3.5.2 ClientConnectionResponse * TerraSwarm::Asynchronous::ClientConnectionResponse::GetNewClientConnectionResponse (const MessageHeader::TSenderId *senderId*, const MessageHeader::TReceiverId *receiverId*, const TRequestResult *requestResult*, const TSystemTime *systemTime*, const TNumberOfClients *numberOfClients*, const TSystemMode *systemMode*) [static]

Creates a new [ClientConnectionResponse](#) message and allocates memory for it.

Warning

Deallocation is the responsibility of the user.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.
<i>requestResult</i>	Result of the connection request.
<i>systemTime</i>	Current system time in epoch format.
<i>numberOfClients</i>	Number of Clients in the system.
<i>systemMode</i>	System working mode.

Returns

Newly allocated message pointer.

Definition at line 25 of file ClientConnectionResponse.cpp.

8.3.5.3 ClientConnectionResponse::TNumberOfClients TerraSwarm::Asynchronous::ClientConnectionResponse::GetNumberOfClients (void) const

Reads the number of clients field.

Returns

Number of clients in the system.

Definition at line 71 of file ClientConnectionResponse.cpp.

8.3.5.4 ClientConnectionResponse::TRequestResult TerraSwarm::Asynchronous::ClientConnectionResponse::GetRequestResult (void) const

Reads the request result field.

Returns

Request result value.

Definition at line 55 of file ClientConnectionResponse.cpp.

8.3.5.5 TDataSize TerraSwarm::Asynchronous::ClientConnectionResponse::GetSize (void) [static]

Returns the size of a [ClientConnectionResponse](#) message.

Returns

Size of the [ClientConnectionResponse](#) message.

Definition at line 87 of file ClientConnectionResponse.cpp.

8.3.5.6 [ClientConnectionResponse::TSystemMode](#) TerraSwarm::Asynchronous::ClientConnectionResponse::GetSystemMode (void) const

Reads the system mode field.

Returns

Current system working mode.

Definition at line 79 of file ClientConnectionResponse.cpp.

8.3.5.7 [ClientConnectionResponse::TSystemTime](#) TerraSwarm::Asynchronous::ClientConnectionResponse::GetSystemTime (void) const

Reads the system time field.

Returns

System time value.

Definition at line 63 of file ClientConnectionResponse.cpp.

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

- S2Sim/TerraswarmLibrary/[ClientConnectionResponse.h](#)
- S2Sim/TerraswarmLibrary/[ClientConnectionResponse.cpp](#)

8.4 TerraSwarm::Synchronous::ClientConnectionResponse Class Reference

This class implements the Response message of the controller to the clients connection request.

```
#include <ClientConnectionResponse.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- enum [RequestResultValues](#) { [RequestAccepted](#) = (TRequestResult)0x00000000, [RequestObjectIdNotFound](#) = (TRequestResult)0x00000001 }
Defines values for the TRequestResult type.
- enum [SystemModeValues](#) { [SimulationMode](#) = (TSystemMode)0x0001, [RealTimeMode](#) = (TSystemMode)0x0002 }
Defines the values for the TSystemMode type.
- typedef bool [TCheckResult](#)
Defines the message check result type.

- typedef unsigned int [TRequestResult](#)
Defines the result of the request type.
- typedef unsigned int [TSystemTime](#)
Defines the time of the system in epoch format.
- typedef unsigned short [TNumberOfClients](#)
Defines the number of clients type.
- typedef unsigned short [TSystemMode](#)
Defines the current system working mode.

Public Member Functions

- [~ClientConnectionResponse](#) (void)
Deallocates the memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory holds a [ClientConnectionResponse](#) message.
- [TRequestResult GetRequestResult](#) (void) const
Reads the RequestResult field.
- [TSystemTime GetSystemTime](#) (void) const
Reads the SystemTime field.
- [TNumberOfClients GetNumberOfClients](#) (void) const
Reads the NumberOfClients field.
- [TSystemMode GetSystemMode](#) (void) const
Reads the SystemMode field.

Static Public Member Functions

- static [ClientConnectionResponse](#) * [GetNewClientConnectionResponse](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::TReceiverId](#) receiverId, const [TRequestResult](#) requestResult, const [TSystemTime](#) systemTime, const [TNumberOfClients](#) numberOfClients, const [TSystemMode](#) systemMode)
Creates a new [ClientConnectionResponse](#) message and allocates memory for it.
- static [TDataSize GetSize](#) (void)
Returns the size of a [ClientConnectionResponse](#) message.

Private Types

- enum [HeaderValues](#) { [MessageType](#) = 0x0001, [MessageId](#) = 0x0005 }
Message Header values.
- enum [FieldSizeValues](#) {
 [RequestResultSize](#) = sizeof([TRequestResult](#)), [SystemTimeSize](#) = sizeof([TSystemTime](#)), [NumberOfClientsSize](#)
 = sizeof([TNumberOfClients](#)), [SystemModeSize](#) = sizeof([TSystemMode](#)),
 [TotalSize](#) = ([RequestResultSize](#) + [SystemTimeSize](#) + [NumberOfClientsSize](#) + [SystemModeSize](#)) }
Size of the message fields.
- enum [FieldIndexValues](#) { [RequestResultIndex](#) = [MessageHeader::MessageHeaderSize](#), [SystemTimeIndex](#) =
 [RequestResultIndex](#) + [RequestResultSize](#), [NumberOfClientsIndex](#) = [SystemTimeIndex](#) + [SystemTimeSize](#),
 [SystemModeIndex](#) = [NumberOfClientsIndex](#) + [NumberOfClientsSize](#) }
Index of the message fields.

- typedef [NetworkByteAccessor](#)
 < [RequestResultIndex](#),
[RequestResultSize](#) > [TRequestResultAccessor](#)
Accessor helper for the RequestResult field.
- typedef [NetworkByteAccessor](#)
 < [SystemTimeIndex](#),
[SystemTimeSize](#) > [TSystemTimeAccessor](#)
Accessor helper for the SystemTime field.
- typedef [NetworkByteAccessor](#)
 < [NumberOfClientsIndex](#),
[NumberOfClientsSize](#) > [TNumberOfClientsAccessor](#)
Accessor helper for the NumberOfClients field.
- typedef [NetworkByteAccessor](#)
 < [SystemModelIndex](#),
[SystemModeSize](#) > [TSystemModeAccessor](#)
Accessor helper for the SystemMode field.

Private Member Functions

- [ClientConnectionResponse](#) (void)
Private constructor to force the usage of the static construction method.

8.4.1 Detailed Description

This class implements the Response message of the controller to the clients connection request.
 Definition at line 220 of file ClientConnectionResponse.h.

8.4.2 Member Typedef Documentation

8.4.2.1 typedef bool TerraSwarm::Synchronous::ClientConnectionResponse::TCheckResult

Defines the message check result type.
 Definition at line 236 of file ClientConnectionResponse.h.

8.4.2.2 typedef unsigned short TerraSwarm::Synchronous::ClientConnectionResponse::TNumberOfClients

Defines the number of clients type.
 Definition at line 269 of file ClientConnectionResponse.h.

8.4.2.3 typedef NetworkByteAccessor<NumberOfClientsIndex, NumberOfClientsSize> TerraSwarm::Synchronous::ClientConnectionResponse::TNumberOfClientsAccessor [private]

Accessor helper for the NumberOfClients field.
 Definition at line 322 of file ClientConnectionResponse.h.

8.4.2.4 typedef unsigned int TerraSwarm::Synchronous::ClientConnectionResponse::TRequestResult

Defines the result of the request type.

Definition at line 250 of file ClientConnectionResponse.h.

8.4.2.5 typedef NetworkByteAccessor<RequestResultIndex, RequestResultSize> TerraSwarm::Synchronous::ClientConnectionResponse::TRequestResultAccessor [private]

Accessor helper for the RequestResult field.

Definition at line 312 of file ClientConnectionResponse.h.

8.4.2.6 typedef unsigned short TerraSwarm::Synchronous::ClientConnectionResponse::TSystemMode

Defines the current system working mode.

Definition at line 274 of file ClientConnectionResponse.h.

8.4.2.7 typedef NetworkByteAccessor<SystemModelIndex, SystemModeSize> TerraSwarm::Synchronous::ClientConnectionResponse::TSystemModeAccessor [private]

Accessor helper for the SystemMode field.

Definition at line 327 of file ClientConnectionResponse.h.

8.4.2.8 typedef unsigned int TerraSwarm::Synchronous::ClientConnectionResponse::TSystemTime

Defines the time of the system in epoch format.

Definition at line 264 of file ClientConnectionResponse.h.

8.4.2.9 typedef NetworkByteAccessor<SystemTimeIndex, SystemTimeSize> TerraSwarm::Synchronous::ClientConnectionResponse::TSystemTimeAccessor [private]

Accessor helper for the SystemTime field.

Definition at line 317 of file ClientConnectionResponse.h.

8.4.3 Member Enumeration Documentation

8.4.3.1 enum TerraSwarm::Synchronous::ClientConnectionResponse::CheckResultValues

Defines the values for TCheckResult.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 241 of file ClientConnectionResponse.h.

8.4.3.2 enum TerraSwarm::Synchronous::ClientConnectionResponse::FieldIndexValues [private]

Index of the message fields.

Enumerator

RequestResultIndex
SystemTimeIndex
NumberOfClientsIndex
SystemModelIndex

Definition at line 301 of file ClientConnectionResponse.h.

8.4.3.3 enum TerraSwarm::Synchronous::ClientConnectionResponse::FieldSizeValues [private]

Size of the message fields.

Enumerator

RequestResultSize
SystemTimeSize
NumberOfClientsSize
SystemModeSize
TotalSize

Definition at line 289 of file ClientConnectionResponse.h.

8.4.3.4 enum TerraSwarm::Synchronous::ClientConnectionResponse::HeaderValues [private]

Message Header values.

Enumerator

MessageType
MessageId

Definition at line 226 of file ClientConnectionResponse.h.

8.4.3.5 enum TerraSwarm::Synchronous::ClientConnectionResponse::RequestResultValues

Defines values for the TRequestResult type.

Enumerator

RequestAccepted The request is accepted.
RequestObjectIdNotFound The requested object name is not found and rejected.

Definition at line 255 of file ClientConnectionResponse.h.

8.4.3.6 enum TerraSwarm::Synchronous::ClientConnectionResponse::SystemModeValues

Defines the values for the TSystemMode type.

Enumerator

SimulationMode Simulation working mode where the system is started artificially.

RealTimeMode Real time working mode where the system is working in real time. Not implemented yet.

Definition at line 279 of file ClientConnectionResponse.h.

8.4.4 Constructor & Destructor Documentation

8.4.4.1 TerraSwarm::Synchronous::ClientConnectionResponse::ClientConnectionResponse (void) [private]

Private constructor to force the usage of the static construction method.

Definition at line 97 of file ClientConnectionResponse.cpp.

8.4.4.2 TerraSwarm::Synchronous::ClientConnectionResponse::~~ClientConnectionResponse (void)

Deallocates the memory.

Definition at line 101 of file ClientConnectionResponse.cpp.

8.4.5 Member Function Documentation

8.4.5.1 ClientConnectionResponse::TCheckResult TerraSwarm::Synchronous::ClientConnectionResponse::CheckMessage (void) const

Checks whether the current memory holds a [ClientConnectionResponse](#) message.

Returns

Result of the check.

Definition at line 126 of file ClientConnectionResponse.cpp.

8.4.5.2 ClientConnectionResponse * TerraSwarm::Synchronous::ClientConnectionResponse::GetNewClientConnectionResponse (const MessageHeader::TSenderId senderId, const MessageHeader::TReceiverId receiverId, const TRequestResult requestResult, const TSystemTime systemTime, const TNumberOfClients numberOfClients, const TSystemMode systemMode) [static]

Creates a new [ClientConnectionResponse](#) message and allocates memory for it.

Warning

Deallocation is the responsibility of the user.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.
<i>requestResult</i>	Result of the connection request.
<i>systemTime</i>	Current system time.
<i>numberOfClients</i>	Current number of clients.
<i>systemMode</i>	Current system working mode.

Returns

Newly allocated message pointer.

Definition at line 107 of file ClientConnectionResponse.cpp.

8.4.5.3 ClientConnectionResponse::TNumberOfClients TerraSwarm::Synchronous::ClientConnectionResponse::GetNumberOfClients (void) const

Reads the NumberOfClients field.

Returns

Number of clients in the system.

Definition at line 153 of file ClientConnectionResponse.cpp.

8.4.5.4 ClientConnectionResponse::TRequestResult TerraSwarm::Synchronous::ClientConnectionResponse::GetRequestResult (void) const

Reads the RequestResult field.

Returns

Result of the connection request.

Definition at line 137 of file ClientConnectionResponse.cpp.

8.4.5.5 TDataSize TerraSwarm::Synchronous::ClientConnectionResponse::GetSize (void) [static]

Returns the size of a [ClientConnectionResponse](#) message.

Returns

Size of the [ClientConnectionResponse](#) message.

Definition at line 169 of file ClientConnectionResponse.cpp.

8.4.5.6 ClientConnectionResponse::TSystemMode TerraSwarm::Synchronous::ClientConnectionResponse::GetSystemMode (void) const

Reads the SystemMode field.

Returns

The current system working mode.

Definition at line 161 of file ClientConnectionResponse.cpp.

8.4.5.7 ClientConnectionResponse::TSystemTime TerraSwarm::Synchronous::ClientConnectionResponse::GetSystemTime (void) const

Reads the SystemTime field.

Returns

Current system time.

Definition at line 145 of file ClientConnectionResponse.cpp.

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

- S2Sim/TerraswarmLibrary/[ClientConnectionResponse.h](#)
- S2Sim/TerraswarmLibrary/[ClientConnectionResponse.cpp](#)

8.5 TerraSwarm::Asynchronous::ClientData Class Reference

[Asynchronous](#) Client Data message from the client to indicate its consumption for a time interval.

```
#include <ClientData.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef bool [TCheckResult](#)
Defines the message check result type.
- typedef unsigned int [TStartTime](#)
Starting time of the sent data points.
- typedef unsigned int [TTimeResolution](#)
Time interval between consecutive data points.
- typedef unsigned int [TNumberOfDataPoints](#)
Number of data points in the message.
- typedef unsigned int [TDataPoint](#)
General type for a data point representing consumption for this case.

Public Member Functions

- [~ClientData](#) (void)
Deallocates the memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory contains a [ClientData](#) message.

- [TStartTime](#) [GetStartTime](#) (void) const
Reads the StartTime field.
- [TTimeResolution](#) [GetTimeResolution](#) (void) const
Reads the TimeResolution field.
- [TNumberOfDataPoints](#) [GetNumberOfDataPoints](#) (void) const
Reads the NumberOfDataPoints field.
- [TDataPoint](#) * [GetDataPoints](#) (void) const
Gets the pointer to the data points in the message.

Static Public Member Functions

- static [ClientData](#) * [GetNewClientData](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::TReceiverId](#) receiverId, const [TStartTime](#) startTime, const [TTimeResolution](#) timeResolution, const [TNumberOfDataPoints](#) numberOfDataPoints, [TDataPoint](#) *dataPoints)
Creates a new Asynchronous [ClientData](#) message and allocates memory for it.

Private Types

- enum [HeaderValues](#) { [MessageType](#) = 0x0001, [MessageId](#) = 0x0003 }
Message Header values.
- enum [FieldSizeValues](#) { [StartTimeSize](#) = sizeof([TStartTime](#)), [TimeResolutionSize](#) = sizeof([TTimeResolution](#)), [NumberOfDataPointsSize](#) = sizeof([TNumberOfDataPoints](#)), [DataPointSize](#) = sizeof([TDataPoint](#)) }
Size of the message fields.
- enum [FieldIndexValues](#) { [StartTimeIndex](#) = [MessageHeader::MessageHeaderSize](#), [TimeResolutionIndex](#) = [StartTimeIndex](#) + [StartTimeSize](#), [NumberOfDataPointsIndex](#) = [TimeResolutionIndex](#) + [TimeResolutionSize](#), [DataStartIndex](#) = [NumberOfDataPointsIndex](#) + [NumberOfDataPointsSize](#) }
Index of the message fields.
- typedef [NetworkByteAccessor](#) < [StartTimeIndex](#), [StartTimeSize](#) > [TStartTimeAccessor](#)
Accessor helper for StartTime field.
- typedef [NetworkByteAccessor](#) < [TimeResolutionIndex](#), [TimeResolutionSize](#) > [TTimeResolutionAccessor](#)
Accessor helper for TimeResolution field.
- typedef [NetworkByteAccessor](#) < [NumberOfDataPointsIndex](#), [NumberOfDataPointsSize](#) > [TNumberOfDataPointsAccessor](#)
Accessor helper for NumberOfDataPoints field.

Private Member Functions

- [ClientData](#) (void)
Private constructor to force the usage of the static construction method.

8.5.1 Detailed Description

Asynchronous Client Data message from the client to indicate its consumption for a time interval.

Definition at line 23 of file ClientData.h.

8.5.2 Member Typedef Documentation

8.5.2.1 typedef bool TerraSwarm::Asynchronous::ClientData::TCheckResult

Defines the message check result type.

Definition at line 39 of file ClientData.h.

8.5.2.2 typedef unsigned int TerraSwarm::Asynchronous::ClientData::TDataPoint

General type for a data point representing consumption for this case.

Definition at line 68 of file ClientData.h.

8.5.2.3 typedef unsigned int TerraSwarm::Asynchronous::ClientData::TNumberOfDataPoints

Number of data points in the message.

Definition at line 63 of file ClientData.h.

8.5.2.4 typedef NetworkByteAccessor<NumberOfDataPointsIndex, NumberOfDataPointsSize> TerraSwarm::Asynchronous::ClientData::TNumberOfDataPointsAccessor [private]

Accessor helper for NumberOfDataPoints field.

Definition at line 106 of file ClientData.h.

8.5.2.5 typedef unsigned int TerraSwarm::Asynchronous::ClientData::TStartTime

Starting time of the sent data points.

Definition at line 53 of file ClientData.h.

8.5.2.6 typedef NetworkByteAccessor<StartTimeIndex, StartTimeSize> TerraSwarm::Asynchronous::ClientData::TStartTimeAccessor [private]

Accessor helper for StartTime field.

Definition at line 96 of file ClientData.h.

8.5.2.7 typedef unsigned int TerraSwarm::Asynchronous::ClientData::TTimeResolution

Time interval between consecutive data points.

Definition at line 58 of file ClientData.h.

8.5.2.8 `typedef NetworkByteAccessor<TimeResolutionIndex, TimeResolutionSize>`
`TerraSwarm::Asynchronous::ClientData::TTimeResolutionAccessor` [private]

Accessor helper for TimeResolution field.

Definition at line 101 of file ClientData.h.

8.5.3 Member Enumeration Documentation

8.5.3.1 `enum TerraSwarm::Asynchronous::ClientData::CheckResultValues`

Defines the values for TCheckResult.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 44 of file ClientData.h.

8.5.3.2 `enum TerraSwarm::Asynchronous::ClientData::FieldIndexValues` [private]

Index of the message fields.

Enumerator

StartTimeIndex

TimeResolutionIndex

NumberOfDataPointsIndex

DataStartIndex

Definition at line 85 of file ClientData.h.

8.5.3.3 `enum TerraSwarm::Asynchronous::ClientData::FieldSizeValues` [private]

Size of the message fields.

Enumerator

StartTimeSize

TimeResolutionSize

NumberOfDataPointsSize

DataPointSize

Definition at line 74 of file ClientData.h.

8.5.3.4 enum TerraSwarm::Asynchronous::ClientData::HeaderValues [private]

Message Header values.

Enumerator

MessageType

MessageId

Definition at line 29 of file ClientData.h.

8.5.4 Constructor & Destructor Documentation

8.5.4.1 TerraSwarm::Asynchronous::ClientData::ClientData (void) [private]

Private constructor to force the usage of the static construction method.

Definition at line 15 of file ClientData.cpp.

8.5.4.2 TerraSwarm::Asynchronous::ClientData::~~ClientData (void)

Deallocates the memory.

Definition at line 19 of file ClientData.cpp.

8.5.5 Member Function Documentation

8.5.5.1 ClientData::TCheckResult TerraSwarm::Asynchronous::ClientData::CheckMessage (void) const

Checks whether the current memory contains a [ClientData](#) message.

Returns

Result of the check.

Definition at line 49 of file ClientData.cpp.

8.5.5.2 ClientData::TDataPoint * TerraSwarm::Asynchronous::ClientData::GetDataPoints (void) const

Gets the pointer to the data points in the message.

Returns

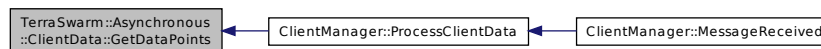
Pointer to the beginning of data points.

Definition at line 84 of file ClientData.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.5.5.3 ClientData * TerraSwarm::Asynchronous::ClientData::GetNewClientData (const MessageHeader::TSenderId senderId, const MessageHeader::TReceiverId receiverId, const TStartTime startTime, const TTimeResolution timeResolution, const TNumberOfDataPoints numberOfDataPoints, TDataPoint * dataPoints) [static]

Creates a new Asynchronous [ClientData](#) message and allocates memory for it.

Warning

Deallocation is the responsibility of the user.

Parameters

<i>senderId</i>	Id of the Sender.
<i>receiverId</i>	Id of the Receiver.
<i>startTime</i>	Starting time for the first consumption data.
<i>timeResolution</i>	Time interval between consecutive data points.
<i>numberOfDataPoints</i>	Number of data points in the message.
<i>dataPoints</i>	Pointer to the buffer holding the data points.

Returns

Newly allocated [ClientData](#) message.

Definition at line 25 of file ClientData.cpp.

8.5.5.4 ClientData::TNumberOfDataPoints TerraSwarm::Asynchronous::ClientData::GetNumberOfDataPoints (void) const

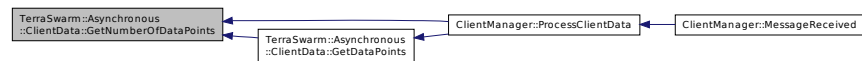
Reads the NumberOfDataPoints field.

Returns

Number of data points in the message.

Definition at line 76 of file ClientData.cpp.

Here is the caller graph for this function:



8.5.5.5 ClientData::TStartTime TerraSwarm::Asynchronous::ClientData::GetStartTime (void) const

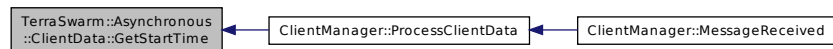
Reads the StartTime field.

Returns

StartTime value.

Definition at line 60 of file ClientData.cpp.

Here is the caller graph for this function:



8.5.5.6 ClientData::TTimeResolution TerraSwarm::Asynchronous::ClientData::GetTimeResolution (void) const

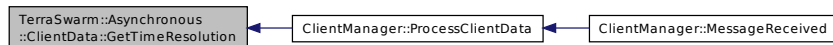
Reads the TimeResolution field.

Returns

TimeResolution value.

Definition at line 68 of file ClientData.cpp.

Here is the caller graph for this function:



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

- S2Sim/TerraswarmLibrary/[ClientData.h](#)
- S2Sim/TerraswarmLibrary/[ClientData.cpp](#)

8.6 TerraSwarm::Synchronous::ClientData Class Reference

[Synchronous](#) Client Data message from the client to indicate its consumption for a time interval.

```
#include <ClientData.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef bool [TCheckResult](#)
Defines the message check result type.
- typedef unsigned int [TDataPoint](#)
Defines the type for a general data point, consumption in this case.

Public Member Functions

- [~ClientData](#) (void)
Deallocates the memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory contains a [ClientData](#) message.
- [TDataPoint GetDataPoint](#) (void) const
Reads the data point in the message.

Static Public Member Functions

- static [ClientData](#) * [GetNewClientData](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::T-ReceiverId](#) receiverId, [TDataPoint](#) dataPoint)
Constructs a new [ClientData](#) message and allocates the necessary memory for it.

Private Types

- enum [HeaderValues](#) { [MessageType](#) = 0x0003, [MessageId](#) = 0x0005 }
Message Header Values.
- enum [FieldSizeValues](#) { [DataPointSize](#) = sizeof(TDataPoint) }
Size of the fields within the message.
- enum [FieldIndexValues](#) { [DataStartIndex](#) = MessageHeader::MessageHeaderSize }
Index of the fields within the message.
- typedef [NetworkByteAccessor](#) < [DataStartIndex](#), [DataPointSize](#) > [TDataPointAccessor](#)
Accessor helper for the DataPoint field.

Private Member Functions

- [ClientData](#) (void)

Private constructor to force the usage of the static construction method.

8.6.1 Detailed Description

[Synchronous](#) Client Data message from the client to indicate its consumption for a time interval.

Definition at line 188 of file ClientData.h.

8.6.2 Member Typedef Documentation

8.6.2.1 typedef bool TerraSwarm::Synchronous::ClientData::TCheckResult

Defines the message check result type.

Definition at line 204 of file ClientData.h.

8.6.2.2 typedef unsigned int TerraSwarm::Synchronous::ClientData::TDataPoint

Defines the type for a general data point, consumption in this case.

Definition at line 218 of file ClientData.h.

8.6.2.3 typedef NetworkByteAccessor<DataStartIndex, DataPointSize> TerraSwarm::Synchronous::ClientData::TDataPointAccessor [private]

Accessor helper for the DataPoint field.

Definition at line 240 of file ClientData.h.

8.6.3 Member Enumeration Documentation

8.6.3.1 enum TerraSwarm::Synchronous::ClientData::CheckResultValues

Defines the values for TCheckResult.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 209 of file ClientData.h.

8.6.3.2 enum TerraSwarm::Synchronous::ClientData::FieldIndexValues [private]

Index of the fields within the message.

Enumerator

DataStartIndex

Definition at line 232 of file ClientData.h.

8.6.3.3 enum **TerraSwarm::Synchronous::ClientData::FieldSizeValues** [private]

Size of the fields within the message.

Enumerator

DataPointSize

Definition at line 224 of file ClientData.h.

8.6.3.4 enum **TerraSwarm::Synchronous::ClientData::HeaderValues** [private]

Message Header Values.

Enumerator

MessageType

MessageId

Definition at line 194 of file ClientData.h.

8.6.4 Constructor & Destructor Documentation

8.6.4.1 TerraSwarm::Synchronous::ClientData::ClientData (void) [private]

Private constructor to force the usage of the static construction method.

Definition at line 97 of file ClientData.cpp.

8.6.4.2 TerraSwarm::Synchronous::ClientData::~~ClientData (void)

Deallocates the memory.

Definition at line 101 of file ClientData.cpp.

8.6.5 Member Function Documentation

8.6.5.1 ClientData::TCheckResult TerraSwarm::Synchronous::ClientData::CheckMessage (void) const

Checks whether the current memory contains a [ClientData](#) message.

Returns

Result of the check.

Definition at line 122 of file ClientData.cpp.

8.6.5.2 ClientData::TDataPoint TerraSwarm::Synchronous::ClientData::GetDataPoint (void) const

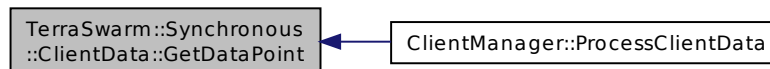
Reads the data point in the message.

Returns

Data point value.

Definition at line 133 of file ClientData.cpp.

Here is the caller graph for this function:



8.6.5.3 ClientData * TerraSwarm::Synchronous::ClientData::GetNewClientData (const MessageHeader::TSenderId senderId, const MessageHeader::TReceiverId receiverId, TDataPoint dataPoint) [static]

Constructs a new [ClientData](#) message and allocates the necessary memory for it.

Warning

Deallocation is the responsibility of the user.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.
<i>dataPoint</i>	Consumption of the user for the next time interval.

Returns

Newly allocated [ClientData](#) message.

Definition at line 107 of file ClientData.cpp.

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

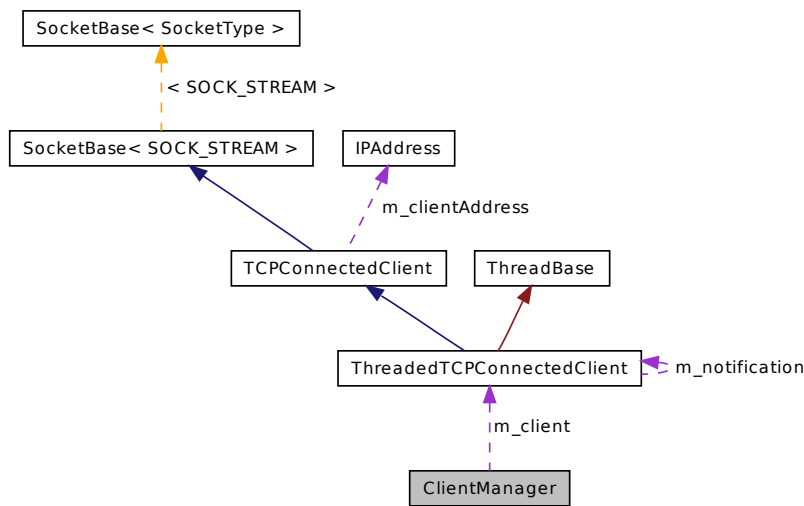
- S2Sim/TerraswarmLibrary/[ClientData.h](#)
- S2Sim/TerraswarmLibrary/[ClientData.cpp](#)

8.7 ClientManager Class Reference

Manages the connection with a client.

```
#include <ClientManager.h>
```

Collaboration diagram for ClientManager:



Public Types

- typedef
`Synchronous::SetCurrentPrice::TPrice TPrice`
Type used to represent a price signal.
- typedef
`Synchronous::SetCurrentPrice::TInterval TInterval`
Type used to represent a time interval.

Public Member Functions

- `ClientManager` (void)
Default constructor.
- `ClientManager` (`ThreadedTCPConnectedClient` *client)
Constructs the client manager with a connection.
- `ClientManager` (const `ClientManager` ©)
Copy constructor.
- `~ClientManager` (void)
Destructor, deletes the connection.
- `ClientManager` & `operator=` (const `ClientManager` ©)
Copies the contents of another client manager.
- void `ConnectionBroken` (void)
Handles the broken connection case.
- void `MessageReceived` (void *data, const size_t dataSize)
Called when a message is received.
- void `SetCurrentPrice` (const `TPrice` price, const `TInterval` beginInterval, const `TInterval` endInterval)

- Sends a price signal to the client.*
 - void [PriceProposal](#) (const [TPrice](#) price, const [TInterval](#) beginInterval, const [TInterval](#) endInterval)
- Sends a price proposal to the client.*
 - bool [IsSynchronous](#) (void) const
- Method to check for Synchronous Client type.*
 - bool [IsAsynchronous](#) (void) const
- Method to check for Asynchronous Client type.*

Private Types

- enum [ClientTypeValues](#) { [AsynchronousClient](#) = ([TClientType](#))0x01, [SynchronousClient](#) = ([TClientType](#))0x02 }
- There are two client types.*
- typedef unsigned char [TClientType](#)
- Defines the client type.*
- typedef [MessageHeader::TId](#) [TId](#)
- Type of the Unique ID of the client in the system.*

Private Member Functions

- void [ProcessClientConnectionRequest](#) ([Asynchronous::ClientConnectionRequest](#) *data)
- Processes an asynchronous client connection request.*
- void [ProcessClientConnectionRequest](#) ([Synchronous::ClientConnectionRequest](#) *data)
- Processes an synchronous client connection request.*
- void [ProcessClientData](#) ([Asynchronous::ClientData](#) *data)
- Processes the consumption information.*
- void [ProcessClientData](#) ([Synchronous::ClientData](#) *data)
- Processes the consumption information.*
- void [ProcessGetPrice](#) ([Synchronous::GetPrice](#) *data)
- Processes the price signal request.*
- void [ProcessDemandNegotiation](#) ([Synchronous::DemandNegotiation](#) *data)
- Processes the demand negotiation response.*

Private Attributes

- [ThreadedTCPConnectedClient](#) * [m_client](#)
- Pointer to the TCP Client connection managing class instance.*
- [TId](#) [m_clientId](#)
- Actual assigned Unique ID of the managed client.*
- [TClientType](#) [m_clientType](#)
- Type of the managed client.*

Static Private Attributes

- static [TId](#) [nextClientId](#) = 1
- Static variable that holds the next Unique ID.*

8.7.1 Detailed Description

Manages the connection with a client.

This class manages the connection to a single client including message processing and message construction.

Definition at line 33 of file ClientManager.h.

8.7.2 Member Typedef Documentation

8.7.2.1 `typedef unsigned char ClientManager::TClientType` [private]

Defines the client type.

Definition at line 39 of file ClientManager.h.

8.7.2.2 `typedef MessageHeader::TId ClientManager::TId` [private]

Type of the Unique ID of the client in the system.

Definition at line 55 of file ClientManager.h.

8.7.2.3 `typedef Synchronous::SetCurrentPrice::TInterval ClientManager::TInterval`

Type used to represent a time interval.

Definition at line 65 of file ClientManager.h.

8.7.2.4 `typedef Synchronous::SetCurrentPrice::TPrice ClientManager::TPrice`

Type used to represent a price signal.

Definition at line 61 of file ClientManager.h.

8.7.3 Member Enumeration Documentation

8.7.3.1 `enum ClientManager::ClientTypeValues` [private]

There are two client types.

- Asynchronous client that connects, delivers its data and disconnects.
- Synchronous client that connects and delivers data in a time synchronous way.

Enumerator

AsynchronousClient Asynchronous client type.

SynchronousClient Synchronous client type.

Definition at line 46 of file ClientManager.h.

8.7.4 Constructor & Destructor Documentation

8.7.4.1 ClientManager::ClientManager (void)

Default constructor.

Initializes the variables.

Definition at line 12 of file ClientManager.cpp.

8.7.4.2 ClientManager::ClientManager (ThreadedTCPConnectedClient * client)

Constructs the client manager with a connection.

This constructor initializes the class and sets the connection handle to the accepted connection.

Parameters

<i>client</i>	Pointer to the accepted TCP connection.
---------------	---

Definition at line 18 of file ClientManager.cpp.

8.7.4.3 ClientManager::ClientManager (const ClientManager & copy)

Copy constructor.

Copies the contents of another client manager. Written to enable usage within stdlib containers.

Parameters

<i>copy</i>	Client manager to be copied.
-------------	------------------------------

Definition at line 26 of file ClientManager.cpp.

8.7.4.4 ClientManager::~ClientManager (void)

Destructor, deletes the connection.

The destructor deletes the accepted connection since it is not managed anymore.

Todo This should be replaced by a reference counting memory manager.

Definition at line 34 of file ClientManager.cpp.

8.7.5 Member Function Documentation

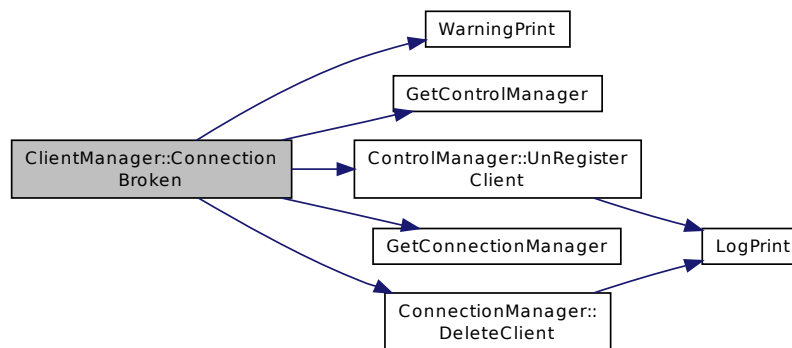
8.7.5.1 void ClientManager::ConnectionBroken (void)

Handles the broken connection case.

If the connection with the client is broken, the stored information should be deleted. This function is called by [ClientManager::m_client](#).

Definition at line 58 of file ClientManager.cpp.

Here is the call graph for this function:



8.7.5.2 `bool ClientManager::IsAsynchronous (void) const` `[inline]`

Method to check for Asynchronous Client type.

See Also

[m_clientType](#)

Returns

Returns whether the client is asynchronous.

Definition at line 252 of file `ClientManager.h`.

8.7.5.3 `bool ClientManager::IsSynchronous (void) const` `[inline]`

Method to check for Synchronous Client type.

See Also

[m_clientType](#)

Returns

Returns whether the client is synchronous.

Definition at line 241 of file `ClientManager.h`.

8.7.5.4 `void ClientManager::MessageReceived (void * data, const size_t dataSize)`

Called when a message is received.

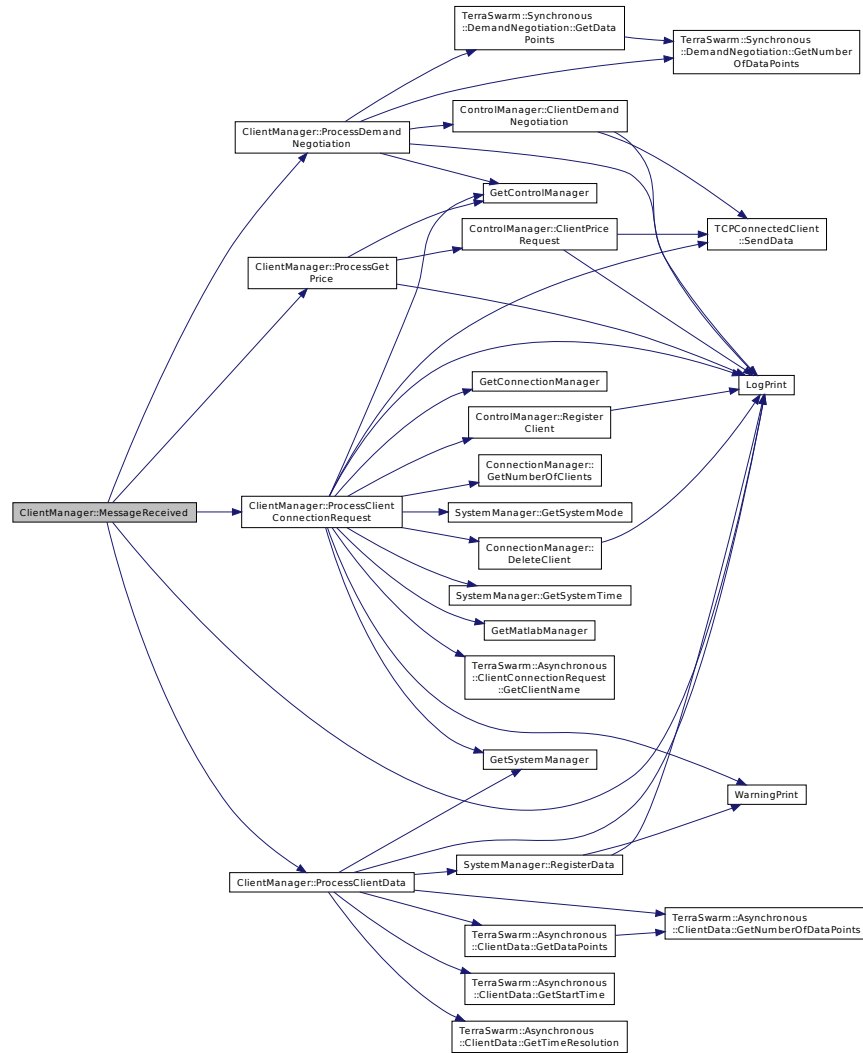
This function is called by `m_client` when a message is received. It checks for all possible message types and calls the necessary processing method.

Parameters

<i>data</i>	Pointer to the received data.
<i>dataSize</i>	Size of the received data.

Definition at line 70 of file ClientManager.cpp.

Here is the call graph for this function:



8.7.5.5 ClientManager & ClientManager::operator= (const ClientManager & copy)

Copies the contents of another client manager.

Overloads the equal operator and copies the content of another client manager.

Parameters

<i>copy</i>	Instance to be copied.
-------------	------------------------

Returns

Returns a reference to itself.

Definition at line 42 of file ClientManager.cpp.

8.7.5.6 void ClientManager::PriceProposal (const TPrice price, const TInterval beginInterval, const TInterval endInterval)

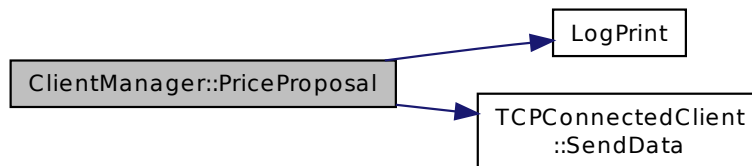
Sends a price proposal to the client.

Parameters

<i>price</i>	Proposed price.
<i>beginInterval</i>	Indicates the beginning time for the proposed price.
<i>endInterval</i>	Indicates the ending time for the proposed price.

Definition at line 250 of file ClientManager.cpp.

Here is the call graph for this function:



8.7.5.7 void ClientManager::ProcessClientConnectionRequest (Asynchronous::ClientConnectionRequest * data) [private]

Processes an asynchronous client connection request.

This function processes a received asynchronous client connection request. There are multiple steps in the acceptance procedure:

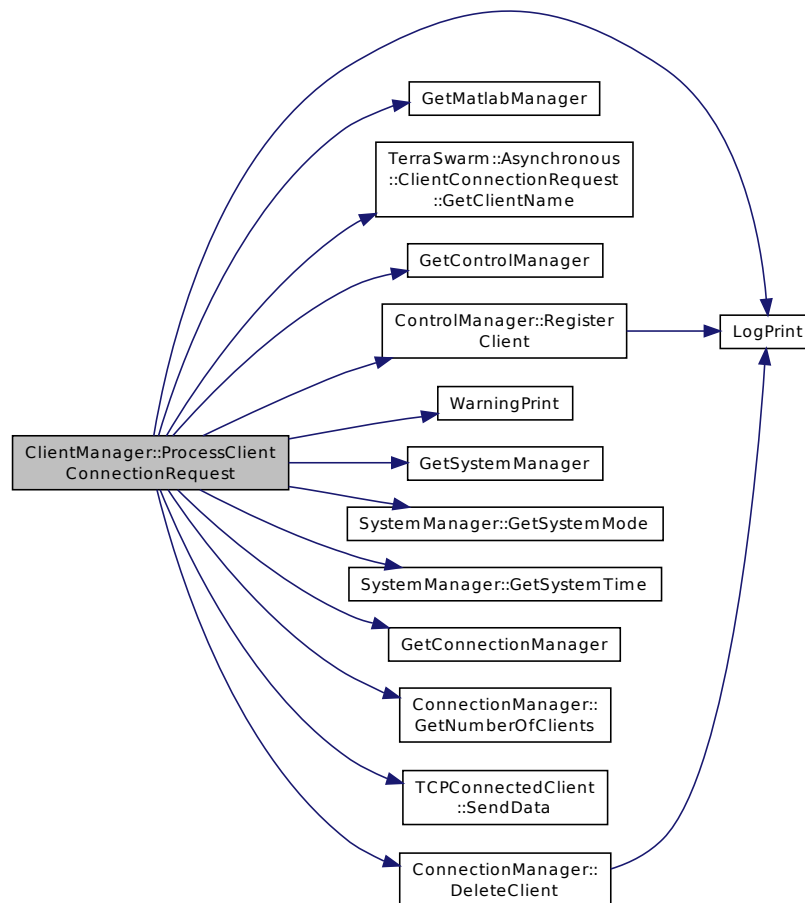
- Checking with OpenDSS whether the object actually exists.
- If the object exists, an acceptance message is sent back.
- Next client id is incremented.
- The client is registered to the [ControlManager](#).

Parameters

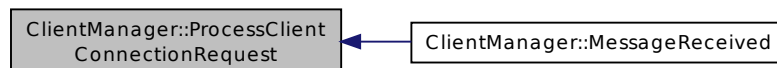
<i>data</i>	Received message structure in TerraSwarm::Asynchronous::ClientConnectionRequest .
-------------	---

Definition at line 108 of file ClientManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.5.8 `void ClientManager::ProcessClientConnectionRequest (Synchronous::ClientConnectionRequest * data)`
`[private]`

Processes an synchronous client connection request.

This function processes a received synchronous client connection request. There are multiple steps in the acceptance procedure:

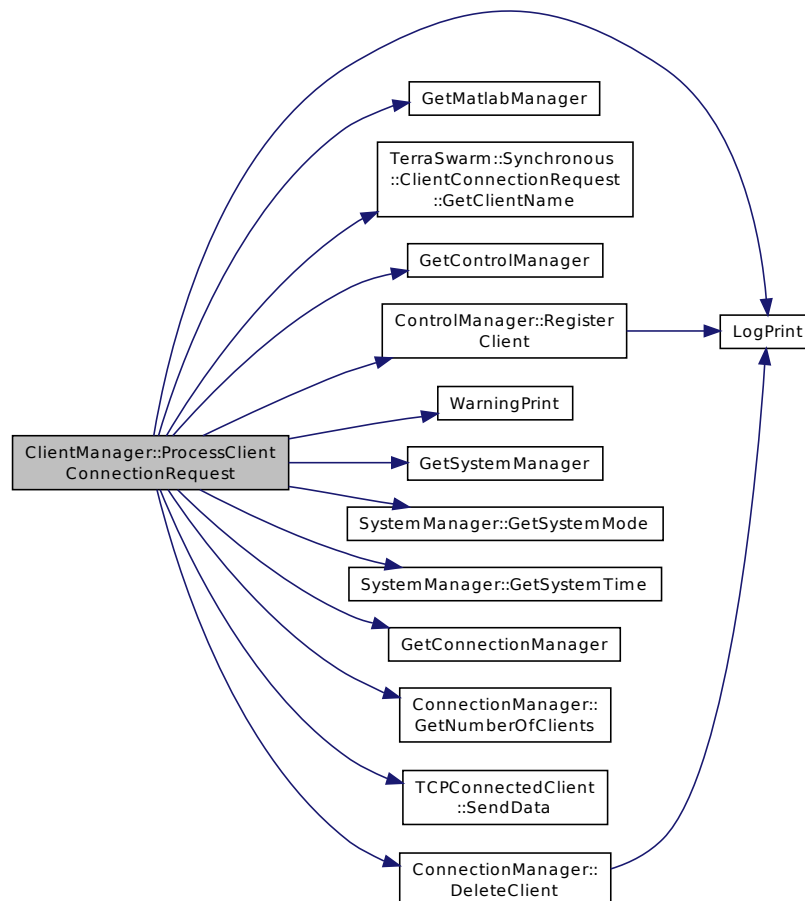
- Checking with OpenDSS whether the object actually exists.
- If the object exists, an acceptance message is sent back.
- Next client id is incremented.
- The client is registered to the [ControlManager](#).

Parameters

<i>data</i>	Received message structure in TerraSwarm::Synchronous::ClientConnectionRequest .
-------------	--

Definition at line 163 of file ClientManager.cpp.

Here is the call graph for this function:



8.7.5.9 void ClientManager::ProcessClientData (Asynchronous::ClientData * data) [private]

Processes the consumption information.

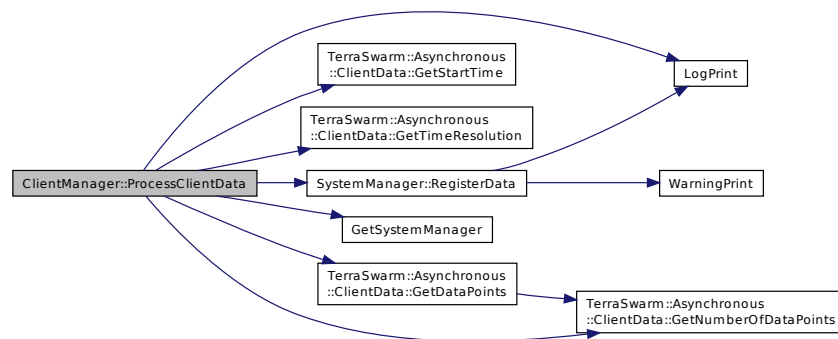
This function processes the received consumption information of the asynchronous client. The data is registered to the [SystemManager](#).

Parameters

<i>data</i>	Received message structure in TerraSwarm::Asynchronous::ClientData .
-------------	--

Definition at line 150 of file ClientManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.5.10 void ClientManager::ProcessClientData (Synchronous::ClientData * data) [private]

Processes the consumption information.

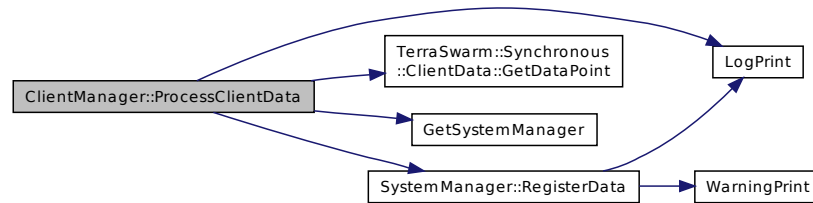
This function processes the received consumption information of the synchronous client. The data is registered to the [SystemManager](#).

Parameters

<i>data</i>	Received message structure in <code>TerraSwarmSynchronous::ClientData</code> .
-------------	--

Definition at line 204 of file ClientManager.cpp.

Here is the call graph for this function:



8.7.5.11 void ClientManager::ProcessDemandNegotiation (Synchronous::DemandNegotiation * *data*) [private]

Processes the demand negotiation response.

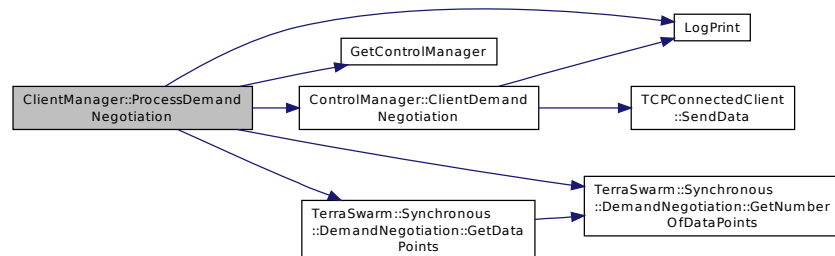
The client sends a demand negotiation response to the simulator. This response is sent to [ControlManager](#).

Parameters

<i>data</i>	Received message structure in TerraSwarm::Synchronous::DemandNegotiation .
-------------	--

Definition at line 223 of file ClientManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.5.12 void ClientManager::ProcessGetPrice (Synchronous::GetPrice * data) [private]

Processes the price signal request.

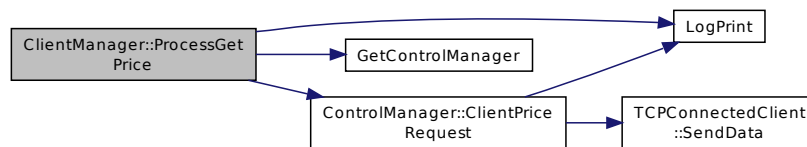
The price signal request by the synchronous client is sent to [ControlManager](#).

Parameters

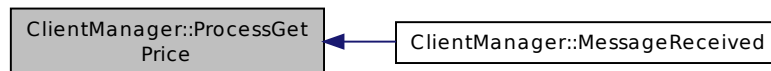
<i>data</i>	Received message structure in TerraSwarm::Synchronous::GetPrice .
-------------	---

Definition at line 214 of file ClientManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.5.13 void ClientManager::SetCurrentPrice (const TPrice price, const TInterval beginInterval, const TInterval endInterval)

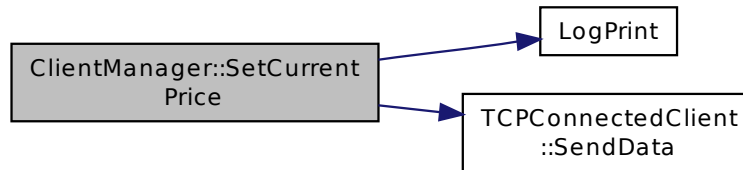
Sends a price signal to the client.

Parameters

<i>price</i>	Indicates the price.
<i>beginInterval</i>	Indicates the beginning time for the price.
<i>endInterval</i>	Indicates the ending time for the price.

Definition at line 234 of file ClientManager.cpp.

Here is the call graph for this function:



8.7.6 Member Data Documentation

8.7.6.1 ThreadedTCPConnectedClient* ClientManager::m_client [private]

Pointer to the TCP Client connection managing class instance.

This variable is used for all communication purposes.

Definition at line 75 of file ClientManager.h.

8.7.6.2 TId ClientManager::m_clientId [private]

Actual assigned Unique ID of the managed client.

Definition at line 79 of file ClientManager.h.

8.7.6.3 TClientType ClientManager::m_clientType [private]

Type of the managed client.

See Also

[{ClientManager::ClientTypeValues}](#)

Definition at line 83 of file ClientManager.h.

8.7.6.4 ClientManager::TId ClientManager::nextClientId = 1 [static], [private]

Static variable that holds the next Unique ID.

Incremented at each new connection. Returns to 0 if all values have been used.

Definition at line 71 of file ClientManager.h.

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

- [S2Sim/ClientManager.h](#)
- [S2Sim/ClientManager.cpp](#)

8.8 CompileCheck< expression, Reason > Class Template Reference

This class has two specializations.

```
#include <CompileTimeCheckerLibrary.h>
```

Public Types

- enum { **Result** = true }
Enumeration for Result displaying.

Static Public Member Functions

- static void **Check** (void)
Empty function for visual purposes.

8.8.1 Detailed Description

```
template<bool expression, typename Reason = class NoReason>class CompileCheck< expression, Reason >
```

This class has two specializations.

If the expression is true, the static **Check()** function will be public and compilation will continue.

Template Parameters

<i>expression</i>	This is a boolean expression that can be evaluated in compile time.
<i>Reason</i>	This can be the name of any class that will be used for debug information only.

Definition at line 19 of file CompileTimeCheckerLibrary.h.

8.8.2 Member Enumeration Documentation

8.8.2.1 `template<bool expression, typename Reason = class NoReason> anonymous enum`

Enumeration for Result displaying.

Enumerator

Result The check passed.

Definition at line 25 of file CompileTimeCheckerLibrary.h.

8.8.3 Member Function Documentation

8.8.3.1 `template<bool expression, typename Reason = class NoReason> static void CompileCheck< expression, Reason >::Check(void) [inline], [static]`

Empty function for visual purposes.

Definition at line 36 of file CompileTimeCheckerLibrary.h.

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

- S2Sim/TerraswarmLibrary/[CompileTimeCheckerLibrary.h](#)

8.9 CompileCheck< false, Reason > Class Template Reference

Second specialization of the class, when the expression is not true.

```
#include <CompileTimeCheckerLibrary.h>
```

Private Types

- enum { [Result](#) = true }
Enumeration for Result displaying.

Static Private Member Functions

- static void [Check](#) (void)
Empty function for visual purposes.

8.9.1 Detailed Description

```
template<typename Reason>class CompileCheck< false, Reason >
```

Second specialization of the class, when the expression is not true.

The static [Check\(\)](#) function is this time private and the compilation will fail.

Definition at line 43 of file CompileTimeCheckerLibrary.h.

8.9.2 Member Enumeration Documentation

```
8.9.2.1 template<typename Reason > anonymous enum [private]
```

Enumeration for Result displaying.

Enumerator

Result The check has failed.

Definition at line 49 of file CompileTimeCheckerLibrary.h.

8.9.3 Member Function Documentation

```
8.9.3.1 template<typename Reason > static void CompileCheck< false, Reason >::Check ( void ) [inline],  
[static], [private]
```

Empty function for visual purposes.

Definition at line 60 of file CompileTimeCheckerLibrary.h.

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

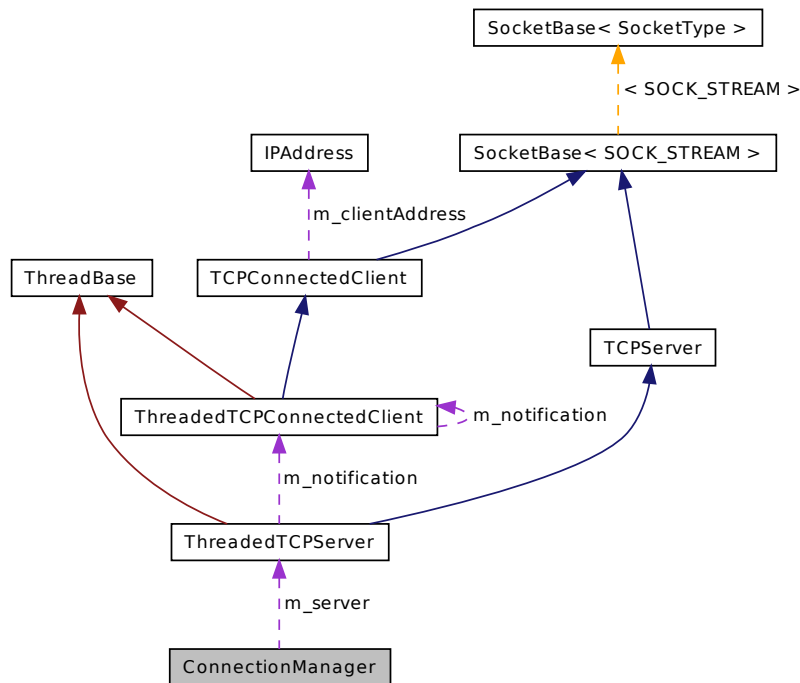
- S2Sim/TerraswarmLibrary/[CompileTimeCheckerLibrary.h](#)

8.10 ConnectionManager Class Reference

Manages connections to all clients.

```
#include <ConnectionManager.h>
```

Collaboration diagram for ConnectionManager:



Public Types

- typedef [TClientId](#) [TNumberOfClients](#)
Represents the type for number of clients.

Public Member Functions

- void [IncomingConnection](#) ([ThreadedTCPConnectedClient](#) *newClient)
Incoming connection handler.
- void [BrokenConnection](#) ([ThreadedTCPConnectedClient](#) *brokenClient)
Broken connection handler.
- void [IncomingMessage](#) ([ThreadedTCPConnectedClient](#) *senderClient, void *buffer, size_t size)
New message processor.

- void [DeleteClient](#) ([ClientManager](#) *clientManager)
Deletes the [ClientManager](#) from all lists and finally deletes itself.
- [TNumberOfClients GetNumberOfClients](#) (void) const
Returns the current number of clients connected.

Private Types

- typedef
[TerraSwarm::MessageHeader::TId TClientId](#)
Type representing the Unique Id of the client.
- typedef std::map
< [ThreadedTCPConnectedClient](#)
*, [ClientManager](#) * > [TClientList](#)
Type mapping TCP client connection pointers to their respective managers.
- typedef std::map
< [ClientManager](#)
*, [ThreadedTCPConnectedClient](#) * > [TReversedClientList](#)
Type mapping the client managers to their respective TCP client connections.
- typedef std::map< [TClientId](#),
[ClientManager](#) * > [TClientIdList](#)
Type mapping the client unique id's to respective managers.

Private Member Functions

- [ConnectionManager](#) (void)
Private constructor for singleton implementation.

Private Attributes

- [TClientList m_clientList](#)
Holds the Connection->Manager based list.
- [TReversedClientList m_reversedClientList](#)
Holds the Manager->Connection based list for speed.
- [ThreadedTCPServer m_server](#)
Instance of the TCP server.

Friends

- [ConnectionManager](#) & [GetConnectionManager](#) (void)
Friend method for singleton implementation.

8.10.1 Detailed Description

Manages connections to all clients.

This class manages the connections to all clients. It implements a TCP server and answers any connection attempts. It forwards received data to the correct [ClientManager](#) instance for processing.

Definition at line 48 of file [ConnectionManager.h](#).

8.10.2 Member Typedef Documentation

8.10.2.1 `typedef TerraSwarm::MessageHeader::TId ConnectionManager::TClientId` [private]

Type representing the Unique Id of the client.

Definition at line 62 of file ConnectionManager.h.

8.10.2.2 `typedef std::map<TClientId, ClientManager*> ConnectionManager::TClientIdList` [private]

Type mapping the client unique id's to respective managers.

Definition at line 77 of file ConnectionManager.h.

8.10.2.3 `typedef std::map<ThreadedTCPConnectedClient*, ClientManager*> ConnectionManager::TClientList` [private]

Type mapping TCP client connection pointers to their respective managers.

Definition at line 67 of file ConnectionManager.h.

8.10.2.4 `typedef TClientId ConnectionManager::TNumberOfClients`

Represents the type for number of clients.

Definition at line 83 of file ConnectionManager.h.

8.10.2.5 `typedef std::map<ClientManager*, ThreadedTCPConnectedClient*> ConnectionManager::TReversed-ClientList` [private]

Type mapping the client managers to their respective TCP client connections.

Definition at line 72 of file ConnectionManager.h.

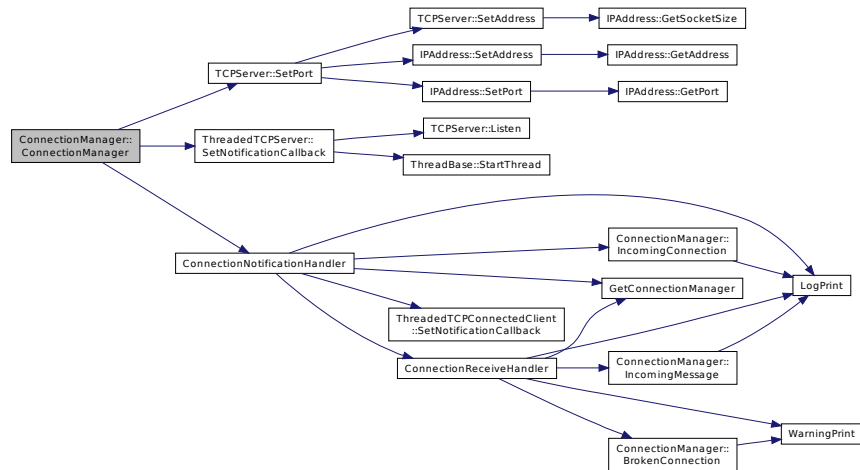
8.10.3 Constructor & Destructor Documentation

8.10.3.1 `ConnectionManager::ConnectionManager (void)` [private]

Private constructor for singleton implementation.

Definition at line 45 of file ConnectionManager.cpp.

Here is the call graph for this function:



8.10.4 Member Function Documentation

8.10.4.1 void ConnectionManager::BrokenConnection (ThreadedTCPConnectedClient * *brokenClient*)

Broken connection handler.

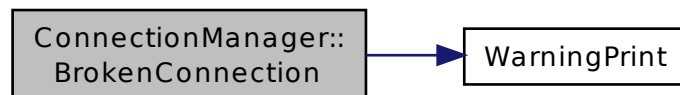
This method handles a broken connection to an existing client. The [ClientManager](#) is notified, leaving the handling to it.

Parameters

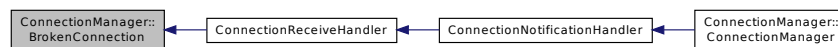
<i>brokenClient</i>	TCP Information of the broken connection.
---------------------	---

Definition at line 66 of file ConnectionManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.10.4.2 void ConnectionManager::DeleteClient (ClientManager * clientManager)

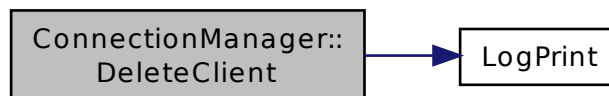
Deletes the [ClientManager](#) from all lists and finally deletes itself.

Parameters

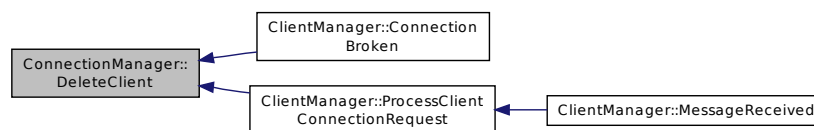
<i>clientManager</i>	ClientManager to be deleted.
----------------------	--

Definition at line 84 of file ConnectionManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.10.4.3 TNumberOfClients ConnectionManager::GetNumberOfClients (void) const [inline]

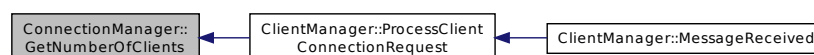
Returns the current number of clients connected.

Returns

Number of connected clients.

Definition at line 154 of file ConnectionManager.h.

Here is the caller graph for this function:



8.10.4.4 void ConnectionManager::IncomingConnection (ThreadedTCPConnectedClient * *newClient*)

Incoming connection handler.

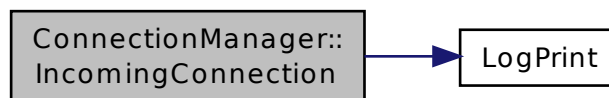
This method handles a new connection from a new client. A new [ClientManager](#) is created and the information is added to the lists for book-keeping.

Parameters

<i>newClient</i>	TCP Information of the new connection.
------------------	--

Definition at line 54 of file ConnectionManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.10.4.5 void ConnectionManager::IncomingMessage (ThreadedTCPConnectedClient * *senderClient*, void * *buffer*, size_t *size*)

New message processor.

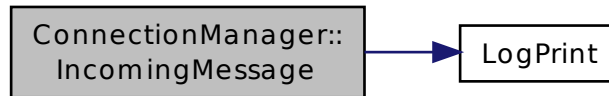
This method takes the received information and relays it to the respective [ClientManager](#).

Parameters

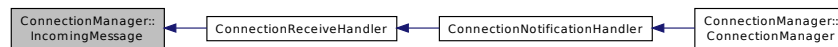
<i>senderClient</i>	TCP Information of the sender.
<i>buffer</i>	Buffer containing the received message.
<i>size</i>	Size of the received message.

Definition at line 75 of file ConnectionManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.10.5 Friends And Related Function Documentation

8.10.5.1 ConnectionManager& GetConnectionManager (void) [friend]

Friend method for singleton implementation.

Returns

Returns the reference to the [ConnectionManager](#).

Definition at line 11 of file `ConnectionManager.cpp`.

8.10.6 Member Data Documentation

8.10.6.1 TClientList ConnectionManager::m_clientList [private]

Holds the `Connection->Manager` based list.

Definition at line 89 of file `ConnectionManager.h`.

8.10.6.2 TReversedClientList ConnectionManager::m_reversedClientList [private]

Holds the `Manager->Connection` based list for speed.

Definition at line 94 of file `ConnectionManager.h`.

8.10.6.3 ThreadedTCPServer ConnectionManager::m_server [private]

Instance of the TCP server.

Definition at line 99 of file ConnectionManager.h.

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

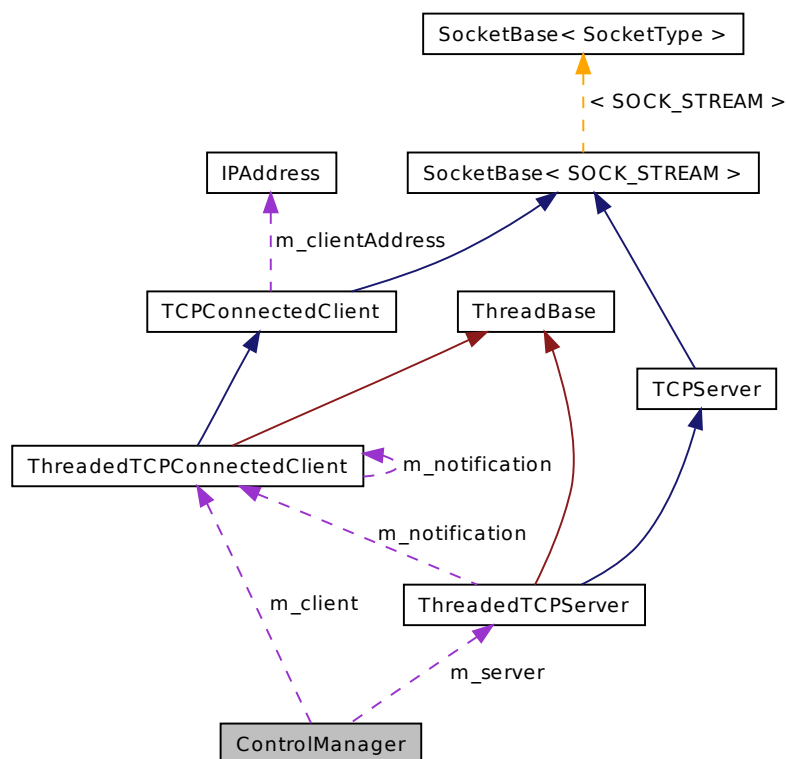
- S2Sim/ConnectionManager.h
- S2Sim/ConnectionManager.cpp

8.11 ControlManager Class Reference

Manages the connection with the External Controller.

```
#include <ControlManager.h>
```

Collaboration diagram for ControlManager:



Public Types

- typedef [Asynchronous::ClientData::TDataPoint TVoltage](#)
Defines the type of Voltage information.
- typedef [Asynchronous::ClientData::TDataPoint TWattage](#)

- Defines the type of Wattage/Power information.*

 - typedef `MessageHeader::TId TNumberOfClients`
- Defines the type of number of clients.*

 - typedef `MessageHeader::TId TClientId`
- Redefines the type for unique id for rapid development.*

 - typedef `Asynchronous::ClientConnectionRequest::TClientName TClientName`
- Redefines the type for object name for rapid development.*

 - typedef `Asynchronous::ClientData::TDataPoint TPrice`
- Defines the type for the price feedback signal.*

 - typedef `Asynchronous::ClientData::TDataPoint TDataPoint`
- Redefines the general data type for rapid development.*

 - typedef `Asynchronous::ClientData::TNumberOfDataPoints TNumberOfDataPoints`
- Redefines the number of data points type for rapid development.*

Public Member Functions

- void `SetClient` (`ThreadedTCPConnectedClient *client`)
Sets the connection information to the accepted External Controller.
- void `ProcessData` (`void *data`, `const size_t size`)
Processes the received message from the External Controller.
- void `MakeDecision` (`void`)
Starts the decision process by sending necessary information to External Controller.
- void `WaitUntilReady` (`void`)
Stops the calling thread until a ready signal is received from the External Controller.
- void `RegisterClient` (`const TClientId clientId`, `const TClientName &clientName`, `ClientManager *clientManager`)
Registers a new client to its maps for later referencing.
- void `UnRegisterClient` (`const TClientId clientId`)
Deletes a client from the maps when connection is broken.
- void `ClientPriceRequest` (`const TClientId clientId`)
Relays the price request of a client to the External Controller.
- void `ClientDemandNegotiation` (`const TClientId clientId`, `const TNumberOfDataPoints numberOfDataPoints`, `TDataPoint *dataPoints`)
Relays the demand negotiation response of the client to the External Controller.
- `TClientName GetClientName` (`const TClientId clientId`)
Returns the name of the object.

Private Types

- enum `MessageTypeValues` {
`MakeDecisionType` = 0x00000001, `DecisionFinishedType` = 0x00000002, `SetPriceType` = 0x00000003, `Send-PriceProposalType` = 0x00000004,
`PriceRequestType` = 0x00000005, `DemandNegotiationType` = 0x00000006 }
Defines the available values for different message types.
- typedef `size_t TDataSize`

- Defines the type for size of a data chunk.*
 - typedef unsigned int [TMessageType](#)
 - Defines the message type field used in the communication for message processing.*
- typedef std::map< [TClientId](#), [TClientName](#) > [TClientIdMap](#)
 - Defines the type holding a ClientId->ClientName mapping.*
- typedef std::map< [TClientId](#), [ClientManager](#) * > [TClientManagerMap](#)
 - Defines the type holding a ClientId->[ClientManager](#) mapping.*

Private Member Functions

- [ControlManager](#) (void)
 - Private constructor for singleton implementation.*

Private Attributes

- [ThreadedTCPServer](#) m_server
 - Implements the TCP server for external controller communication.*
- [ThreadedTCPConnectedClient](#) * m_client
 - Pointer to the currently accepted external controller communication handler.*
- Semaphore [m_readySemaphore](#)
 - Semaphore that is released when the frame is finished by the external controller.*
- [TClientIdMap](#) m_clientIdMap
 - Map containing the unique id to object name mapping.*
- [TClientManagerMap](#) m_clientManagerMap
 - Map containing the unique id to [ClientManager](#) mapping.*

Friends

- [ControlManager](#) & [GetControlManager](#) (void)
 - Friend method for singleton implementation.*

8.11.1 Detailed Description

Manages the connection with the External Controller.

This class manages the connection to the External Controller. All received commands are processed and necessary updates are sent back.

Definition at line 63 of file [ControlManager.h](#).

8.11.2 Member Typedef Documentation

8.11.2.1 typedef [MessageHeader::TId](#) [ControlManager::TClientId](#)

Redefines the type for unique id for rapid development.

Definition at line 92 of file [ControlManager.h](#).

8.11.2.2 `typedef std::map<TClientId, TClientName> ControlManager::TClientIdMap [private]`

Defines the type holding a ClientId->ClientName mapping.

Definition at line 141 of file ControlManager.h.

8.11.2.3 `typedef std::map<TClientId, ClientManager*> ControlManager::TClientManagerMap [private]`

Defines the type holding a ClientId->ClientManager mapping.

Definition at line 146 of file ControlManager.h.

8.11.2.4 `typedef Asynchronous::ClientConnectionRequest::TClientName ControlManager::TClientName`

Redefines the type for object name for rapid development.

Definition at line 97 of file ControlManager.h.

8.11.2.5 `typedef Asynchronous::ClientData::TDataPoint ControlManager::TDataPoint`

Redefines the general data type for rapid development.

Definition at line 107 of file ControlManager.h.

8.11.2.6 `typedef size_t ControlManager::TDataSize [private]`

Defines the type for size of a data chunk.

Definition at line 118 of file ControlManager.h.

8.11.2.7 `typedef unsigned int ControlManager::TMessageType [private]`

Defines the message type field used in the communication for message processing.

Definition at line 123 of file ControlManager.h.

8.11.2.8 `typedef MessageHeader::TId ControlManager::TNumberOfClients`

Defines the type of number of clients.

Definition at line 87 of file ControlManager.h.

8.11.2.9 `typedef Asynchronous::ClientData::TNumberOfDataPoints ControlManager::TNumberOfDataPoints`

Redefines the number of data points type for rapid development.

Definition at line 112 of file ControlManager.h.

8.11.2.10 `typedef Asynchronous::ClientData::TDataPoint ControlManager::TPrice`

Defines the type for the price feedback signal.

Definition at line 102 of file ControlManager.h.

8.11.2.11 typedef Asynchronous::ClientData::TDataPoint ControlManager::TVoltage

Defines the type of Voltage information.

Definition at line 77 of file ControlManager.h.

8.11.2.12 typedef Asynchronous::ClientData::TDataPoint ControlManager::TWattage

Defines the type of Wattage/Power information.

Definition at line 82 of file ControlManager.h.

8.11.3 Member Enumeration Documentation

8.11.3.1 enum ControlManager::MessageTypeValues [private]

Defines the available values for different message types.

Enumerator

MakeDecisionType Sent to external controller to indicate the beginning of a frame.

DecisionFinishedType Received from external controller for the end of a frame.

SetPriceType Sent by external controller to send a price signal to a client.

SendPriceProposalType Sent by external controller to send a price proposal to a client.

PriceRequestType Sent to external controller to indicate a price request by a client.

DemandNegotiationType Sent to external controller to indicate a price proposal response by a client.

Definition at line 128 of file ControlManager.h.

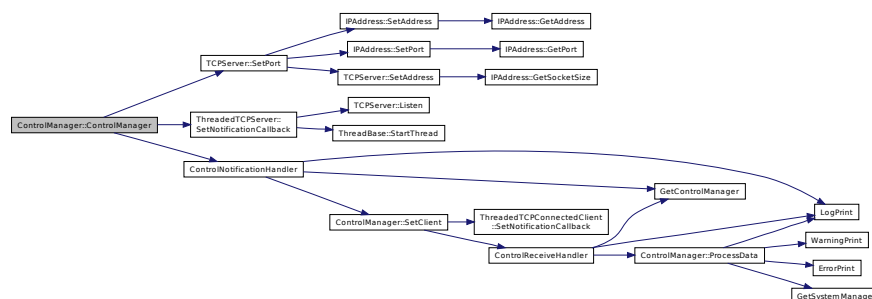
8.11.4 Constructor & Destructor Documentation

8.11.4.1 ControlManager::ControlManager(void) [private]

Private constructor for singleton implementation.

Definition at line 35 of file ControlManager.cpp.

Here is the call graph for this function:



8.11.5 Member Function Documentation

8.11.5.1 `void ControlManager::ClientDemandNegotiation (const TClientId clientId, const TNumberOfDataPoints numberOfDataPoints, TDataPoint * dataPoints)`

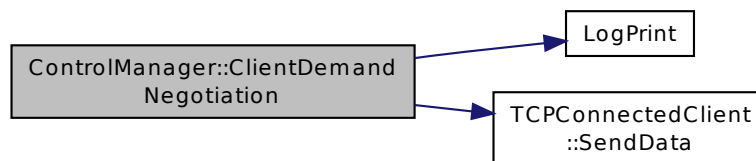
Relays the demand negotiation response of the client to the External Controller.

Parameters

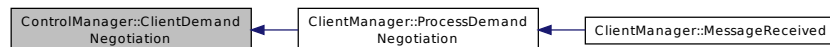
<i>clientId</i>	Unique client id of the responding client.
<i>numberOfDataPoints</i>	Number of consumption points sent.
<i>dataPoints</i>	Buffer for the consumption points.

Definition at line 231 of file ControlManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.5.2 `void ControlManager::ClientPriceRequest (const TClientId clientId)`

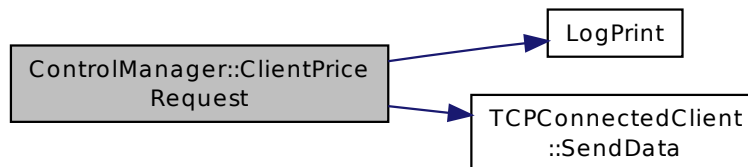
Relays the price request of a client to the External Controller.

Parameters

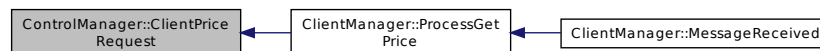
<i>clientId</i>	Unique client id of the requesting client.
-----------------	--

Definition at line 203 of file ControlManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.5.3 TClientName ControlManager::GetClientName (const TClientId *clientId*) [inline]

Returns the name of the object.

Todo Let's make this not inline for debugging.

Parameters

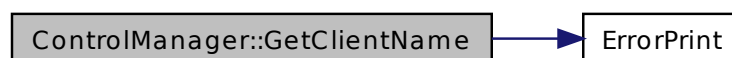
<i>clientId</i>	Unique client id for the requested object.
-----------------	--

Returns

Object name of the requested client id.

Definition at line 296 of file `ControlManager.h`.

Here is the call graph for this function:



8.11.5.4 void ControlManager::MakeDecision (void)

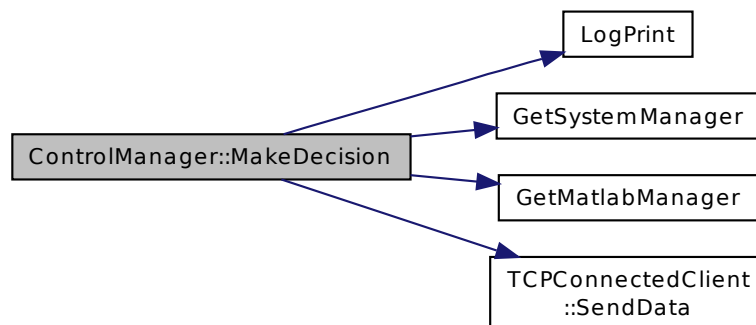
Starts the decision process by sending necessary information to External Controller.

This function starts the synchronous client decision process by sending the required information for each client. The information contains:

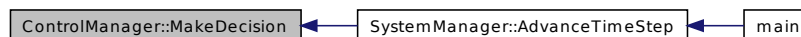
- Message Size for easier processing.
- Total number of synchronous clients.
- System mode.
- System time.
- For each client:
 - Client unique id
 - Client terminal voltage

Definition at line 132 of file ControlManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.5.5 void ControlManager::ProcessData (void * data, const size_t size)

Processes the received message from the External Controller.

This function processes the received message from the External Controller. Note that the received buffer is not guaranteed to hold a single message. This function processes the buffer in a while loop until all received messages are processed in order not to lose any data. The work flow is as follows:

- Check the received size for a connection drop (gracefull or not).
- Check the message type and process the respective data structure.
- Repeat process until the remaining unprocessed message size is zero.

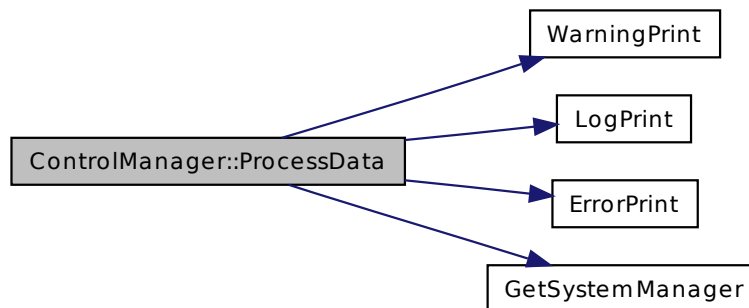
Todo Divide the function into multiple functions, processing each message separately.

Parameters

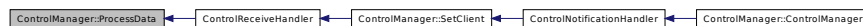
<i>data</i>	Buffer for the received message.
<i>size</i>	Size of the received message.

Definition at line 44 of file ControlManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.5.6 `void ControlManager::RegisterClient (const TClientId clientId, const TClientName & clientName, ClientManager * clientManager) [inline]`

Registers a new client to its maps for later referencing.

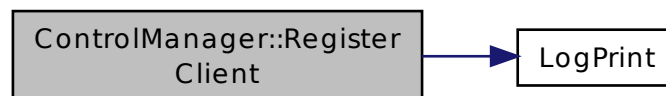
Todo This shouldn't be inline.

Parameters

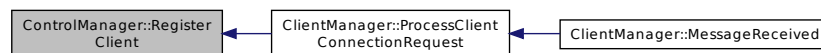
<i>clientId</i>	Unique client id.
<i>clientName</i>	Object name.
<i>clientManager</i>	ClientManager instance managing the client related works.

Definition at line 244 of file ControlManager.h.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.5.7 void ControlManager::SetClient (ThreadedTCPConnectedClient * client) [inline]

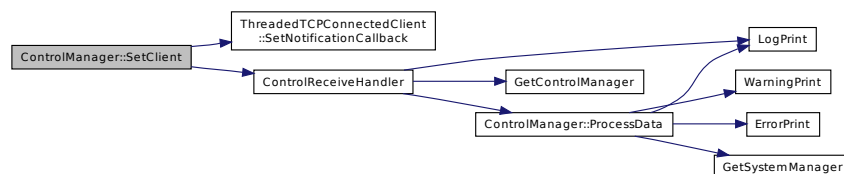
Sets the connection information to the accepted External Controller.

Parameters

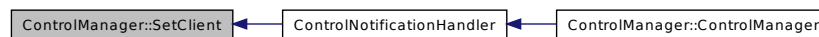
<i>client</i>	Connection information fo the newly accepted External Controller.
---------------	---

Definition at line 187 of file ControlManager.h.

Here is the call graph for this function:



Here is the caller graph for this function:



8.11.5.8 void ControlManager::UnRegisterClient (const TClientId *clientId*) [inline]

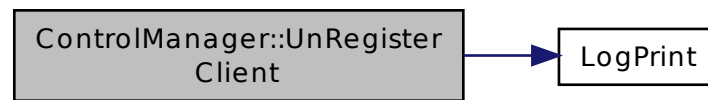
Deletes a client from the maps when connection is broken.

Parameters

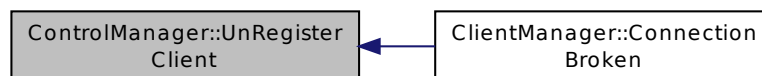
<i>clientId</i>	Unique client id of the broken connection.
-----------------	--

Definition at line 259 of file ControlManager.h.

Here is the call graph for this function:



Here is the caller graph for this function:



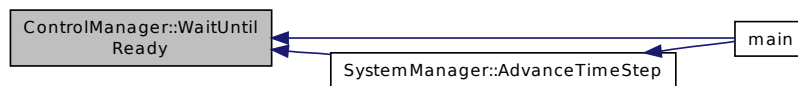
8.11.5.9 void ControlManager::WaitUntilReady (void) [inline]

Stops the calling thread until a ready signal is received from the External Controller.

Todo This doesn't need to be inline.

Definition at line 229 of file ControlManager.h.

Here is the caller graph for this function:



8.11.6 Friends And Related Function Documentation

8.11.6.1 ControlManager& GetControlManager (void) [friend]

Friend method for singleton implementation.

Returns

Returns the reference to the [ControlManager](#).
Only instance of [ControlManager](#).

Definition at line 11 of file ControlManager.cpp.

8.11.7 Member Data Documentation

8.11.7.1 ThreadedTCPConnectedClient* ControlManager::m_client [private]

Pointer to the currently accepted external controller communication handler.

Definition at line 157 of file ControlManager.h.

8.11.7.2 TClientIdMap ControlManager::m_clientIdMap [private]

Map containing the unique id to object name mapping.

Definition at line 167 of file ControlManager.h.

8.11.7.3 TClientManagerMap ControlManager::m_clientManagerMap [private]

Map containing the unique id to [ClientManager](#) mapping.

Definition at line 172 of file ControlManager.h.

8.11.7.4 Semaphore ControlManager::m_readySemaphore [private]

Semaphore that is released when the frame is finished by the external controller.

It stops all synchronous control until the next frame starts. This handles the process synchronization with the External Controller in a local way.

Definition at line 162 of file ControlManager.h.

8.11.7.5 ThreadedTCPServer ControlManager::m_server [private]

Implements the TCP server for external controller communication.

Definition at line 152 of file ControlManager.h.

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

- S2Sim/[ControlManager.h](#)
- S2Sim/[ControlManager.cpp](#)

8.12 TerraSwarm::Synchronous::DemandNegotiation Class Reference

Defines the [DemandNegotiation](#) message sent from the client to the Controller as a response to the price proposal.

```
#include <DemandNegotiation.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef bool [TCheckResult](#)
Defines the message check result type.
- typedef unsigned int [TNumberOfDataPoints](#)
Defines the type for number of data points.
- typedef unsigned int [TDataPoint](#)
Defines a general data point, used for consumption in this message.

Public Member Functions

- [~DemandNegotiation](#) (void)
Deallocates the memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory contains a [DemandNegotiation](#) message.
- [TNumberOfDataPoints GetNumberOfDataPoints](#) (void) const
Reads the number of data points in the message.
- [TDataPoint * GetDataPoints](#) (void) const
Gets the pointer to the data points field.

Static Public Member Functions

- static [DemandNegotiation * GetNewDemandNegotiation](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::TReceiverId](#) receiverId, const [TNumberOfDataPoints](#) numberOfDataPoints, [TDataPoint](#) *dataPoints)
Creates a new [DemandNegotiation](#) message and allocates the memory.

Private Types

- enum [HeaderValues](#) { [MessageType](#) = 0x0003, [MessageId](#) = 0x0004 }
Message Header values.
- enum [FieldSizeValues](#) { [NumberOfDataPointsSize](#) = sizeof(TNumberOfDataPoints), [DataPointSize](#) = sizeof(T-DataPoint) }
Size of the fields in the message.
- enum [FieldIndexValues](#) { [NumberOfDataPointsIndex](#) = MessageHeader::MessageHeaderSize, [DataStartIndex](#) = NumberOfDataPointsIndex + NumberOfDataPointsSize }
Index of the fields in the message.
- typedef [NetworkByteAccessor](#)
< [NumberOfDataPointsIndex](#),
[NumberOfDataPointsSize](#) > [TNumberOfDataPointsAccessor](#)
Accessor Helper for NumberOfDataPoints field.

Private Member Functions

- [DemandNegotiation](#) (void)
Private constructor to force the usage of the static creation method.

8.12.1 Detailed Description

Defines the [DemandNegotiation](#) message sent from the client to the Controller as a response to the price proposal.
Definition at line 23 of file DemandNegotiation.h.

8.12.2 Member Typedef Documentation

8.12.2.1 typedef bool TerraSwarm::Synchronous::DemandNegotiation::TCheckResult

Defines the message check result type.
Definition at line 39 of file DemandNegotiation.h.

8.12.2.2 typedef unsigned int TerraSwarm::Synchronous::DemandNegotiation::TDataPoint

Defines a general data point, used for consumption in this message.
Definition at line 58 of file DemandNegotiation.h.

8.12.2.3 typedef unsigned int TerraSwarm::Synchronous::DemandNegotiation::TNumberOfDataPoints

Defines the type for number of data points.
Definition at line 53 of file DemandNegotiation.h.

8.12.2.4 `typedef NetworkByteAccessor<NumberOfDataPointsIndex, NumberOfDataPointsSize>`
`TerraSwarm::Synchronous::DemandNegotiation::TNumberOfDataPointsAccessor` [private]

Accessor Helper for NumberOfDataPoints field.

Definition at line 82 of file DemandNegotiation.h.

8.12.3 Member Enumeration Documentation

8.12.3.1 `enum TerraSwarm::Synchronous::DemandNegotiation::CheckResultValues`

Defines the values for TCheckResult.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 44 of file DemandNegotiation.h.

8.12.3.2 `enum TerraSwarm::Synchronous::DemandNegotiation::FieldIndexValues` [private]

Index of the fields in the message.

Enumerator

NumberOfDataPointsIndex

DataStartIndex

Definition at line 73 of file DemandNegotiation.h.

8.12.3.3 `enum TerraSwarm::Synchronous::DemandNegotiation::FieldSizeValues` [private]

Size of the fields in the message.

Enumerator

NumberOfDataPointsSize

DataPointSize

Definition at line 64 of file DemandNegotiation.h.

8.12.3.4 `enum TerraSwarm::Synchronous::DemandNegotiation::HeaderValues` [private]

Message Header values.

Enumerator

MessageType

MessageId

Definition at line 29 of file DemandNegotiation.h.

8.12.4 Constructor & Destructor Documentation

8.12.4.1 TerraSwarm::Synchronous::DemandNegotiation::DemandNegotiation (void) [private]

Private constructor to force the usage of the static creation method.

Definition at line 14 of file DemandNegotiation.cpp.

8.12.4.2 TerraSwarm::Synchronous::DemandNegotiation::~~DemandNegotiation (void)

Deallocates the memory.

Definition at line 18 of file DemandNegotiation.cpp.

8.12.5 Member Function Documentation

8.12.5.1 DemandNegotiation::TCheckResult TerraSwarm::Synchronous::DemandNegotiation::CheckMessage (void) const

Checks whether the current memory contains a [DemandNegotiation](#) message.

Returns

Result of the check.

Definition at line 42 of file DemandNegotiation.cpp.

8.12.5.2 DemandNegotiation::TDataPoint * TerraSwarm::Synchronous::DemandNegotiation::GetDataPoints (void) const

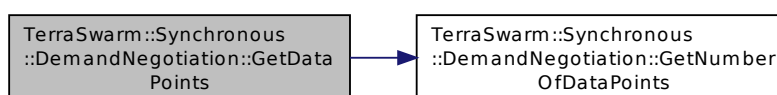
Gets the pointer to the data points field.

Returns

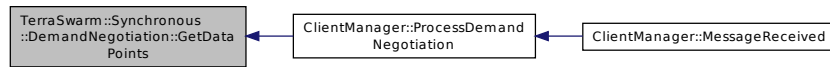
Pointer to the start of the data points field.

Definition at line 61 of file DemandNegotiation.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.5.3 DemandNegotiation * TerraSwarm::Synchronous::DemandNegotiation (const MessageHeader::TSenderId *senderId*, const MessageHeader::TReceiverId *receiverId*, const TNumberOfDataPoints *numberOfDataPoints*, TDataPoint * *dataPoints*) [static]

Creates a new [DemandNegotiation](#) message and allocates the memory.

Warning

Dellocation is the responsibility of the user.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.
<i>numberOfDataPoints</i>	Number of data points in the message.
<i>dataPoints</i>	Pointer to the buffer holding the data points.

Returns

Pointer to the newly allocated message.

Definition at line 24 of file DemandNegotiation.cpp.

8.12.5.4 DemandNegotiation::TNumberOfDataPoints TerraSwarm::Synchronous::DemandNegotiation::GetNumberOfDataPoints (void) const

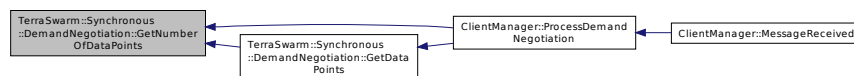
Reads the number of data points in the message.

Returns

Number of data points.

Definition at line 53 of file DemandNegotiation.cpp.

Here is the caller graph for this function:



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

- S2Sim/TerraswarmLibrary/[DemandNegotiation.h](#)
- S2Sim/TerraswarmLibrary/[DemandNegotiation.cpp](#)

8.13 TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size > Class Template Reference

Template class that uses the correct conversion function according to the size of the data.

Static Public Member Functions

- static TInput [NetworkToHost](#) (const TInput &value)
Converts the network byte order to host byte order.
- static TInput [HostToNetwork](#) (const TInput &value)
Converts the host byte order to the network byte order.

8.13.1 Detailed Description

```
template<TByteIndex byteIndex, TDataSize dataSize>template<typename TInput, TDataSize size = sizeof( TInput )>class TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size >
```

Template class that uses the correct conversion function according to the size of the data.

The default specialization is for any size not equal to 1,2 or 4.

Template Parameters

<i>TInput</i>	Type of the field that is to be accessed.
<i>size</i>	Size of TInput, done automatically.

Definition at line 49 of file NetworkByteAccessor.h.

8.13.2 Member Function Documentation

```
8.13.2.1 template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput , TDataSize size = sizeof( TInput )>static TInput TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size >::HostToNetwork( const TInput & value ) [inline],[static]
```

Converts the host byte order to the network byte order.

Parameters

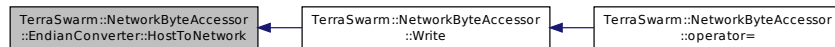
<i>value</i>	Value to be converted.
--------------	------------------------

Returns

Converted value.

Definition at line 73 of file NetworkByteAccessor.h.

Here is the caller graph for this function:



8.13.2.2 `template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput, TDataSize size = sizeof(TInput)> static TInput TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size >::NetworkToHost (const TInput & value) [inline],[static]`

Converts the network byte order to host byte order.

Parameters

<i>value</i>	Values to be converted.
--------------	-------------------------

Returns

Converted result.

Definition at line 60 of file NetworkByteAccessor.h.

Here is the caller graph for this function:



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

- S2Sim/TerraswarmLibrary/[NetworkByteAccessor.h](#)

8.14 TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< T-Input, 1 > Class Template Reference

Template specialization for a type with size 1 (char).

Static Public Member Functions

- static TInput [NetworkToHost](#) (const TInput value)
- static TInput [HostToNetwork](#) (const TInput value)

8.14.1 Detailed Description

template<TByteIndex byteIndex, TDataSize dataSize>template<typename TInput>class TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 1 >

Template specialization for a type with size 1 (char).

No conversion is performed.

Definition at line 83 of file NetworkByteAccessor.h.

8.14.2 Member Function Documentation

8.14.2.1 template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > static TInput TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 1 >::HostToNetwork (const TInput value) [inline], [static]

Definition at line 92 of file NetworkByteAccessor.h.

8.14.2.2 template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > static TInput TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 1 >::NetworkToHost (const TInput value) [inline], [static]

Definition at line 87 of file NetworkByteAccessor.h.

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

- S2Sim/TerraswarmLibrary/[NetworkByteAccessor.h](#)

8.15 TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 > Class Template Reference

Template specialization for a type with size 2 (short).

Static Public Member Functions

- static TInput [NetworkToHost](#) (const TInput value)
- static TInput [HostToNetwork](#) (const TInput value)

8.15.1 Detailed Description

template<TByteIndex byteIndex, TDataSize dataSize>template<typename TInput>class TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 >

Template specialization for a type with size 2 (short).

Short conversion is performed.

Definition at line 102 of file NetworkByteAccessor.h.

8.15.2 Member Function Documentation

8.15.2.1 `template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > static TInput TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 >::HostToNetwork (const TInput value) [inline], [static]`

Definition at line 111 of file NetworkByteAccessor.h.

8.15.2.2 `template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > static TInput TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 >::NetworkToHost (const TInput value) [inline], [static]`

Definition at line 106 of file NetworkByteAccessor.h.

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

- S2Sim/TerraswarmLibrary/[NetworkByteAccessor.h](#)

8.16 TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< T-Input, 4 > Class Template Reference

Template specialization for a type with size 4 (int).

Static Public Member Functions

- static TInput [NetworkToHost](#) (const TInput value)
- static TInput [HostToNetwork](#) (const TInput value)

8.16.1 Detailed Description

`template<TByteIndex byteIndex, TDataSize dataSize>template<typename TInput>class TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 4 >`

Template specialization for a type with size 4 (int).

Integer conversion is performed.

Definition at line 121 of file NetworkByteAccessor.h.

8.16.2 Member Function Documentation

8.16.2.1 `template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > static TInput TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 4 >::HostToNetwork (const TInput value) [inline], [static]`

Definition at line 130 of file NetworkByteAccessor.h.

```
8.16.2.2  template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > static TInput
          TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 4 >::NetworkToHost (
          const TInput value )  [inline], [static]
```

Definition at line 125 of file NetworkByteAccessor.h.

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

- S2Sim/TerraswarmLibrary/NetworkByteAccessor.h

8.17 TerraSwarm::Synchronous::GetPrice Class Reference

Defines the [GetPrice](#) message sent from the client to the Controller to get the current price value.

```
#include <GetPrice.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef bool [TCheckResult](#)
Defines the message check result type.

Public Member Functions

- [~GetPrice](#) (void)
Deallocates the memory.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory contains the [GetPrice](#) message.

Static Public Member Functions

- static [GetPrice](#) * [GetNewGetPrice](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::TReceiverId](#) receiverId)
Creates a new [GetPrice](#) message and allocates the necessary memory.
- static [TDataSize GetSize](#) (void)
Returns the size of a [GetPrice](#) message.

Private Types

- enum [HeaderValues](#) { [MessageType](#) = 0x0003, [Messageld](#) = 0x0001 }
Message Header values.
- enum [FieldSizeValues](#) { [TotalSize](#) = (0) }
Defines the Field Size values.

Private Member Functions

- [GetPrice](#) (void)

Private constructor to force the usage of the static creation method.

8.17.1 Detailed Description

Defines the [GetPrice](#) message sent from the client to the Controller to get the current price value.

Definition at line 22 of file `GetPrice.h`.

8.17.2 Member Typedef Documentation

8.17.2.1 `typedef bool TerraSwarm::Synchronous::GetPrice::TCheckResult`

Defines the message check result type.

Definition at line 38 of file `GetPrice.h`.

8.17.3 Member Enumeration Documentation

8.17.3.1 `enum TerraSwarm::Synchronous::GetPrice::CheckResultValues`

Defines the values for `TCheckResult`.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 43 of file `GetPrice.h`.

8.17.3.2 `enum TerraSwarm::Synchronous::GetPrice::FieldSizeValues` `[private]`

Defines the Field Size values.

Enumerator

TotalSize No data field.

Definition at line 53 of file `GetPrice.h`.

8.17.3.3 `enum TerraSwarm::Synchronous::GetPrice::HeaderValues` `[private]`

Message Header values.

Enumerator

MessageType

MessageId

Definition at line 28 of file `GetPrice.h`.

8.17.4 Constructor & Destructor Documentation

8.17.4.1 TerraSwarm::Synchronous::GetPrice::GetPrice (void) [private]

Private constructor to force the usage of the static creation method.

Definition at line 14 of file GetPrice.cpp.

8.17.4.2 TerraSwarm::Synchronous::GetPrice::~~GetPrice (void)

Deallocates the memory.

Definition at line 18 of file GetPrice.cpp.

8.17.5 Member Function Documentation

8.17.5.1 GetPrice::TCheckResult TerraSwarm::Synchronous::GetPrice::CheckMessage (void) const

Checks whether the current memory contains the [GetPrice](#) message.

Returns

Result of the check.

Definition at line 35 of file GetPrice.cpp.

8.17.5.2 GetPrice * TerraSwarm::Synchronous::GetPrice::GetNewGetPrice (const MessageHeader::TSenderId senderId, const MessageHeader::TReceiverId receiverId) [static]

Creates a new [GetPrice](#) message and allocates the necessary memory.

Warning

Deallocation is the responsibility of the user.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.

Returns

Pointer to the newly allocated message.

Definition at line 24 of file GetPrice.cpp.

8.17.5.3 TDataSize TerraSwarm::Synchronous::GetPrice::GetSize (void) [static]

Returns the size of a [GetPrice](#) message.

Returns

Size of the [GetPrice](#) message.

Definition at line 46 of file [GetPrice.cpp](#).

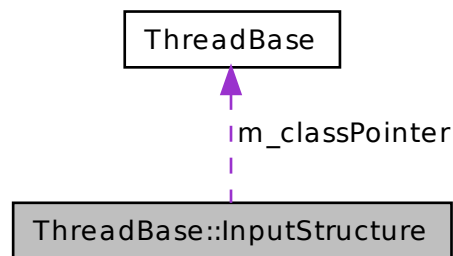
The documentation for this class was generated from the following files:

- [S2Sim/TerraswarmLibrary/GetPrice.h](#)
- [S2Sim/TerraswarmLibrary/GetPrice.cpp](#)

8.18 ThreadBase::InputStructure Struct Reference

Special Input structure sent to the wrapper function: [PosixThreadCover\(\)](#).

Collaboration diagram for ThreadBase::InputStructure:

**Public Member Functions**

- [InputStructure](#) (void *mainInput, [ThreadBase](#) *classPointer)
Simple constructor to assign the values.
- [InputStructure](#) (void *mainInput, [ThreadBase](#) *classPointer)
Simple constructor to assign the values.

Public Attributes

- void * [m_mainInput](#)
Actual input to the thread body.
- [ThreadBase](#) * [m_classPointer](#)
Pointer to the [ThreadBase](#) class to gain access to the real thread body.

8.18.1 Detailed Description

Special Input structure sent to the wrapper function: [PosixThreadCover\(\)](#).

This structure is used to recover the class pointer and the actual input to the thread body.

Definition at line 53 of file PosixThreadBase.h.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 ThreadBase::InputStructure::InputStructure (void * *mainInput*, ThreadBase * *classPointer*) `[inline]`

Simple constructor to assign the values.

Parameters

<i>mainInput</i>	Actual input to the thread body.
<i>classPointer</i>	Pointer to the ThreadBase class.

Definition at line 71 of file PosixThreadBase.h.

8.18.2.2 ThreadBase::InputStructure::InputStructure (void * *mainInput*, ThreadBase * *classPointer*) `[inline]`

Simple constructor to assign the values.

Parameters

<i>mainInput</i>	Actual input to the thread body.
<i>classPointer</i>	Pointer to the ThreadBase class.

Definition at line 72 of file WindowsThreadBase.h.

8.18.3 Member Data Documentation

8.18.3.1 ThreadBase * ThreadBase::InputStructure::m_classPointer

Pointer to the [ThreadBase](#) class to gain access to the real thread body.

Definition at line 63 of file PosixThreadBase.h.

8.18.3.2 void * ThreadBase::InputStructure::m_mainInput

Actual input to the thread body.

This is stored in this structure because the OS only allows a single input to the thread body, which is structure itself.

Definition at line 58 of file PosixThreadBase.h.

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

- S2Sim/ThreadLibrary/[PosixThreadBase.h](#)
- S2Sim/ThreadLibrary/[WindowsThreadBase.h](#)

8.19 IPAddress Class Reference

This class is an abstraction of the OS IP Address structure.

```
#include <IPAddress.h>
```

Public Types

- typedef unsigned int [TAddress](#)
IP Address value.
- typedef unsigned short [TPort](#)
Port value.
- typedef socklen_t [TSocketSize](#)
Size of the socket structure.

Public Member Functions

- [IPAddress](#) (void)
Sets the address family and clears the structure.
- [IPAddress](#) (const [IPAddress](#) ©)
Binary copy of the structure.
- [~IPAddress](#) (void)
No use.
- [IPAddress](#) & [operator=](#) (const [IPAddress](#) &rhs)
Assignment operator for binary copy.
- bool [operator==](#) (const [IPAddress](#) &rhs) const
Checks whether the addresses and ports are the same.
- void [SetAddress](#) (const [TAddress](#) address)
Sets the address field of the structure.
- [TAddress](#) [GetAddress](#) (void) const
Reads the address field of the structure.
- void [SetPort](#) (const [TPort](#) port)
Sets the port field of the structure.
- [TPort](#) [GetPort](#) (void) const
Reads the port field of the structure.
- [operator sockaddr *](#) (void)
Returns the cast version of the address structure to be used in OS utilities.

Static Public Member Functions

- static [TSocketSize](#) [GetSocketSize](#) (void)
Returns the size of the socket structure.

Private Types

- typedef sockaddr_in [TAddressStruct](#)
BSD Socket structure.

Private Member Functions

- [TAddress](#) & [GetAddress](#) (void)
Returns a reference to the address value.
- [TPort](#) & [GetPort](#) (void)
Return a reference to the port value.

Private Attributes

- [TAddressStruct m_address](#)
OS Address structure to be abstracted.

8.19.1 Detailed Description

This class is an abstraction of the OS IP Address structure.

Byte order conversion is handled automatically.

Definition at line 30 of file IPAddress.h.

8.19.2 Member Typedef Documentation

8.19.2.1 `typedef unsigned int IPAddress::TAddress`

IP Address value.

Definition at line 42 of file IPAddress.h.

8.19.2.2 `typedef sockaddr_in IPAddress::TAddressStruct` `[private]`

BSD Socket structure.

Definition at line 36 of file IPAddress.h.

8.19.2.3 `typedef unsigned short IPAddress::TPort`

Port value.

Definition at line 47 of file IPAddress.h.

8.19.2.4 `typedef socklen_t IPAddress::TSocketSize`

Size of the socket structure.

Definition at line 52 of file IPAddress.h.

8.19.3 Constructor & Destructor Documentation

8.19.3.1 `IPAddress::IPAddress (void)`

Sets the address family and clears the structure.

Definition at line 10 of file IPAddress.cpp.

8.19.3.2 IPAddress::IPAddress (const IPAddress & copy)

Binary copy of the structure.

Parameters

<i>copy</i>	Address to be copied.
-------------	-----------------------

Definition at line 16 of file IPAddress.cpp.

8.19.3.3 IPAddress::~~IPAddress (void)

No use.

Definition at line 21 of file IPAddress.cpp.

8.19.4 Member Function Documentation

8.19.4.1 IPAddress::TAddress & IPAddress::GetAddress (void) [private]

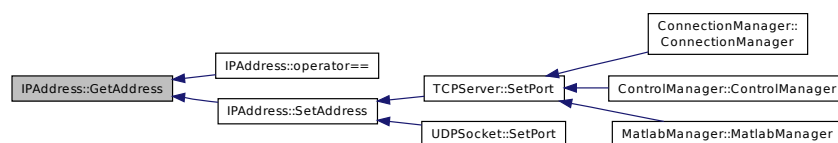
Returns a reference to the address value.

Returns

Reference to the address field.

Definition at line 53 of file IPAddress.cpp.

Here is the caller graph for this function:



8.19.4.2 IPAddress::TAddress IPAddress::GetAddress (void) const

Reads the address field of the structure.

Returns

4 byte IP Address.

Definition at line 63 of file IPAddress.cpp.

8.19.4.3 IPAddress::TPort & IPAddress::GetPort (void) [private]

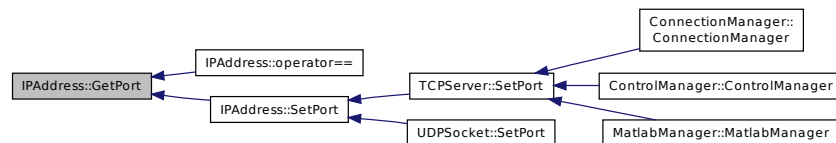
Return a reference to the port value.

Returns

Reference to the port field.

Definition at line 85 of file IPAddress.cpp.

Here is the caller graph for this function:



8.19.4.4 IPAddress::TPort IPAddress::GetPort (void) const

Reads the port field of the structure.

Returns

2 byte port value.

Definition at line 79 of file IPAddress.cpp.

8.19.4.5 static TSocketSize IPAddress::GetSocketSize (void) [inline],[static]

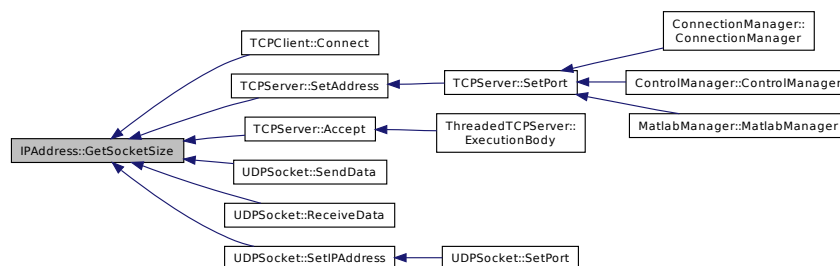
Returns the size of the socket structure.

Returns

Size of the socket structure.

Definition at line 160 of file IPAddress.h.

Here is the caller graph for this function:



8.19.4.6 IPAddress::operator sockaddr * (void)

Returns the cast version of the address structure to be used in OS utilities.

Returns

Address of the internal structure.

Definition at line 90 of file IPAddress.cpp.

8.19.4.7 IPAddress & IPAddress::operator= (const IPAddress & rhs)

Assignment operator for binary copy.

Parameters

<i>rhs</i>	Right hand side of (=).
------------	-------------------------

Returns

Returns a reference to itself.

Definition at line 26 of file IPAddress.cpp.

8.19.4.8 bool IPAddress::operator== (const IPAddress & rhs) const

Checks whether the addresses and ports are the same.

Parameters

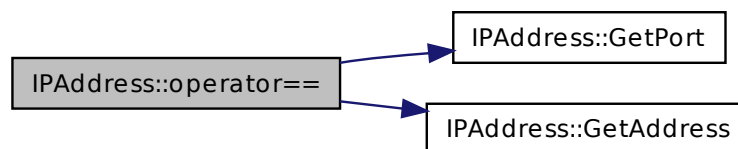
<i>rhs</i>	Right hand size of (==).
------------	--------------------------

Returns

Returns the equality between the classes.

Definition at line 37 of file IPAddress.cpp.

Here is the call graph for this function:



8.19.4.9 void IPAddress::SetAddress (const TAddress address)

Sets the address field of the structure.

Parameters

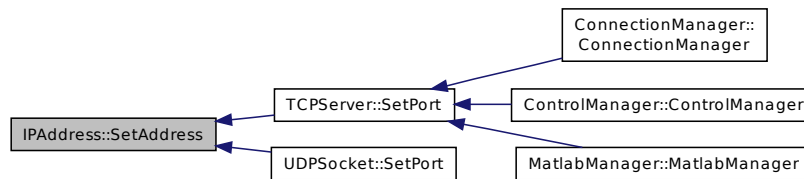
<i>address</i>	4 byte IP Address.
----------------	--------------------

Definition at line 47 of file IPAddress.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.19.4.10 void IPAddress::SetPort (const TPort *port*)

Sets the port field of the structure.

Parameters

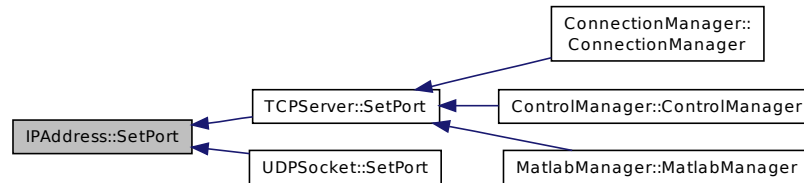
<i>port</i>	2 byte port value.
-------------	--------------------

Definition at line 73 of file IPAddress.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.19.5 Member Data Documentation

8.19.5.1 `TAddressStruct IPAddress::m_address` [private]

OS Address structure to be abstracted.

Definition at line 58 of file `IPAddress.h`.

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

- `S2Sim/SocketLibrary/`[IPAddress.h](#)
- `S2Sim/SocketLibrary/`[IPAddress.cpp](#)

8.20 MatlabManager Class Reference

Manages the connection to the OpenDSS-Matlab controller.

```
#include <MatlabManager.h>
```

```
classDiagram
    class SocketBaseTemplate["SocketBase< SocketType >"]
    class SocketStream["SocketBase< SOCK_STREAM >"]
    class IPAddress
    class ThreadBase
    class TCPConnectedClient
    class ThreadedTCPConnectedClient
    class TCPServer
    class ThreadedTCPServer
    class MatlabManager

    SocketBaseTemplate <|-- SocketStream
    SocketStream <|-- TCPConnectedClient
    SocketStream <|-- TCPServer
    ThreadBase <|-- TCPConnectedClient
    ThreadBase <|-- ThreadedTCPConnectedClient
    ThreadBase <|-- ThreadedTCPServer
    ThreadedTCPConnectedClient <|-- ThreadedTCPServer

    SocketStream ..> SocketBaseTemplate : < SOCK_STREAM >
    TCPConnectedClient ..> IPAddress : m_clientAddress
    ThreadedTCPConnectedClient ..> ThreadedTCPConnectedClient : m_notification
    ThreadedTCPConnectedClient ..> ThreadedTCPServer : m_notification
    ThreadedTCPConnectedClient ..> ThreadedTCPServer : m_client
    ThreadedTCPServer ..> ThreadedTCPServer : m_server
    ThreadedTCPServer ..> ThreadedTCPConnectedClient : m_server
    ThreadedTCPServer ..> ThreadedTCPConnectedClient : m_server
```

- typedef std::string TClientName
Defines the type for an object name.
- typedef
Asynchronous::ClientData::TDataPoint TWattage
Defines the consumption information type.
- typedef
Asynchronous::ClientData::TDataPoint TVoltage
Defines the voltage information type.

- void **SetClient** (**ThreadedTCPConnectedClient** *client)
Sets the OpenDSS controller connection information.
- bool **IsClientPresent** (const **TClientName** &clientName)
Checks for the presence of a client by communicating with OpenDSS controller.
- void **SetWattage** (const **TClientName** &clientName, const **TWattage** wattage)

- Sets the consumption of a client by communicating with OpenDSS controller.*
- [TWattage GetWattage](#) (const [TClientName](#) &clientName)
Gets the consumption of an object by communicating with OpenDSS controller.
- [TVoltage GetVoltage](#) (const [TClientName](#) &clientName)
Gets the terminal voltage of an object by communicating with OpenDSS controller.
- void [AdvanceTimeStep](#) (void)
Sends a signal to OpenDSS controller to indicate the end of a time step.
- void [ProcessData](#) (void *buffer, size_t size)
Processes the received message from OpenDSS controller.

Private Types

- enum [MessageTypeValues](#) {
[ClientCheckRequestType](#) = ([TMessageType](#))0x00000001, [ClientCheckResultType](#) = ([TMessageType](#))0x00000002, [ClientSetWattageType](#) = ([TMessageType](#))0x00000003, [ClientGetWattageType](#) = ([TMessageType](#))0x00000004,
[ClientWattageResultType](#) = ([TMessageType](#))0x00000005, [AdvanceTimeStepType](#) = ([TMessageType](#))0x00000006, [ClientGetVoltageType](#) = ([TMessageType](#))0x00000007, [ClientVoltageResultType](#) = ([TMessageType](#))0x00000008 }
Defines the values that can be used for TMessageType.
- enum [ClientCheckResultValues](#) { [ClientExists](#) = ([TClientCheckResult](#))0x00000001, [ClientDoesNotExist](#) = ([TClientCheckResult](#))0x00000002 }
Defines the values for TClientCheckResult.
- typedef unsigned int [TMessageType](#)
This type defines the message type used for message processing.
- typedef unsigned int [TClientCheckResult](#)
Defines the type that is received from the client existence check.

Private Member Functions

- [MatlabManager](#) (void)
Private constructor to implement the singleton.

Private Attributes

- [ThreadedTCPServer m_server](#)
Implements the TCP server managing the connection to OpenDSS controller.
- [ThreadedTCPConnectedClient * m_client](#)
Pointer to the connection management with the OpenDSS controller.
- Semaphore [m_clientPresenceSemaphore](#)
Semaphore used to signal that the presence check result is received.
- Semaphore [m_clientWattageSemaphore](#)
Semaphore used to signal that the wattage get result is received.
- Semaphore [m_clientVoltageSemaphore](#)
Semaphore used to signal that the voltage get result is received.
- bool [m_clientPresentInformation](#)
Temporary storage for the client presence information.

- [TWattage m_clientWattageInformation](#)
Temporary storage for the client wattage information.
- [TVoltage m_clientVoltageInformation](#)
Temporary storage for the client voltage information.

Friends

- [MatlabManager](#) & [GetMatlabManager](#) (void)
Friend method to implement the singleton for [MatlabManager](#).

8.20.1 Detailed Description

Manages the connection to the OpenDSS-Matlab controller.

This class manages the communication to the MATLAB controller, that controls the OpenDSS related information processing through the DLL interface.

Definition at line 48 of file MatlabManager.h.

8.20.2 Member Typedef Documentation

8.20.2.1 `typedef unsigned int MatlabManager::TClientCheckResult` [private]

Defines the type that is received from the client existence check.

Definition at line 82 of file MatlabManager.h.

8.20.2.2 `typedef std::string MatlabManager::TClientName`

Defines the type for an object name.

Definition at line 97 of file MatlabManager.h.

8.20.2.3 `typedef unsigned int MatlabManager::TMessageType` [private]

This type defines the message type used for message processing.

Definition at line 62 of file MatlabManager.h.

8.20.2.4 `typedef Asynchronous::ClientData::TDataPoint MatlabManager::TVoltage`

Defines the voltage information type.

Definition at line 107 of file MatlabManager.h.

8.20.2.5 `typedef Asynchronous::ClientData::TDataPoint MatlabManager::TWattage`

Defines the consumption information type.

Definition at line 102 of file MatlabManager.h.

8.20.3 Member Enumeration Documentation

8.20.3.1 enum MatlabManager::ClientCheckResultValues [private]

Defines the values for TClientCheckResult.

Enumerator

ClientExists Indicates that the object exists.

ClientDoesNotExist Indicates that the object does not exist.

Definition at line 87 of file MatlabManager.h.

8.20.3.2 enum MatlabManager::MessageTypeValues [private]

Defines the values that can be used for TMessageType.

Enumerator

ClientCheckRequestType Sent to the OpenDSS controller to check whether a client exists.

ClientCheckResultType Response of the OpenDSS controller to the client existence check.

ClientSetWattageType Sent to the OpenDSS controller to set the consumption of an object.

ClientGetWattageType Sent to the OpenDSS controller to get the consumption of an object.

ClientWattageResultType Response of the OpenDSS controller to the client consumption get request.

AdvanceTimeStepType Sent to the OpenDSS controller to indicate the end of a time step.

ClientGetVoltageType Sent to the OpenDSS controller to get the terminal voltage of an object.

ClientVoltageResultType Response of the OpenDSS controller to the voltage get request.

Definition at line 67 of file MatlabManager.h.

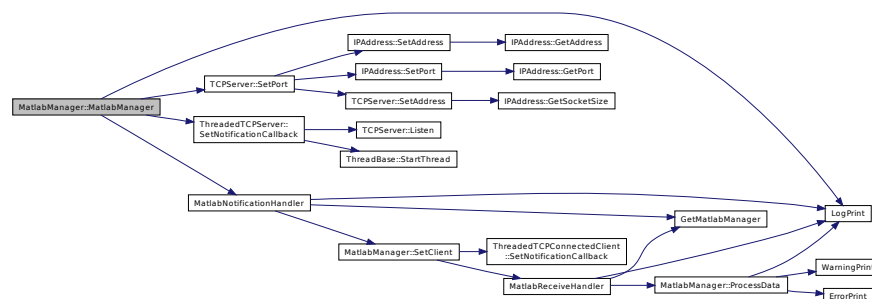
8.20.4 Constructor & Destructor Documentation

8.20.4.1 MatlabManager::MatlabManager (void) [private]

Private constructor to implement the singleton.

Definition at line 40 of file MatlabManager.cpp.

Here is the call graph for this function:



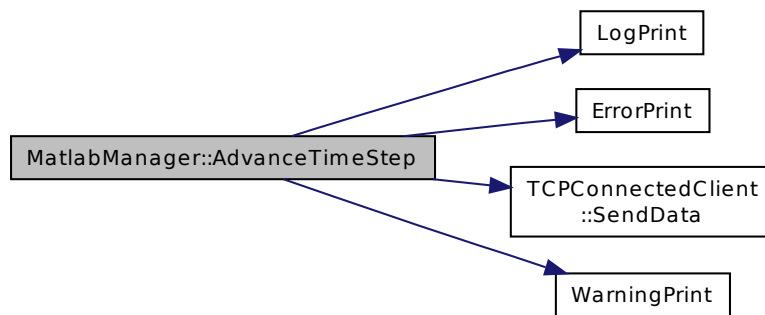
8.20.5 Member Function Documentation

8.20.5.1 void MatlabManager::AdvanceTimeStep (void)

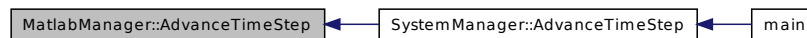
Sends a signal to OpenDSS controller to indicate the end of a time step.

Definition at line 341 of file MatlabManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.5.2 MatlabManager::TVoltage MatlabManager::GetVoltage (const TClientName & clientName)

Gets the terminal voltage of an object by communicating with OpenDSS controller.

It may block the function call.

Parameters

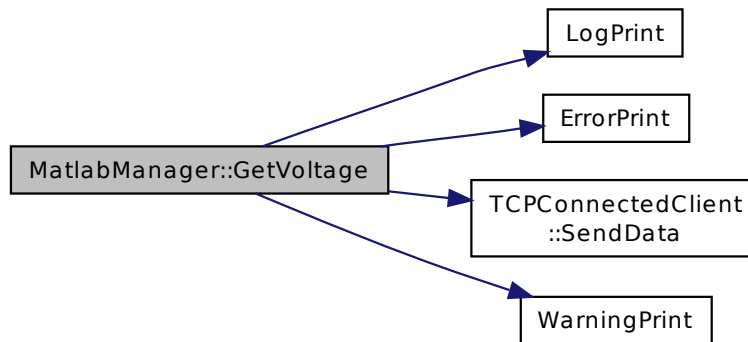
<i>clientName</i>	Name of the object.
-------------------	---------------------

Returns

Terminal voltage of the client.

Definition at line 291 of file MatlabManager.cpp.

Here is the call graph for this function:

**8.20.5.3 MatlabManager::TWattage MatlabManager::GetWattage (const TClientName & *clientName*)**

Gets the consumption of an object by communicating with OpenDSS controller.

It may block the function call.

Parameters

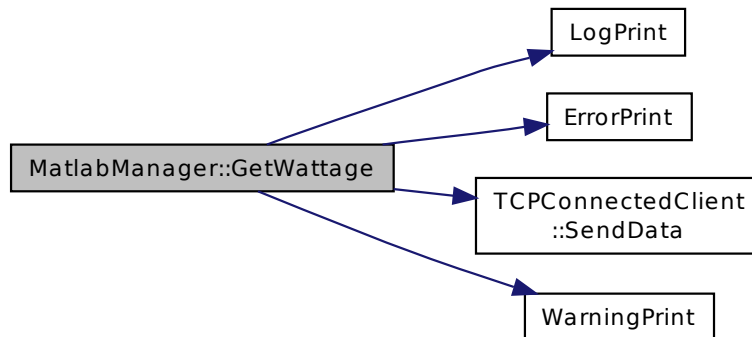
<i>clientName</i>	Name of the object.
-------------------	---------------------

Returns

Consumption of the selected object.

Definition at line 241 of file MatlabManager.cpp.

Here is the call graph for this function:

**8.20.5.4** `bool MatlabManager::IsClientPresent (const TClientName & clientName)`

Checks for the presence of a client by communicating with OpenDSS controller.

It may block the function call.

Parameters

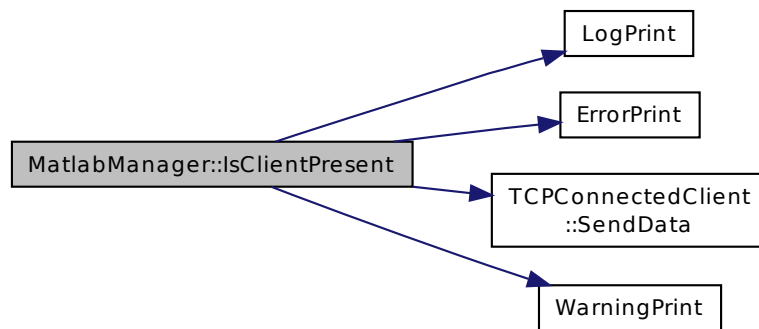
<i>clientName</i>	Name of the object to be checked.
-------------------	-----------------------------------

Returns

Indicates the presence of the client.

Definition at line 145 of file MatlabManager.cpp.

Here is the call graph for this function:



8.20.5.5 void MatlabManager::ProcessData (void * *buffer*, size_t *size*)

Processes the received message from OpenDSS controller.

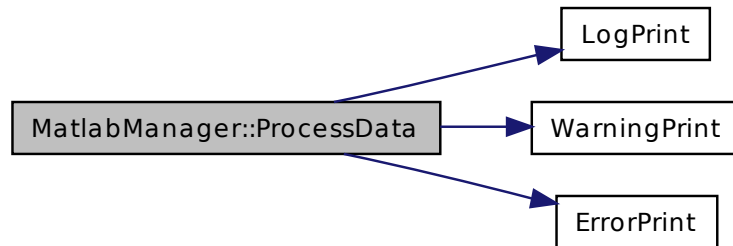
This function processes all received messages from the OpenDSS controller. Note that, it is not guaranteed that the received buffer contains only a single message. This function uses a while loop to process all the messages to avoid data loss.

Parameters

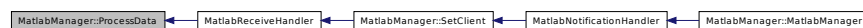
<i>buffer</i>	Buffer containing the received data.
<i>size</i>	Size of the received data.

Definition at line 56 of file MatlabManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.5.6 void MatlabManager::SetClient (ThreadedTCPConnectedClient * client) [inline]

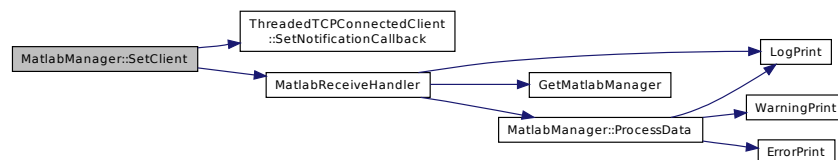
Sets the OpenDSS controller connection information.

Parameters

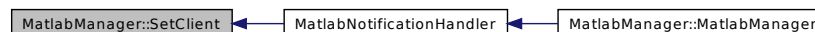
<i>client</i>	<#client description#>
---------------	------------------------

Definition at line 163 of file `MatlabManager.h`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.5.7 void MatlabManager::SetWattage (const TClientName & *clientName*, const TWattage *wattage*)

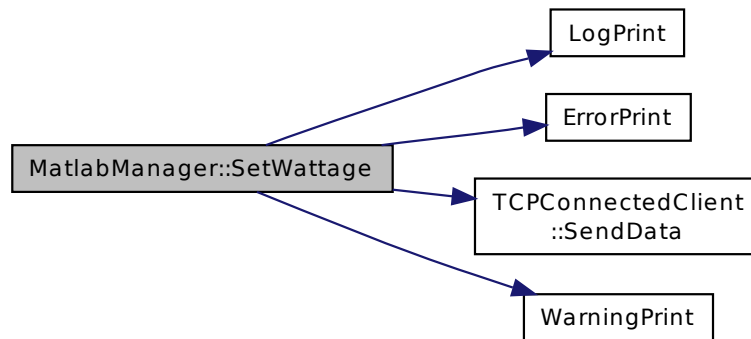
Sets the consumption of a client by communicating with OpenDSS controller.

Parameters

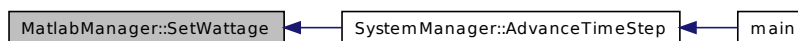
<i>clientName</i>	Name of the object.
<i>wattage</i>	Consumption of the selected object.

Definition at line 195 of file MatlabManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.6 Friends And Related Function Documentation

8.20.6.1 MatlabManager& GetMatlabManager (void) [friend]

Friend method to implement the singleton for [MatlabManager](#).

Returns

Returns the only instance of [MatlabManager](#).
The only instance of [MatlabManager](#).

Definition at line 13 of file MatlabManager.cpp.

8.20.7 Member Data Documentation

8.20.7.1 ThreadedTCPConnectedClient* MatlabManager::m_client [private]

Pointer to the connection management with the OpenDSS controller.

Definition at line 118 of file MatlabManager.h.

8.20.7.2 Semaphore MatlabManager::m_clientPresenceSemaphore [private]

Semaphore used to signal that the presence check result is received.

Definition at line 123 of file MatlabManager.h.

8.20.7.3 bool MatlabManager::m_clientPresentInformation [private]

Temporary storage for the client presence information.

Definition at line 138 of file MatlabManager.h.

8.20.7.4 TVoltage MatlabManager::m_clientVoltageInformation [private]

Temporary storage for the client voltage information.

Definition at line 148 of file MatlabManager.h.

8.20.7.5 Semaphore MatlabManager::m_clientVoltageSemaphore [private]

Semaphore used to signal that the voltage get result is received.

Definition at line 133 of file MatlabManager.h.

8.20.7.6 TWattage MatlabManager::m_clientWattageInformation [private]

Temporary storage for the client wattage information.

Definition at line 143 of file MatlabManager.h.

8.20.7.7 Semaphore MatlabManager::m_clientWattageSemaphore [private]

Semaphore used to signal that the wattage get result is received.

Definition at line 128 of file MatlabManager.h.

8.20.7.8 ThreadedTCPServer MatlabManager::m_server [private]

Implements the TCP server managing the connection to OpenDSS controller.

Definition at line 113 of file MatlabManager.h.

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

- [S2Sim/MatlabManager.h](#)
- [S2Sim/MatlabManager.cpp](#)

8.21 TerraSwarm::MessageEnder Class Reference

Class to send the end of message field.

```
#include <MessageEnder.h>
```

Public Types

- enum [SizeValues](#) { [EndOfMessageSize](#) = sizeof(TEndOfMessage) }
Defines the size values of the class.
- enum [CheckResultValues](#) { [CheckSuccess](#) = (TCheckResult)true, [CheckFail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef bool [TCheckResult](#)
Defines the message check result type.

Public Member Functions

- void [SetEndOfMessageField](#) (void)
Sets the EOM field to its value.
- [TCheckResult CheckEndOfMessageField](#) (void) const
Checks the EOM field for the correct value.

Private Types

- enum [EndOfMessageValues](#) { [EndOfMessageValue](#) = (TEndOfMessage)0xFEDCBA98 }
Values for the EOM field.
- typedef unsigned int [TEndOfMessage](#)
End of message field type.
- typedef [NetworkByteAccessor](#)
< 0, [EndOfMessageSize](#) > [TEndOfMessageAccessor](#)
Accessor helper for the EOM field.

8.21.1 Detailed Description

Class to send the end of message field.

Definition at line 20 of file MessageEnder.h.

8.21.2 Member Typedef Documentation

8.21.2.1 typedef bool TerraSwarm::MessageEnder::TCheckResult

Defines the message check result type.

Definition at line 48 of file MessageEnder.h.

8.21.2.2 `typedef unsigned int TerraSwarm::MessageEnder::TEndOfMessage` [private]

End of message field type.

Definition at line 26 of file MessageEnder.h.

8.21.2.3 `typedef NetworkByteAccessor<0,EndOfMessageSize> TerraSwarm::MessageEnder::TEndOfMessage-
Accessor` [private]

Accessor helper for the EOM field.

Definition at line 63 of file MessageEnder.h.

8.21.3 Member Enumeration Documentation

8.21.3.1 `enum TerraSwarm::MessageEnder::CheckResultValues`

Defines the values for TCheckResult.

Enumerator

CheckSuccess Message has correct EOM.

CheckFail Message has incorrect EOM.

Definition at line 53 of file MessageEnder.h.

8.21.3.2 `enum TerraSwarm::MessageEnder::EndOfMessageValues` [private]

Values for the EOM field.

Enumerator

EndOfMessageValue The only defined EOM value.

Definition at line 31 of file MessageEnder.h.

8.21.3.3 `enum TerraSwarm::MessageEnder::SizeValues`

Defines the size values of the class.

Enumerator

EndOfMessageSize Size of the EOM field.

Definition at line 40 of file MessageEnder.h.

8.21.4 Member Function Documentation

8.21.4.1 `MessageEnder::TCheckResult TerraSwarm::MessageEnder::CheckEndOfMessageField (void) const`

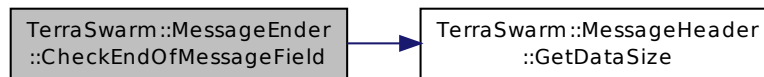
Checks the EOM field for the correct value.

Returns

Result of the check.

Definition at line 23 of file MessageEnder.cpp.

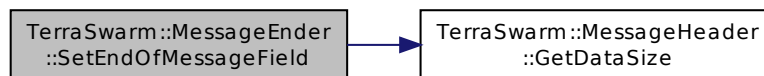
Here is the call graph for this function:

**8.21.4.2 void TerraSwarm::MessageEnder::SetEndOfMessageField (void)**

Sets the EOM field to its value.

Definition at line 14 of file MessageEnder.cpp.

Here is the call graph for this function:



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

- [S2Sim/TerraswarmLibrary/MessageEnder.h](#)
- [S2Sim/TerraswarmLibrary/MessageEnder.cpp](#)

8.22 TerraSwarm::MessageHeader Class Reference

This class defines the common message header for all messages.

```
#include <MessageHeader.h>
```

Public Types

- enum [SizeValues](#) { [MessageHeaderSize](#) }
Defines various size values for the Message Header.
- typedef unsigned short [TId](#)
Unique client id type.

- typedef [TId](#) [TSenderId](#)
Unique Id of the sender.
- typedef [TId](#) [TReceiverId](#)
Unique Id of the receiver.
- typedef unsigned short [TMessageType](#)
Type of the message for message parsing.
- typedef unsigned short [TMessageId](#)
Id of the message for message parsing.

Public Member Functions

- void [PrepareOutgoingMessage](#) (const [TSenderId](#) senderId, const [TReceiverId](#) receiverId, const [TMessageType](#) messageType, const [TMessageId](#) messageId, const [TDataSize](#) dataSize)
Sets all the fields of the message header and prepares it.
- [TSenderId](#) [GetSenderId](#) (void) const
Gets the Sender Id.
- [TReceiverId](#) [GetReceiverId](#) (void) const
Gets the Receiver Id.
- [TMessageType](#) [GetMessageType](#) (void) const
Gets the message type.
- [TMessageId](#) [GetMessageId](#) (void) const
Gets the Message Id.
- [TDataSize](#) [GetDataSize](#) (void) const
Gets the Data Size.

Static Public Member Functions

- static [MessageHeader](#) * [GetNewMessageHeader](#) (void)
Static function that creates a buffer for a new message header.

Private Types

- enum [StartOfMessageValues](#) { [StartOfMessageDefaultValue](#) = (TStartOfMessage)0x12345678 }
- enum [HeaderFieldSizeValues](#) {
[StartOfMessageSize](#) = (TDataSize)sizeof(TStartOfMessage), [IdSize](#) = (TDataSize)sizeof(TId), [SenderIdSize](#) = [IdSize](#), [ReceiverIdSize](#) = [IdSize](#),
[SequenceNumberSize](#) = (TDataSize)sizeof(TSequenceNumber), [MessageTypeSize](#) = (TDataSize)sizeof(TMessageType), [MessageIdSize](#) = (TDataSize)sizeof(TMessageId), [DataSizeSize](#) = (TDataSize)sizeof(TDataSize) }
Size of the header fields.
- enum [HeaderFieldIndexValues](#) {
[StartOfMessageIndex](#) = (TByteIndex)0, [SenderIdIndex](#) = (TByteIndex)([StartOfMessageIndex](#) + [StartOfMessageSize](#)), [ReceiverIdIndex](#) = (TByteIndex)([SenderIdIndex](#) + [SenderIdSize](#)), [SequenceNumberIndex](#) = (TByteIndex)([ReceiverIdIndex](#) + [ReceiverIdSize](#)),
[MessageTypeIndex](#) = (TByteIndex)([SequenceNumberIndex](#) + [SequenceNumberSize](#)), [MessageIdIndex](#) = (TByteIndex)([MessageTypeIndex](#) + [MessageTypeSize](#)), [DataSizeIndex](#) = (TByteIndex)([MessageIdIndex](#) + [MessageIdSize](#)), [DataIndex](#) = (TByteIndex)([DataSizeIndex](#) + [DataSizeSize](#)) }

Byte index values for the header fields.

- typedef unsigned int [TStartOfMessage](#)
Indicates the start of the message.
- typedef unsigned int [TSequenceNumber](#)
Defines the message sequence number.
- typedef [NetworkByteAccessor](#)
< [StartOfMessageIndex](#),
[StartOfMessageSize](#) > [TStartOfMessageAccessor](#)
Manages the accessing of the Start of Message field.
- typedef [NetworkByteAccessor](#)
< [SenderIdIndex](#), [SenderIdSize](#) > [TSenderIdAccessor](#)
Manages the accessing of the Sender Id field.
- typedef [NetworkByteAccessor](#)
< [ReceiverIdIndex](#),
[ReceiverIdSize](#) > [TReceiverIdAccessor](#)
Manages the accessing of the Receiver Id field.
- typedef [NetworkByteAccessor](#)
< [SequenceNumberIndex](#),
[SequenceNumberSize](#) > [TSequenceNumberAccessor](#)
Manages the accessing of the Sequence Number field.
- typedef [NetworkByteAccessor](#)
< [MessageTypeIndex](#),
[MessageTypeSize](#) > [TMessageTypeAccessor](#)
Manages the accessing of the Message Type field.
- typedef [NetworkByteAccessor](#)
< [MessageIdIndex](#),
[MessageIdSize](#) > [TMessageIdAccessor](#)
Manages the accessing of the Message Id field.
- typedef [NetworkByteAccessor](#)
< [DataSizeIndex](#), [DataSizeSize](#) > [TDataSizeAccessor](#)
Manages the accessing of the Data Size field.

Private Member Functions

- [MessageHeader](#) (void)
Not used.
- template<typename Accessor >
Accessor * [Access](#) (void)
Template function to use the accessor of any field to access that field.
- template<typename Accessor >
const Accessor * [Access](#) (void) const
Template function to use the accessor of any field to access that field in a constant fashion.

Static Private Member Functions

- static [TSequenceNumber](#) [GetNextSequenceNumber](#) (const [TSenderId](#) senderId, const [TReceiverId](#) receiverId)
Static function to get the next sequence number as the sequence number is sender unique, not message unique.

8.22.1 Detailed Description

This class defines the common message header for all messages.

Definition at line 19 of file MessageHeader.h.

8.22.2 Member Typedef Documentation

8.22.2.1 `typedef NetworkByteAccessor<DataSizeIndex, DataSizeSize> TerraSwarm::MessageHeader::TData-SizeAccessor [private]`

Manages the accessing of the Data Size field.

Definition at line 140 of file MessageHeader.h.

8.22.2.2 `typedef unsigned short TerraSwarm::MessageHeader::TId`

Unique client id type.

Definition at line 25 of file MessageHeader.h.

8.22.2.3 `typedef unsigned short TerraSwarm::MessageHeader::TMsgId`

Id of the message for message parsing.

Definition at line 45 of file MessageHeader.h.

8.22.2.4 `typedef NetworkByteAccessor<MsgIdIndex, MsgIdSize> TerraSwarm::MessageHeader::T-MsgIdAccessor [private]`

Manages the accessing of the Message Id field.

Definition at line 135 of file MessageHeader.h.

8.22.2.5 `typedef unsigned short TerraSwarm::MessageHeader::TMessageType`

Type of the message for message parsing.

Definition at line 40 of file MessageHeader.h.

8.22.2.6 `typedef NetworkByteAccessor<MessageTypeIndex, MessageTypeSize> TerraSwarm::MessageHeader-::TMessageTypeAccessor [private]`

Manages the accessing of the Message Type field.

Definition at line 130 of file MessageHeader.h.

8.22.2.7 `typedef TId TerraSwarm::MessageHeader::TReceiverId`

Unique Id of the receiver.

Definition at line 35 of file MessageHeader.h.

8.22.2.8 `typedef NetworkByteAccessor<ReceiverIdIndex, ReceiverIdSize> TerraSwarm::MessageHeader::T-ReceiverIdAccessor [private]`

Manages the accessing of the Receiver Id field.

Definition at line 120 of file MessageHeader.h.

8.22.2.9 `typedef TId TerraSwarm::MessageHeader::TSenderId`

Unique Id of the sender.

Definition at line 30 of file MessageHeader.h.

8.22.2.10 `typedef NetworkByteAccessor<SenderIdIndex, SenderIdSize> TerraSwarm::MessageHeader::T-SenderIdAccessor [private]`

Manages the accessing of the Sender Id field.

Definition at line 115 of file MessageHeader.h.

8.22.2.11 `typedef unsigned int TerraSwarm::MessageHeader::TSequenceNumber [private]`

Defines the message sequence number.

Definition at line 64 of file MessageHeader.h.

8.22.2.12 `typedef NetworkByteAccessor<SequenceNumberIndex, SequenceNumberSize> TerraSwarm::MessageHeader::TSequenceNumberAccessor [private]`

Manages the accessing of the Sequence Number field.

Definition at line 125 of file MessageHeader.h.

8.22.2.13 `typedef unsigned int TerraSwarm::MessageHeader::TStartOfMessage [private]`

Indicates the start of the message.

Definition at line 51 of file MessageHeader.h.

8.22.2.14 `typedef NetworkByteAccessor<StartOfMessageIndex, StartOfMessageSize> TerraSwarm::MessageHeader::TStartOfMessageAccessor [private]`

Manages the accessing of the Start of Message field.

Definition at line 110 of file MessageHeader.h.

8.22.3 Member Enumeration Documentation

8.22.3.1 `enum TerraSwarm::MessageHeader::HeaderFieldIndexValues [private]`

Byte index values for the header fields.

Enumerator

StartOfMessageIndex
SenderIdIndex
ReceiverIdIndex
SequenceNumberIndex
MessageTypeIndex
MessageIdIndex
DataSizeIndex
DataIndex

Definition at line 95 of file MessageHeader.h.

8.22.3.2 enum TerraSwarm::MessageHeader::HeaderFieldSizeValues [private]

Size of the header fields.

Enumerator

StartOfMessageSize
IdSize
SenderIdSize
ReceiverIdSize
SequenceNumberSize
MessageTypeSize
MessageIdSize
DataSizeSize

Definition at line 80 of file MessageHeader.h.

8.22.3.3 enum TerraSwarm::MessageHeader::SizeValues

Defines various size values for the Message Header.

Enumerator

MessageHeaderSize Size of the message header.

Definition at line 146 of file MessageHeader.h.

8.22.3.4 enum TerraSwarm::MessageHeader::StartOfMessageValues [private]

Defines the value for TStartOfMessage.

Enumerator

StartOfMessageDefaultValue Only possible value for SOM.

Definition at line 56 of file MessageHeader.h.

8.22.4 Constructor & Destructor Documentation

8.22.4.1 TerraSwarm::MessageHeader::MessageHeader (void) [private]

Not used.

Definition at line 13 of file MessageHeader.cpp.

8.22.5 Member Function Documentation

8.22.5.1 template<typename Accessor > Accessor* TerraSwarm::MessageHeader::Access (void) [inline], [private]

Template function to use the accessor of any field to access that field.

Template Parameters

<i>Accessor</i>	The accessor type for the desired field.
-----------------	--

Returns

Returns the accessor for the desired field.

Definition at line 170 of file MessageHeader.h.

8.22.5.2 template<typename Accessor > const Accessor* TerraSwarm::MessageHeader::Access (void) const [inline], [private]

Template function to use the accessor of any field to access that field in a constant fashion.

Template Parameters

<i>Accessor</i>	The accessor type for the desired field.
-----------------	--

Returns

Returns the accessor for the desired field.

Definition at line 183 of file MessageHeader.h.

8.22.5.3 TDataSize TerraSwarm::MessageHeader::GetDataSize (void) const [inline]

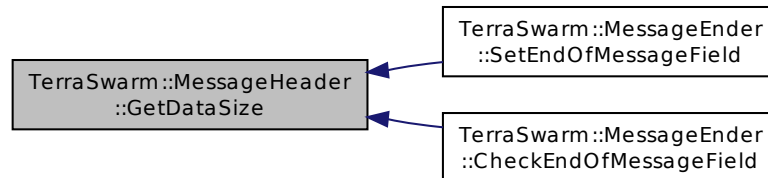
Gets the Data Size.

Returns

Size of the user data.

Definition at line 271 of file MessageHeader.h.

Here is the caller graph for this function:



8.22.5.4 TMessageId TerraSwarm::MessageHeader::GetMessageId (void) const [inline]

Gets the Message Id.

Returns

Message Id field value.

Definition at line 258 of file MessageHeader.h.

8.22.5.5 TMessageType TerraSwarm::MessageHeader::GetMessageType (void) const [inline]

Gets the message type.

Returns

Message Type field value.

Definition at line 245 of file MessageHeader.h.

8.22.5.6 MessageHeader * TerraSwarm::MessageHeader::GetNewMessageHeader (void) [static]

Static function that creates a buffer for a new message header.

Returns

Returns the address of the allocated buffer.

Definition at line 18 of file MessageHeader.cpp.

8.22.5.7 MessageHeader::TSequenceNumber TerraSwarm::MessageHeader::GetNextSequenceNumber (const TSenderId senderId, const TReceiverId receiverId) [static], [private]

Static function to get the next sequence number as the sequence number is sender unique, not message unique.

Parameters

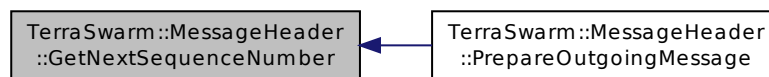
<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the intended receiver.

Returns

Next available sequence number.

Definition at line 41 of file MessageHeader.cpp.

Here is the caller graph for this function:



8.22.5.8 TReceiverId TerraSwarm::MessageHeader::GetReceiverId (void) const [inline]

Gets the Receiver Id.

Returns

Id of the Receiver.

Definition at line 232 of file MessageHeader.h.

8.22.5.9 TSenderId TerraSwarm::MessageHeader::GetSenderId (void) const [inline]

Gets the Sender Id.

Returns

Id of the Sender.

Definition at line 219 of file MessageHeader.h.

8.22.5.10 void TerraSwarm::MessageHeader::PrepareOutgoingMessage (const TSenderId senderId, const TReceiverId receiverId, const TMessageType messageType, const TMessageId messageId, const TDataSize dataSize)

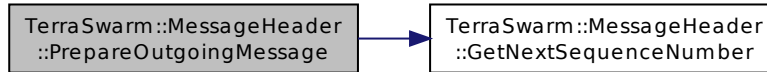
Sets all the fields of the message header and prepares it.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.
<i>messageType</i>	Message type field.
<i>messageId</i>	Message Id field.
<i>dataSize</i>	Data Size field.

Definition at line 25 of file MessageHeader.cpp.

Here is the call graph for this function:



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

- S2Sim/TerraswarmLibrary/MessageHeader.h
- S2Sim/TerraswarmLibrary/MessageHeader.cpp

8.23 TerraSwarm::NetworkByteAccessor< byteIndex, dataSize > Class Template Reference

Template class to automatically convert byte order and help access ordered bytes in the memory.

```
#include <NetworkByteAccessor.h>
```

Classes

- class [EndianConverter](#)
Template class that uses the correct conversion function according to the size of the data.
- class [EndianConverter< TInput, 1 >](#)
Template specialization for a type with size 1 (char).
- class [EndianConverter< TInput, 2 >](#)
Template specialization for a type with size 2 (short).
- class [EndianConverter< TInput, 4 >](#)
Template specialization for a type with size 4 (int).

Public Member Functions

- template<typename TInput >
TInput [Write](#) (const TInput input)
Writes a value to the given address index with correct byte order.
- template<typename TInput >
TInput [operator=](#) (const TInput input)
Same as [Write\(\)](#) to simplify code writing.

- `template<typename TInput >`
`TInput Read (TInput &value) const`
Reads a value from the given address index with correct byte order.

8.23.1 Detailed Description

`template<TByteIndex byteIndex, TDataSize dataSize> class TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >`

Template class to automatically convert byte order and help access ordered bytes in the memory.

This class is especially useful for message processing and preparing.

Template Parameters

<i>byteIndex</i>	Index of the bytes that are to be manipulated/accessed.
<i>dataSize</i>	Number of bytes to be manipulated/accessed.

Definition at line 39 of file NetworkByteAccessor.h.

8.23.2 Member Function Documentation

8.23.2.1 `template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > TInput`
`TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::operator= (const TInput input) [inline]`

Same as [Write\(\)](#) to simplify code writing.

Parameters

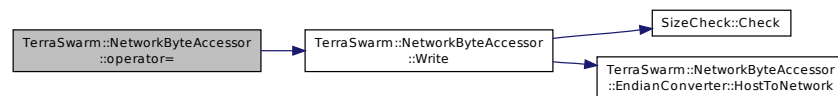
<i>input</i>	Value to be written.
--------------	----------------------

Returns

Actually written value.

Definition at line 164 of file NetworkByteAccessor.h.

Here is the call graph for this function:



8.23.2.2 `template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > TInput`
`TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::Read (TInput & value) const [inline]`

Reads a value from the given address index with correct byte order.

In addition, this function checks for the size of the input for mistakes inc ompile time.

Parameters

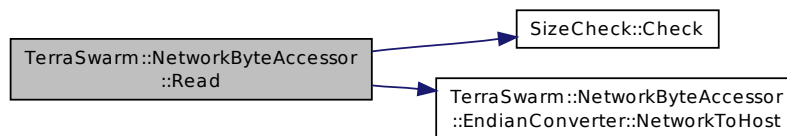
<i>value</i>	Reference to the value to be written to.
--------------	--

Returns

The value read from the address index.

Definition at line 178 of file NetworkByteAccessor.h.

Here is the call graph for this function:



8.23.2.3 `template<TByteIndex byteIndex, TDataSize dataSize> template<typename TInput > TInput
TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::Write (const TInput input) [inline]`

Writes a value to the given address index with correct byte order.

In addition, it checks for the size of the input type for mistakes in compile time.

Parameters

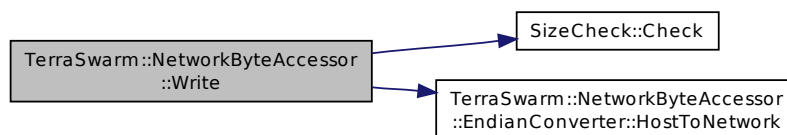
<i>input</i>	Value to be written.
--------------	----------------------

Returns

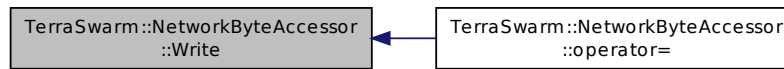
Actually written value.

Definition at line 146 of file NetworkByteAccessor.h.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- S2Sim/TerraswarmLibrary/[NetworkByteAccessor.h](#)

8.24 TerraSwarm::Synchronous::PriceProposal Class Reference

Price Proposal message sent by the controller to the clients to propose a price.

```
#include <PriceProposal.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef bool [TCheckResult](#)
Defines the message check result type.
- typedef unsigned int [TPrice](#)
Type for price signal.
- typedef unsigned int [TInterval](#)
Type for time interval.

Public Member Functions

- [~PriceProposal](#) (void)
Deletes the allocated message.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory contains the [PriceProposal](#) message.
- [TPrice GetPrice](#) (void) const
Reads the price field.
- [TInterval GetIntervalBegin](#) (void) const
Reads the IntervalBegin field in the message.
- [TInterval GetIntervalEnd](#) (void) const
Reads the IntervalEnd field in the message.

Static Public Member Functions

- static `PriceProposal * GetNewPriceProposal` (const `MessageHeader::TSenderId` senderId, const `MessageHeader::TReceiverId` receiverId, const `TPrice` price, const `TInterval` intervalBegin, const `TInterval` intervalEnd)
Creates a new `PriceProposal` message and allocates the necessary memory for it.
- static `TDataSize GetSize` (void)
Returns the size of this message.

Private Types

- enum `HeaderValues` { `MessageType` = 0x0003, `MessageId` = 0x0003 }
Message Header values.
- enum `FieldSizeValues` { `PriceSize` = sizeof(`TPrice`), `IntervalBeginSize` = sizeof(`TInterval`), `IntervalEndSize` = sizeof(`TInterval`), `TotalSize` = (`PriceSize` + `IntervalBeginSize` + `IntervalEndSize`) }
Size of the message fields.
- enum `FieldIndexValues` { `PriceIndex` = `MessageHeader::MessageHeaderSize`, `IntervalBeginIndex` = `PriceIndex` + `PriceSize`, `IntervalEndIndex` = `IntervalBeginIndex` + `IntervalBeginSize` }
Index values for the message fields.
- typedef `NetworkByteAccessor`
 < `PriceIndex`, `PriceSize` > `TPriceAccessor`
Accessor helper for Price field.
- typedef `NetworkByteAccessor`
 < `IntervalBeginIndex`,
 `IntervalBeginSize` > `TIntervalBeginAccessor`
Accessor helper for IntervalBegin field.
- typedef `NetworkByteAccessor`
 < `IntervalEndIndex`,
 `IntervalEndSize` > `TIntervalEndAccessor`
Accessor helper for IntervalEnd field.

Private Member Functions

- `PriceProposal` (void)
Empty private function to force the usage of the static method.

8.24.1 Detailed Description

Price Proposal message sent by the controller to the clients to propose a price.

Definition at line 22 of file `PriceProposal.h`.

8.24.2 Member Typedef Documentation

8.24.2.1 typedef bool TerraSwarm::Synchronous::PriceProposal::TCheckResult

Defines the message check result type.

Definition at line 38 of file `PriceProposal.h`.

8.24.2.2 `typedef unsigned int TerraSwarm::Synchronous::PriceProposal::TInterval`

Type for time interval.

Definition at line 57 of file PriceProposal.h.

8.24.2.3 `typedef NetworkByteAccessor<IntervalBeginIndex, IntervalBeginSize> TerraSwarm::Synchronous::PriceProposal::TIntervalBeginAccessor [private]`

Accessor helper for IntervalBegin field.

Definition at line 89 of file PriceProposal.h.

8.24.2.4 `typedef NetworkByteAccessor<IntervalEndIndex, IntervalEndSize> TerraSwarm::Synchronous::PriceProposal::TIntervalEndAccessor [private]`

Accessor helper for IntervalEnd field.

Definition at line 94 of file PriceProposal.h.

8.24.2.5 `typedef unsigned int TerraSwarm::Synchronous::PriceProposal::TPrice`

Type for price signal.

Definition at line 52 of file PriceProposal.h.

8.24.2.6 `typedef NetworkByteAccessor<PriceIndex, PriceSize> TerraSwarm::Synchronous::PriceProposal::TPriceAccessor [private]`

Accessor helper for Price field.

Definition at line 84 of file PriceProposal.h.

8.24.3 Member Enumeration Documentation

8.24.3.1 `enum TerraSwarm::Synchronous::PriceProposal::CheckResultValues`

Defines the values for TCheckResult.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 43 of file PriceProposal.h.

8.24.3.2 `enum TerraSwarm::Synchronous::PriceProposal::FieldIndexValues [private]`

Index values for the message fields.

Enumerator

PriceIndex

IntervalBeginIndex

IntervalEndIndex

Definition at line 74 of file PriceProposal.h.

8.24.3.3 `enum TerraSwarm::Synchronous::PriceProposal::FieldSizeValues` `[private]`

Size of the message fields.

Enumerator

PriceSize

IntervalBeginSize

IntervalEndSize

TotalSize

Definition at line 63 of file PriceProposal.h.

8.24.3.4 `enum TerraSwarm::Synchronous::PriceProposal::HeaderValues` `[private]`

Message Header values.

Enumerator

MessageType

MessageId

Definition at line 28 of file PriceProposal.h.

8.24.4 Constructor & Destructor Documentation

8.24.4.1 `TerraSwarm::Synchronous::PriceProposal::PriceProposal (void)` `[private]`

Empty private function to force the usage of the static method.

Definition at line 14 of file PriceProposal.cpp.

8.24.4.2 `TerraSwarm::Synchronous::PriceProposal::~~PriceProposal (void)`

Deletes the allocated message.

Definition at line 18 of file PriceProposal.cpp.

8.24.5 Member Function Documentation

8.24.5.1 `PriceProposal::TCheckResult TerraSwarm::Synchronous::PriceProposal::CheckMessage (void) const`

Checks whether the current memory contains the [PriceProposal](#) message.

Returns

Result of the check.

Definition at line 41 of file PriceProposal.cpp.

8.24.5.2 PriceProposal::TInterval TerraSwarm::Synchronous::PriceProposal::GetIntervalBegin (void) const

Reads the IntervalBegin field in the message.

Returns

IntervalBegin value in the message.

Definition at line 60 of file PriceProposal.cpp.

8.24.5.3 PriceProposal::TInterval TerraSwarm::Synchronous::PriceProposal::GetIntervalEnd (void) const

Reads the IntervalEnd field in the message.

Returns

IntervalEnd value in the message.

Definition at line 68 of file PriceProposal.cpp.

8.24.5.4 PriceProposal * TerraSwarm::Synchronous::PriceProposal::GetNewPriceProposal (const MessageHeader::TSenderId *senderId*, const MessageHeader::TReceiverId *receiverId*, const TPrice *price*, const TInterval *intervalBegin*, const TInterval *intervalEnd*) [static]

Creates a new [PriceProposal](#) message and allocates the necessary memory for it.

Warning

Deallocation is the job of the user.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.
<i>price</i>	Proposed price.
<i>intervalBegin</i>	Beginning of the price interval.
<i>intervalEnd</i>	End of the price interval.

Returns

Pointer to the allocated message.

Definition at line 24 of file PriceProposal.cpp.

8.24.5.5 PriceProposal::TPrice TerraSwarm::Synchronous::PriceProposal::GetPrice (void) const

Reads the price field.

Returns

Price value in the message.

Definition at line 52 of file PriceProposal.cpp.

8.24.5.6 TDataSize TerraSwarm::Synchronous::PriceProposal::GetSize (void) [static]

Returns the size of this message.

Returns

Size of a [PriceProposal](#) message.

Definition at line 76 of file PriceProposal.cpp.

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

- S2Sim/TerraswarmLibrary/[PriceProposal.h](#)
- S2Sim/TerraswarmLibrary/[PriceProposal.cpp](#)

8.25 TerraSwarm::Synchronous::SetCurrentPrice Class Reference

Set Current Price message sent for the Controller to the clients to set the current price and advance the time frame.

```
#include <SetCurrentPrice.h>
```

Public Types

- enum [CheckResultValues](#) { [Success](#) = (TCheckResult)true, [Fail](#) = (TCheckResult)false }
Defines the values for TCheckResult.
- typedef bool [TCheckResult](#)
Defines the message check result type.
- typedef unsigned int [TPrice](#)
Type for price signal.
- typedef unsigned int [TInterval](#)
Type for time interval.

Public Member Functions

- [~SetCurrentPrice](#) (void)
Deallocates the memory for the message.
- [TCheckResult CheckMessage](#) (void) const
Checks whether the current memory contains a [SetCurrentPrice](#) message.
- [TPrice GetPrice](#) (void) const

Reads the price form the current message.

- [TInterval GetIntervalBegin](#) (void) const
Reads the IntervalBegin field in the message.
- [TInterval GetIntervalEnd](#) (void) const
Reads the IntervalEnd field in the message.

Static Public Member Functions

- static [SetCurrentPrice](#) * [GetNewSetCurrentPrice](#) (const [MessageHeader::TSenderId](#) senderId, const [MessageHeader::TReceiverId](#) receiverId, const [TPrice](#) price, const [TInterval](#) intervalBegin, const [TInterval](#) intervalEnd)
Creates a new [SetCurrentPrice](#) message and allocates memory for it.
- static [TDataSize](#) [GetSize](#) (void)
Returns the size of a [SetCurrentPrice](#) message.

Private Types

- enum [HeaderValues](#) { [MessageType](#) = 0x0003, [Messageld](#) = 0x0002 }
Message header values.
- enum [FieldSizeValues](#) { [PriceSize](#) = sizeof([TPrice](#)), [IntervalBeginSize](#) = sizeof([TInterval](#)), [IntervalEndSize](#) = sizeof([TInterval](#)), [TotalSize](#) = ([PriceSize](#) + [IntervalBeginSize](#) + [IntervalEndSize](#)) }
Size values for the data fields.
- enum [FieldIndexValues](#) { [PriceIndex](#) = [MessageHeader::MessageHeaderSize](#), [IntervalBeginIndex](#) = [PriceIndex](#) + [PriceSize](#), [IntervalEndIndex](#) = [IntervalBeginIndex](#) + [IntervalBeginSize](#) }
Index values for the fata fields.
- typedef [NetworkByteAccessor](#) < [PriceIndex](#), [PriceSize](#) > [TPriceAccessor](#)
Accessor helper for the Price field.
- typedef [NetworkByteAccessor](#) < [IntervalBeginIndex](#), [IntervalBeginSize](#) > [TIntervalBeginAccessor](#)
Accessor helper for the IntervalBegin field.
- typedef [NetworkByteAccessor](#) < [IntervalEndIndex](#), [IntervalEndSize](#) > [TIntervalEndAccessor](#)
Accessor helper for the IntervalEnd field.

Private Member Functions

- [SetCurrentPrice](#) (void)
No use.

8.25.1 Detailed Description

Set Current Price message sent for the Controller to the clients to set the current price and advance the time frame.

Definition at line 22 of file [SetCurrentPrice.h](#).

8.25.2 Member Typedef Documentation

8.25.2.1 typedef bool TerraSwarm::Synchronous::SetCurrentPrice::TCheckResult

Defines the message check result type.

Definition at line 38 of file SetCurrentPrice.h.

8.25.2.2 typedef unsigned int TerraSwarm::Synchronous::SetCurrentPrice::TInterval

Type for time interval.

Definition at line 57 of file SetCurrentPrice.h.

8.25.2.3 typedef NetworkByteAccessor<IntervalBeginIndex, IntervalBeginSize> TerraSwarm::Synchronous::SetCurrentPrice::TIntervalBeginAccessor [private]

Accessor helper for the IntervalBegin field.

Definition at line 89 of file SetCurrentPrice.h.

8.25.2.4 typedef NetworkByteAccessor<IntervalEndIndex, IntervalEndSize> TerraSwarm::Synchronous::SetCurrentPrice::TIntervalEndAccessor [private]

Accessor helper for the IntervalEnd field.

Definition at line 94 of file SetCurrentPrice.h.

8.25.2.5 typedef unsigned int TerraSwarm::Synchronous::SetCurrentPrice::TPrice

Type for price signal.

Definition at line 52 of file SetCurrentPrice.h.

8.25.2.6 typedef NetworkByteAccessor<PriceIndex, PriceSize> TerraSwarm::Synchronous::SetCurrentPrice::TPriceAccessor [private]

Accessor helper for the Price field.

Definition at line 84 of file SetCurrentPrice.h.

8.25.3 Member Enumeration Documentation

8.25.3.1 enum TerraSwarm::Synchronous::SetCurrentPrice::CheckResultValues

Defines the values for TCheckResult.

Enumerator

Success Message is of correct type and id.

Fail Message has incorrect type or id.

Definition at line 43 of file SetCurrentPrice.h.

8.25.3.2 enum TerraSwarm::Synchronous::SetCurrentPrice::FieldIndexValues [private]

Index values for the data fields.

Enumerator

PriceIndex

IntervalBeginIndex

IntervalEndIndex

Definition at line 74 of file SetCurrentPrice.h.

8.25.3.3 enum TerraSwarm::Synchronous::SetCurrentPrice::FieldSizeValues [private]

Size values for the data fields.

Enumerator

PriceSize

IntervalBeginSize

IntervalEndSize

TotalSize

Definition at line 63 of file SetCurrentPrice.h.

8.25.3.4 enum TerraSwarm::Synchronous::SetCurrentPrice::HeaderValues [private]

Message header values.

Enumerator

MessageType

MessageId

Definition at line 28 of file SetCurrentPrice.h.

8.25.4 Constructor & Destructor Documentation

8.25.4.1 TerraSwarm::Synchronous::SetCurrentPrice::SetCurrentPrice (void) [private]

No use.

Private constructor to force usage of the static creation method.

Definition at line 14 of file SetCurrentPrice.cpp.

8.25.4.2 TerraSwarm::Synchronous::SetCurrentPrice::~~SetCurrentPrice (void)

Deallocates the memory for the message.

Definition at line 18 of file SetCurrentPrice.cpp.

8.25.5 Member Function Documentation

8.25.5.1 SetCurrentPrice::TCheckResult TerraSwarm::Synchronous::SetCurrentPrice::CheckMessage (void) const

Checks whether the current memory contains a [SetCurrentPrice](#) message.

Returns

Result of the check.

Definition at line 41 of file SetCurrentPrice.cpp.

8.25.5.2 SetCurrentPrice::TInterval TerraSwarm::Synchronous::SetCurrentPrice::GetIntervalBegin (void) const

Reads the IntervalBegin field in the message.

Returns

Value of IntervalBegin in the message.

Definition at line 60 of file SetCurrentPrice.cpp.

8.25.5.3 SetCurrentPrice::TInterval TerraSwarm::Synchronous::SetCurrentPrice::GetIntervalEnd (void) const

Reads the IntervalEnd field in the message.

Returns

Value of IntervalEnd in the message.

Definition at line 68 of file SetCurrentPrice.cpp.

8.25.5.4 SetCurrentPrice * TerraSwarm::Synchronous::SetCurrentPrice::GetNewSetCurrentPrice (const MessageHeader::TSenderId *senderId*, const MessageHeader::TReceiverId *receiverId*, const TPrice *price*, const TInterval *intervalBegin*, const TInterval *intervalEnd*) [static]

Creates a new [SetCurrentPrice](#) message and allocates memory for it.

Warning

Deallocation is the responsibility of the user.

Parameters

<i>senderId</i>	Id of the sender.
<i>receiverId</i>	Id of the receiver.
<i>price</i>	Price signal to be set.

<i>intervalBegin</i>	Beginning of the price interval.
<i>intervalEnd</i>	Ending of the price interval.

Returns

Returns a new allocated message.

Definition at line 24 of file SetCurrentPrice.cpp.

8.25.5.5 SetCurrentPrice::TPrice TerraSwarm::Synchronous::SetCurrentPrice::GetPrice (void) const

Reads the price form the current message.

Returns

Price value in the message.

Definition at line 52 of file SetCurrentPrice.cpp.

8.25.5.6 TDataSize TerraSwarm::Synchronous::SetCurrentPrice::GetSize (void) [static]

Returns the size of a [SetCurrentPrice](#) message.

Returns

<#return value description#>

Definition at line 76 of file SetCurrentPrice.cpp.

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

- S2Sim/TerraswarmLibrary/[SetCurrentPrice.h](#)
- S2Sim/TerraswarmLibrary/[SetCurrentPrice.cpp](#)

8.26 SizeCheck< checkedType, checkedSize, Reason > Class Template Reference

This class checks the size of a type with the given size and gives a compile error with the reason parameter is the check fails.

```
#include <CompileTimeCheckerLibrary.h>
```

Public Types

- enum { [Result](#) = CompileCheck<sizeof(checkedType) == checkedSize, Reason>::Result }
Enumeration for Result displaying.

Static Public Member Functions

- static void [Check](#) (void)
Empty class that will only compile if the check passes.

8.26.1 Detailed Description

template<typename checkedType, unsigned int checkedSize, typename Reason = class NoReason>class SizeCheck< checkedType, checkedSize, Reason >

This class checks the size of a type with the given size and gives a compile error with the reason parameter is the check fails.

Template Parameters

<i>checkedType</i>	A type that will be checked for size.
<i>checkedSize</i>	The size that the type should have.
<i>Reason</i>	Any class name that will be used for debug information only.

Definition at line 71 of file CompileTimeCheckerLibrary.h.

8.26.2 Member Enumeration Documentation

8.26.2.1 template<typename checkedType , unsigned int checkedSize, typename Reason = class NoReason> anonymous enum

Enumeration for Result displaying.

Enumerator

Result Result is the result of the compile check class.

Definition at line 77 of file CompileTimeCheckerLibrary.h.

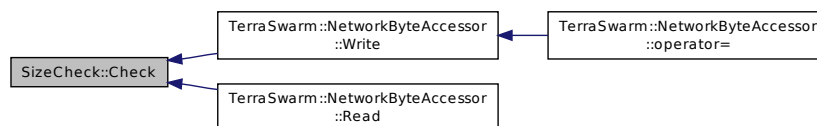
8.26.3 Member Function Documentation

8.26.3.1 template<typename checkedType , unsigned int checkedSize, typename Reason = class NoReason> static void SizeCheck< checkedType, checkedSize, Reason >::Check (void) [inline], [static]

Empty class that will only compile if the check passes.

Definition at line 88 of file CompileTimeCheckerLibrary.h.

Here is the caller graph for this function:



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

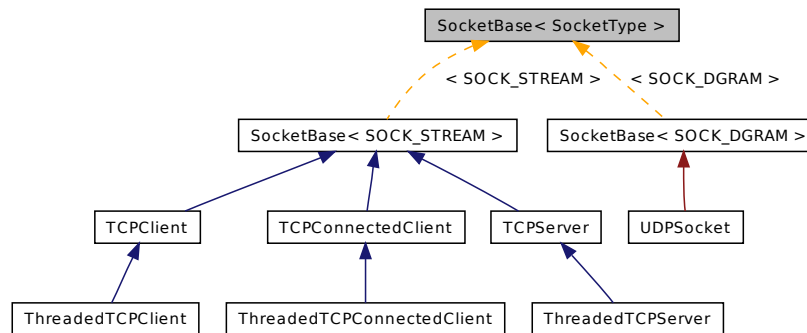
- S2Sim/TerraswarmLibrary/[CompileTimeCheckerLibrary.h](#)

8.27 SocketBase< SocketType > Class Template Reference

This class is an abstraction over the OS socket and defines a base class for socket opening and closing utilities.

```
#include <SocketBase.h>
```

Inheritance diagram for SocketBase< SocketType >:



Public Types

- typedef size_t `TNumberOfBytes`
Defines the type for number of bytes.
- typedef void * `TBuffer`
Defines the type for a buffer pointer.

Protected Types

- enum `SocketIdValues` { `InvalidSocketId` = (`TSocketId`)(-1) }
- typedef int `SOCKET`
Defines the type of a Socket handle.
- typedef `SOCKET` `TSocketId`
Redefines the socket handle type for multiplatform purposes.

Protected Member Functions

- `SocketBase` (const `TSocketId` socketId)
Constructor that receives a handle as input.
- `SocketBase` (const `SocketBase`< `SocketType` > ©)
Copies the handle of another socket.
- `SocketBase` (void)
Sets the handle to the invalid value.
- `~SocketBase` (void)
Closes the socket handle.

Protected Attributes

- [TSocketId m_socketId](#)

Handle to the socket used for OS utilities.

Private Member Functions

- void [OpenSocket](#) (void)

Opens a new socket and sets its reusable options.

- void [CloseSocket](#) (void)

Closes the socket handle.

8.27.1 Detailed Description

```
template<int SocketType>class SocketBase< SocketType >
```

This class is an abstraction over the OS socket and defines a base class for socket opening and closing utilities.

Template Parameters

<i>SocketType</i>	Type of the socket for UDP, TCP or other types.
-------------------	---

Definition at line 33 of file SocketBase.h.

8.27.2 Member Typedef Documentation

8.27.2.1 `template<int SocketType> typedef int SocketBase< SocketType >::SOCKET` [protected]

Defines the type of a Socket handle.

Definition at line 50 of file SocketBase.h.

8.27.2.2 `template<int SocketType> typedef void* SocketBase< SocketType >::TBuffer`

Defines the type for a buffer pointer.

Definition at line 44 of file SocketBase.h.

8.27.2.3 `template<int SocketType> typedef size_t SocketBase< SocketType >::TNumberOfBytes`

Defines the type for number of bytes.

Definition at line 39 of file SocketBase.h.

8.27.2.4 `template<int SocketType> typedef SOCKET SocketBase< SocketType >::TSocketId` [protected]

Redefines the socket handle type for multiplatform purposes.

Definition at line 55 of file SocketBase.h.

8.27.3 Member Enumeration Documentation

8.27.3.1 `template<int SocketType> enum SocketBase::SocketIdValues` `[protected]`

Special values for the TSocketId.

Enumerator

InvalidSocketId Defines an invalid socket handle.

Definition at line 60 of file SocketBase.h.

8.27.4 Constructor & Destructor Documentation

8.27.4.1 `template<int SocketType> SocketBase< SocketType >::SocketBase (const TSocketId socketId)` `[inline]`, `[protected]`

Constructor that receives a handle as input.

It does not open a new socket.

Parameters

<i>socketId</i>	Handle of a socket.
-----------------	---------------------

Definition at line 71 of file SocketBase.h.

8.27.4.2 `template<int SocketType> SocketBase< SocketType >::SocketBase (const SocketBase< SocketType > & copy)` `[inline]`, `[protected]`

Copies the handle of another socket.

Parameters

<i>copy</i>	Socket to be copied.
-------------	----------------------

Definition at line 78 of file SocketBase.h.

8.27.4.3 `template<int SocketType> SocketBase< SocketType >::SocketBase (void)` `[protected]`

Sets the handle to the invalid value.

Definition at line 111 of file SocketBase.h.

Here is the call graph for this function:



8.27.4.4 `template<int SocketType> SocketBase< SocketType >::~~SocketBase (void)` [protected]

Closes the socket handle.

Definition at line 117 of file SocketBase.h.

8.27.5 Member Function Documentation

8.27.5.1 `template<int SocketType> void SocketBase< SocketType >::CloseSocket (void)` [private]

Closes the socket handle.

Definition at line 144 of file SocketBase.h.

8.27.5.2 `template<int SocketType> void SocketBase< SocketType >::OpenSocket (void)` [private]

Opens a new socket and sets its reusable options.

Definition at line 124 of file SocketBase.h.

Here is the caller graph for this function:



8.27.6 Member Data Documentation

8.27.6.1 `template<int SocketType> TSocketId SocketBase< SocketType >::m_socketId` [protected]

Handle to the socket used for OS utilities.

Definition at line 94 of file SocketBase.h.

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

- S2Sim/SocketLibrary/[SocketBase.h](#)

8.28 SystemManager Class Reference

Manages the various components of the system and timing.

```
#include <SystemManager.h>
```

Public Types

- enum [SystemModeValues](#) { [SimulationMode](#) = (TSystemMode)0x0001, [RealTimeMode](#) = (TSystemMode)0x0002 }

- Defines the values for the TSystemMode type.*
 - typedef unsigned int [TSystemTime](#)
 - Defines the type for System Time in epoch format.*
 - typedef unsigned short [TSystemMode](#)
 - Defines the working mode of the system.*
 - typedef [MessageHeader::TId](#) [TClientId](#)
 - Redefines the unique client id type for rapid development.*
 - typedef [Asynchronous::ClientConnectionRequest::TClientName](#) [TClientName](#)
 - Redefines the object name for rapid development.*
 - typedef [Asynchronous::ClientData::TDataPoint](#) [TDataPoint](#)
 - Redefines the data point type for rapid development.*
 - typedef [Asynchronous::ClientData::TNumberOfDataPoints](#) [TNumberOfDataPoints](#)
 - Redefines the number of data points type for rapid development.*
 - typedef [TDataPoint](#) [TVoltage](#)
 - Defines the voltage information type.*
 - typedef [TDataPoint](#) [TWattage](#)
 - Defines the wattage consumption type.*

Public Member Functions

- [TSystemTime](#) [GetSystemTime](#) (void) const
 - Returns the current system time.*
- [TSystemMode](#) [GetSystemMode](#) (void) const
 - Returns the current system working mode.*
- void [RegisterData](#) (const [TClientId](#) clientId, const [TSystemTime](#) startTime, const [TSystemTime](#) resolution, const [TNumberOfDataPoints](#) numberOfDataPoints, [TDataPoint](#) *dataPoints)
 - Used to register multiple consumption information.*
- void [RegisterData](#) (const [TClientId](#) clientId, [TDataPoint](#) dataPoint)
 - Used to register a single consumption information for the next time step.*
- void [AdvanceTimeStep](#) (void)
 - Main time iteration of the system.*

Private Types

- typedef std::map< [TClientId](#), [TDataPoint](#) > [TDataMap](#)
 - Defines the mapping from ClientId->Consumption.*
- typedef std::map< [TSystemTime](#), [TDataMap](#) > [TSystemMap](#)
 - Defines the mapping from Time->(ClientId->Consumption).*

Private Member Functions

- [SystemManager](#) (void)
 - Private constructor for singleton implementation.*

Private Attributes

- [TSystemTime m_systemTime](#)
The current system time, incremented at each time step.
- [TSystemMode m_systemMode](#)
The current working mode of the system.
- [TSystemMap m_systemMap](#)
This variable contains the consumption information for all clients for the future.

Friends

- [SystemManager](#) & [GetSystemManager](#) (void)
Friend function to implement the singleton.

8.28.1 Detailed Description

Manages the various components of the system and timing.

This class coordinates the other components within S2Sim and manages the timing of consumption information from both async and synchronous clients.

Definition at line 37 of file SystemManager.h.

8.28.2 Member Typedef Documentation

8.28.2.1 `typedef MessageHeader::TId SystemManager::TClientId`

Redefines the unique client id type for rapid development.

Definition at line 69 of file SystemManager.h.

8.28.2.2 `typedef Asynchronous::ClientConnectionRequest::TClientName SystemManager::TClientName`

Redefines the object name for rapid development.

Definition at line 74 of file SystemManager.h.

8.28.2.3 `typedef std::map<TClientId, TDataPoint> SystemManager::TDataMap [private]`

Defines the mapping from ClientId->Consumption.

Definition at line 100 of file SystemManager.h.

8.28.2.4 `typedef Asynchronous::ClientData::TDataPoint SystemManager::TDataPoint`

Redefines the data point type for rapid development.

Definition at line 79 of file SystemManager.h.

8.28.2.5 `typedef Asynchronous::ClientData::TNumberOfDataPoints SystemManager::TNumberOfDataPoints`

Redefines the number of data points type for rapid development.

Definition at line 84 of file SystemManager.h.

8.28.2.6 `typedef std::map<TSystemTime, TDataMap> SystemManager::TSystemMap` `[private]`

Defines the mapping from Time->(ClientId->Consumption).

Definition at line 105 of file SystemManager.h.

8.28.2.7 `typedef unsigned short SystemManager::TSystemMode`

Defines the working mode of the system.

Definition at line 55 of file SystemManager.h.

8.28.2.8 `typedef unsigned int SystemManager::TSystemTime`

Defines the type for System Time in epoch format.

Definition at line 50 of file SystemManager.h.

8.28.2.9 `typedef TDataPoint SystemManager::TVoltage`

Defines the voltage information type.

Definition at line 89 of file SystemManager.h.

8.28.2.10 `typedef TDataPoint SystemManager::TWattage`

Defines the wattage consumption type.

Definition at line 94 of file SystemManager.h.

8.28.3 Member Enumeration Documentation

8.28.3.1 `enum SystemManager::SystemModeValues`

Defines the values for the TSystemMode type.

Enumerator

SimulationMode Indicates that the system expects an external signal to start.

RealTimeMode Indicates that the system is working in real-time, even without external signaling. Not implemented.

Definition at line 60 of file SystemManager.h.

8.28.4 Constructor & Destructor Documentation

8.28.4.1 SystemManager::SystemManager (void) [private]

Private constructor for singleton implementation.

Definition at line 19 of file SystemManager.cpp.

8.28.5 Member Function Documentation

8.28.5.1 void SystemManager::AdvanceTimeStep (void)

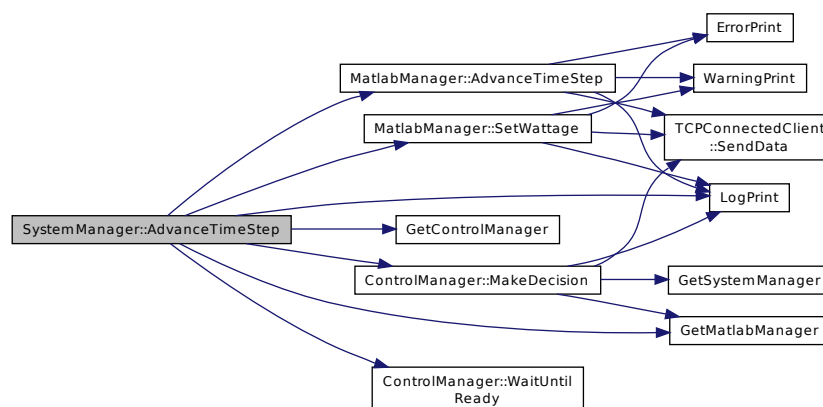
Main time iteration of the system.

This method is the main time iteration of the whole system. The workflow is as follows:

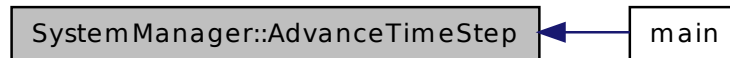
- Delete the previous time step information.
- Get the current time consumption information.
- Set the consumption information in OpenDSS.
- Advance the time in OpenDSS.
- Invoke the External Controller for a decision.
- Wait for the External Controller to finish its decision.

Definition at line 63 of file SystemManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.28.5.2 `TSystemMode SystemManager::GetSystemMode (void) const` `[inline]`

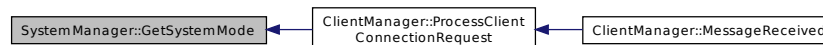
Returns the current system working mode.

Returns

Current system working mode.

Definition at line 147 of file `SystemManager.h`.

Here is the caller graph for this function:



8.28.5.3 `TSystemTime SystemManager::GetSystemTime (void) const` `[inline]`

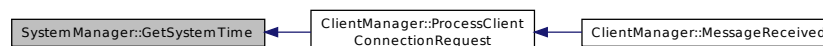
Returns the current system time.

Returns

Current system time.

Definition at line 136 of file `SystemManager.h`.

Here is the caller graph for this function:



8.28.5.4 `void SystemManager::RegisterData (const TClientId clientId, const TSystemTime startTime, const TSystemTime resolution, const TNumberOfDataPoints numberOfDataPoints, TDataPoint * dataPoints)`

Used to register multiple consumption information.

This method is mostly used for asynchronous consumption registration. Multiple consumption data points are fed into the [SystemManager::m_systemMap](#).

Parameters

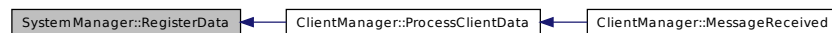
<i>clientId</i>	Unique client id for the consumer.
<i>startTime</i>	Starting time of the consumption map.
<i>resolution</i>	Time resolution between consecutive consumptions.
<i>numberOfDataPoints</i>	Number of consumption data points.
<i>dataPoints</i>	Buffer containing consumption data points.

Definition at line 27 of file SystemManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.28.5.5 `void SystemManager::RegisterData (const TClientId clientId, TDataPoint dataPoint)`

Used to register a single consumption information for the next time step.

The method is mostly used for synchronous consumption registration. The consumption for the next time interval is registered.

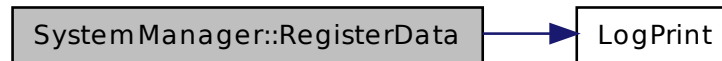
Parameters

<i>clientId</i>	Unique client id of the consumer.
-----------------	-----------------------------------

<i>dataPoint</i>	Consumption for the next time interval.
------------------	---

Definition at line 53 of file SystemManager.cpp.

Here is the call graph for this function:



8.28.6 Friends And Related Function Documentation

8.28.6.1 SystemManager& GetSystemManager (void) [friend]

Friend function to implement the singleton.

Returns

Returns the only instance of [SystemManager](#).
The only instance of [SystemManager](#).

Definition at line 13 of file SystemManager.cpp.

8.28.7 Member Data Documentation

8.28.7.1 TSystemMap SystemManager::m_systemMap [private]

This variable contains the consumption information for all clients for the future.

It is a mapping from time->Client/Data. This allows us to get the consumption of any client at any time. This simplifies the asynchronous client consumption drastically.

Definition at line 121 of file SystemManager.h.

8.28.7.2 TSystemMode SystemManager::m_systemMode [private]

The current working mode of the system.

Definition at line 116 of file SystemManager.h.

8.28.7.3 TSystemTime SystemManager::m_systemTime [private]

The current system time, incremented at each time step.

Definition at line 111 of file SystemManager.h.

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

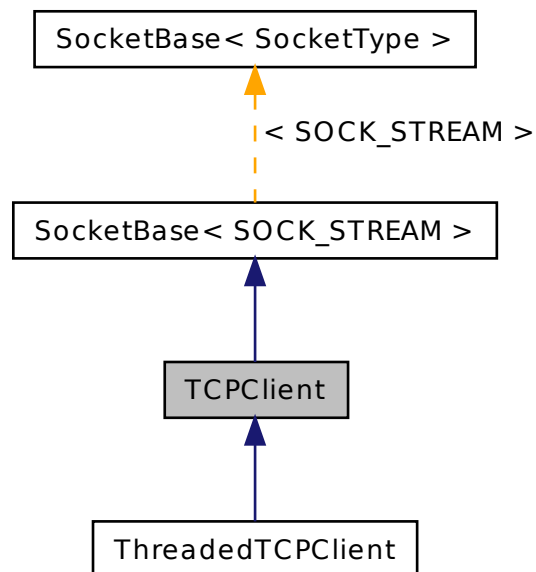
- [S2Sim/SystemManager.h](#)
- [S2Sim/SystemManager.cpp](#)

8.29 TCPClient Class Reference

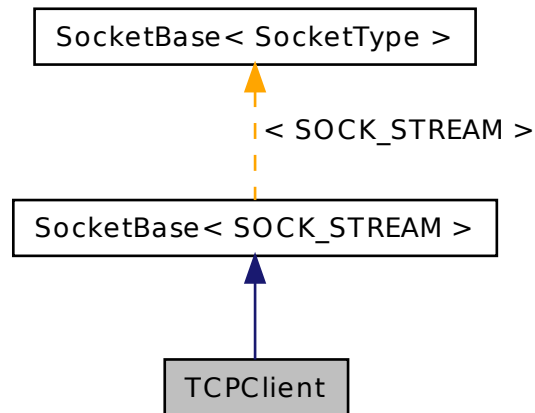
This class defines a TCP client that can connect to a TCP server and communicate.

```
#include <TCPClient.h>
```

Inheritance diagram for TCPClient:



Collaboration diagram for TCPClient:



Public Member Functions

- `TCPClient` (void)
No use.
- `~TCPClient` (void)
No use.
- bool `Connect` (IPAddress &address)
Starts a TCP connection to a server.
- `TNumberOfBytes SendData` (const TBuffer buffer, const TNumberOfBytes length)
Sends data to the server.
- `TNumberOfBytes ReceiveData` (TBuffer buffer, const TNumberOfBytes receptionLength)
Received data from the server.

Private Types

- typedef `SocketBase< SOCK_STREAM > TBaseType`
Defines the base type for rapid development.

Additional Inherited Members

8.29.1 Detailed Description

This class defines a TCP client that can connect to a TCP server and communicate.

All OS related utilities are abstracted.

Definition at line 17 of file TCPClient.h.

8.29.2 Member Typedef Documentation

8.29.2.1 typedef SocketBase<SOCK_STREAM> TCPClient::TBaseType [private]

Defines the base type for rapid development.

Definition at line 66 of file TCPClient.h.

8.29.3 Constructor & Destructor Documentation

8.29.3.1 TCPClient::TCPClient (void)

No use.

Definition at line 10 of file TCPClient.cpp.

8.29.3.2 TCPClient::~~TCPClient (void)

No use.

Definition at line 14 of file TCPClient.cpp.

8.29.4 Member Function Documentation

8.29.4.1 bool TCPClient::Connect (IPAddress & address)

Starts a TCP connection to a server.

Parameters

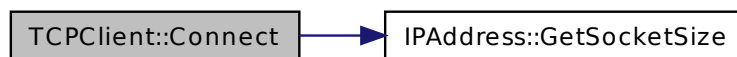
<i>address</i>	IPAddress of the server.
----------------	--------------------------

Returns

Success of the communication.

Definition at line 31 of file TCPClient.cpp.

Here is the call graph for this function:



8.29.4.2 TCPClient::TNumberOfBytes TCPClient::ReceiveData (TBuffer buffer, const TNumberOfBytes receptionLength)

Received data from the server.

This is a blocking call.

Parameters

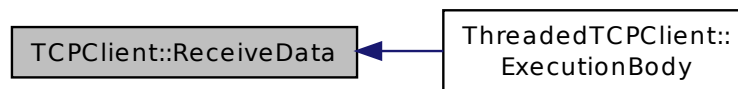
<i>buffer</i>	Buffer that will hold the received data.
<i>receptionLength</i>	Length of the buffer and maximum receivable data.

Returns

The actual number of bytes received.

Definition at line 25 of file TCPClient.cpp.

Here is the caller graph for this function:



8.29.4.3 TCPClient::TNumberOfBytes TCPClient::SendData (const TBuffer *buffer*, const TNumberOfBytes *length*)

Sends data to the server.

Parameters

<i>buffer</i>	Buffer, holding the data to be sent.
<i>length</i>	Length of the data to be sent.

Returns

Actually sent number of bytes.

Definition at line 19 of file TCPClient.cpp.

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

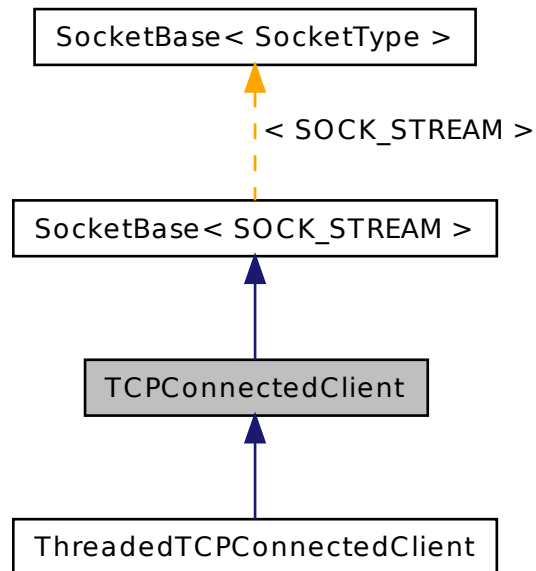
- S2Sim/SocketLibrary/[TCPClient.h](#)
- S2Sim/SocketLibrary/[TCPClient.cpp](#)

8.30 TCPConnectedClient Class Reference

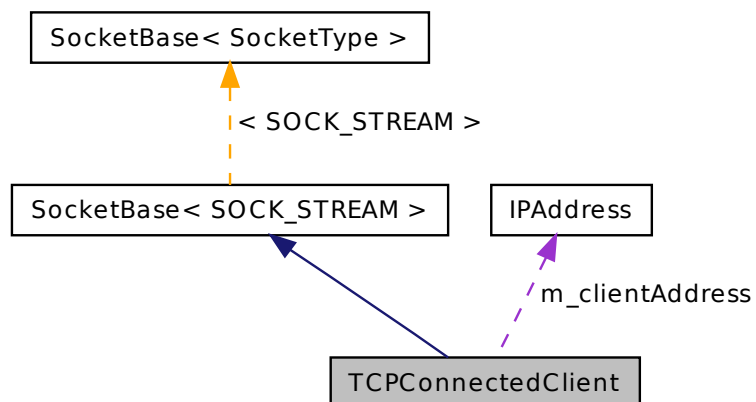
Manages the connection to a connected client on the server side.

```
#include <TCPConnectedClient.h>
```

Inheritance diagram for TCPConnectedClient:



Collaboration diagram for TCPConnectedClient:



Public Member Functions

- [TCPConnectedClient](#) (const [TSocketId](#) socketId, [IPAddress](#) &clientAddress)
Initializes the class with a ready socket and address.
- [TCPConnectedClient](#) (const [TCPConnectedClient](#) ©)
Copies the contents of another instance.
- [~TCPConnectedClient](#) (void)
Not used.
- [TNumberOfBytes SendData](#) (const [TBuffer](#) buffer, const [TNumberOfBytes](#) length)
Sends data to the connected client.
- [TNumberOfBytes ReceiveData](#) ([TBuffer](#) buffer, const [TNumberOfBytes](#) receptionLength)
Receives data from the connected client.
- [IPAddress GetClientAddress](#) (void) const
Returns the IP address of the client.

Private Types

- typedef [SocketBase](#)< SOCK_STREAM > [TBaseType](#)
Defines the base class for rapid development.

Private Attributes

- [IPAddress m_clientAddress](#)
IP Address of the client.

Additional Inherited Members

8.30.1 Detailed Description

Manages the connection to a connected client on the server side.

When a client is accepted, an instance of the class is initialized to further maintain the connection.

Definition at line 17 of file TCPConnectedClient.h.

8.30.2 Member Typedef Documentation

8.30.2.1 typedef [SocketBase](#)<SOCK_STREAM> [TCPConnectedClient::TBaseType](#) [private]

Defines the base class for rapid development.

Definition at line 72 of file TCPConnectedClient.h.

8.30.3 Constructor & Destructor Documentation

8.30.3.1 [TCPConnectedClient::TCPConnectedClient](#) (const [TSocketId](#) socketId, [IPAddress](#) & clientAddress)

Initializes the class with a ready socket and address.

Parameters

<i>socketId</i>	Handle to the socket.
<i>clientAddress</i>	IPAddress of the connected client.

Definition at line 10 of file TCPConnectedClient.cpp.

8.30.3.2 TCPConnectedClient::TCPConnectedClient (const TCPConnectedClient & copy)

Copies the contents of another instance.

Parameters

<i>copy</i>	Instance to be copied.
-------------	------------------------

Definition at line 16 of file TCPConnectedClient.cpp.

8.30.3.3 TCPConnectedClient::~~TCPConnectedClient (void)

Not used.

Definition at line 20 of file TCPConnectedClient.cpp.

8.30.4 Member Function Documentation

8.30.4.1 IPAddress TCPConnectedClient::GetClientAddress (void) const

Returns the IP address of the client.

Returns

IP address of the client.

Definition at line 38 of file TCPConnectedClient.cpp.

8.30.4.2 TCPConnectedClient::TNumberOfBytes TCPConnectedClient::ReceiveData (TBuffer buffer, const TNumberOfBytes receptionLength)

Receives data from the connected client.

This is a blocking call.

Parameters

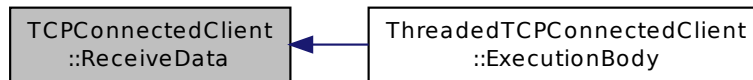
<i>buffer</i>	Buffer, where the received data will be stored int.
<i>receptionLength</i>	Length of the buffer and the maximum receivable data size.

Returns

The actual number of bytes received.

Definition at line 31 of file TCPConnectedClient.cpp.

Here is the caller graph for this function:



8.30.4.3 TCPConnectedClient::TNumberOfBytes TCPConnectedClient::SendData (const TBuffer *buffer*, const TNumberOfBytes *length*)

Sends data to the connected client.

Parameters

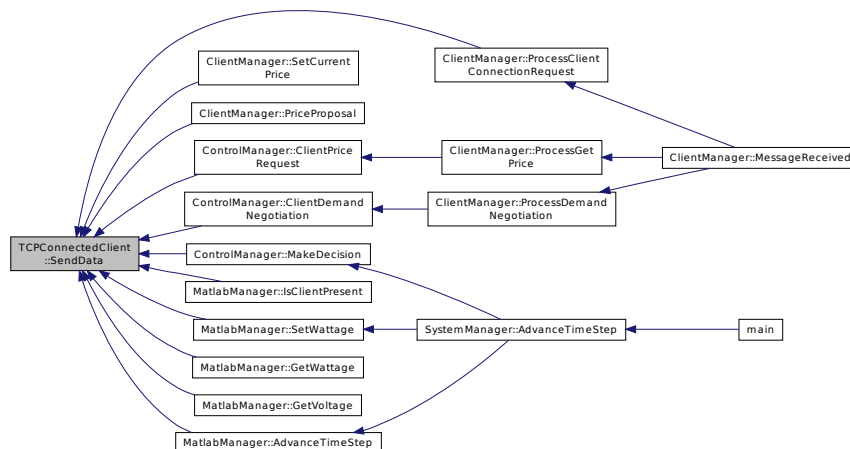
<i>buffer</i>	Buffer, holding the data to be sent.
<i>length</i>	Length of the data to be sent.

Returns

Actual number of bytes sent.

Definition at line 25 of file TCPConnectedClient.cpp.

Here is the caller graph for this function:



8.30.5 Member Data Documentation

8.30.5.1 IPAddress TCPConnectedClient::m_clientAddress [private]

IP Address of the client.

Definition at line 78 of file TCPConnectedClient.h.

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

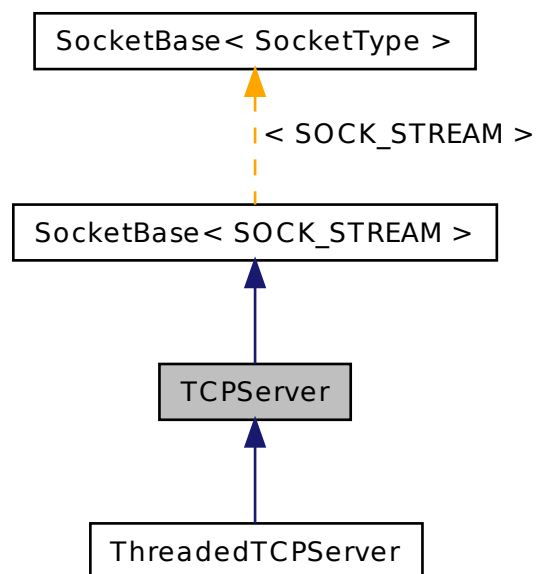
- S2Sim/SocketLibrary/[TCPConnectedClient.h](#)
- S2Sim/SocketLibrary/[TCPConnectedClient.cpp](#)

8.31 TCPServer Class Reference

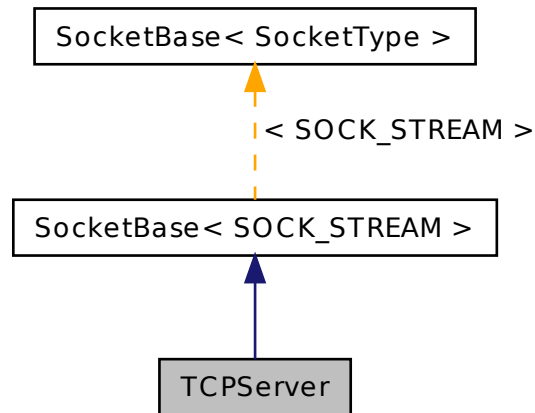
Defines a TCP server that can listen to connection attempts and can accept them for communication.

```
#include <TCPServer.h>
```

Inheritance diagram for TCPServer:



Collaboration diagram for TCPServer:



Public Member Functions

- `TCPServer` (void)
Not used.
- `~TCPServer` (void)
Not used.
- void `SetAddress` (IPAddress &address)
Sets the IP address of the server.
- void `SetPort` (const IPAddress::TPort port)
Sets the listening port of the server.
- bool `Listen` (void)
This method makes the server listen to incoming connection attempts.
- `ThreadedTCPConnectedClient * Accept` (void)
This method makes the TCP server accept any incoming connection attempt.

Private Types

- typedef `SocketBase< SOCK_STREAM > TBaseType`
Defines the type of the base class for rapid development.

Additional Inherited Members

8.31.1 Detailed Description

Defines a TCP server that can listen to connection attempts and can accept them for communication.

Definition at line 18 of file TCPServer.h.

8.31.2 Member Typedef Documentation

8.31.2.1 `typedef SocketBase<SOCK_STREAM> TCPServer::TBaseType` [private]

Defines the type of the base class for rapid development.

Definition at line 67 of file TCPServer.h.

8.31.3 Constructor & Destructor Documentation

8.31.3.1 `TCPServer::TCPServer (void)`

Not used.

Definition at line 10 of file TCPServer.cpp.

8.31.3.2 `TCPServer::~~TCPServer (void)`

Not used.

Definition at line 14 of file TCPServer.cpp.

8.31.4 Member Function Documentation

8.31.4.1 `ThreadedTCPConnectedClient * TCPServer::Accept (void)`

This method makes the TCP server accept any incoming connection attempt.

This is a blocking call. If any client is accepted, a [ThreadedTCPConnectedClient](#) instance is returned for client management and communication.

Returns

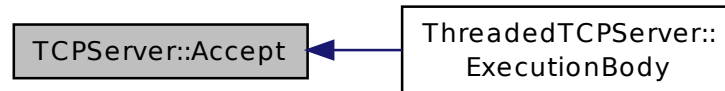
An instance of [ThreadedTCPConnectedClient](#) for client management and communication.

Definition at line 48 of file TCPServer.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.31.4.2 bool TCPServer::Listen (void)

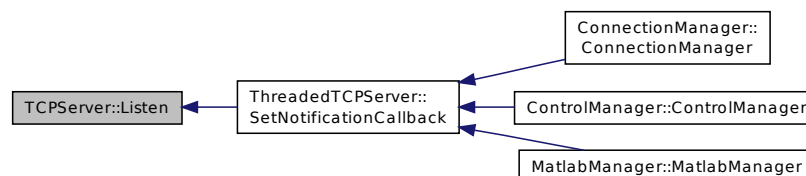
This method makes the server listen to incoming connection attempts.

Returns

Returns the success of the listen attempt.

Definition at line 37 of file `TCPServer.cpp`.

Here is the caller graph for this function:



8.31.4.3 void TCPServer::SetAddress (IPAddress & address)

Sets the IP address of the server.

This method is not recommended as the IP address of the computer is mostly set, while only the port number is required. If you are not sure, use [TCPServer::SetPort\(\)](#) instead.

Parameters

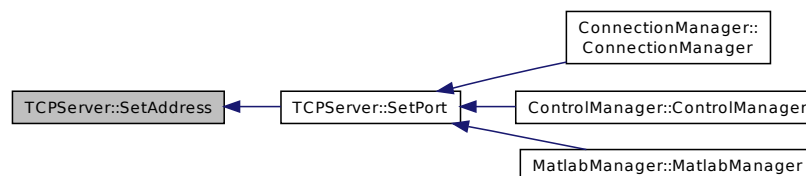
<i>address</i>	IP Address to be used.
----------------	------------------------

Definition at line 19 of file `TCPServer.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.31.4.4 void TCPServer::SetPort (const IPAddress::TPort port)

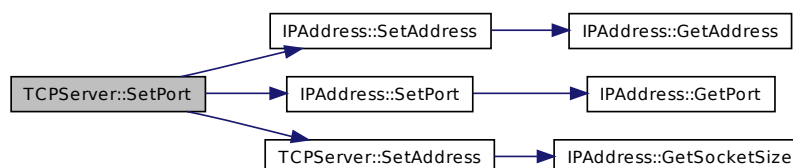
Sets the listening port of the server.

Parameters

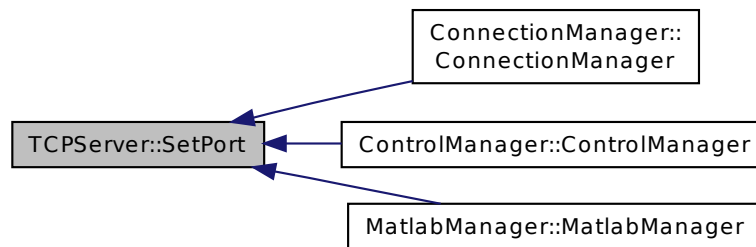
<i>port</i>	Port to be listened to for incoming connection attempts.
-------------	--

Definition at line 28 of file `TCPServer.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

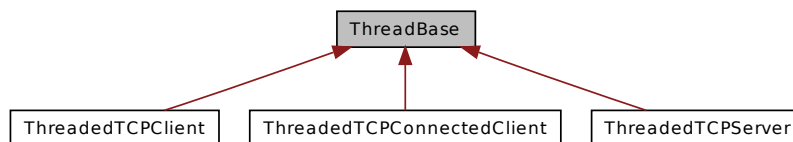
- S2Sim/SocketLibrary/[TCPServer.h](#)
- S2Sim/SocketLibrary/[TCPServer.cpp](#)

8.32 ThreadBase Class Reference

Provides a base class that can be inherited from to gain threading capabilities.

```
#include <PosixThreadBase.h>
```

Inheritance diagram for ThreadBase:



Classes

- struct [InputStructure](#)
Special Input structure sent to the wrapper function: [PosixThreadCover\(\)](#).

Public Types

- enum [StartedValues](#) { [isStarted](#) = (TStarted)true, [isNotStarted](#) = (TStarted)false, [isStarted](#) = (TStarted)true, [isNotStarted](#) = (TStarted)false }
Defines values for the TStarted type.

- enum [StartedValues](#) { [isStarted](#) = (TStarted)true, [isNotStarted](#) = (TStarted)false, [isStarted](#) = (TStarted)true, [isNotStarted](#) = (TStarted)false }
- Defines values for the TStarted type.*
- typedef size_t [TStackSize](#)
- Defines the Stack Size type for the thread.*
- typedef bool [TStarted](#)
- Defines a type indicating whether the thread is started.*
- typedef size_t [TStackSize](#)
- Defines the Stack Size type for the thread.*
- typedef bool [TStarted](#)
- Defines a type indicating whether the thread is started.*

Public Member Functions

- [ThreadBase](#) (void)
- Initializes values but does not create a thread.*
- [ThreadBase](#) (void *input)
- Initializes values and starts the thread with the given input.*
- virtual [~ThreadBase](#) (void)
- Stops the thread and frees the allocated attribute.*
- [ThreadBase](#) (const [ThreadBase](#) ©)
- Copies the contents of another thread.*
- void [SetStackSize](#) (const [TStackSize](#) stackSize)
- Sets the stack size of the thread.*
- [TStackSize](#) [GetStackSize](#) (void) const
- Gets the stack size of the thread.*
- void [StartThread](#) (void *input)
- Creates a thread and starts executing with the given input.*
- [ThreadBase](#) (void)
- Initializes the values but does not create a thread.*
- virtual [~ThreadBase](#) (void)
- Closes the thread handle.*
- [ThreadBase](#) (const [ThreadBase](#) ©)
- Copies the contents of the other class.*
- void [SetStackSize](#) (const [TStackSize](#) stackSize)
- Sets the stack size of the thread.*
- [TStackSize](#) [GetStackSize](#) (void) const
- Gets the stack size of the thread.*
- void [StartThread](#) (void *input)
- Creates a thread and starts executing with the given input.*

Private Types

- typedef pthread_t [TThreadId](#)
Type for the thread handle used for the OS utilities.
- typedef pthread_attr_t [TThreadAttribute](#)
Type for the thread attributes.
- typedef DWORD [TThreadId](#)
Defines the Thread Id type.
- typedef HANDLE [TThreadHandle](#)
Defines the Thread Handle type.

Private Member Functions

- void * [ExecuteThread](#) (void *input)
Function called by the wrapper function [PosixThreadCover\(\)](#). The method executes the overridden execution body method with the actual input and returns the output of the execution body through the OS.
- virtual void * [ExecutionBody](#) (void *input)=0
Purely virtual method to be overridden by the inheriting class to implement its own thread body.
- void * [ExecuteThread](#) (void *input)
Function called by the wrapper function [PosixThreadCover\(\)](#). The method executes the overridden execution body method with the actual input and returns the output of the execution body through the OS.
- virtual void * [ExecutionBody](#) (void *)=0
Purely virtual method to be overridden by the inheriting class to implement its own thread body.

Private Attributes

- [TStarted m_started](#)
Indicates whether the thread is started to execute.
- [TThreadId m_threadId](#)
Variable holding the handle to the thread.
- [TThreadAttribute m_threadAttribute](#)
Variable holding the attributes of the thread.
- [TThreadHandle m_threadHandle](#)
Variable holding the handle to the thread.
- [TStackSize m_stackSize](#)
Variable holding the stack size of the thread.

Friends

- void * [PosixThreadCover](#) (void *)
Friend wrapper function that is actually called by the OS as the thread body.
- DWORD WINAPI [WindowsThreadCover](#) (LPVOID)
Friend wrapper function that is actually called by the OS as the thread body.

8.32.1 Detailed Description

Provides a base class that can be inherited from to gain threading capabilities.

Any class that wants to have threading capability can inherit from this class and override its purely virtual method [ThreadBase::ExecutionBody](#). This overridden method will be the threads main body.

Definition at line 30 of file PosixThreadBase.h.

8.32.2 Member Typedef Documentation

8.32.2.1 `typedef size_t ThreadBase::TStackSize`

Defines the Stack Size type for the thread.

Definition at line 81 of file PosixThreadBase.h.

8.32.2.2 `typedef size_t ThreadBase::TStackSize`

Defines the Stack Size type for the thread.

Definition at line 82 of file WindowsThreadBase.h.

8.32.2.3 `typedef bool ThreadBase::TStarted`

Defines a type indicating whether the thread is started.

Definition at line 86 of file PosixThreadBase.h.

8.32.2.4 `typedef bool ThreadBase::TStarted`

Defines a type indicating whether the thread is started.

Definition at line 87 of file WindowsThreadBase.h.

8.32.2.5 `typedef pthread_attr_t ThreadBase::TThreadAttribute` `[private]`

Type for the thread attributes.

Definition at line 48 of file PosixThreadBase.h.

8.32.2.6 `typedef HANDLE ThreadBase::TThreadHandle` `[private]`

Defines the Thread Handle type.

Definition at line 49 of file WindowsThreadBase.h.

8.32.2.7 `typedef pthread_t ThreadBase::TThreadId` `[private]`

Type for the thread handle used for the OS utilities.

Definition at line 43 of file PosixThreadBase.h.

8.32.2.8 `typedef DWORD ThreadBase::TThreadId` `[private]`

Defines the Thread Id type.

Definition at line 44 of file WindowsThreadBase.h.

8.32.3 Member Enumeration Documentation

8.32.3.1 `enum ThreadBase::StartedValues`

Defines values for the TStarted type.

Enumerator

isStarted Indicates that the thread is started.

isNotStarted Indicates that the thread is not started yet.

isStarted Indicates that the thread is started.

isNotStarted Indicates that the thread is not started yet.

Definition at line 91 of file PosixThreadBase.h.

8.32.3.2 `enum ThreadBase::StartedValues`

Defines values for the TStarted type.

Enumerator

isStarted Indicates that the thread is started.

isNotStarted Indicates that the thread is not started yet.

isStarted Indicates that the thread is started.

isNotStarted Indicates that the thread is not started yet.

Definition at line 92 of file WindowsThreadBase.h.

8.32.4 Constructor & Destructor Documentation

8.32.4.1 `ThreadBase::ThreadBase (void)` `[inline]`

Initializes values but does not create a thread.

Definition at line 139 of file PosixThreadBase.h.

8.32.4.2 `ThreadBase::ThreadBase (void * input)` `[inline]`

Initializes values and starts the thread with the given input.

Parameters

<i>input</i>	Input to the thread body.
--------------	---------------------------

Definition at line 150 of file PosixThreadBase.h.

Here is the call graph for this function:



8.32.4.3 virtual ThreadBase::~~ThreadBase (void) [inline],[virtual]

Stops the thread and frees the allocates attribute.

Definition at line 160 of file PosixThreadBase.h.

8.32.4.4 ThreadBase::ThreadBase (const ThreadBase & copy) [inline]

Copies the contents of another thread.

Parameters

<i>copy</i>	Instance to be copied.
-------------	------------------------

Definition at line 174 of file PosixThreadBase.h.

8.32.4.5 ThreadBase::ThreadBase (void) [inline]

Initializes the values but does not create a thread.

Definition at line 145 of file WindowsThreadBase.h.

8.32.4.6 virtual ThreadBase::~~ThreadBase (void) [inline],[virtual]

Closes the thread handle.

Definition at line 155 of file WindowsThreadBase.h.

8.32.4.7 ThreadBase::ThreadBase (const ThreadBase & copy) [inline]

Copies the contents of the other class.

Parameters

<i>copy</i>	Class to be copied.
-------------	---------------------

Definition at line 168 of file WindowsThreadBase.h.

8.32.5 Member Function Documentation

8.32.5.1 void* ThreadBase::ExecuteThread (void * *input*) [inline], [private]

Function called by the wrapper function [PosixThreadCover\(\)](#).The method executes the overridden execution body method with the actual input and returns the output of the execution body through the OS.

Parameters

<i>input</i>	Actual input to the thread body.
--------------	----------------------------------

Returns

Returns the exact thing that the thread body returns.

Definition at line 121 of file PosixThreadBase.h.

Here is the call graph for this function:



8.32.5.2 void* ThreadBase::ExecuteThread (void * *input*) [inline], [private]

Function called by the wrapper function [PosixThreadCover\(\)](#).The method executes the overridden execution body method with the actual input and returns the output of the execution body through the OS.

Parameters

<i>input</i>	Actual input to the thread body.
--------------	----------------------------------

Returns

Returns the exact thing that the thread body returns.

Definition at line 127 of file WindowsThreadBase.h.

Here is the call graph for this function:



8.32.5.3 `virtual void* ThreadBase::ExecutionBody (void * input)` `[private],[pure virtual]`

Purely virtual method to be overridden by the inheriting class to implement its own thread body.

Parameters

<i>input</i>	Input to the thread body.
--------------	---------------------------

Returns

Return is optional.

Implemented in [ThreadedTCPClient](#), [ThreadedTCPConnectedClient](#), and [ThreadedTCPServer](#).

Here is the caller graph for this function:



8.32.5.4 `virtual void* ThreadBase::ExecutionBody (void *)` `[private],[pure virtual]`

Purely virtual method to be overridden by the inheriting class to implement its own thread body.

Parameters

<i>input</i>	Input to the thread body.
--------------	---------------------------

Returns

Return is optional.

Implemented in [ThreadedTCPClient](#), [ThreadedTCPConnectedClient](#), and [ThreadedTCPServer](#).

8.32.5.5 `TStackSize ThreadBase::GetStackSize (void) const` `[inline]`

Gets the stack size of the thread.

Returns

Returns the current stack size.

Definition at line 197 of file WindowsThreadBase.h.

8.32.5.6 `TStackSize ThreadBase::GetStackSize (void) const` `[inline]`

Gets the stack size of the thread.

Returns

Stack size of the thread.

Definition at line 202 of file PosixThreadBase.h.

8.32.5.7 `void ThreadBase::SetStackSize (const TStackSize stackSize)` `[inline]`

Sets the stack size of the thread.

Parameters

<i>stackSize</i>	Desired stack size.
------------------	---------------------

Definition at line 181 of file WindowsThreadBase.h.

8.32.5.8 `void ThreadBase::SetStackSize (const TStackSize stackSize)` `[inline]`

Sets the stack size of the thread.

Parameters

<i>stackSize</i>	Desired stack size.
------------------	---------------------

Definition at line 186 of file PosixThreadBase.h.

8.32.5.9 `void ThreadBase::StartThread (void * input)` `[inline]`

Creates a thread and starts executing with the given input.

Parameters

<i>input</i>	Input to the thread body.
--------------	---------------------------

Definition at line 213 of file WindowsThreadBase.h.

8.32.5.10 `void ThreadBase::StartThread (void * input)` `[inline]`

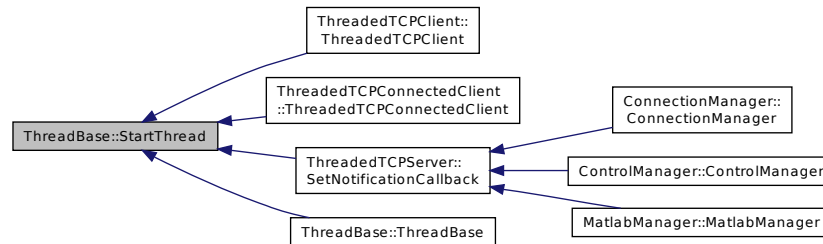
Creates a thread and starts executing with the given input.

Parameters

<i>input</i>	Input to the thread body.
--------------	---------------------------

Definition at line 220 of file PosixThreadBase.h.

Here is the caller graph for this function:



8.32.6 Friends And Related Function Documentation

8.32.6.1 void* PosixThreadCover (void *) [friend]

Friend wrapper function that is actually called by the OS as the thread body.

Uses the input to gain access to the class and the real thread input.

Parameters

<i>void*</i>	Special input structure containing the class address and the real input to the execution body.
--------------	--

It takes the actual class pointer and the input into the execution function from its input and executes the "actual thread body" with the desired input. This allows us to use the OS thread utilities to run a C++ class method, rather than a plain C function.

Parameters

<i>void*</i>	Special input structure containing the class address and the real input to the execution body.
--------------	--

Returns

Returns whatever the class method returns.

8.32.6.2 DWORD WINAPI WindowsThreadCover (LPVOID) [friend]

Friend wrapper function that is actually called by the OS as the thread body.

Uses the input to gain access to the class and the real thread input.

Parameters

<i>void*</i>	Special input structure containing the class address and the real input to the execution body.
--------------	--

It takes the actual class pointer and the input into the execution function from its input and executes the "actual thread body" with the desired input. This allows us to use the OS thread utilities to run a C++ class method, rather than a plain C function.

Parameters

<i>void*</i>	Special input structure containing the class address and the real input to the execution body.
--------------	--

Returns

Returns whatever the class method returns.

8.32.7 Member Data Documentation

8.32.7.1 TStackSize ThreadBase::m_stackSize [private]

Variable holding the stack size of the thread.

Definition at line 117 of file WindowsThreadBase.h.

8.32.7.2 TStarted ThreadBase::m_started [private]

Indicates whether the thread is started to execute.

Definition at line 101 of file PosixThreadBase.h.

8.32.7.3 TThreadAttribute ThreadBase::m_threadAttribute [private]

Variable holding the attributes of the thread.

Definition at line 111 of file PosixThreadBase.h.

8.32.7.4 TThreadHandle ThreadBase::m_threadHandle [private]

Variable holding the handle to the thread.

Definition at line 112 of file WindowsThreadBase.h.

8.32.7.5 TThreadId ThreadBase::m_threadId [private]

Variable holding the handle to the thread.

Variable holding the id of the thread.

Definition at line 106 of file PosixThreadBase.h.

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

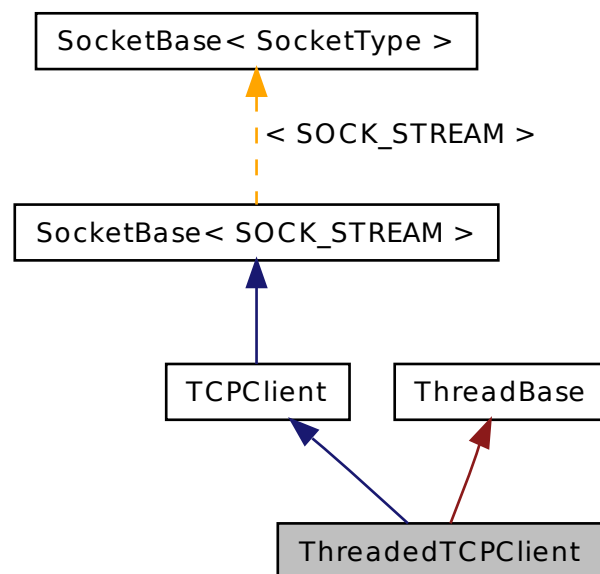
- S2Sim/ThreadLibrary/[PosixThreadBase.h](#)
- S2Sim/ThreadLibrary/[WindowsThreadBase.h](#)

8.33 ThreadedTCPClient Class Reference

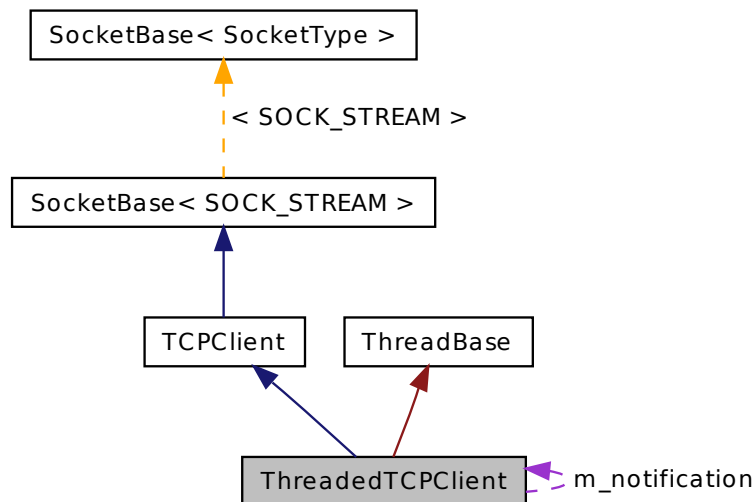
This is a TCP client class that receives data in a separate thread in the background.

```
#include <ThreadedTCPClient.h>
```

Inheritance diagram for ThreadedTCPClient:



Collaboration diagram for ThreadedTCPClient:



Public Types

- typedef void(* [TNotification](#))(ThreadedTCPClient *, [TBuffer](#) data, const [TNumberOfBytes](#) size)
Defines the callback type.

Public Member Functions

- [ThreadedTCPClient](#) (void)
Starts the thread and initializes the variables.
- [ThreadedTCPClient](#) (const [TCPClient](#) &client)
Copies only the TCP Client part and starts a new thread.
- [~ThreadedTCPClient](#) (void) throw ()
Stops execution.
- void [SetNotificationCallback](#) ([TNotification](#) notification)
Sets the notification callback function.

Private Member Functions

- void * [ExecutionBody](#) (void *input)
The execution body of the thread.

Private Attributes

- [TNotification m_notification](#)

Notification callback to be called.

- bool [m_started](#)

Boolean variable to stop the thread.

- Semaphore [m_allowingSemaphore](#)

This semaphore is released only if there is a legitimate callback function.

Additional Inherited Members

8.33.1 Detailed Description

This is a TCP client class that receives data in a separate thread in the background.

In order to use the class, a notification callback needs to be set that will be called when a data is received.

Definition at line 18 of file ThreadedTCPClient.h.

8.33.2 Member Typedef Documentation

8.33.2.1 `typedef void(* ThreadedTCPClient::TNotification)(ThreadedTCPClient *, TBuffer data, const TNumberOfBytes size)`

Defines the callback type.

The function should have three inputs:

- ThreadedTCPClient*: The function will be notified who the sender is.
- TBuffer: The function will be provided the data received in a buffer on the stack. The function is responsible of copying the data or it will be overwritten.
- TNumberOfBytes: The function will be provided how many bytes have been actually received.

Definition at line 27 of file ThreadedTCPClient.h.

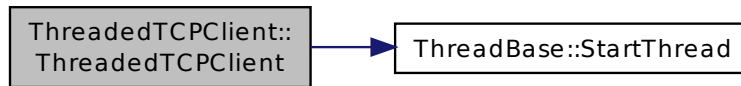
8.33.3 Constructor & Destructor Documentation

8.33.3.1 `ThreadedTCPClient::ThreadedTCPClient (void)`

Starts the thread and initializes the variables.

Definition at line 10 of file ThreadedTCPClient.cpp.

Here is the call graph for this function:

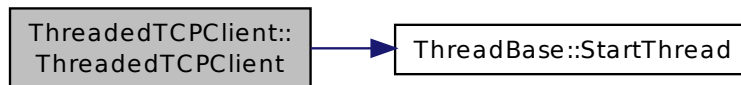


8.33.3.2 ThreadedTCPClient::ThreadedTCPClient (const TCPClient & *client*)

Copies only the TCP Client part and starts a new thread.

Definition at line 15 of file ThreadedTCPClient.cpp.

Here is the call graph for this function:



8.33.3.3 ThreadedTCPClient::~~ThreadedTCPClient (void) throw ()

Stops execution.

Definition at line 20 of file ThreadedTCPClient.cpp.

8.33.4 Member Function Documentation

8.33.4.1 void * ThreadedTCPClient::ExecutionBody (void * *input*) [private], [virtual]

The execution body of the thread.

The thread waits for a semaphore, which is released only once when the notification callback is set. If the semaphore is taken, the thread will start receiving data from the server.

Parameters

<i>input</i>	Not used.
--------------	-----------

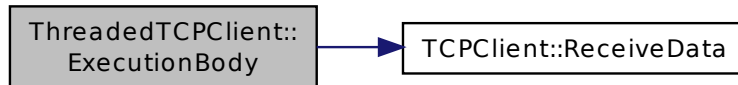
Returns

Not used. NULL.

Implements [ThreadBase](#).

Definition at line 26 of file ThreadedTCPClient.cpp.

Here is the call graph for this function:



8.33.4.2 void ThreadedTCPClient::SetNotificationCallback (TNotification *notification*) [inline]

Sets the notification callback function.

If the semaphore was already released, it is taken back. If the new callback is legit, it is released again.

Parameters

<i>notification</i>	Notification callback function.
---------------------	---------------------------------

Definition at line 77 of file ThreadedTCPClient.h.

8.33.5 Member Data Documentation

8.33.5.1 Semaphore ThreadedTCPClient::m_allowingSemaphore [private]

This semaphore is released only if there is a legitimate callback function.

Definition at line 53 of file ThreadedTCPClient.h.

8.33.5.2 TNotification ThreadedTCPClient::m_notification [private]

Notification callback to be called.

Definition at line 43 of file ThreadedTCPClient.h.

8.33.5.3 bool ThreadedTCPClient::m_started [private]

Boolean variable to stop the thread.

Definition at line 48 of file ThreadedTCPClient.h.

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

- S2Sim/SocketLibrary/[ThreadedTCPClient.h](#)

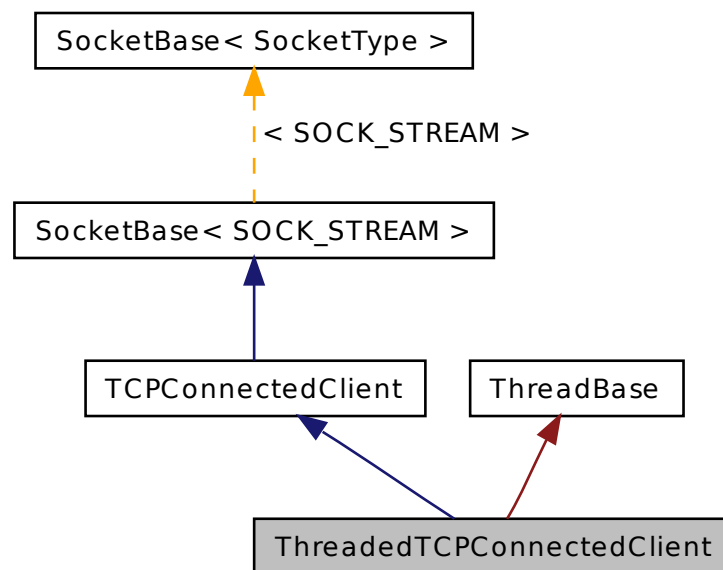
- S2Sim/SocketLibrary/[ThreadedTCPClient.cpp](#)

8.34 ThreadedTCPConnectedClient Class Reference

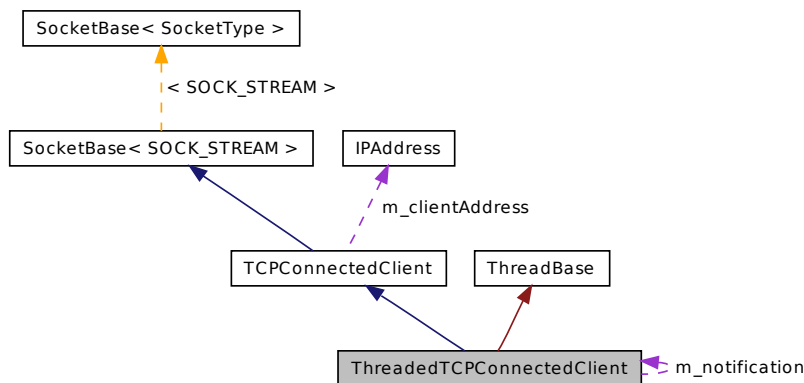
Manages the connection to an accepted client on the server side and receives data in a separate thread in the background.

```
#include <ThreadedTCPConnectedClient.h>
```

Inheritance diagram for ThreadedTCPConnectedClient:



Collaboration diagram for ThreadedTCPConnectedClient:



Public Types

- typedef void(* TNotification)(ThreadedTCPConnectedClient *, TBuffer data, const TNumberOfBytes size)
Defines the callback type.

Public Member Functions

- ThreadedTCPConnectedClient (const TSocketId socketId, IPAddress &clientAddress)
Initializes the class with a ready socket and address, and starts the reception thread.
- ThreadedTCPConnectedClient (const TCPConnectedClient &client)
Copies the TCP connection information and starts a new thread.
- ~ThreadedTCPConnectedClient (void) throw ()
Stops the execution of the thread.
- void SetNotificationCallback (TNotification notification)
Sets the notification callback function.

Private Member Functions

- void * ExecutionBody (void *input)
The execution body of the thread.

Private Attributes

- TNotification m_notification
Notification callback to be called.
- bool m_started
Boolean variable to stop the thread.
- Semaphore m_allowingSemaphore
This semaphore is released only if there is a legitimate callback function.

Additional Inherited Members

8.34.1 Detailed Description

Manages the connection to an accepted client on the server side and receives data in a separate thread in the background.

Definition at line 18 of file ThreadedTCPConnectedClient.h.

8.34.2 Member Typedef Documentation

8.34.2.1 `typedef void(* ThreadedTCPConnectedClient::TNotification)(ThreadedTCPConnectedClient *, TBuffer data, const TNumberOfBytes size)`

Defines the callback type.

The function should have three inputs:

- ThreadedTCPConnectedClient*: The function will be notified who the sender is.
- TBuffer: The function will be provided the data received in a buffer on the stack. The function is responsible of copying the data or it will be overwritten.
- TNumberOfBytes: The function will be provided how many bytes have been actually received.

Definition at line 27 of file ThreadedTCPConnectedClient.h.

8.34.3 Constructor & Destructor Documentation

8.34.3.1 `ThreadedTCPConnectedClient::ThreadedTCPConnectedClient (const TSocketId socketId, IPAddress & clientAddress)`

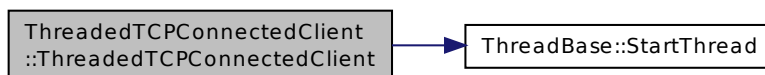
Initializes the class with a ready socket and address, and starts the reception thread.

Parameters

<i>socketId</i>	Handle to the socket.
<i>clientAddress</i>	IPAddress of the connected client.

Definition at line 10 of file ThreadedTCPConnectedClient.cpp.

Here is the call graph for this function:

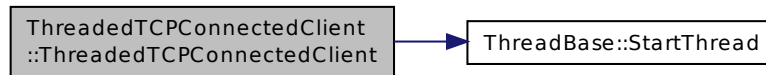


8.34.3.2 ThreadedTCPConnectedClient::ThreadedTCPConnectedClient (const TCPConnectedClient & client)

Copies the TCP connection information and starts a new thread.

Definition at line 18 of file ThreadedTCPConnectedClient.cpp.

Here is the call graph for this function:



8.34.3.3 ThreadedTCPConnectedClient::~~ThreadedTCPConnectedClient (void) throw

Stops the execution of the thread.

Definition at line 26 of file ThreadedTCPConnectedClient.cpp.

8.34.4 Member Function Documentation

8.34.4.1 void * ThreadedTCPConnectedClient::ExecutionBody (void * input) [private], [virtual]

The execution body of the thread.

The thread waits for a semaphore, which is released only once when the notification callback is set. If the semaphore is taken, the thread will start receiving data from the server.

Parameters

<i>input</i>	Not used.
--------------	-----------

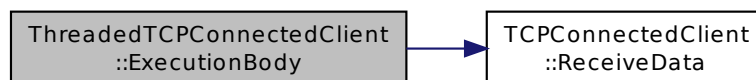
Returns

Not used. NULL.

Implements [ThreadBase](#).

Definition at line 32 of file ThreadedTCPConnectedClient.cpp.

Here is the call graph for this function:



8.34.4.2 void ThreadedTCPConnectedClient::SetNotificationCallback (TNotification notification) [inline]

Sets the notification callback function.

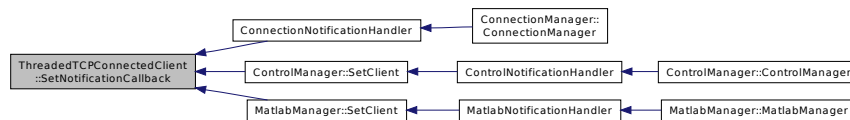
If the semaphore was already released, it is taken back. If the new callback is legit, it is released again.

Parameters

<i>notification</i>	Notification callback function.
---------------------	---------------------------------

Definition at line 79 of file ThreadedTCPConnectedClient.h.

Here is the caller graph for this function:



8.34.5 Member Data Documentation

8.34.5.1 Semaphore ThreadedTCPConnectedClient::m_allowingSemaphore [private]

This semaphore is released only if there is a legitimate callback function.

Definition at line 53 of file ThreadedTCPConnectedClient.h.

8.34.5.2 TNotification ThreadedTCPConnectedClient::m_notification [private]

Notification callback to be called.

Definition at line 43 of file ThreadedTCPConnectedClient.h.

8.34.5.3 bool ThreadedTCPConnectedClient::m_started [private]

Boolean variable to stop the thread.

Definition at line 48 of file ThreadedTCPConnectedClient.h.

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

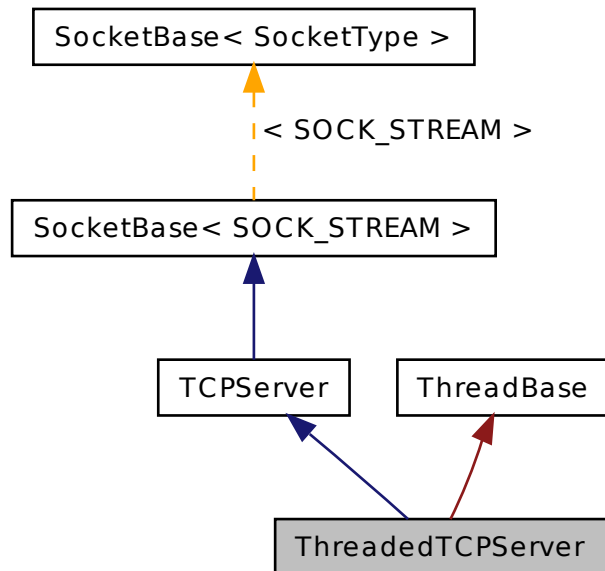
- S2Sim/SocketLibrary/[ThreadedTCPConnectedClient.h](#)
- S2Sim/SocketLibrary/[ThreadedTCPConnectedClient.cpp](#)

8.35 ThreadedTCPServer Class Reference

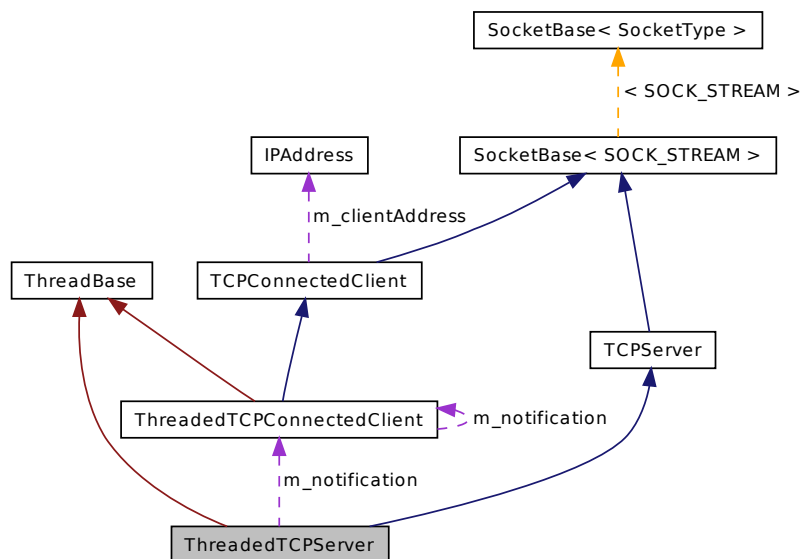
Defines the threaded version of [TCPServer](#) that accepts clients in another thread in the background.

```
#include <ThreadedTCPServer.h>
```

Inheritance diagram for ThreadedTCPServer:



Collaboration diagram for ThreadedTCPServer:



Public Types

- typedef void(* TNotification)(ThreadedTCPConnectedClient *)

Defines the type of the callback notification function.

Public Member Functions

- ThreadedTCPServer (void)
Initializes the variables.
- ~ThreadedTCPServer (void) throw ()
Stops the execution of the thread.
- void SetNotificationCallback (TNotification notification)
Sets the notification callback.

Private Member Functions

- void * ExecutionBody (void *input)
The execution body of the thread.

Private Attributes

- TNotification m_notification
Callback function.
- bool m_started
Indicates whether the thread is running.

Additional Inherited Members

8.35.1 Detailed Description

Defines the threaded version of [TCPServer](#) that accepts clients in another thread in the background.

When a connection is established, the user will be notified through a callback function.

Definition at line 18 of file ThreadedTCPServer.h.

8.35.2 Member Typedef Documentation

8.35.2.1 typedef void(* ThreadedTCPServer::TNotification)(ThreadedTCPConnectedClient *)

Defines the type of the callback notification function.

The callback should only have a single input. The function will be provided the pointer of its caller.

Definition at line 24 of file ThreadedTCPServer.h.

8.35.3 Constructor & Destructor Documentation

8.35.3.1 ThreadedTCPServer::ThreadedTCPServer (void)

Initializes the variables.

Definition at line 10 of file ThreadedTCPServer.cpp.

8.35.3.2 ThreadedTCPServer::~~ThreadedTCPServer (void) throw ()

Stops the execution of the thread.

Definition at line 14 of file ThreadedTCPServer.cpp.

8.35.4 Member Function Documentation

8.35.4.1 void * ThreadedTCPServer::ExecutionBody (void * *input*) [private],[virtual]

The execution body of the thread.

The thread will start accepting connections.

Parameters

<i>input</i>	Not used.
--------------	-----------

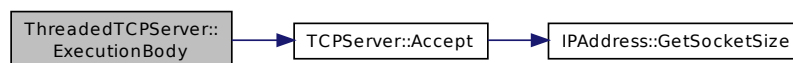
Returns

Not used. NULL.

Implements [ThreadBase](#).

Definition at line 20 of file ThreadedTCPServer.cpp.

Here is the call graph for this function:



8.35.4.2 void ThreadedTCPServer::SetNotificationCallback (TNotification *notification*) [inline]

Sets the notification callback.

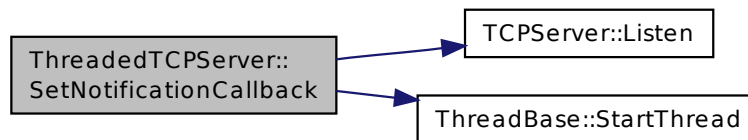
If the notification callback function is legit, the thread is started. This function can be set only once. Adding a guard has currently no meaning but overhead.

Parameters

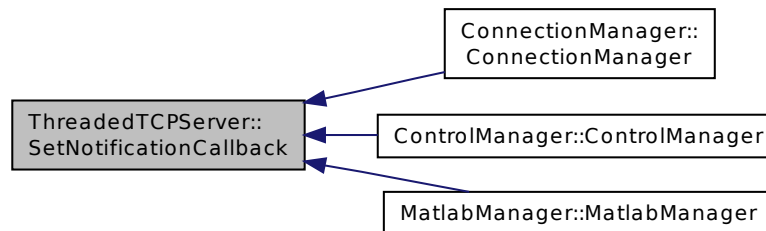
<i>notification</i>	Callback to the notification function.
---------------------	--

Definition at line 64 of file ThreadedTCPServer.h.

Here is the call graph for this function:



Here is the caller graph for this function:



8.35.5 Member Data Documentation

8.35.5.1 TNotification ThreadedTCPServer::m_notification [private]

Callback function.

Definition at line 40 of file ThreadedTCPServer.h.

8.35.5.2 bool ThreadedTCPServer::m_started [private]

Indicates whether the thread is running.

Definition at line 45 of file ThreadedTCPServer.h.

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

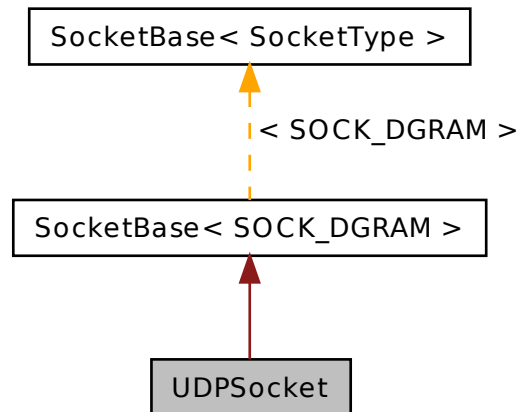
- S2Sim/SocketLibrary/[ThreadedTCPServer.h](#)
- S2Sim/SocketLibrary/[ThreadedTCPServer.cpp](#)

8.36 UDPSocket Class Reference

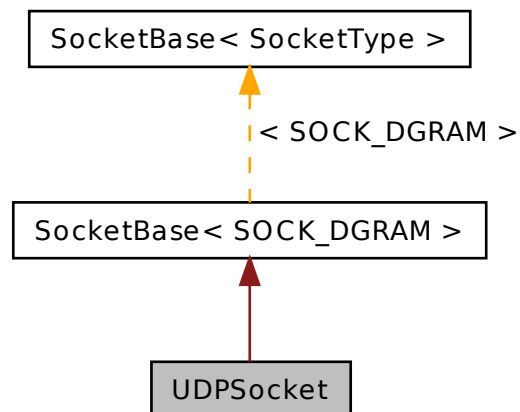
Manages a UDP connection.

```
#include <UDPSocket.h>
```

Inheritance diagram for UDPSocket:



Collaboration diagram for UDPSocket:



Public Member Functions

- [UDPSocket](#) (void)
Not used.
- [~UDPSocket](#) (void)
Not used.
- void [SetIPAddress](#) ([IPAddress](#) &address)
Sets the IP Address of the socket.
- void [SetPort](#) (const [IPAddress::TPort](#) port)
Sets the port of the UDP Socket.
- [TNumberOfBytes SendData](#) (const [TBuffer](#) buffer, const [TNumberOfBytes](#) length, [IPAddress](#) &destination)
Sends a data to another UDP socket.
- [TNumberOfBytes ReceiveData](#) ([TBuffer](#) buffer, const [TNumberOfBytes](#) receptionLength, [IPAddress](#) &destination-Address)
Receives data from another UDP Socket.

Private Types

- typedef [SocketBase](#)< SOCK_DGRAM > [TBaseType](#)
Defines the base class for rapid development.

Additional Inherited Members

8.36.1 Detailed Description

Manages a UDP connection.

If the port is selected, this socket can send to any UDP socket.

Definition at line 17 of file UDPSocket.h.

8.36.2 Member Typedef Documentation

8.36.2.1 typedef [SocketBase](#)<SOCK_DGRAM> [UDPSocket::TBaseType](#) [private]

Defines the base class for rapid development.

Definition at line 74 of file UDPSocket.h.

8.36.3 Constructor & Destructor Documentation

8.36.3.1 [UDPSocket::UDPSocket](#) (void)

Not used.

Definition at line 10 of file UDPSocket.cpp.

8.36.3.2 UDPSocket::~UDPSocket (void)

Not used.

Definition at line 14 of file UDPSocket.cpp.

8.36.4 Member Function Documentation

8.36.4.1 UDPSocket::TNumberOfBytes UDPSocket::ReceiveData (TBuffer *buffer*, const TNumberOfBytes *receptionLength*, IPAddress & *destinationAddress*)

Receives data from another UDP Socket.

This is a blocking call.

Parameters

<i>buffer</i>	An allocated buffer where the received data will be stored in.
<i>receptionLength</i>	Length of the buffer and the maximum reception length.
<i>destination-Address</i>	IP Address, from which the data will be received.

Returns

Returns the actual received number of bytes.

Definition at line 25 of file UDPSocket.cpp.

Here is the call graph for this function:



8.36.4.2 UDPSocket::TNumberOfBytes UDPSocket::SendData (const TBuffer *buffer*, const TNumberOfBytes *length*, IPAddress & *destination*)

Sends a data to another UDP socket.

Parameters

<i>buffer</i>	Buffer, containing the data to be sent.
<i>length</i>	Length of the data to be sent.
<i>destination</i>	IP Address of the destination socket.

Returns

Returns the actual number of bytes sent.

Definition at line 19 of file UDPSocket.cpp.

Here is the call graph for this function:

**8.36.4.3 void UDPSocket::SetIPAddress (IPAddress & address)**

Sets the IP Address of the socket.

This method is not recommended as the IP Address of the user is already set. If the user is not sure, [UDPSocket::SetPort\(\)](#) is the usual usage.

Parameters

<i>address</i>	IP Address of the socket.
----------------	---------------------------

Definition at line 32 of file UDPSocket.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.4.4 void UDPSocket::SetPort (const IPAddress::TPort *port*)

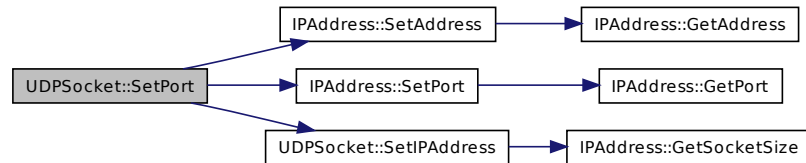
Sets the port of the UDP Socket.

Parameters

<i>port</i>	The port that the destination will send its data to.
-------------	--

Definition at line 38 of file UDPSocket.cpp.

Here is the call graph for this function:



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

- S2Sim/SocketLibrary/[UDPSocket.h](#)
- S2Sim/SocketLibrary/[UDPSocket.cpp](#)

Chapter 9

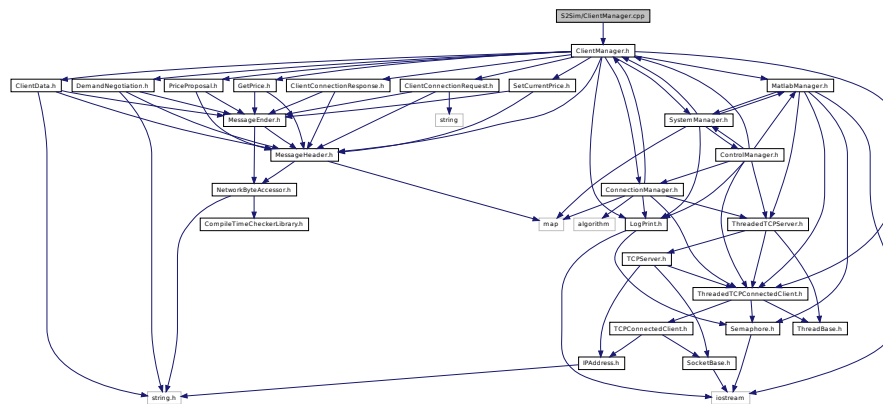
File Documentation

9.1 S2Sim/ClientManager.cpp File Reference

Source file for the [ClientManager](#) class.

```
#include "ClientManager.h"
```

Include dependency graph for ClientManager.cpp:



9.1.1 Detailed Description

Source file for the [ClientManager](#) class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

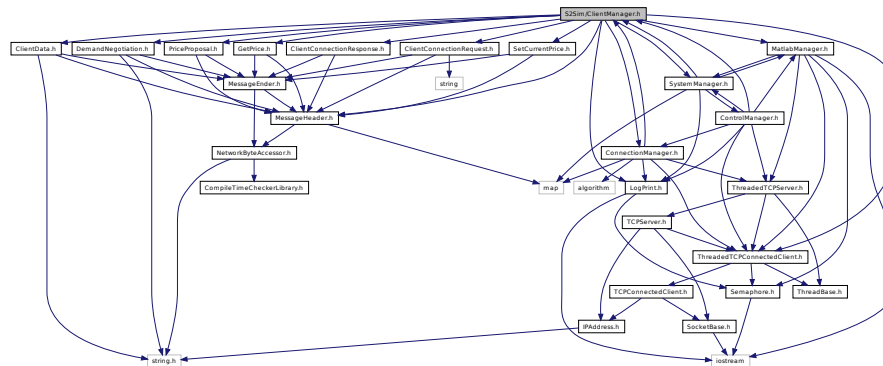
Definition in file [ClientManager.cpp](#).

9.2 S2Sim/ClientManager.h File Reference

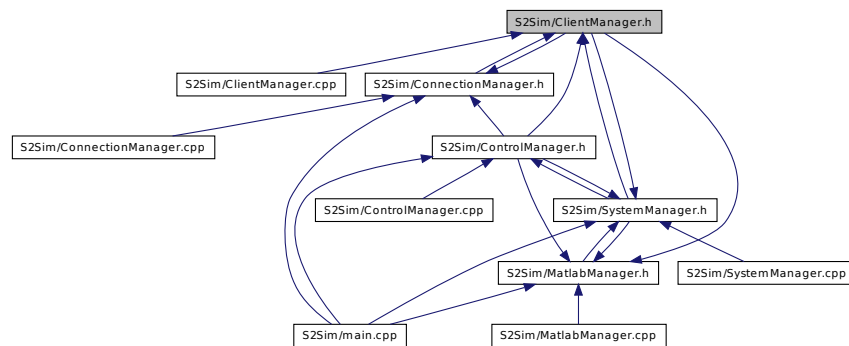
Header file for the [ClientManager](#) class.

```
#include "MessageHeader.h"
#include "ClientConnectionRequest.h"
#include "ClientConnectionResponse.h"
#include "SetCurrentPrice.h"
#include "PriceProposal.h"
#include "ClientData.h"
#include "GetPrice.h"
#include "DemandNegotiation.h"
#include "ConnectionManager.h"
#include "MatlabManager.h"
#include "SystemManager.h"
#include "ThreadedTCPConnectedClient.h"
#include "LogPrint.h"
```

Include dependency graph for ClientManager.h:



This graph shows which files directly or indirectly include this file:



9.3.1 Detailed Description

This file implements the [ConnectionManager](#) class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

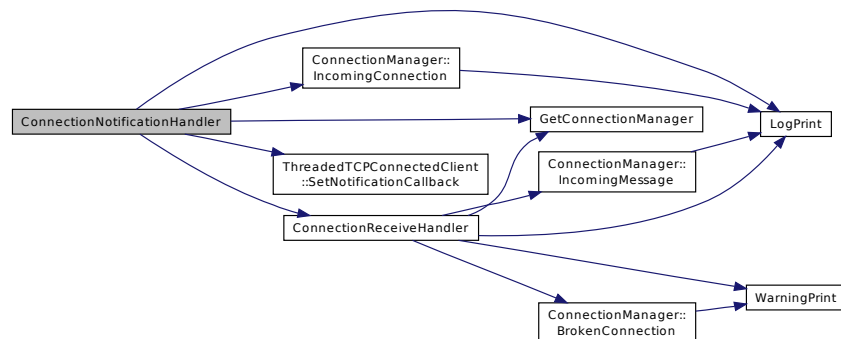
Definition in file [ConnectionManager.cpp](#).

9.3.2 Function Documentation

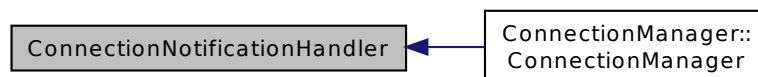
9.3.2.1 void ConnectionNotificationHandler (ThreadedTCPConnectedClient * *acceptedClient*)

Definition at line 18 of file ConnectionManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.2.2 void ConnectionReceiveHandler (ThreadedTCPConnectedClient * *client*, void * *buffer*, size_t *size*)

Wrapper callback method for reception notification.

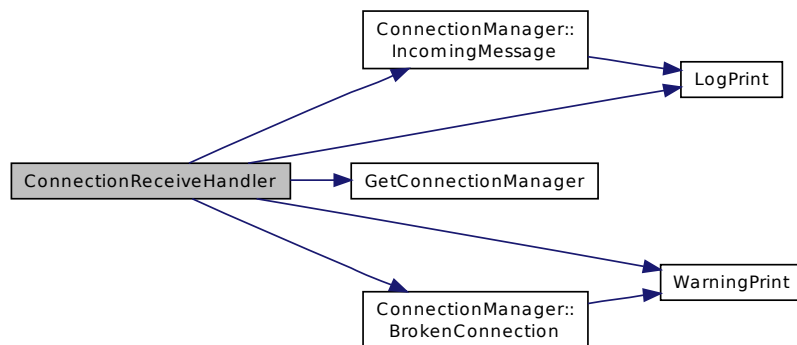
This method is called to notify that a data has been received. This method relays the information directly to [ConnectionManager](#).

Parameters

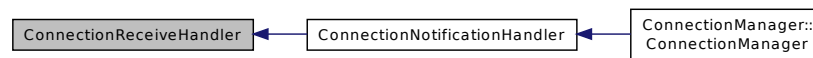
<i>client</i>	Client information of the sender.
<i>buffer</i>	Buffer containing the message.
<i>size</i>	Size of the message.

Definition at line 28 of file ConnectionManager.cpp.

Here is the call graph for this function:

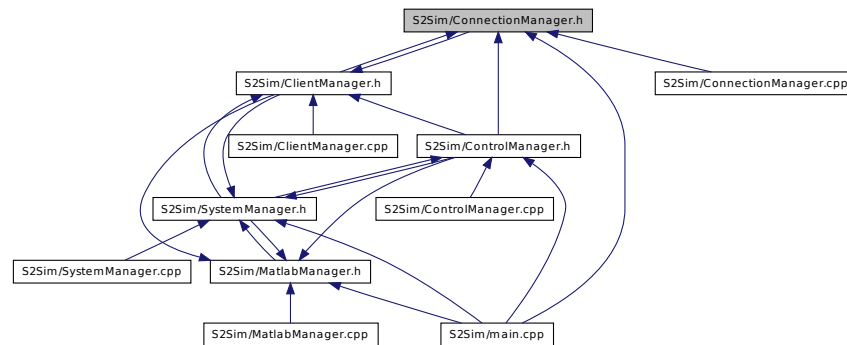


Here is the caller graph for this function:



9.3.2.3 ConnectionManager& GetConnectionManager (void)

This graph shows which files directly or indirectly include this file:



Classes

- class [ConnectionManager](#)
Manages connections to all clients.

Functions

- void [ConnectionReceiveHandler](#) ([ThreadedTCPConnectedClient](#) *client, void *buffer, size_t size)
Wrapper callback method for reception notification.
- void [ConnectionNotificationHandler](#) ([TCPConnectedClient](#) *acceptedClient)
Wrapper callback method for connection notification.
- [ConnectionManager](#) & [GetConnectionManager](#) (void)

9.4.1 Detailed Description

This file defines the [ConnectionManager](#) class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

Definition in file [ConnectionManager.h](#).

9.4.2 Function Documentation

9.4.2.1 void ConnectionNotificationHandler ([TCPConnectedClient](#) * *acceptedClient*)

Wrapper callback method for connection notification.

This method is called to notify that a new connection is established. This method relays the notification directly to the [ConnectionManager](#).

Parameters

<i>acceptedClient</i>	Information about the newly accepted client.
-----------------------	--

9.4.2.2 void ConnectionReceiveHandler (ThreadedTCPConnectedClient * *client*, void * *buffer*, size_t *size*)

Wrapper callback method for reception notification.

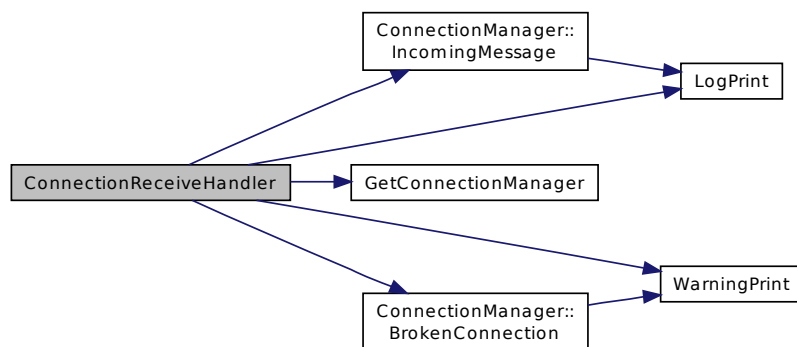
This method is called to notify that a data has been received. This method relays the information directly to [ConnectionManager](#).

Parameters

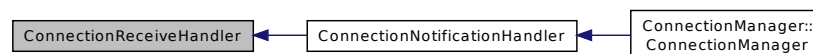
<i>client</i>	Client information of the sender.
<i>buffer</i>	Buffer containing the message.
<i>size</i>	Size of the message.

Definition at line 28 of file ConnectionManager.cpp.

Here is the call graph for this function:



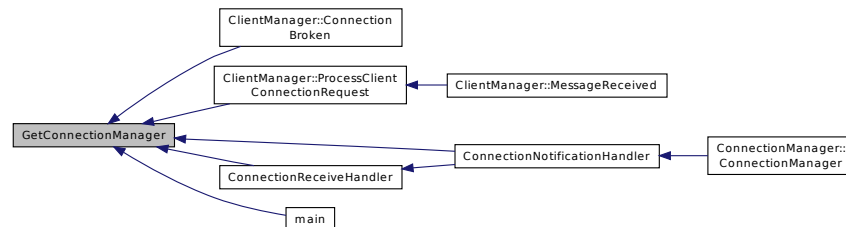
Here is the caller graph for this function:



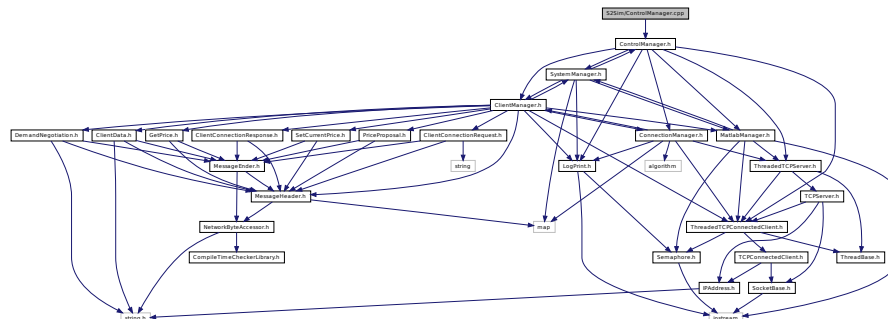
9.4.2.3 ConnectionManager& GetConnectionManager (void)

Returns the reference to the [ConnectionManager](#).

Here is the caller graph for this function:



Include dependency graph for ControlManager.cpp:



- **ControlManager** & **GetControlManager** (void)
*Returns the only instance of **ControlManager**.*
- void **ControlNotificationHandler** (**ThreadedTCPConnectedClient** *acceptedClient)
- void **ControlReceiveHandler** (**ThreadedTCPConnectedClient** *client, void *buffer, size_t size)
Wrapper callback method for reception notification.

This file implements the `ControlManager` class.

Date

Oct 27, 2013

Author

: Alper Sinan Akyurek

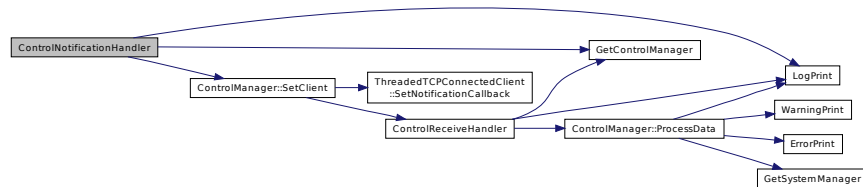
Definition in file [ControlManager.cpp](#).

9.5.2 Function Documentation

9.5.2.1 void ControlNotificationHandler (ThreadedTCPConnectedClient * *acceptedClient*)

Definition at line 18 of file ControlManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

9.5.2.2 void ControlReceiveHandler (ThreadedTCPConnectedClient * *client*, void * *buffer*, size_t *size*)

Wrapper callback method for reception notification.

This method is called to notify that a new data is received from the external controller. The data is relayed directly to [ControlManager](#).

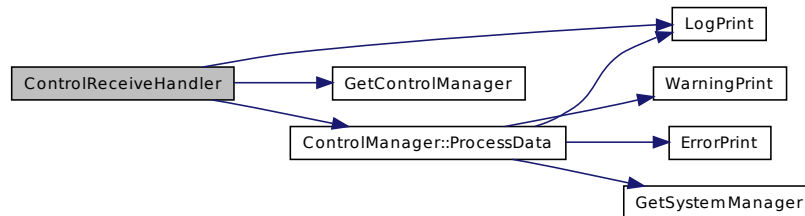
Parameters

<i>client</i>	Connection information of the external controller.
---------------	--

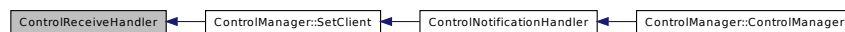
<i>buffer</i>	Buffer containing the received data.
<i>size</i>	Size of the received data.

Definition at line 27 of file ControlManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



9.5.2.3 ControlManager& GetControlManager (void)

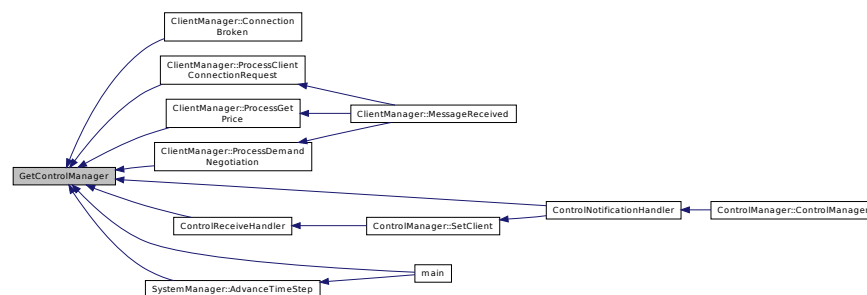
Returns the only instance of [ControlManager](#).

Returns

Returns the reference to the [ControlManager](#).

Definition at line 11 of file ControlManager.cpp.

Here is the caller graph for this function:



Returns the only instance of [ControlManager](#).

- void [ControlNotificationHandler](#) ([TCPConnectedClient](#) *acceptedClient)

Wrapper callback method for connection notification.

- void [ControlReceiveHandler](#) ([ThreadedTCPConnectedClient](#) *client, void *buffer, size_t size)

Wrapper callback method for reception notification.

9.6.1 Detailed Description

This file defines the [ControlManager](#) class.

Date

Oct 27, 2013

Author

: Alper Sinan Akyurek

Definition in file [ControlManager.h](#).

9.6.2 Function Documentation

9.6.2.1 void ControlNotificationHandler ([TCPConnectedClient](#) * *acceptedClient*)

Wrapper callback method for connection notification.

This method is called to notify that a new external controller connection is accepted. The information is relayed directly to [ControlManager](#).

Parameters

<i>acceptedClient</i>	Information about the newly accepted external controller.
-----------------------	---

9.6.2.2 void ControlReceiveHandler ([ThreadedTCPConnectedClient](#) * *client*, void * *buffer*, size_t *size*)

Wrapper callback method for reception notification.

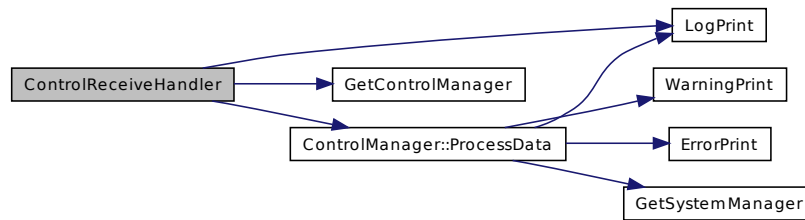
This method is called to notify that a new data is received from the external controller. The data is relayed directly to [ControlManager](#).

Parameters

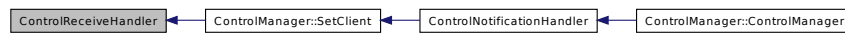
<i>client</i>	Connection information of the external controller.
<i>buffer</i>	Buffer containing the received data.
<i>size</i>	Size of the received data.

Definition at line 27 of file [ControlManager.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



9.6.2.3 ControlManager& GetControlManager (void)

Returns the only instance of [ControlManager](#).

Friend method for singleton implementation.

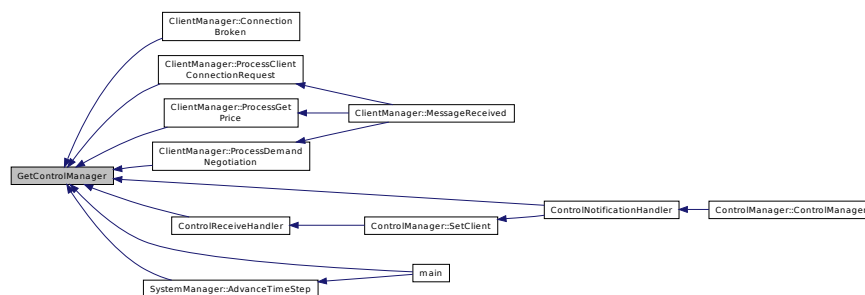
Returns

Only instance of [ControlManager](#).

Returns the reference to the [ControlManager](#).

Definition at line 11 of file `ControlManager.cpp`.

Here is the caller graph for this function:

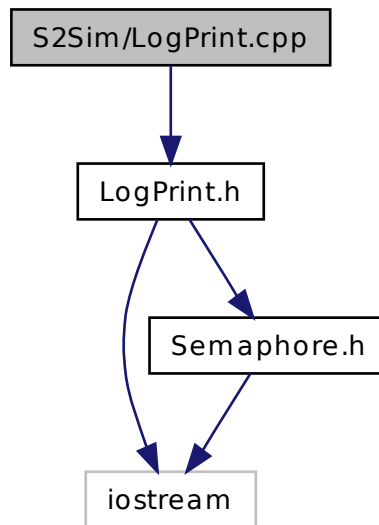


9.7 S2Sim/LogPrint.cpp File Reference

Defines logging related parameters.

```
#include "LogPrint.h"
```

Include dependency graph for LogPrint.cpp:



Variables

- unsigned int `logLevel` = 3

This is a small hack to suppress compiler warning for default macro parameter initialization.

- bool `functionPrint` = true

Boolean to indicate whether the function name should be printed for each log.

9.7.1 Detailed Description

Defines logging related parameters.

Date

Oct 28, 2013

Author

: Alper Sinan Akyurek

Definition in file [LogPrint.cpp](#).

9.7.2 Variable Documentation

9.7.2.1 bool functionPrint = true

Boolean to indicate whether the function name should be printed for each log.

Definition at line 11 of file LogPrint.cpp.

9.7.2.2 unsigned int logLevel = 3

This is a small hack to suppress compiler warning for default macro parameter initialization.

Logging level on how detailed the logs should be.

- 1: All logs
- 2: Warnings and errors
- 3: Errors only
- >3: No logging

Definition at line 10 of file LogPrint.cpp.

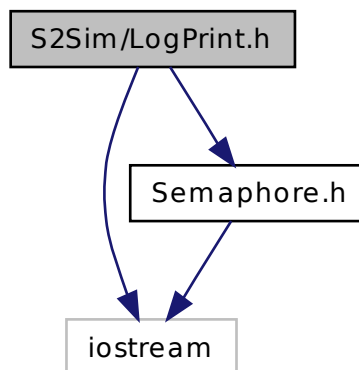
9.8 S2Sim/LogPrint.h File Reference

Defines logging related functions.

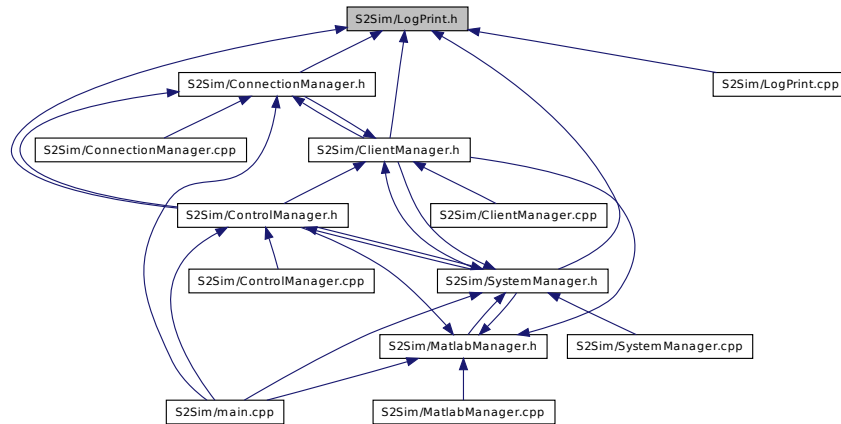
```
#include <iostream>
```

```
#include "Semaphore.h"
```

Include dependency graph for LogPrint.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define LOG_FUNCTION_START() LogPrint("-START-");`
Macro to indicate the start of a function for verbose debugging.
- `#define LOG_FUNCTION_END() LogPrint("-END-");`
Macro to indicate the end of a function for verbose debugging.

Functions

- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 , typename I7 , typename I8 >`
`void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, I7 input7, I8 input8, const char *function=__PRETTY_FUNCTION__)`
8 parameter log message
- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 , typename I7 >`
`void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, I7 input7, const char *function=__PRETTY_FUNCTION__)`
7 parameter log message
- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 >`
`void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, const char *function=__PRETTY_FUNCTION__)`
6 parameter log message
- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 >`
`void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, const char *function=__PRETTY_FUNCTION__)`
5 parameter log message
- `template<typename I1 , typename I2 , typename I3 , typename I4 >`
`void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, const char *function=__PRETTY_FUNCTION__)`
4 parameter log message
- `template<typename I1 , typename I2 , typename I3 >`
`void LogPrint (I1 input1, I2 input2, I3 input3, const char *function=__PRETTY_FUNCTION__)`
3 parameter log message

- `template<typename I1 , typename I2 >`
`void LogPrint (I1 input1, I2 input2, const char *function=__PRETTY_FUNCTION__)`
2 parameter log message
- `template<typename I1 >`
`void LogPrint (I1 input1, const char *function=__PRETTY_FUNCTION__)`
1 parameter log message
- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 >`
`void WarningPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, const char *function=__PRETTY_FUNCTION__)`
6 parameter warning message
- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 >`
`void WarningPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, const char *function=__PRETTY_FUNCTION__)`
5 parameter warning message
- `template<typename I1 , typename I2 , typename I3 , typename I4 >`
`void WarningPrint (I1 input1, I2 input2, I3 input3, I4 input4, const char *function=__PRETTY_FUNCTION__)`
4 parameter warning message
- `template<typename I1 , typename I2 , typename I3 >`
`void WarningPrint (I1 input1, I2 input2, I3 input3, const char *function=__PRETTY_FUNCTION__)`
3 parameter warning message
- `template<typename I1 , typename I2 >`
`void WarningPrint (I1 input1, I2 input2, const char *function=__PRETTY_FUNCTION__)`
2 parameter warning message
- `template<typename I1 >`
`void WarningPrint (I1 input1, const char *function=__PRETTY_FUNCTION__)`
1 parameter warning message
- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 >`
`void ErrorPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, const char *function=__PRETTY_FUNCTION__)`
6 parameter error message
- `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 >`
`void ErrorPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, const char *function=__PRETTY_FUNCTION__)`
5 parameter error message
- `template<typename I1 , typename I2 , typename I3 , typename I4 >`
`void ErrorPrint (I1 input1, I2 input2, I3 input3, I4 input4, const char *function=__PRETTY_FUNCTION__)`
4 parameter error message
- `template<typename I1 , typename I2 , typename I3 >`
`void ErrorPrint (I1 input1, I2 input2, I3 input3, const char *function=__PRETTY_FUNCTION__)`
3 parameter error message
- `template<typename I1 , typename I2 >`
`void ErrorPrint (I1 input1, I2 input2, const char *function=__PRETTY_FUNCTION__)`
2 parameter error message
- `template<typename I1 >`
`void ErrorPrint (I1 input1, const char *function=__PRETTY_FUNCTION__)`
1 parameter error message

Variables

- unsigned int [logLevel](#)

This is a small hack to suppress compiler warning for default macro parameter initialization.

- bool [functionPrint](#)

Boolean to indicate whether the function name should be printed for each log.

9.8.1 Detailed Description

Defines logging related functions.

Date

Oct 28, 2013

Author

: Alper Sinan Akyurek

Definition in file [LogPrint.h](#).

9.8.2 Macro Definition Documentation

9.8.2.1 `#define LOG_FUNCTION_END() LogPrint("-END-");`

Macro to indicate the end of a function for verbose debugging.

Definition at line 41 of file LogPrint.h.

9.8.2.2 `#define LOG_FUNCTION_START() LogPrint("-START-");`

Macro to indicate the start of a function for verbose debugging.

Definition at line 36 of file LogPrint.h.

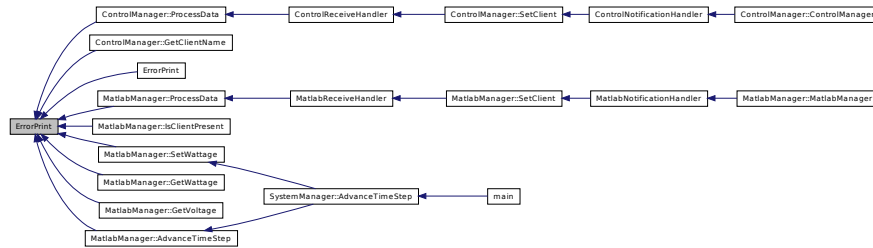
9.8.3 Function Documentation

9.8.3.1 `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 > void ErrorPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, const char * function = __PRETTY_FUNCTION__)`

6 parameter error message

Definition at line 200 of file LogPrint.h.

Here is the caller graph for this function:



9.8.3.2 `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 > void ErrorPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, const char * function = __PRETTY_FUNCTION__)`

5 parameter error message

Definition at line 220 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.3 `template<typename I1 , typename I2 , typename I3 , typename I4 > void ErrorPrint (I1 input1, I2 input2, I3 input3, I4 input4, const char * function = __PRETTY_FUNCTION__)`

4 parameter error message

Definition at line 229 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.4 `template<typename I1 , typename I2 , typename I3 > void ErrorPrint (I1 input1, I2 input2, I3 input3, const char * function = __PRETTY_FUNCTION__)`

3 parameter error message

Definition at line 238 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.5 `template<typename I1 , typename I2 > void ErrorPrint (I1 input1, I2 input2, const char * function = __PRETTY_FUNCTION__)`

2 parameter error message

Definition at line 247 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.6 `template<typename I1 > void ErrorPrint (I1 input1, const char * function = __PRETTY_FUNCTION__)`

1 parameter error message

Definition at line 256 of file LogPrint.h.

Here is the call graph for this function:

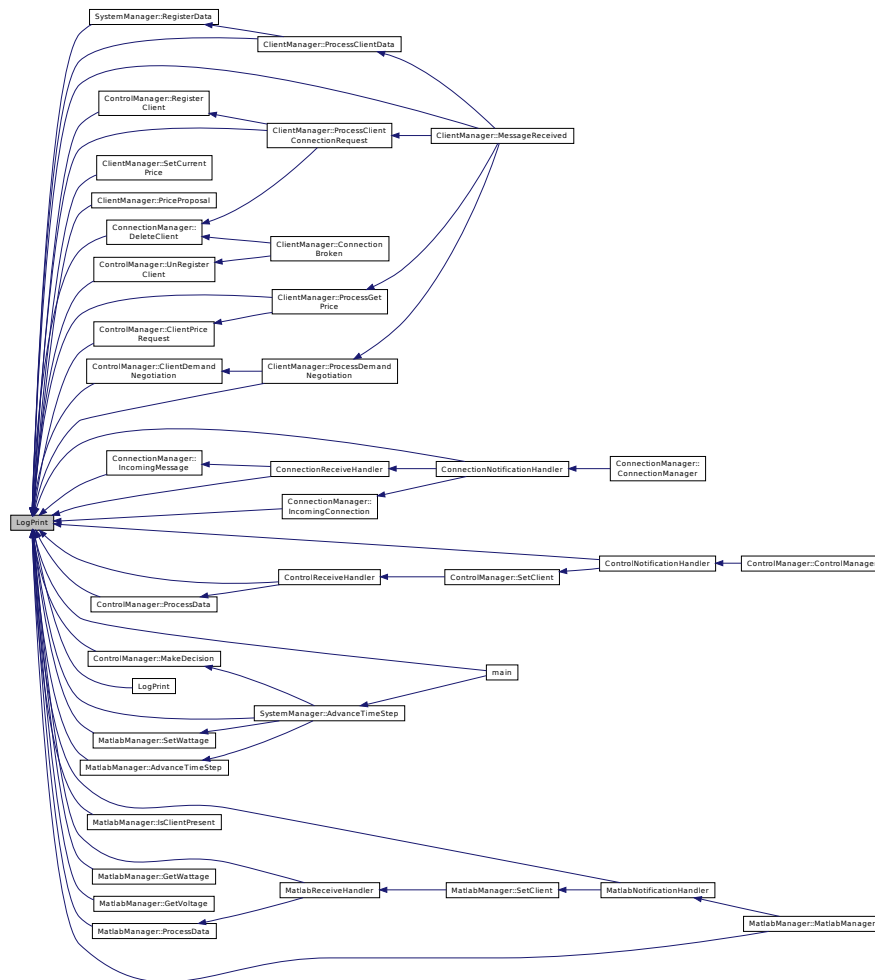


```
9.8.3.7 template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 , typename I7 , typename I8 >  
void LogPrint ( I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, I7 input7, I8 input8, const char * function =  
    __PRETTY_FUNCTION__ )
```

8 parameter log message

Definition at line 47 of file LogPrint.h.

Here is the caller graph for this function:



```

9.8.3.8 template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 , typename I7 > void
LogPrint ( I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, I7 input7, const char * function =
__PRETTY_FUNCTION__ )

```

7 parameter log message

Definition at line 72 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.9 `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 > void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, const char * function = __PRETTY_FUNCTION__)`

6 parameter log message

Definition at line 81 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.10 `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 > void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, const char * function = __PRETTY_FUNCTION__)`

5 parameter log message

Definition at line 90 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.11 `template<typename I1 , typename I2 , typename I3 , typename I4 > void LogPrint (I1 input1, I2 input2, I3 input3, I4 input4, const char * function = __PRETTY_FUNCTION__)`

4 parameter log message

Definition at line 99 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.12 `template<typename I1 , typename I2 , typename I3 > void LogPrint (I1 input1, I2 input2, I3 input3, const char * function = __PRETTY_FUNCTION__)`

3 parameter log message

Definition at line 108 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.13 `template<typename I1 , typename I2 > void LogPrint (I1 input1, I2 input2, const char * function = __PRETTY_FUNCTION__)`

2 parameter log message

Definition at line 117 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.14 `template<typename I1 > void LogPrint (I1 input1, const char * function = __PRETTY_FUNCTION__)`

1 parameter log message

Definition at line 126 of file LogPrint.h.

Here is the call graph for this function:

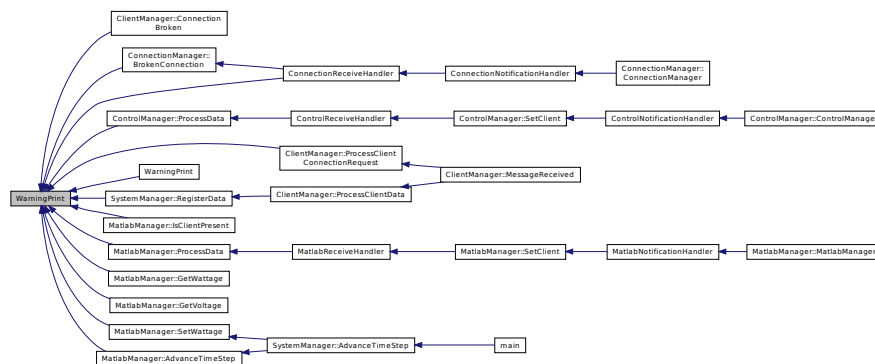


9.8.3.15 `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 , typename I6 > void WarningPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, I6 input6, const char * function = __PRETTY_FUNCTION__)`

6 parameter warning message

Definition at line 135 of file LogPrint.h.

Here is the caller graph for this function:



9.8.3.16 `template<typename I1 , typename I2 , typename I3 , typename I4 , typename I5 > void WarningPrint (I1 input1, I2 input2, I3 input3, I4 input4, I5 input5, const char * function = __PRETTY_FUNCTION__)`

5 parameter warning message

Definition at line 155 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.17 `template<typename I1 , typename I2 , typename I3 , typename I4 > void WarningPrint (I1 input1, I2 input2, I3 input3, I4 input4, const char * function = __PRETTY_FUNCTION__)`

4 parameter warning message

Definition at line 164 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.18 `template<typename I1 , typename I2 , typename I3 > void WarningPrint (I1 input1, I2 input2, I3 input3, const char * function = __PRETTY_FUNCTION__)`

3 parameter warning message

Definition at line 173 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.19 `template<typename I1 , typename I2 > void WarningPrint (I1 input1, I2 input2, const char * function = __PRETTY_FUNCTION__)`

2 parameter warning message

Definition at line 182 of file LogPrint.h.

Here is the call graph for this function:



9.8.3.20 `template<typename I1 > void WarningPrint (I1 input1, const char * function = __PRETTY_FUNCTION__)`

1 parameter warning message

Definition at line 191 of file LogPrint.h.

Here is the call graph for this function:



9.8.4 Variable Documentation

9.8.4.1 bool functionPrint

Boolean to indicate whether the function name should be printed for each log.

Definition at line 11 of file LogPrint.cpp.

9.8.4.2 unsigned int logLevel

This is a small hack to suppress compiler warning for default macro parameter initialization.

Logging level on how detailed the logs should be.

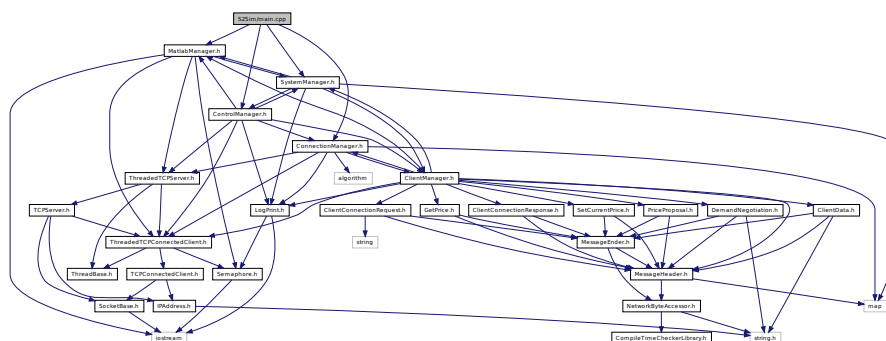
- 1: All logs
- 2: Warnings and errors
- 3: Errors only
- >3: No logging

Definition at line 10 of file LogPrint.cpp.

9.9 S2Sim/main.cpp File Reference

Main file that starts an infinite running loop on the `SystemManager::AdvanceTimeStep` method.

```
#include "MatlabManager.h"
#include "ConnectionManager.h"
#include "ControlManager.h"
#include "SystemManager.h"
Include dependency graph for main.cpp:
```



Functions

- `int main (int argc, char **argv)`

9.9.1 Detailed Description

Main file that starts an infinite running loop on the `SystemManager::AdvanceTimeStep` method.

Date

Sep 19, 2013

Author

: Alper Sinan Akyurek

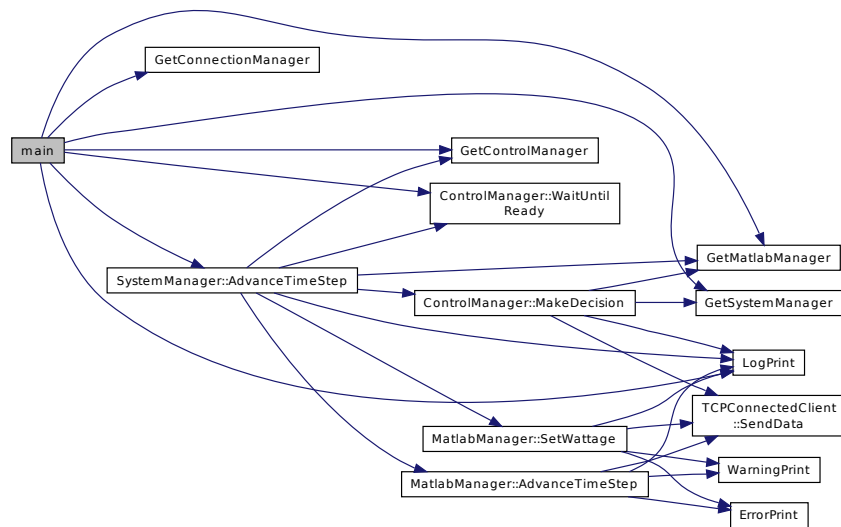
Definition in file [main.cpp](#).

9.9.2 Function Documentation

9.9.2.1 `int main (int argc, char ** argv)`

Definition at line 20 of file main.cpp.

Here is the call graph for this function:

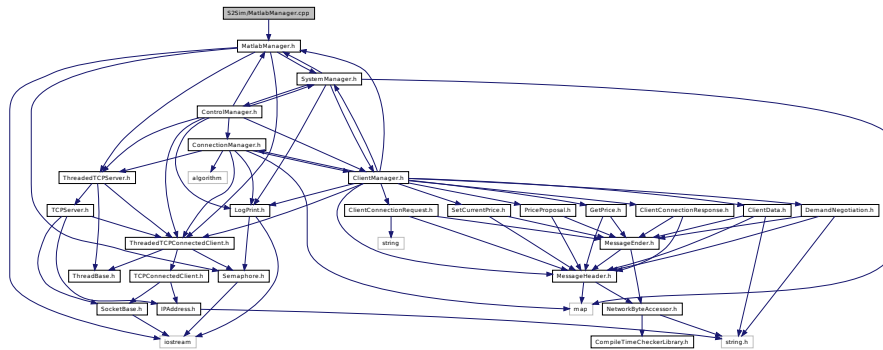


9.10 S2Sim/MatlabManager.cpp File Reference

Implements the [MatlabManager](#) class.

```
#include "MatlabManager.h"
```

Include dependency graph for MatlabManager.cpp:



Functions

- [MatlabManager](#) & [GetMatlabManager](#) (void)
This function returns the only instance of [MatlabManager](#).
- void [MatlabNotificationHandler](#) ([ThreadedTCPConnectedClient](#) *acceptedClient)
- void [MatlabReceiveHandler](#) ([ThreadedTCPConnectedClient](#) *client, [ThreadedTCPConnectedClient::TBuffer](#) buffer, [ThreadedTCPConnectedClient::TNumberOfBytes](#) size)
Wrapper callback function that forwards the received data to [MatlabManager](#).

9.10.1 Detailed Description

Implements the [MatlabManager](#) class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

Definition in file [MatlabManager.cpp](#).

9.10.2 Function Documentation

9.10.2.1 [MatlabManager](#)& [GetMatlabManager](#) (void)

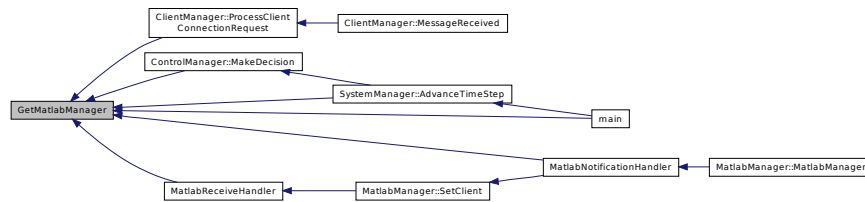
This function returns the only instance of [MatlabManager](#).

Returns

Returns the only instance of [MatlabManager](#).

Definition at line 13 of file MatlabManager.cpp.

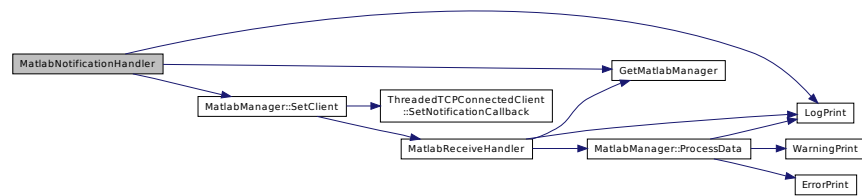
Here is the caller graph for this function:



9.10.2.2 void MatlabNotificationHandler (ThreadedTCPConnectedClient * *acceptedClient*)

Definition at line 20 of file MatlabManager.cpp.

Here is the call graph for this function:



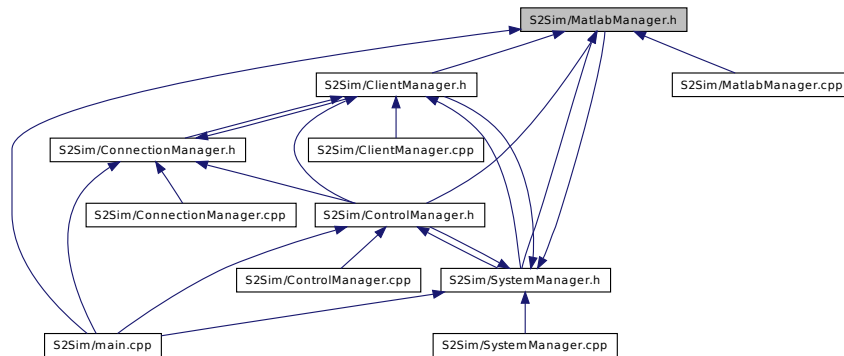
Here is the caller graph for this function:



9.10.2.3 void MatlabReceiveHandler (ThreadedTCPConnectedClient * *client*, ThreadedTCPConnectedClient::TBuffer *buffer*, ThreadedTCPConnectedClient::TNumberOfBytes *size*)

Wrapper callback function that forwards the received data to [MatlabManager](#).

This graph shows which files directly or indirectly include this file:



Classes

- class [MatlabManager](#)
Manages the connection to the OpenDSS-Matlab controller.

Functions

- [MatlabManager](#) & [GetMatlabManager](#) (void)
This function returns the only instance of [MatlabManager](#).
- void [MatlabReceiveHandler](#) ([ThreadedTCPConnectedClient](#) *client, [ThreadedTCPConnectedClient::TBuffer](#) buffer, [ThreadedTCPConnectedClient::TNumberOfBytes](#) size)
Wrapper callback function that forwards the received data to [MatlabManager](#).

9.11.1 Detailed Description

Defines the [MatlabManager](#) class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

Definition in file [MatlabManager.h](#).

9.11.2 Function Documentation

9.11.2.1 [MatlabManager](#)& [GetMatlabManager](#) (void)

This function returns the only instance of [MatlabManager](#).

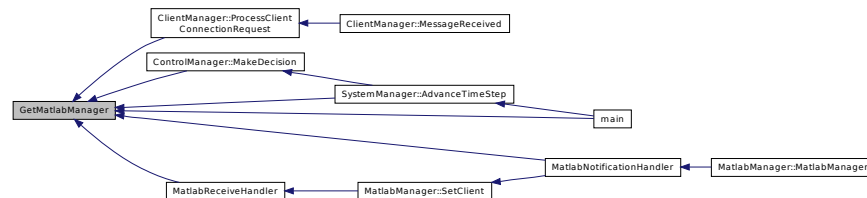
Friend method to implement the singleton for [MatlabManager](#).

Returns

The only instance of [MatlabManager](#).
Returns the only instance of [MatlabManager](#).

Definition at line 13 of file MatlabManager.cpp.

Here is the caller graph for this function:



9.11.2.2 void MatlabReceiveHandler (ThreadedTCPConnectedClient * *client*, ThreadedTCPConnectedClient::TBuffer *buffer*, ThreadedTCPConnectedClient::TNumberOfBytes *size*)

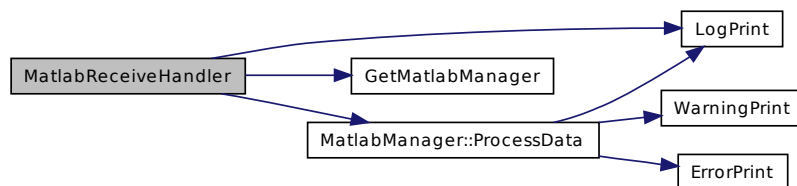
Wrapper callback function that forwards the received data to [MatlabManager](#).

Parameters

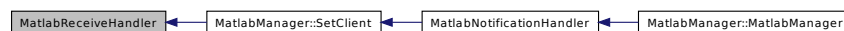
<i>client</i>	Connection information of OpenDSS-Matlab controller.
<i>buffer</i>	Buffer containing the received data.
<i>size</i>	Size of the received data.

Definition at line 29 of file MatlabManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

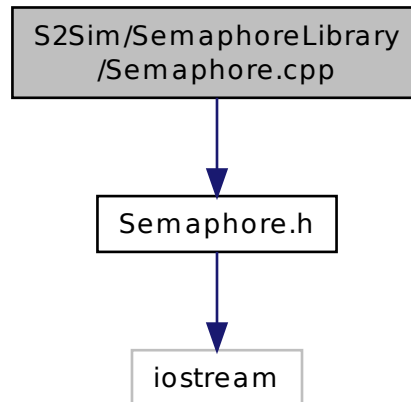


9.12 S2Sim/SemaphoreLibrary/Semaphore.cpp File Reference

Implements the Semaphore class under Windows 32/64bit and POSIX enables OSes.

```
#include "Semaphore.h"
```

Include dependency graph for Semaphore.cpp:



9.12.1 Detailed Description

Implements the Semaphore class under Windows 32/64bit and POSIX enables OSes.

Date

Jan 20, 2014

Author

Alper Sinan Akyurek

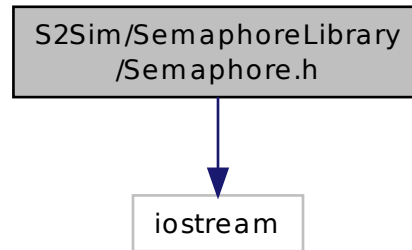
Definition in file [Semaphore.cpp](#).

9.13 S2Sim/SemaphoreLibrary/Semaphore.h File Reference

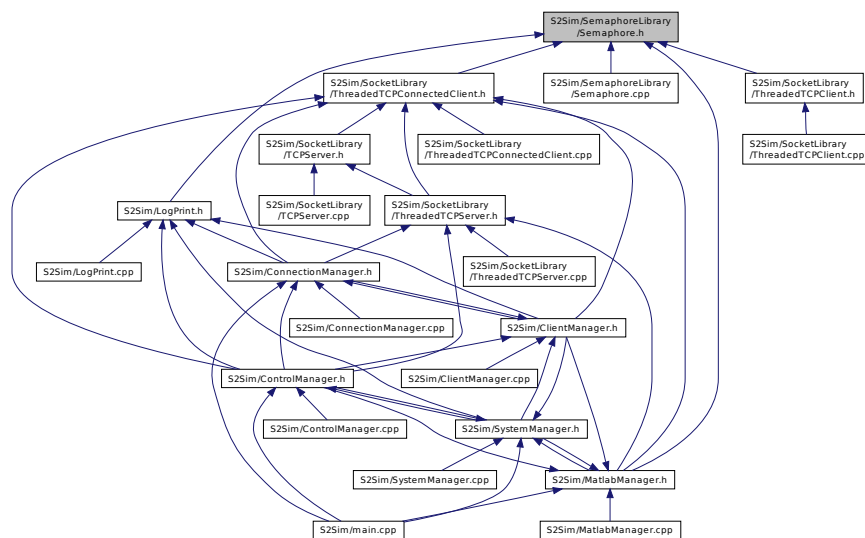
Defines the Semaphore class under Windows 32/64bit and POSIX enables OSes.

```
#include <iostream>
```

Include dependency graph for Semaphore.h:



This graph shows which files directly or indirectly include this file:



9.13.1 Detailed Description

Defines the Semaphore class under Windows 32/64bit and POSIX enables OSes.

Date

Jan 20, 2014

Author

: Alper Sinan Akyurek

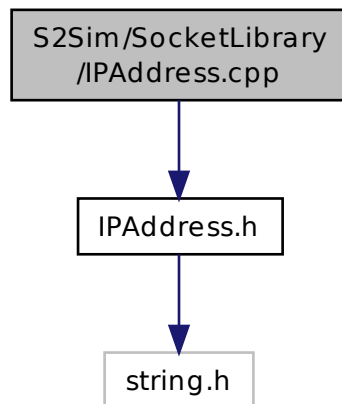
Definition in file [Semaphore.h](#).

9.14 S2Sim/SocketLibrary/IpAddress.cpp File Reference

Implements the [IpAddress](#) class.

```
#include "IpAddress.h"
```

Include dependency graph for IpAddress.cpp:



9.14.1 Detailed Description

Implements the [IpAddress](#) class.

Date

Jan 21, 2014

Author

: Alper Sinan Akyurek

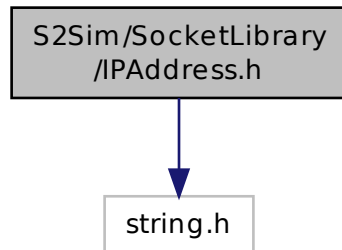
Definition in file [IpAddress.cpp](#).

9.15 S2Sim/SocketLibrary/IpAddress.h File Reference

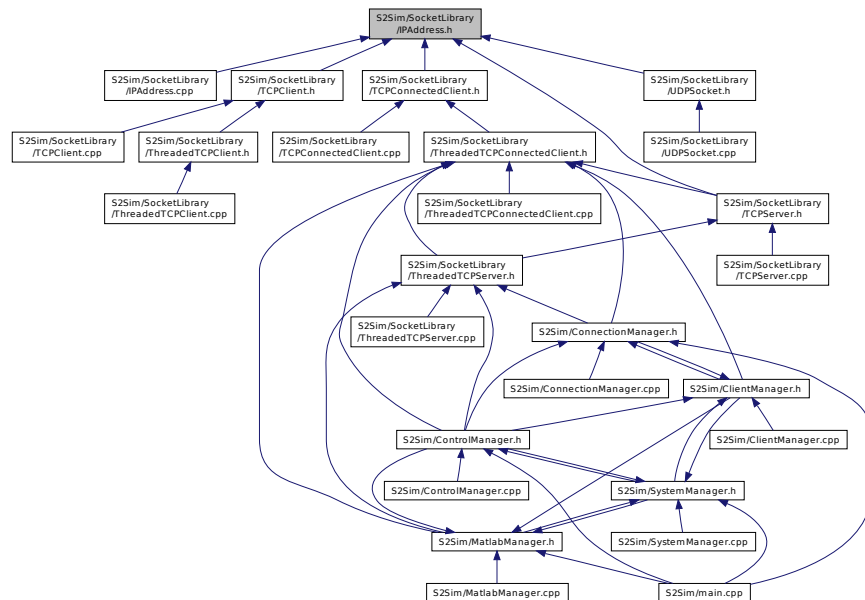
Defines the [IpAddress](#) class.

```
#include <string.h>
```

Include dependency graph for IPAddress.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [IPAddress](#)

This class is an abstraction of the OS IP Address structure.

9.15.1 Detailed Description

Defines the [IPAddress](#) class.

Date

Jan 21, 2014

Author

: Alper Sinan Akyurek

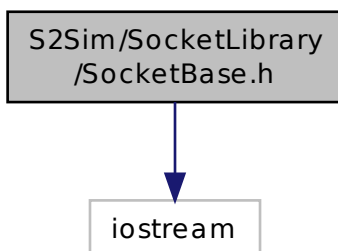
Definition in file [IPAddress.h](#).

9.16 S2Sim/SocketLibrary/SocketBase.h File Reference

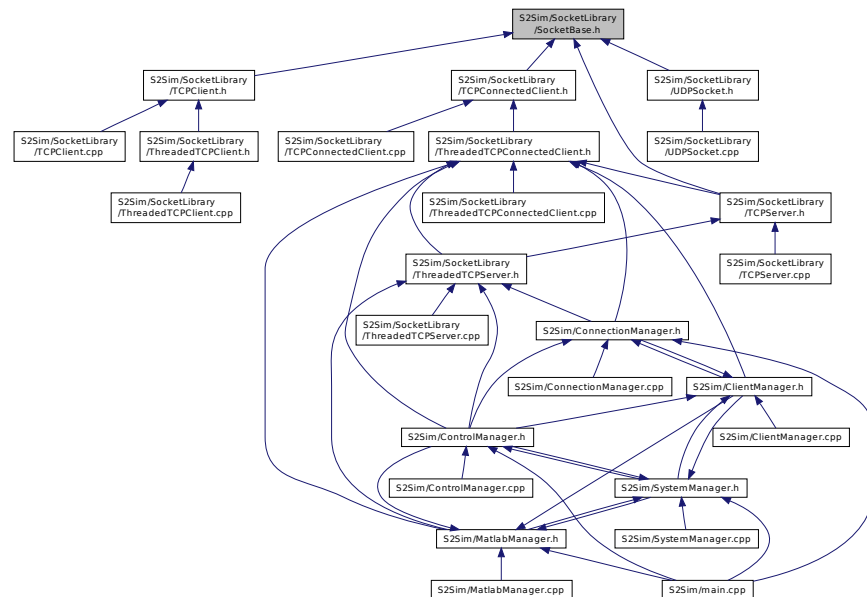
Defines the [SocketBase](#) template class.

```
#include <iostream>
```

Include dependency graph for SocketBase.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SocketBase< SocketType >](#)

This class is an abstraction over the OS socket and defines a base class for socket opening and closing utilities.

9.16.1 Detailed Description

Defines the [SocketBase](#) template class.

Date

Jun 21, 2013

Author

: Alper Sinan Akyurek

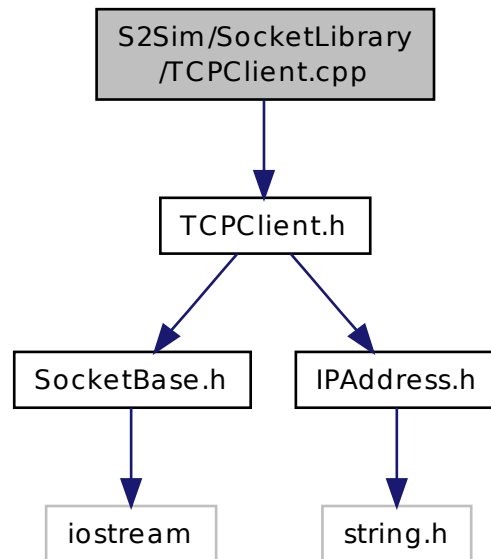
Definition in file [SocketBase.h](#).

9.17 S2Sim/SocketLibrary/TCPClient.cpp File Reference

Implements the [TCPClient](#) class.

```
#include "TCPClient.h"
```

Include dependency graph for TCPClient.cpp:



9.17.1 Detailed Description

Implements the [TCPClient](#) class.

Date

Jan 21, 2014

Author

: Alper Sinan Akyurek

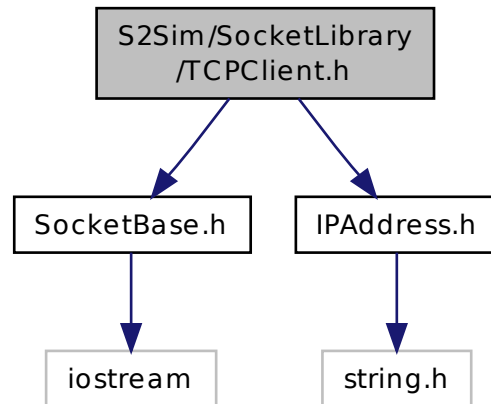
Definition in file [TCPClient.cpp](#).

9.18 S2Sim/SocketLibrary/TCPClient.h File Reference

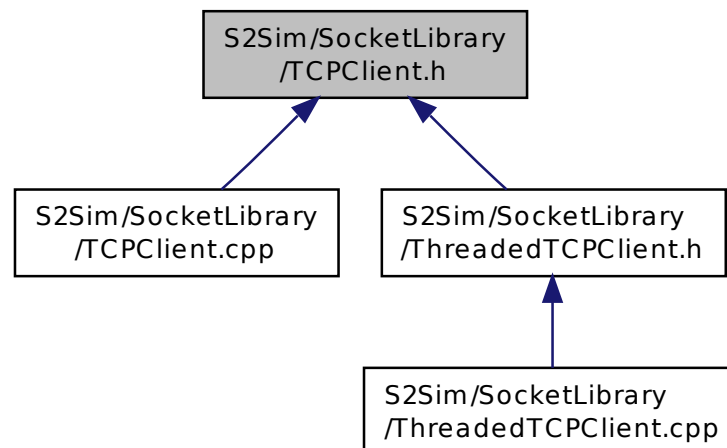
Defines the [TCPClient](#) class.

```
#include "SocketBase.h"
#include "IPAddress.h"
```

Include dependency graph for TCPClient.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TCPClient](#)

This class defines a TCP client that can connect to a TCP server and communicate.

9.18.1 Detailed Description

Defines the [TCPClient](#) class.

Date

Jan 21, 2014

Author

: Alper Sinan Akyurek

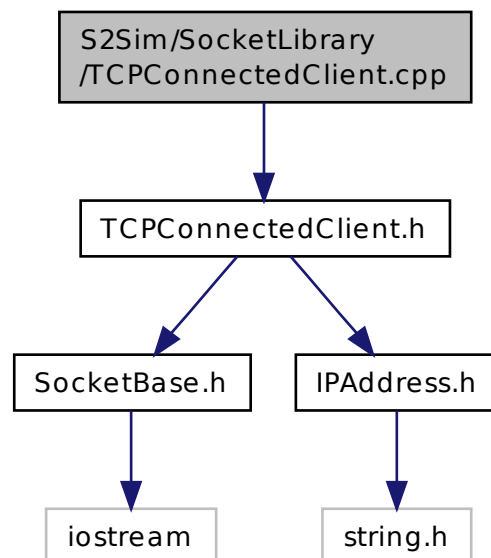
Definition in file [TCPClient.h](#).

9.19 S2Sim/SocketLibrary/TCPConnectedClient.cpp File Reference

Implements the [TCPConnectedClient](#) class.

```
#include "TCPConnectedClient.h"
```

Include dependency graph for TCPConnectedClient.cpp:



9.19.1 Detailed Description

Implements the [TCPConnectedClient](#) class. Jan 21, 2014

Author

: Alper Sinan Akyurek

Definition in file [TCPConnectedClient.cpp](#).

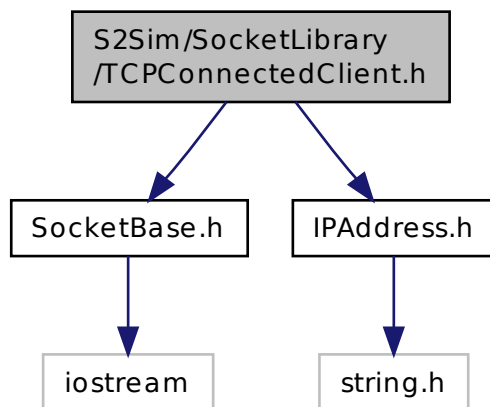
9.20 S2Sim/SocketLibrary/TCPConnectedClient.h File Reference

Defines the [TCPConnectedClient](#) class.

```
#include "SocketBase.h"
```

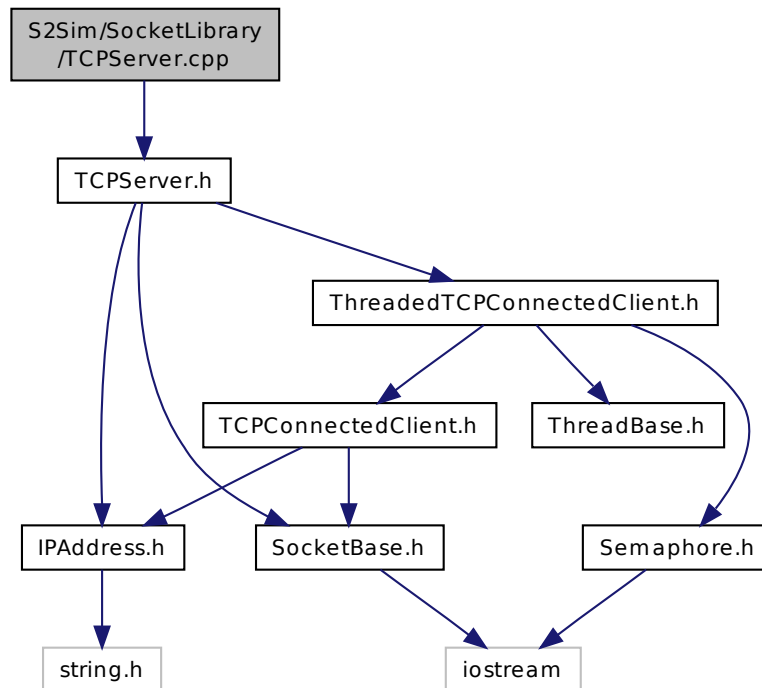
```
#include "IPAddress.h"
```

Include dependency graph for TCPConnectedClient.h:




```
#include "TCPServer.h"
```

Include dependency graph for TCPServer.cpp:



9.21.1 Detailed Description

Implements the [TCPServer](#) class.

Date

Jan 21, 2014

Author

: Alper Sinan Akyurek

Definition in file [TCPServer.cpp](#).

9.22 S2Sim/SocketLibrary/TCPServer.h File Reference

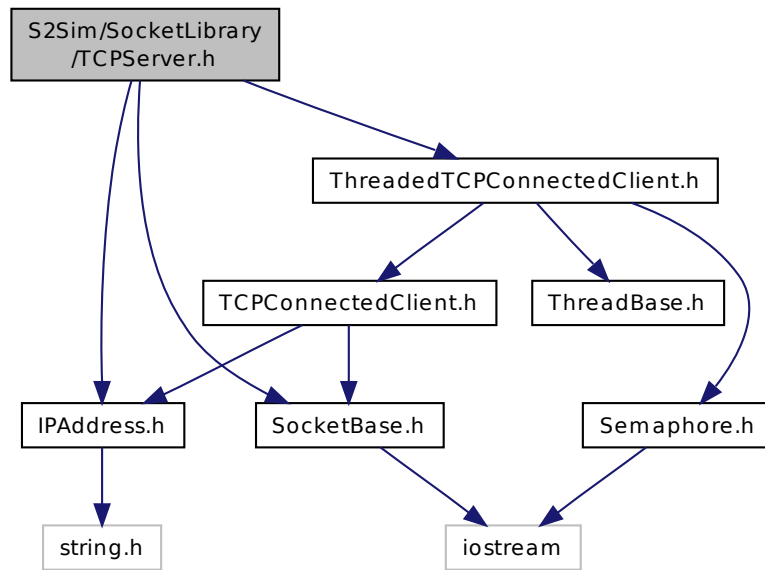
Defines the [TCPServer](#) class.

```
#include "SocketBase.h"
```

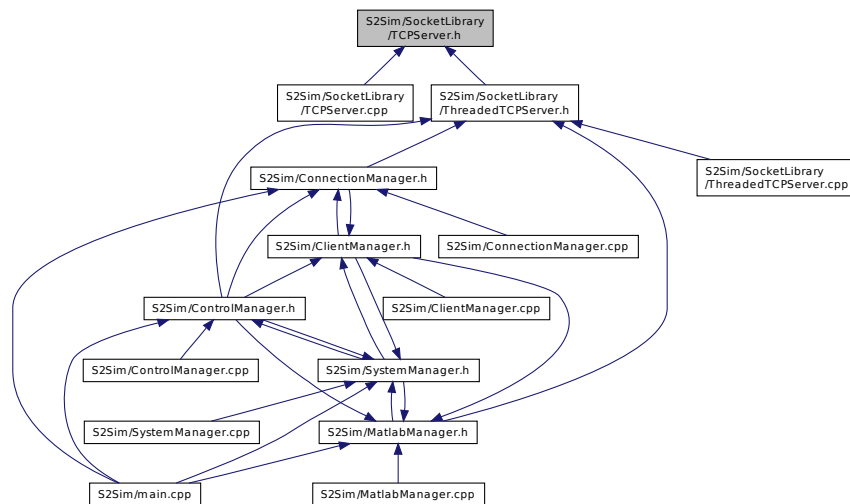
```
#include "IPAddress.h"
```

```
#include "ThreadedTCPConnectedClient.h"
```

Include dependency graph for TCPServer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TCPServer](#)

Defines a TCP server that can listen to connection attempts and can accept them for communication.

9.22.1 Detailed Description

Defines the [TCPServer](#) class.

Date

Jan 21, 2014

Author

: Alper Sinan Akyurek

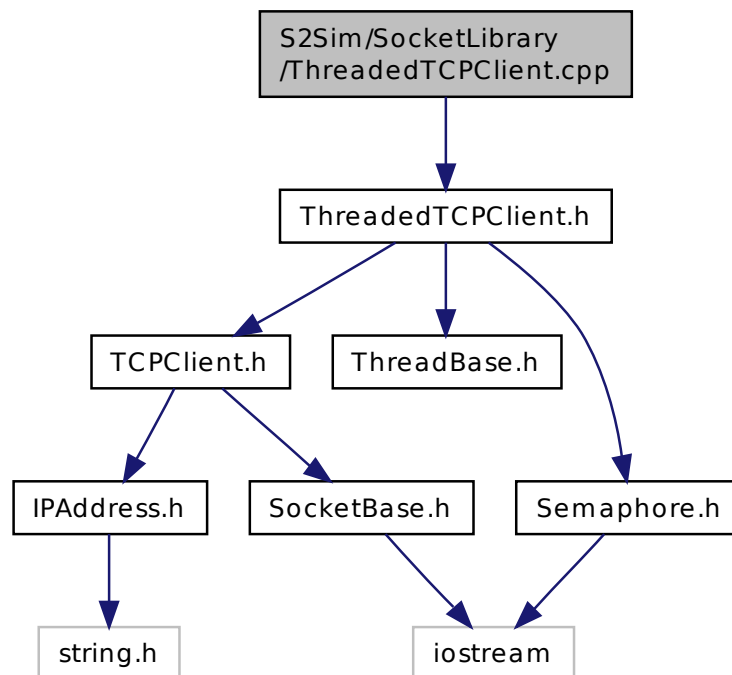
Definition in file [TCPServer.h](#).

9.23 S2Sim/SocketLibrary/ThreadedTCPClient.cpp File Reference

Implements the [ThreadedTCPClient](#) class.

```
#include "ThreadedTCPClient.h"
```

Include dependency graph for ThreadedTCPClient.cpp:



9.23.1 Detailed Description

Implements the [ThreadedTCPClient](#) class.

Date

Jan 22, 2014

Author

: Alper Sinan Akyurek

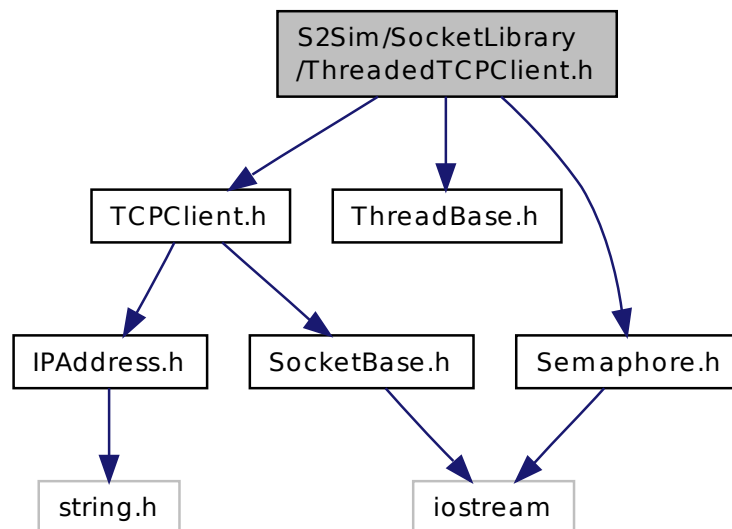
Definition in file [ThreadedTCPClient.cpp](#).

9.24 S2Sim/SocketLibrary/ThreadedTCPClient.h File Reference

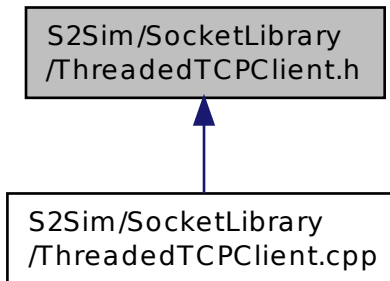
Defines the [ThreadedTCPClient](#) class.

```
#include "TCPClient.h"  
#include "ThreadBase.h"  
#include "Semaphore.h"
```

Include dependency graph for ThreadedTCPClient.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ThreadedTCPClient](#)

This is a TCP client class that receives data in a separate thread in the background.

9.24.1 Detailed Description

Defines the [ThreadedTCPClient](#) class.

Date

Jan 22, 2014

Author

: Alper Sinan Akyurek

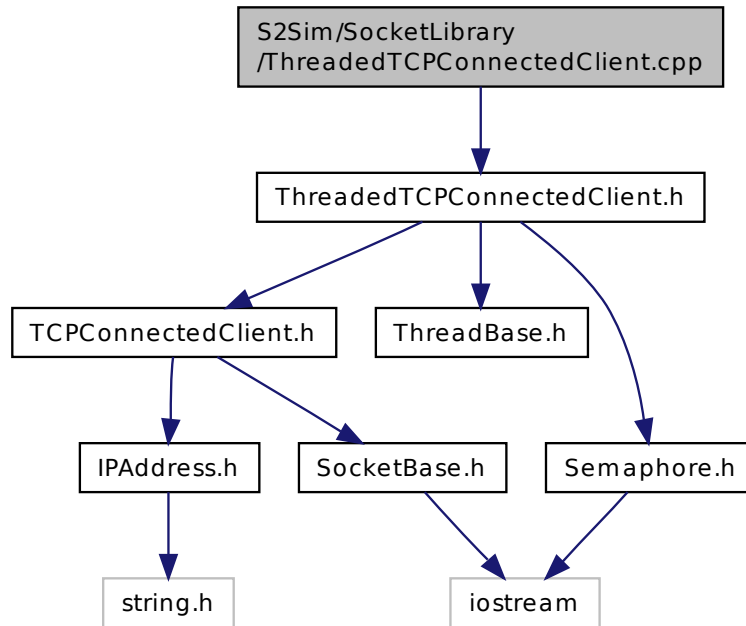
Definition in file [ThreadedTCPClient.h](#).

9.25 S2Sim/SocketLibrary/ThreadedTCPConnectedClient.cpp File Reference

Implements the [ThreadedTCPConnectedClient](#) class.

```
#include "ThreadedTCPConnectedClient.h"
```

Include dependency graph for ThreadedTCPConnectedClient.cpp:



9.25.1 Detailed Description

Implements the [ThreadedTCPConnectedClient](#) class.

Date

Jan 22, 2014

Author

: Alper Sinan Akyurek

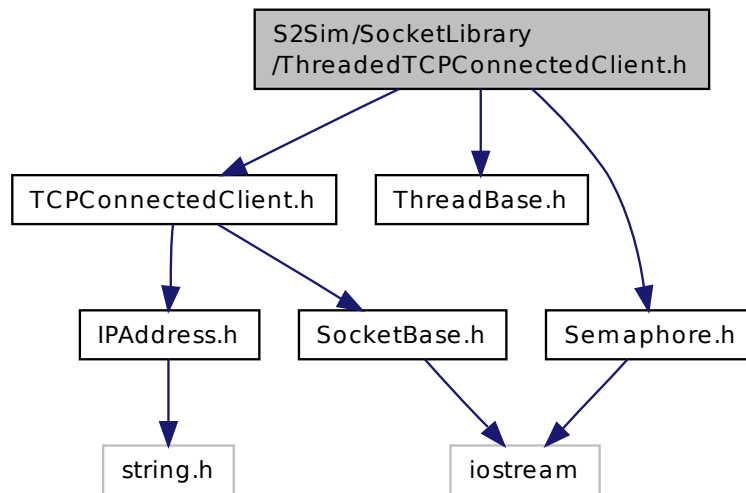
Definition in file [ThreadedTCPConnectedClient.cpp](#).

9.26 S2Sim/SocketLibrary/ThreadedTCPConnectedClient.h File Reference

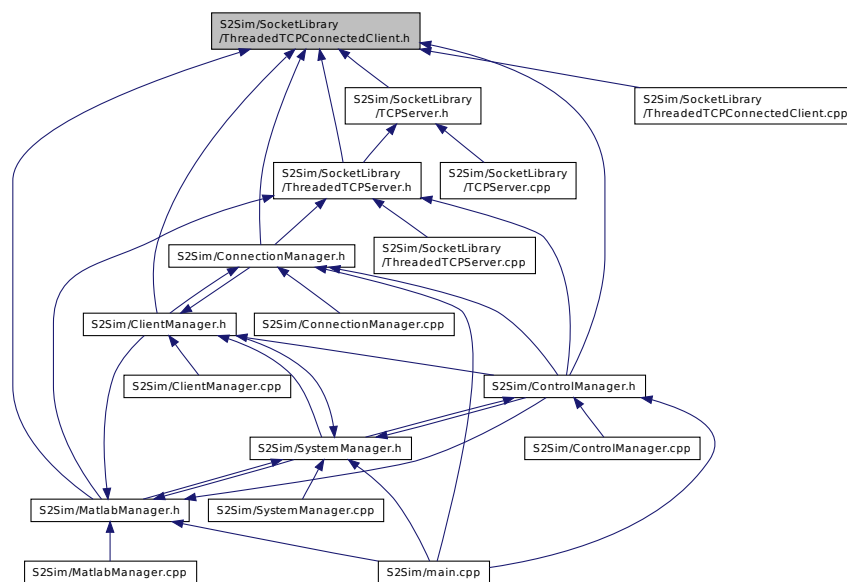
Defeines the [ThreadedTCPConnectedClient](#) class.

```
#include "TCPConnectedClient.h"
#include "ThreadBase.h"
#include "Semaphore.h"
```

Include dependency graph for ThreadedTCPConnectedClient.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ThreadedTCPConnectedClient

Manages the connection to an accepted client on the server side and receives data in a separate thread in the background.

9.26.1 Detailed Description

Defines the [ThreadedTCPConnectedClient](#) class.

Date

Jan 22, 2014

Author

: Alper Sinan Akyurek

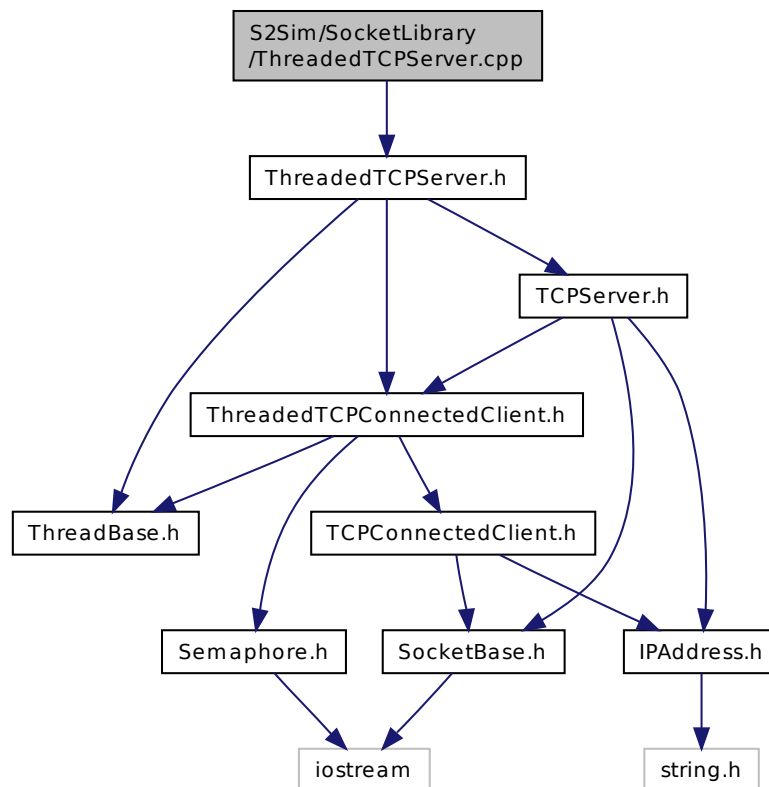
Definition in file [ThreadedTCPConnectedClient.h](#).

9.27 S2Sim/SocketLibrary/ThreadedTCPServer.cpp File Reference

Implements the [ThreadedTCPServer](#) class.

```
#include "ThreadedTCPServer.h"
```

Include dependency graph for ThreadedTCPServer.cpp:



9.27.1 Detailed Description

Implements the [ThreadedTCPServer](#) class.

Date

Jan 22, 2014

Author

: Alper Sinan Akyurek

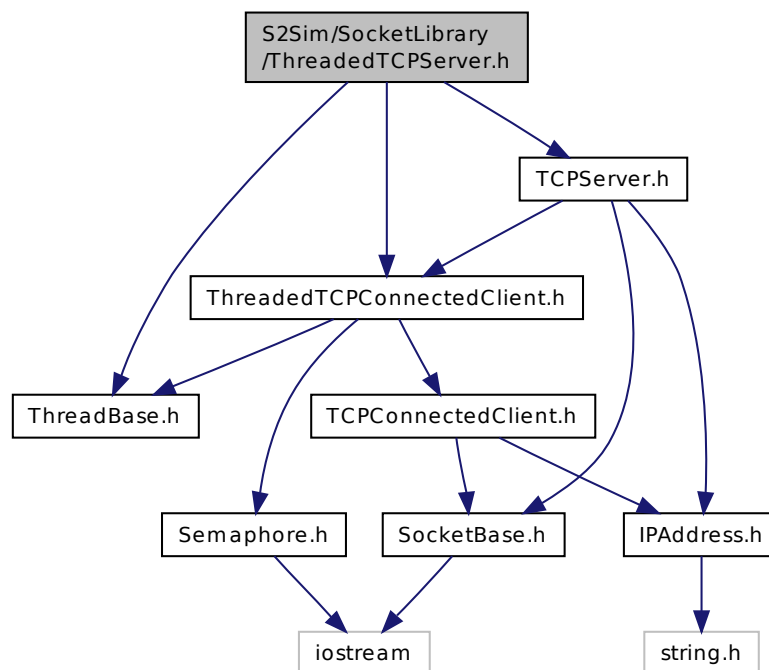
Definition in file [ThreadedTCPServer.cpp](#).

9.28 S2Sim/SocketLibrary/ThreadedTCPServer.h File Reference

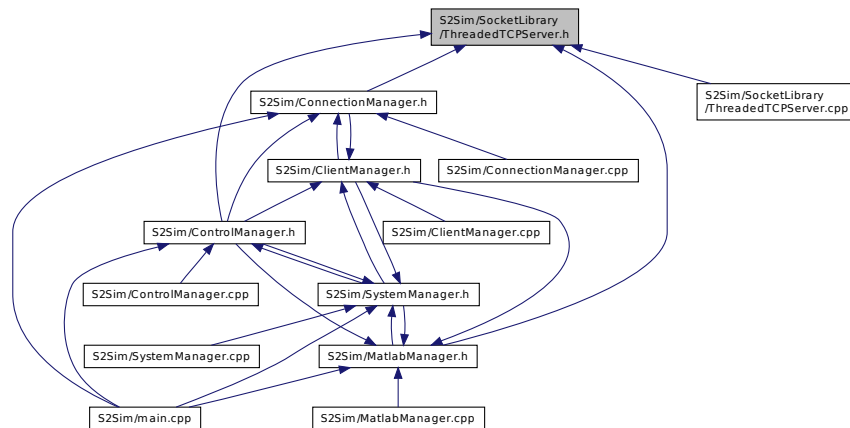
Defines the [ThreadedTCPServer](#) class.

```
#include "ThreadBase.h"
#include "TCPServer.h"
#include "ThreadedTCPConnectedClient.h"
```

Include dependency graph for ThreadedTCPServer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ThreadedTCPServer](#)

Defines the threaded version of [TCPServer](#) that accepts clients in another thread in the background.

9.28.1 Detailed Description

Defines the [ThreadedTCPServer](#) class.

Date

Jan 22, 2014

Author

: Alper Sinan Akyurek

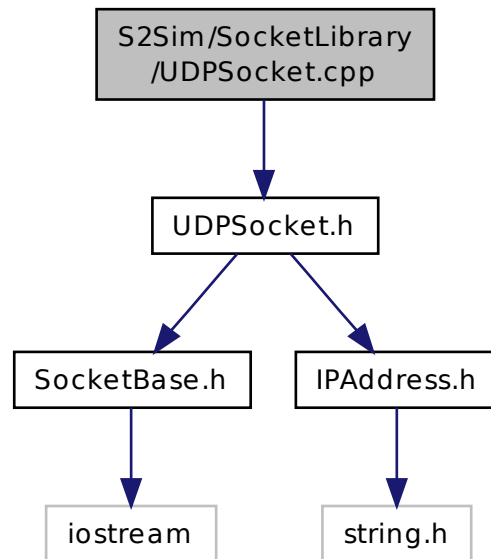
Definition in file [ThreadedTCPServer.h](#).

9.29 S2Sim/SocketLibrary/UDPSocket.cpp File Reference

Implements the [UDPSocket](#) class.


```
#include "UDPSocket.h"
```

Include dependency graph for UDPSocket.cpp:



9.29.1 Detailed Description

Implements the [UDPSocket](#) class.

Date

Jun 21, 2013

Author

: Alper Sinan Akyurek

Definition in file [UDPSocket.cpp](#).

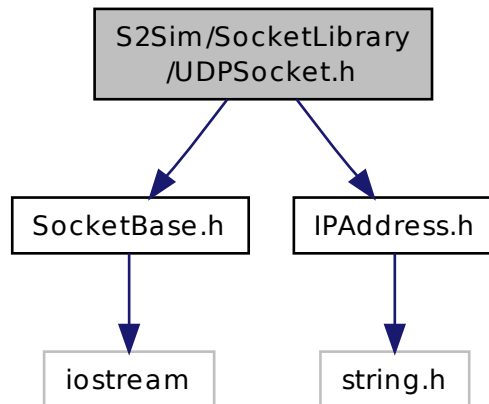
9.30 S2Sim/SocketLibrary/UDPSocket.h File Reference

Defines the [UDPSocket](#) class.

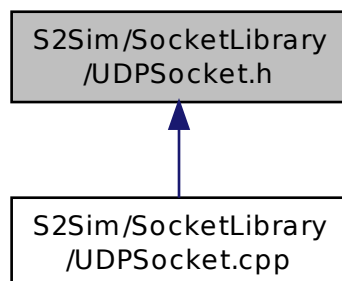
```
#include "SocketBase.h"
```

```
#include "IPAddress.h"
```

Include dependency graph for UDPSocket.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `UDPSocket`
Manages a UDP connection.

9.30.1 Detailed Description

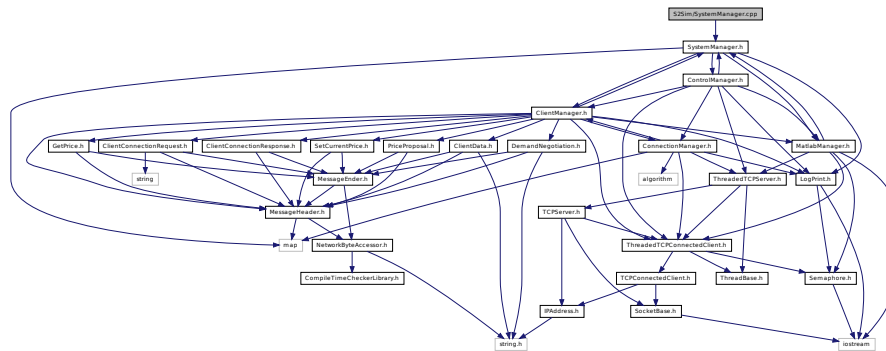
Defines the `UDPSocket` class.

Jun 21, 2013

: Alper Sinan Akyurek

9.31 S2Sim/SystemManager.cpp File Reference

Include dependency graph for SystemManager.cpp:



- **SystemManager & GetSystemManager** (void)

9.31.1 Detailed Description

Oct 25, 2013

: Alper Sinan Akyurek

Generated on Fri Feb 21 2014 15:21:35 for S2Sim by Doxygen

9.31.2 Function Documentation

9.31.2.1 SystemManager& GetSystemManager (void)

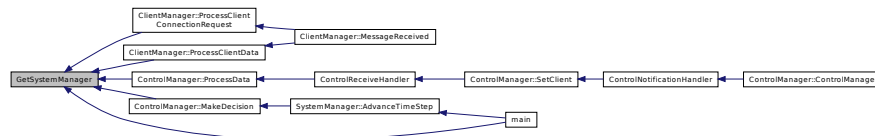
Singleton function that returns the only instance of [SystemManager](#).

Returns

Returns the only instance of [SystemManager](#).

Definition at line 13 of file SystemManager.cpp.

Here is the caller graph for this function:

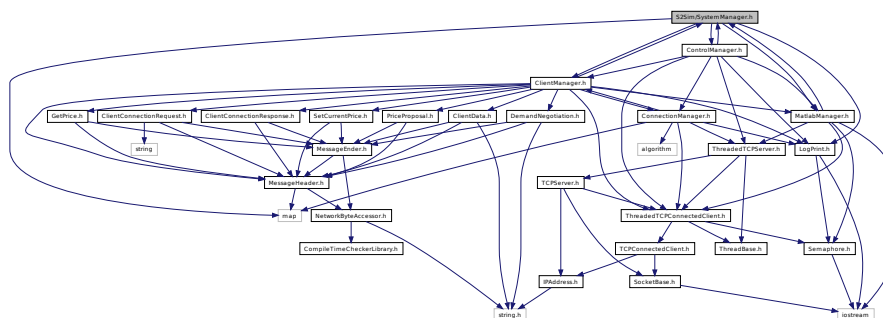


9.32 S2Sim/SystemManager.h File Reference

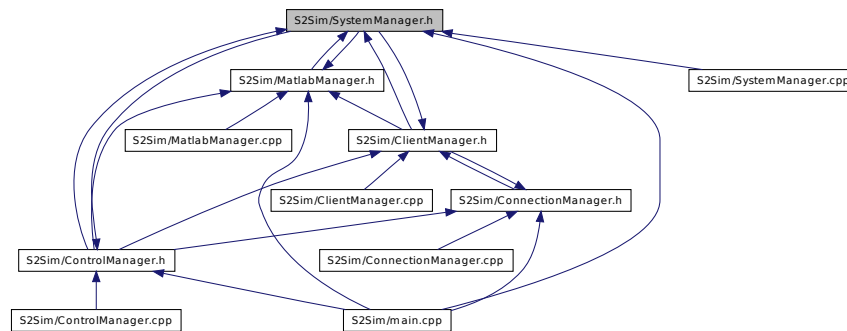
Defines the [SystemManager](#) class.

```
#include <map>
#include "ClientManager.h"
#include "MatlabManager.h"
#include "ControlManager.h"
#include "LogPrint.h"
```

Include dependency graph for SystemManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [SystemManager](#)
Manages the various components of the system and timing.

Functions

- [SystemManager](#) & [GetSystemManager](#) (void)
Singleton function that returns the only instance of [SystemManager](#).

9.32.1 Detailed Description

Defines the [SystemManager](#) class.

Date

Oct 25, 2013

Author

: Alper Sinan Akyurek

Definition in file [SystemManager.h](#).

9.32.2 Function Documentation

9.32.2.1 SystemManager& GetSystemManager (void)

Singleton function that returns the only instance of [SystemManager](#).

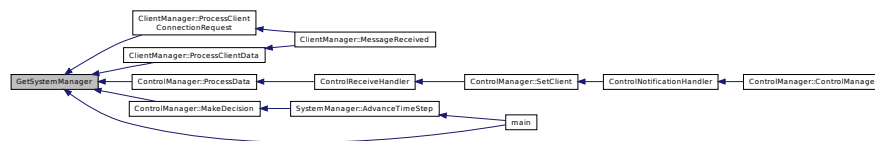
Friend function to implement the singleton.

Returns

The only instance of [SystemManager](#).
Returns the only instance of [SystemManager](#).

Definition at line 13 of file `SystemManager.cpp`.

Here is the caller graph for this function:

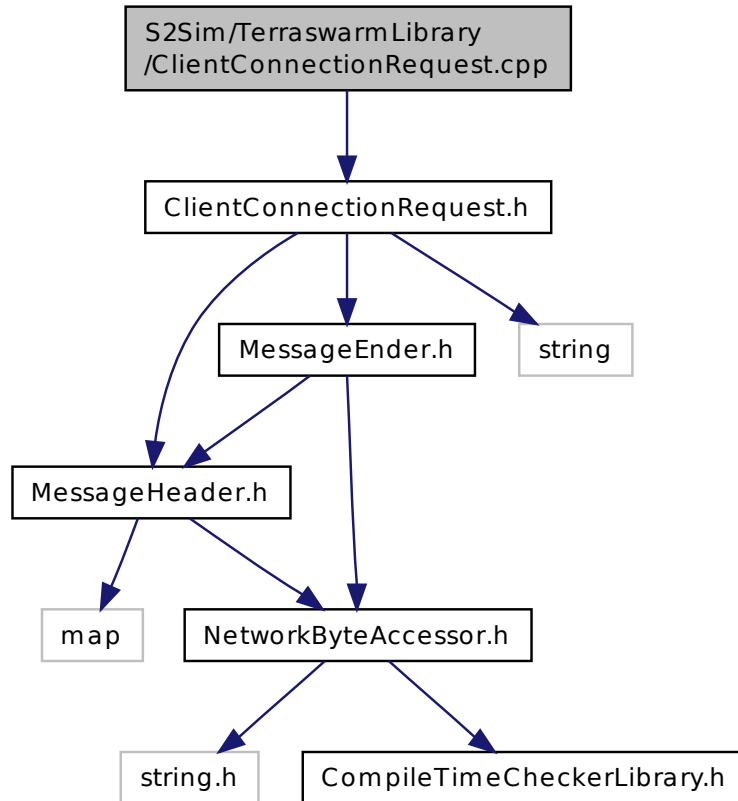


9.33 S2Sim/TerraswarmLibrary/ClientConnectionRequest.cpp File Reference

Implements the `ClientConnectionRequest` class.

```
#include "ClientConnectionRequest.h"
```

Include dependency graph for ClientConnectionRequest.cpp:



Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Asynchronous](#)
Asynchronous client messages are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.33.1 Detailed Description

Implements the `ClientConnectionRequest` class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

Definition in file [ClientConnectionRequest.cpp](#).

9.34 S2Sim/TerraswarmLibrary/ClientConnectionRequest.h File Reference

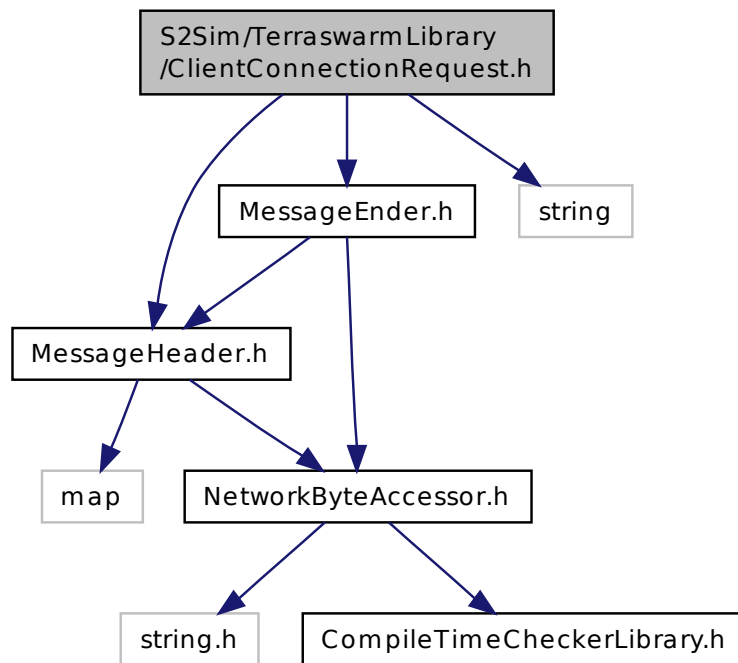
Defines the ClientConnectionRequest classes.

```
#include "MessageHeader.h"
```

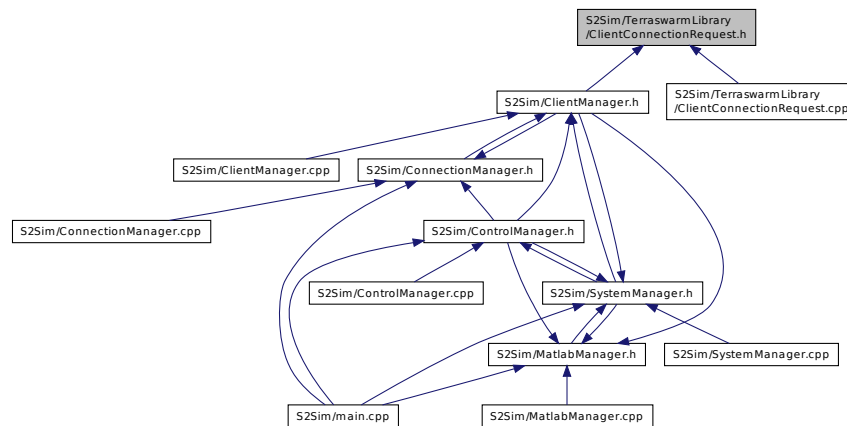
```
#include "MessageEnder.h"
```

```
#include <string>
```

Include dependency graph for ClientConnectionRequest.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::Asynchronous::ClientConnectionRequest](#)
This class implements the asynchronous client connection request message.
- class [TerraSwarm::Synchronous::ClientConnectionRequest](#)
This class implements the synchronous client connection request message.

Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Asynchronous](#)
Asynchronous client messages are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.34.1 Detailed Description

Defines the ClientConnectionRequest classes.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

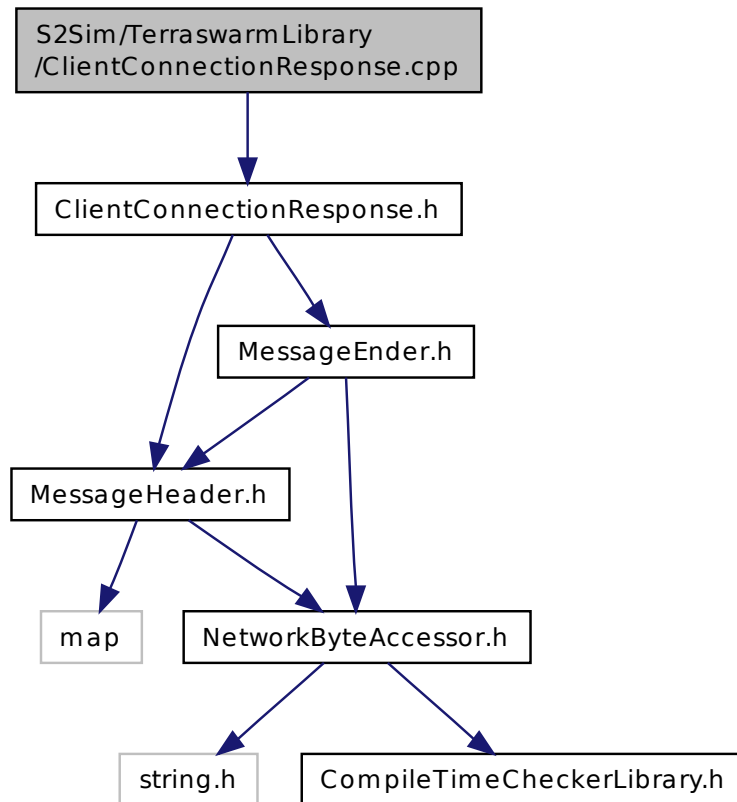
Definition in file [ClientConnectionRequest.h](#).

9.35 S2Sim/TerraswarmLibrary/ClientConnectionResponse.cpp File Reference

Implements the Async and Synchronous ClientConnectionResponse message and classes.

```
#include "ClientConnectionResponse.h"
```

Include dependency graph for ClientConnectionResponse.cpp:



Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Asynchronous](#)
Asynchronous client messages are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.35.1 Detailed Description

Implements the Async and Synchronous ClientConnectionResponse message and classes.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

Definition in file [ClientConnectionResponse.cpp](#).

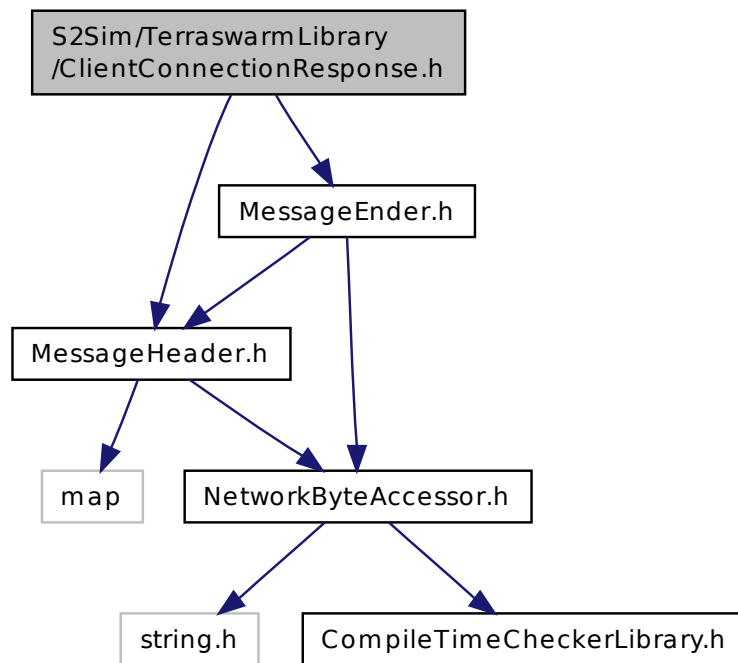
9.36 S2Sim/TerraswarmLibrary/ClientConnectionResponse.h File Reference

Defines the Async and Synchronous ClientConnectionResponse messages and classes.

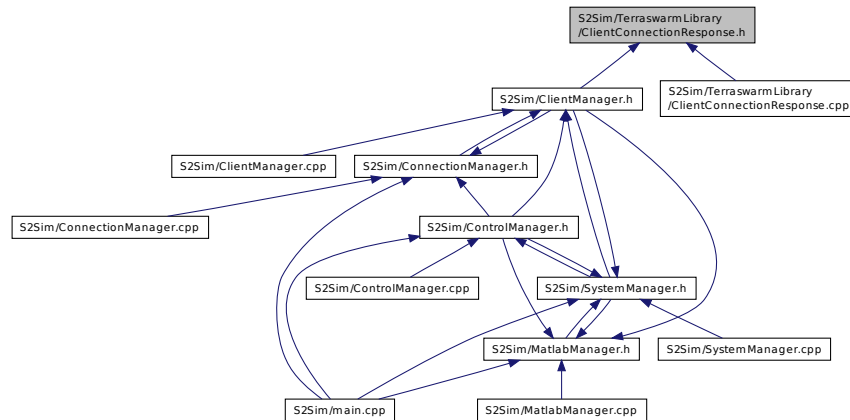
#include "MessageHeader.h"

#include "MessageEnder.h"

Include dependency graph for ClientConnectionResponse.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::Asynchronous::ClientConnectionResponse](#)
This class implements the Response message of the controller to the clients connection request.
- class [TerraSwarm::Synchronous::ClientConnectionResponse](#)
This class implements the Response message of the controller to the clients connection request.

Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Asynchronous](#)
Asynchronous client messages are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.36.1 Detailed Description

Defines the Asnyc and Synchronous ClientConnectionResponse messages and classes.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

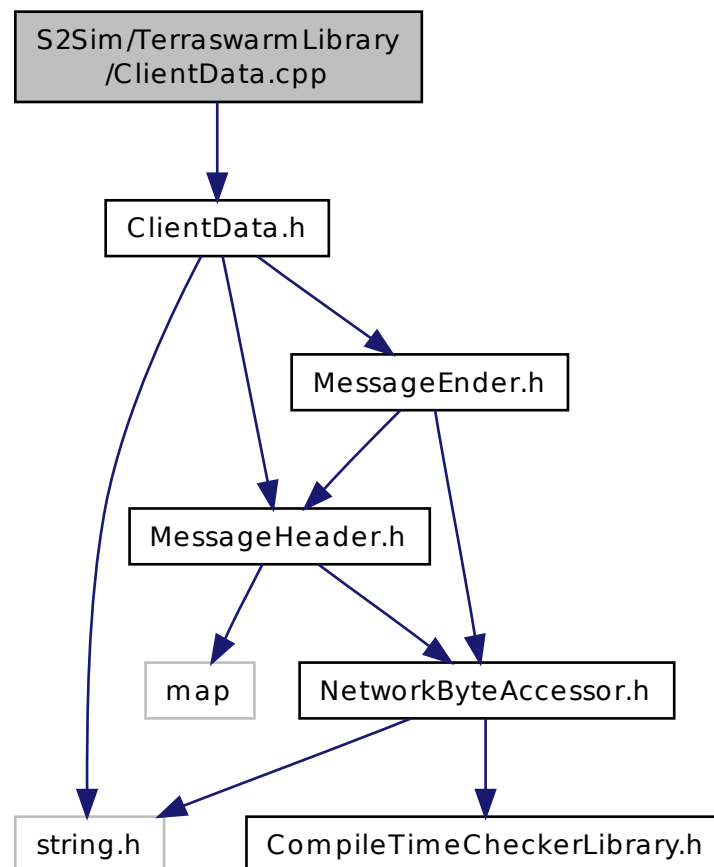
Definition in file [ClientConnectionResponse.h](#).

9.37 S2Sim/TerraswarmLibrary/ClientData.cpp File Reference

Implements the Async and Synchronous ClientData classes and messages.

```
#include "ClientData.h"
```

Include dependency graph for ClientData.cpp:



Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Asynchronous](#)
Asynchronous client messages are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.37.1 Detailed Description

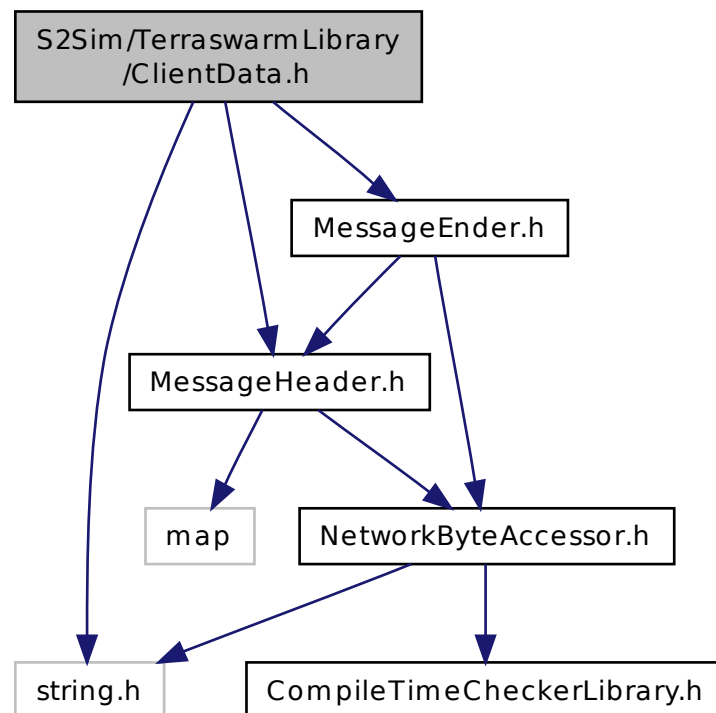
Implements the Async and Synchronous ClientData classes and messages. Created on: Oct 13, 2013 Author: Alper
Definition in file [ClientData.cpp](#).

9.38 S2Sim/TerraswarmLibrary/ClientData.h File Reference

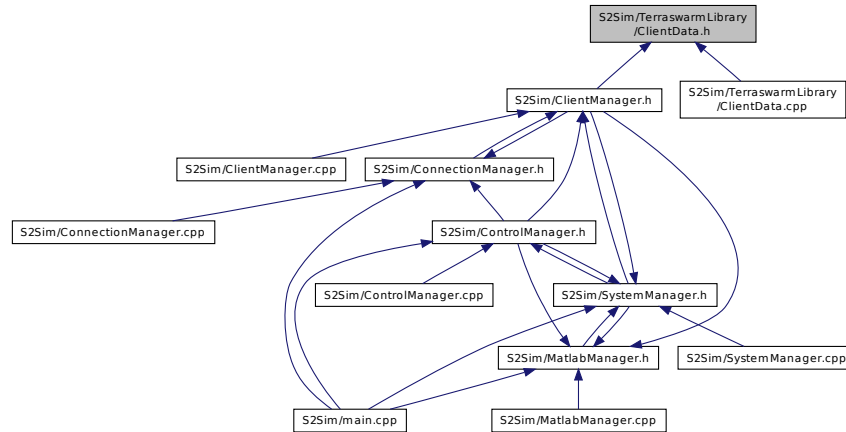
Defines the Aysnchronous and Synchronous ClientData class and messages.

```
#include "MessageHeader.h"  
#include "MessageEnder.h"  
#include <string.h>
```

Include dependency graph for ClientData.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::Asynchronous::ClientData](#)
Asynchronous Client Data message from the client to indicate its consumption for a time interval.
- class [TerraSwarm::Synchronous::ClientData](#)
Synchronous Client Data message from the client to indicate its consumption for a time interval.

Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Asynchronous](#)
Asynchronous client messages are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

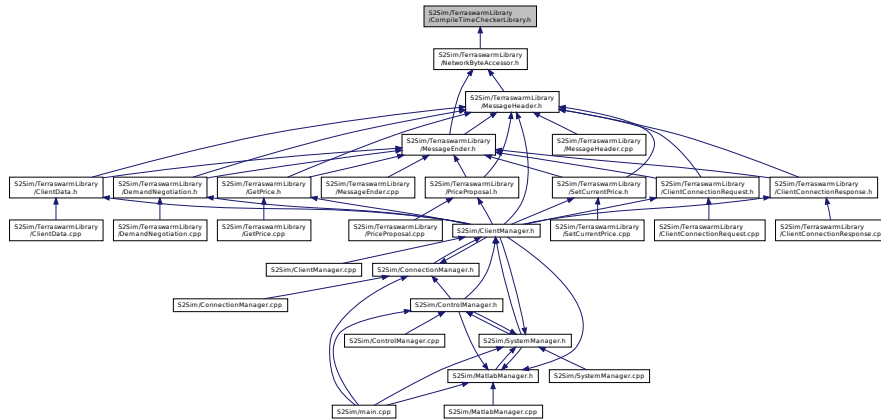
9.38.1 Detailed Description

Defines the Aysnchronous and Synchronous ClientData class and messages. Created on: Oct 13, 2013 Author: Alper
Definition in file [ClientData.h](#).

9.39 S2Sim/TerraswarmLibrary/CompileTimeCheckerLibrary.h File Reference

This file contains template classes to do compile time checking.

This graph shows which files directly or indirectly include this file:



Classes

- class [CompileCheck< expression, Reason >](#)

This class has two specializations.

- class [CompileCheck< false, Reason >](#)

Second specialization of the class, when the expression is not true.

- class [SizeCheck< checkedType, checkedSize, Reason >](#)

This class checks the size of a type with the given size and gives a compile error with the reason parameter is the check fails.

9.39.1 Detailed Description

This file contains template classes to do compile time checking. Created on: Oct 8, 2013 Author: Alper

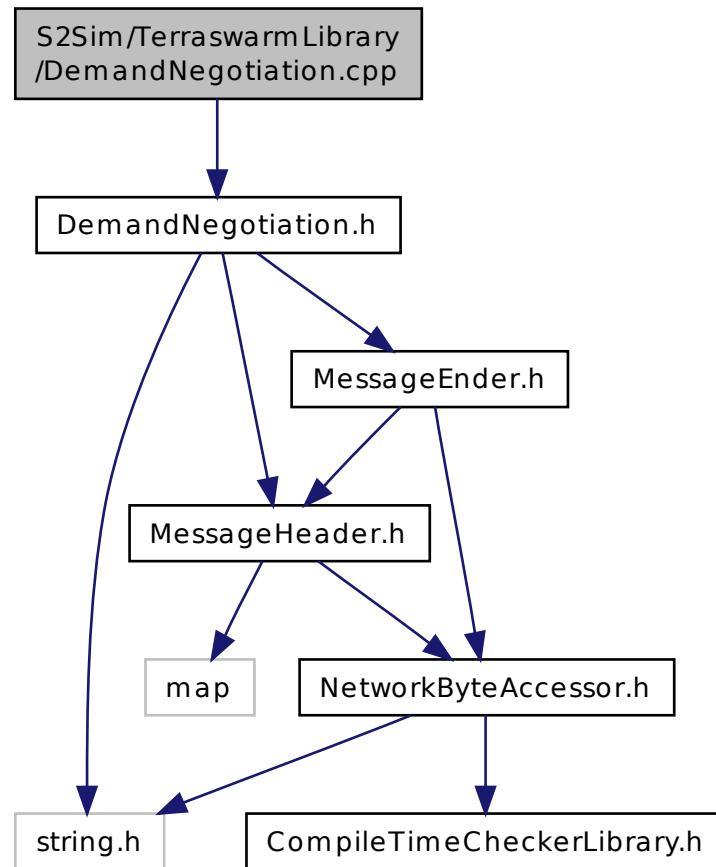
Definition in file [CompileTimeCheckerLibrary.h](#).

9.40 S2Sim/TerraswarmLibrary/DemandNegotiation.cpp File Reference

Implements the DemandNegotiation class.


```
#include "DemandNegotiation.h"
```

Include dependency graph for DemandNegotiation.cpp:



Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.40.1 Detailed Description

Implements the `DemandNegotiation` class.

Date

Oct 31, 2013

Author

: Alper Sinan Akyurek

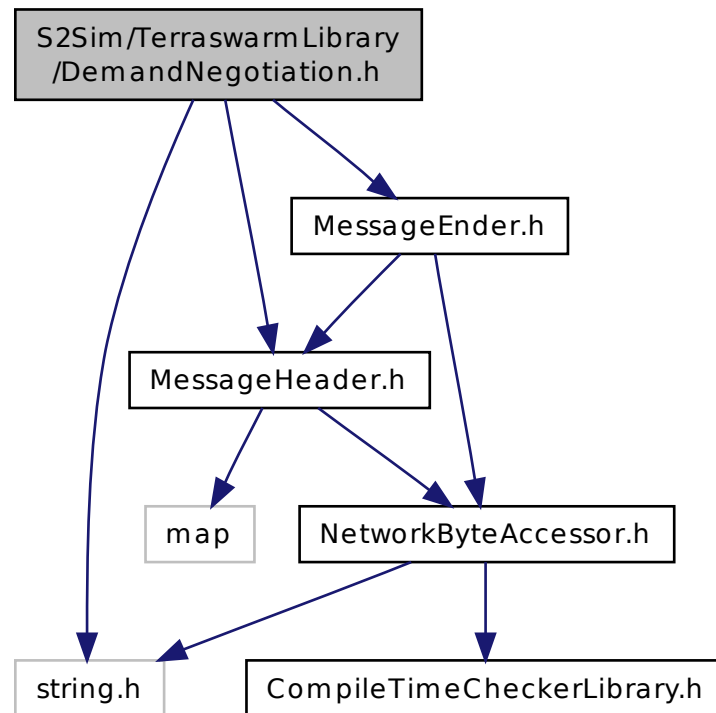
Definition in file [DemandNegotiation.cpp](#).

9.41 S2Sim/TerraswarmLibrary/DemandNegotiation.h File Reference

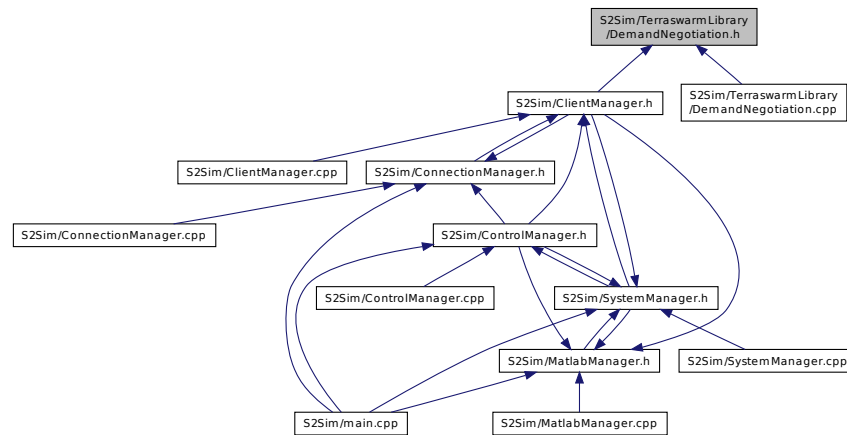
Defines the DemandNegotiation class and message.

```
#include "MessageHeader.h"  
#include "MessageEnder.h"  
#include <string.h>
```

Include dependency graph for DemandNegotiation.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::Synchronous::DemandNegotiation](#)
Defines the [DemandNegotiation](#) message sent from the client to the Controller as a response to the price proposal.

Namespaces

- [TerraSwarm](#)
[TerraSwarm](#) related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
[Synchronous](#) client message are defined under this namespace.

9.41.1 Detailed Description

Defines the DemandNegotiation class and message.

Date

Oct 31, 2013

Author

: Alper Sinan Akyurek

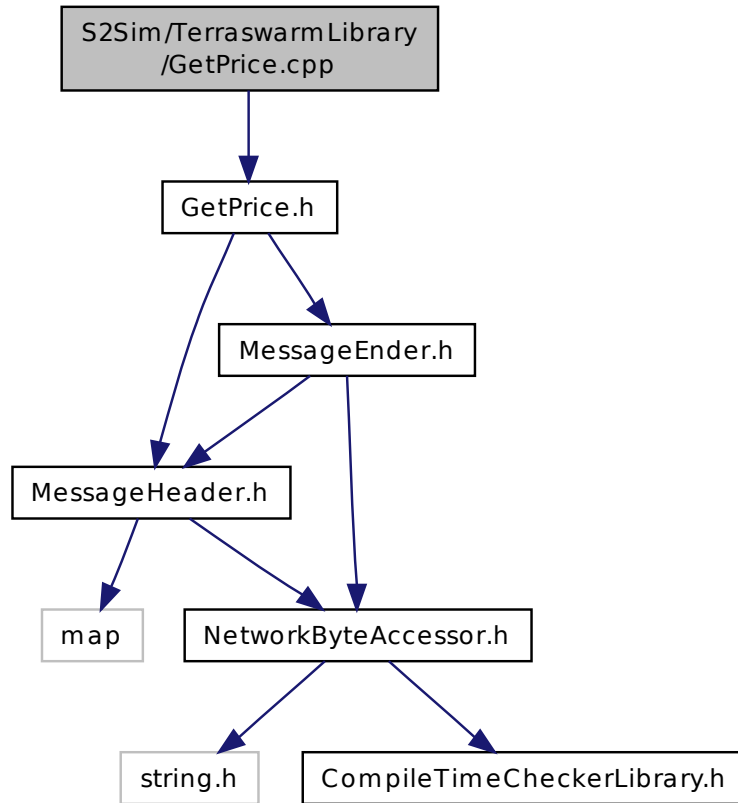
Definition in file [DemandNegotiation.h](#).

9.42 S2Sim/TerraswarmLibrary/GetPrice.cpp File Reference

Implements the GetPrice class.

```
#include "GetPrice.h"
```

Include dependency graph for GetPrice.cpp:



Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.42.1 Detailed Description

Implements the `GetPrice` class.

Date

Oct 31, 2013

Author

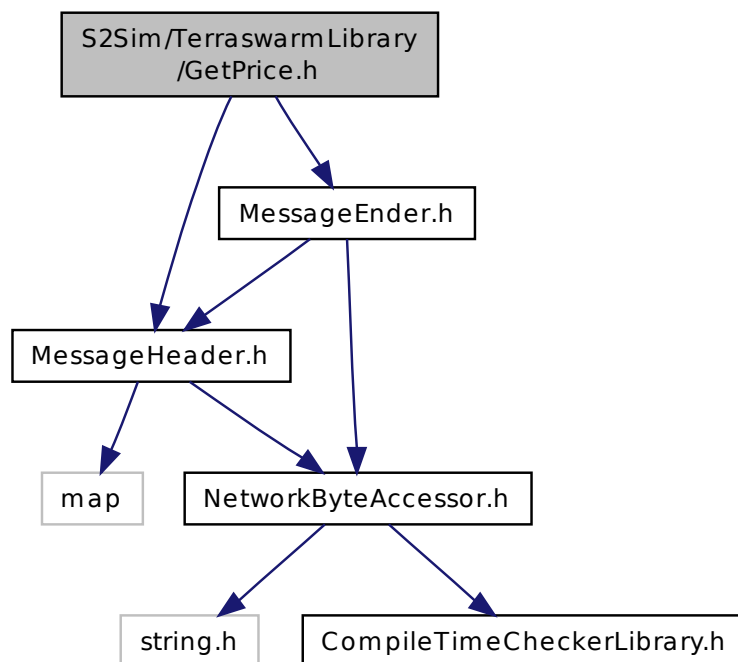
: Alper Sinan Akyurek

Definition in file [GetPrice.cpp](#).

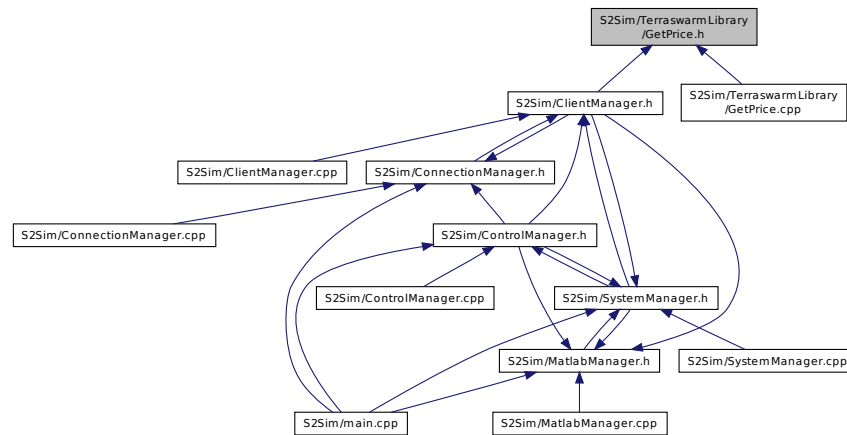
9.43 S2Sim/TerraswarmLibrary/GetPrice.h File Reference

Defines the GetPrice class and message.

```
#include "MessageHeader.h"  
#include "MessageEnder.h"  
Include dependency graph for GetPrice.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::Synchronous::GetPrice](#)
Defines the [GetPrice](#) message sent from the client to the Controller to get the current price value.

Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.43.1 Detailed Description

Defines the GetPrice class and message.

Date

Oct 31, 2013

Author

: Alper Sinan Akyurek

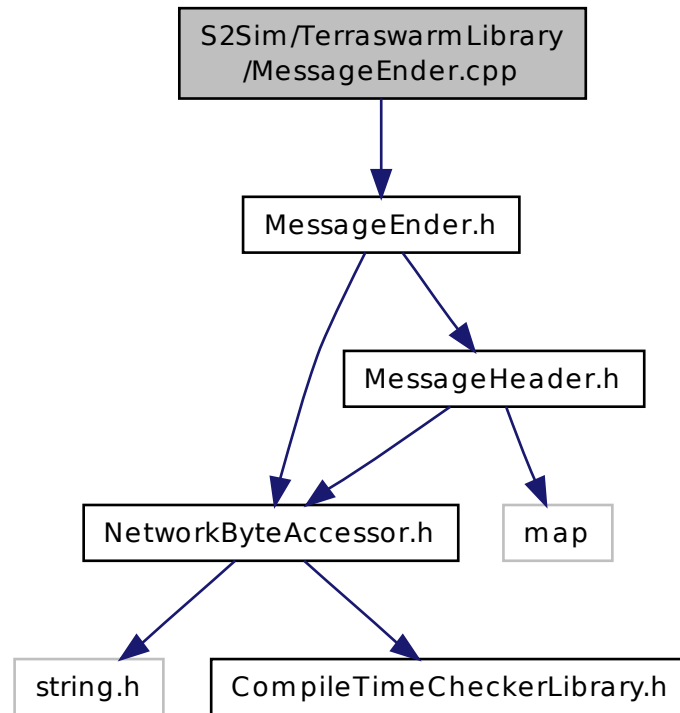
Definition in file [GetPrice.h](#).

9.44 S2Sim/TerraswarmLibrary/MessageEnder.cpp File Reference

Implements a MessageEnder class.

```
#include "MessageEnder.h"
```

Include dependency graph for MessageEnder.cpp:



Namespaces

- [TerraSwarm](#)

[TerraSwarm](#) related classes are defined under this namespace.

9.44.1 Detailed Description

Implements a `MessageEnder` class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

Definition in file [MessageEnder.cpp](#).

Namespaces

- [TerraSwarm](#)

[TerraSwarm](#) related classes are defined under this namespace.

9.45.1 Detailed Description

Defines a MessageEnder class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

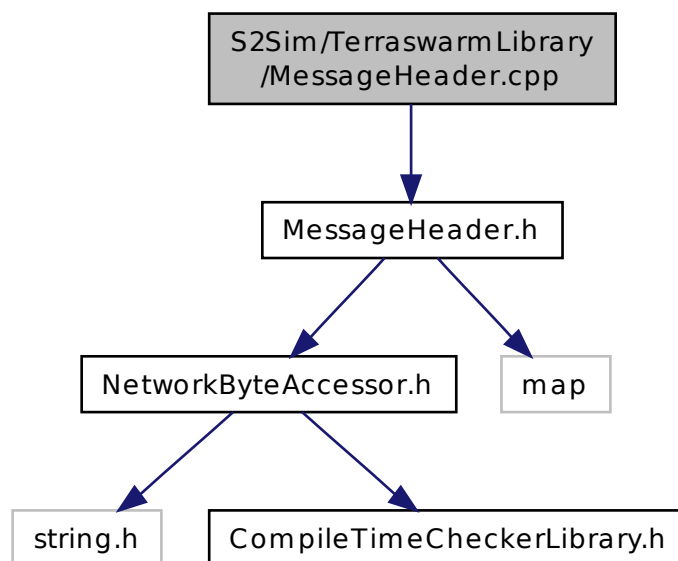
Definition in file [MessageEnder.h](#).

9.46 S2Sim/TerraswarmLibrary/MessageHeader.cpp File Reference

Implements the MessageHeader class.

```
#include "MessageHeader.h"
```

Include dependency graph for MessageHeader.cpp:



Namespaces

- [TerraSwarm](#)

[TerraSwarm](#) related classes are defined under this namespace.

9.46.1 Detailed Description

Implements the MessageHeader class.

Date

Oct 13, 2013

Author

: Alper Sinan Akyurek

Definition in file [MessageHeader.cpp](#).

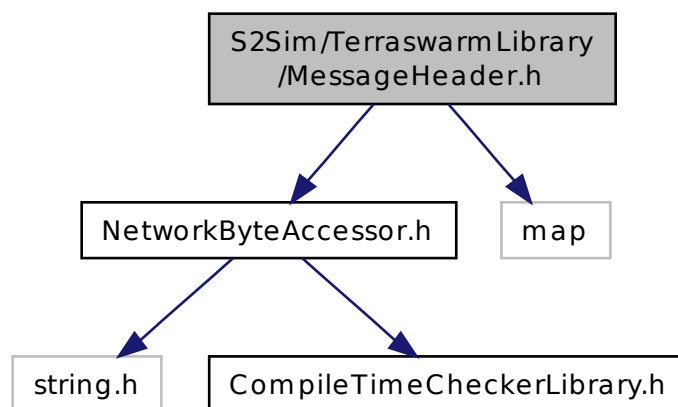
9.47 S2Sim/TerraswarmLibrary/MessageHeader.h File Reference

Defines the MessageHeader class.

```
#include "NetworkByteAccessor.h"
```

```
#include <map>
```

Include dependency graph for MessageHeader.h:



- class TerraSwarm::MessageHeader

Namespaces

- TerraSwarm* related classes are defined under this namespace.

Defines the MessageHeader class.

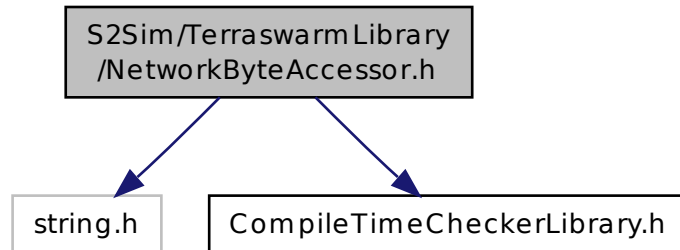
Oct 13, 2013

: Alper Sinan Akyurek

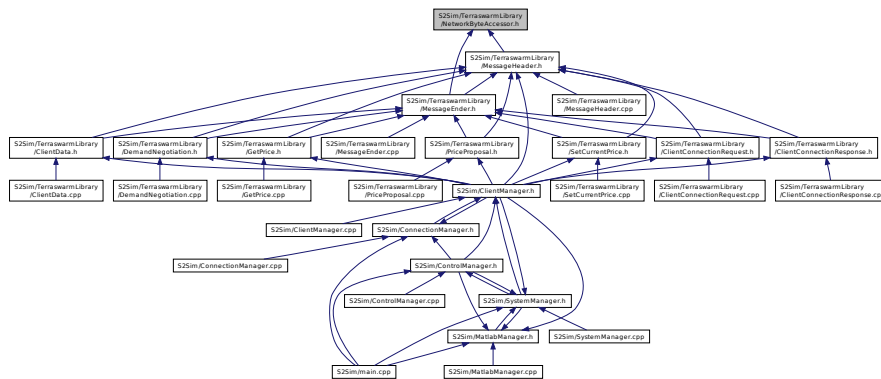
9.48 S2Sim/TerraswarmLibrary/NetworkByteAccessor.h File Reference

```
#include <string.h>
#include "CompileTimeCheckerLibrary.h"
```

Include dependency graph for NetworkByteAccessor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >](#)
Template class to automatically convert byte order and help access ordered bytes in the memory.
- class [TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, size >](#)
Template class that uses the correct conversion function according to the size of the data.
- class [TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 1 >](#)
Template specialization for a type with size 1 (char).
- class [TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 2 >](#)
Template specialization for a type with size 2 (short).
- class [TerraSwarm::NetworkByteAccessor< byteIndex, dataSize >::EndianConverter< TInput, 4 >](#)
Template specialization for a type with size 4 (int).

Namespaces

- [TerraSwarm](#)

TerraSwarm related classes are defined under this namespace.

Typedefs

- typedef unsigned int [TerraSwarm::TByteIndex](#)

Index of a byte in memory.

- typedef unsigned int [TerraSwarm::TDataSize](#)

Size of a data in memory.

9.48.1 Detailed Description

Defines the NetworkByteAccessor class for byte order conversion and easier data access.

Date

Oct 8, 2013

Author

: Alper Sinan Akyurek

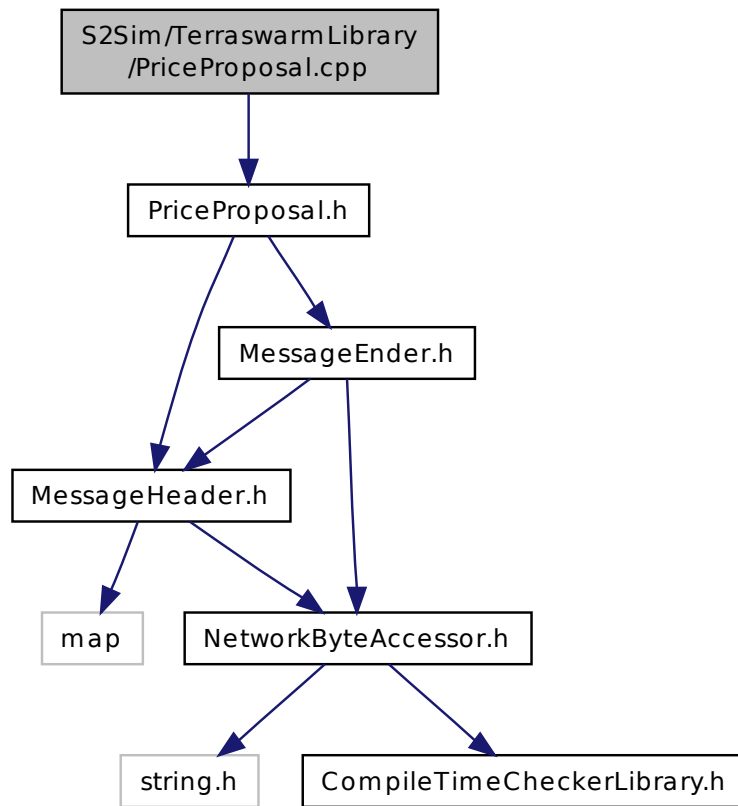
Definition in file [NetworkByteAccessor.h](#).

9.49 S2Sim/TerraswarmLibrary/PriceProposal.cpp File Reference

Implements the PriceProposal class.

```
#include "PriceProposal.h"
```

Include dependency graph for PriceProposal.cpp:



Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.49.1 Detailed Description

Implements the `PriceProposal` class.

Date

Oct 31, 2013

Author

: Alper Sinan Akyurek

Definition in file [PriceProposal.cpp](#).

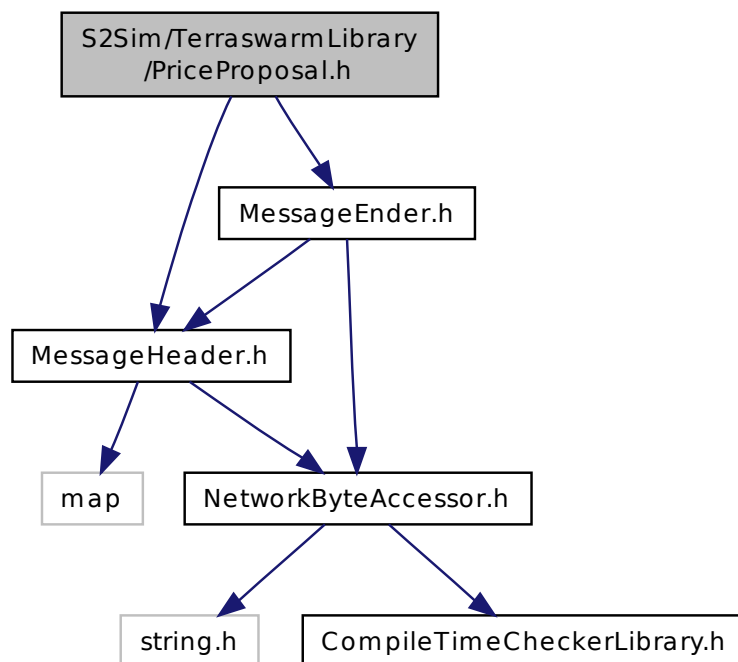
9.50 S2Sim/TerraswarmLibrary/PriceProposal.h File Reference

Defines the PriceProposal class.

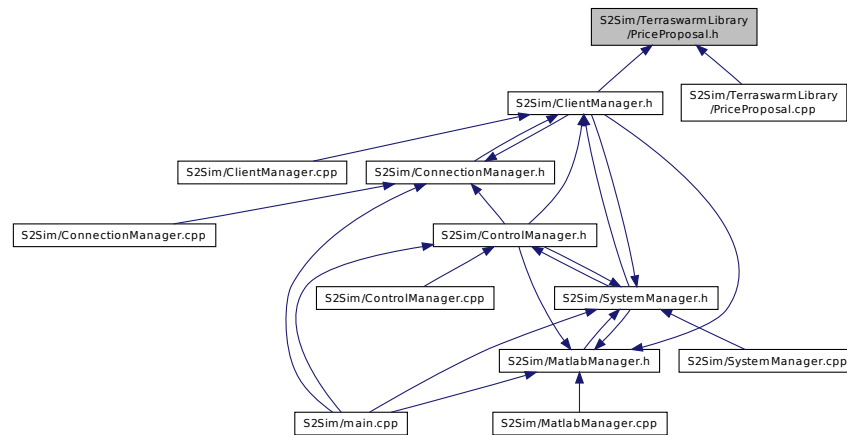
```
#include "MessageHeader.h"
```

```
#include "MessageEnder.h"
```

Include dependency graph for PriceProposal.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::Synchronous::PriceProposal](#)
Price Proposal message sent by the controller to the clients to propose a price.

Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.50.1 Detailed Description

Defines the PriceProposal class.

Date

Oct 31, 2013

Author

: Alper Sinan Akyurek

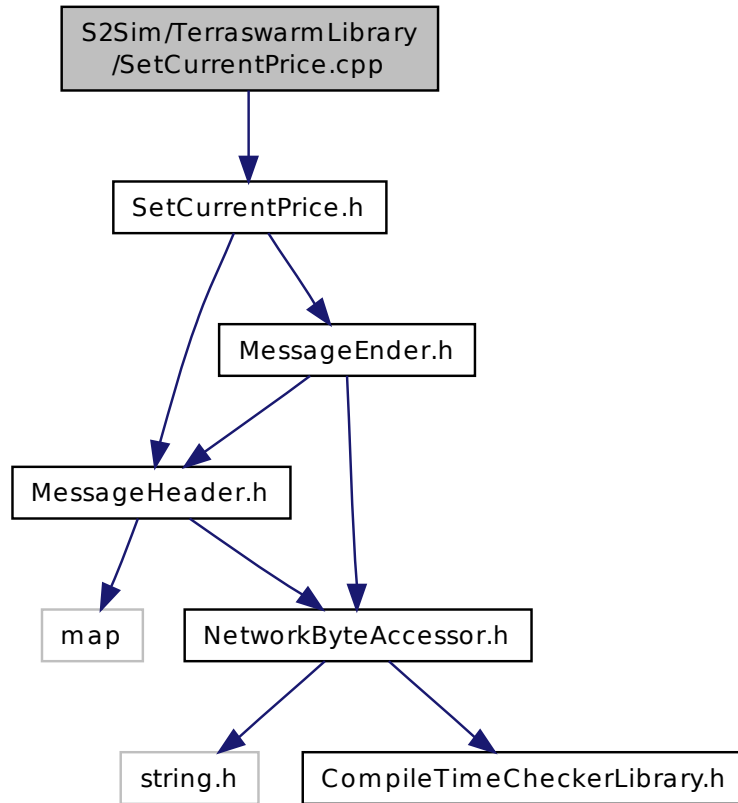
Definition in file [PriceProposal.h](#).

9.51 S2Sim/TerraswarmLibrary/SetCurrentPrice.cpp File Reference

Implements the SetCurrentPrice class.


```
#include "SetCurrentPrice.h"
```

Include dependency graph for SetCurrentPrice.cpp:



Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.51.1 Detailed Description

Implements the `SetCurrentPrice` class.

Date

Oct 31, 2013

Author

: Alper Sinan Akyurek

Definition in file [SetCurrentPrice.cpp](#).

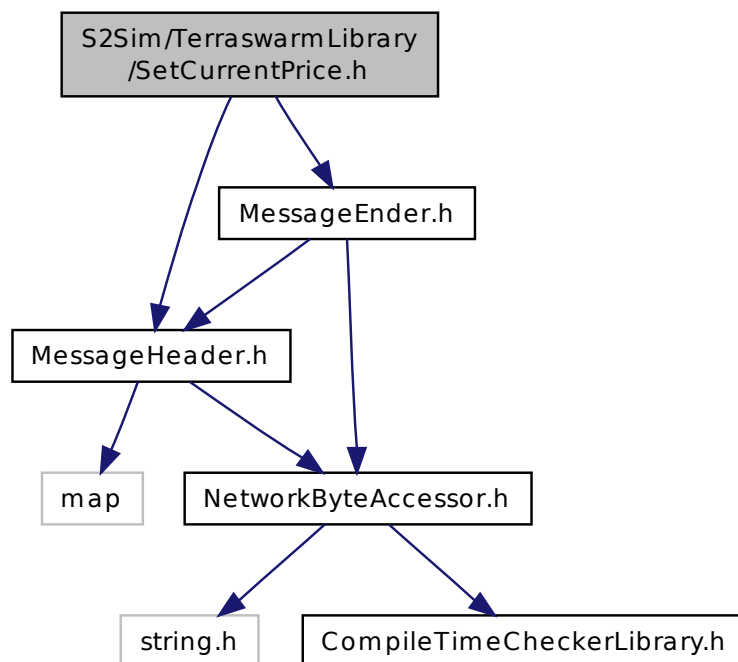
9.52 S2Sim/TerraswarmLibrary/SetCurrentPrice.h File Reference

Defines the SetCurrentPrice class.

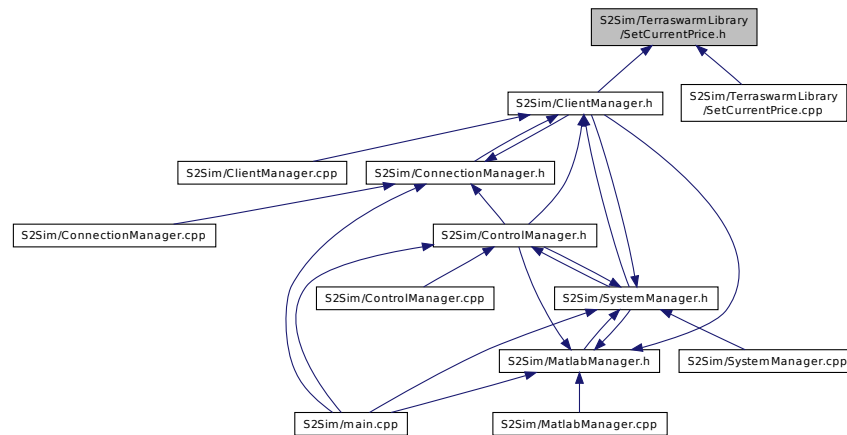
```
#include "MessageHeader.h"
```

```
#include "MessageEnder.h"
```

Include dependency graph for SetCurrentPrice.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [TerraSwarm::Synchronous::SetCurrentPrice](#)
Set Current Price message sent for the Controller to the clients to set the current price and advance the time frame.

Namespaces

- [TerraSwarm](#)
TerraSwarm related classes are defined under this namespace.
- [TerraSwarm::Synchronous](#)
Synchronous client message are defined under this namespace.

9.52.1 Detailed Description

Defines the SetCurrentPrice class.

Date

Oct 31, 2013

Author

: Alper Sinan Akyurek

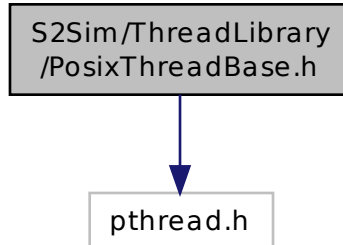
Definition in file [SetCurrentPrice.h](#).

9.53 S2Sim/ThreadLibrary/PosixThreadBase.h File Reference

Defines the [ThreadBase](#) class.

```
#include <pthread.h>
```

Include dependency graph for PosixThreadBase.h:



Classes

- class [ThreadBase](#)
Provides a base class that can be inherited from to gain threading capabilities.
- struct [ThreadBase::InputStructure](#)
Special Input structure sent to the wrapper function: [PosixThreadCover\(\)](#).

Functions

- void * [PosixThreadCover](#) (void *)
Wrapper function called by the operating system as the thread body.

9.53.1 Detailed Description

Defines the [ThreadBase](#) class.

Date

Dec 21, 2013 : Alper Sinan Akyurek

Definition in file [PosixThreadBase.h](#).

9.53.2 Function Documentation

9.53.2.1 void* PosixThreadCover (void *)

Wrapper function called by the operating system as the thread body.

Friend wrapper function that is actually called by the OS as the thread body.

It takes the actual class pointer and the input into the execution function from its input and executes the "actual thread body" with the desired input. This allows us to use the OS thread utilities to run a C++ class method, rather than a plain C function.

Parameters

<i>void*</i>	Special input structure containing the class address and the real input to the execution body.
--------------	--

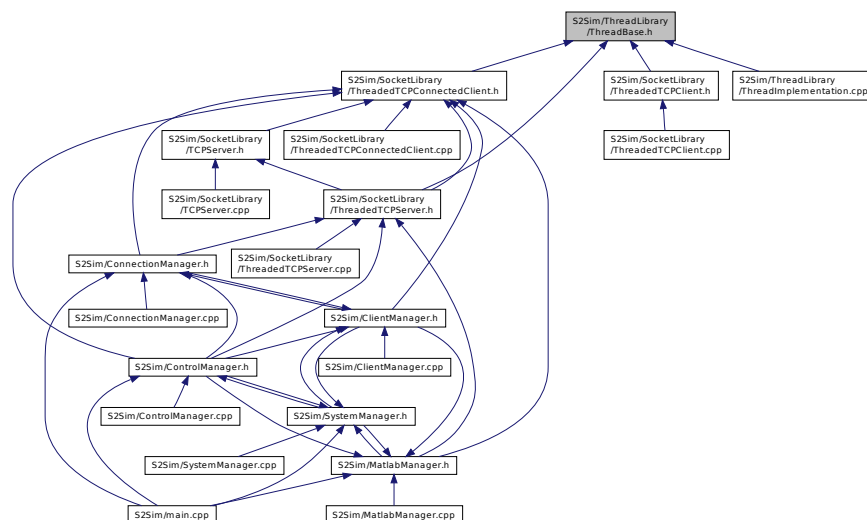
Returns

Returns whatever the class method returns.

9.54 S2Sim/ThreadLibrary/ThreadBase.h File Reference

This file selects the right implementation according to the current OS.

This graph shows which files directly or indirectly include this file:



9.54.1 Detailed Description

This file selects the right implementation according to the current OS.

Date

Dec 21, 2013

Author

: Alper Sinan Akyurek

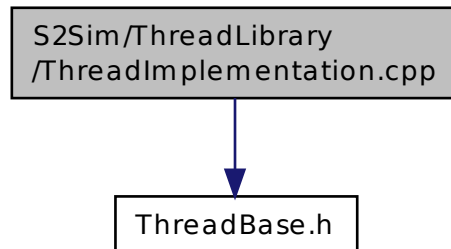
Definition in file [ThreadBase.h](#).

9.55 S2Sim/ThreadLibrary/ThreadImplementation.cpp File Reference

Implements the Thread Wrapper functions.

```
#include "ThreadBase.h"
```

Include dependency graph for ThreadImplementation.cpp:



9.55.1 Detailed Description

Implements the Thread Wrapper functions.

Date

Dec 21, 2013

Author

: Alper Sinan Akyurek

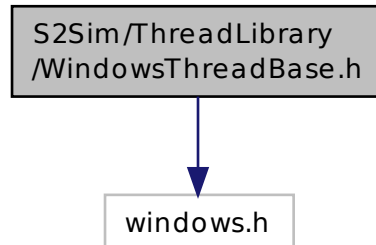
Definition in file [ThreadImplementation.cpp](#).

9.56 S2Sim/ThreadLibrary/WindowsThreadBase.h File Reference

Definition of the [ThreadBase](#) class under Windows implementation.

```
#include <windows.h>
```

Include dependency graph for WindowsThreadBase.h:



Classes

- class [ThreadBase](#)
Provides a base class that can be inherited from to gain threading capabilities.
- struct [ThreadBase::InputStructure](#)
Special Input structure sent to the wrapper function: [PosixThreadCover\(\)](#).

Functions

- DWORD WINAPI [WindowsThreadCover](#) (LPVOID)
Wrapper function called by the operating system as the thread body.

9.56.1 Detailed Description

Definition of the [ThreadBase](#) class under Windows implementation.

Date

Dec 21, 2013

Author

: Alper Sinan Akyurek

Definition in file [WindowsThreadBase.h](#).

9.56.2 Function Documentation

9.56.2.1 DWORD WINAPI WindowsThreadCover (LPVOID)

Wrapper function called by the operating system as the thread body.

Friend wrapper function that is actually called by the OS as the thread body.

It takes the actual class pointer and the input into the execution function from its input and executes the "actual thread body" with the desired input. This allows us to use the OS thread utilities to run a C++ class method, rather than a plain C function.

Parameters

<i>void*</i>	Special input structure containing the class address and the real input to the execution body.
--------------	--

Returns

Returns whatever the class method returns.