System.Net.Sockets Library Reference

September 24, 2014

Contents

Description

Provides support for TCP sockets.

Copyrights

Namespaces

Namespace	Description
Global	Interface to C runtime library that provides the sockets implementation for the platform.
System.Net.Sockets	Provides support for TCP sockets.

1 Usage

1.0.1 Referencing the Sockets Library

Right-click a project node in IDE | Project References... | Add System Extension Library Reference... | enable *System.Net.Sockets* check box

or add following line to your project's .cmp file:

```
reference <ext/System.Net.Sockets/System.Net.Sockets.cml>;
```

2 Global Namespace

Interface to C runtime library that provides the sockets implementation for the platform.

2.1 Functions

Function	Description
accept_socket(int)	Accepts a connection to a TCP socket.
begin_connect()	Preliminary operation for connect_socket(const char*, const char*, int*, int*) operation. Locks a mutex, because getting error after connect is not thread-safe.
begin_get_socket_error_str()	Preliminary operation for get_socket_error_str(int) operation. Locks a mutex, because retrieving socket error string is not thread-safe.
bind_socket(int, int)	Binds a socket to a port.
close_socket(int)	Closes a socket.
connect_socket(const char*, const char*, int*, int*)	Creates a new TCP connection.
create_tcp_socket()	Creates a TCP socket.
done_sockets()	Uninitializes the socket library.
end_connect()	Closing operation for connect_socket(const char*, const char*, int*, int*) operation. Unlocks a mutex.
end_get_socket_error_str()	Closing operation for get_socket_error_str(int) operation. Unlocks a mutex.
get_addrinfo_error(int)	Returns error description when getaddrinfo call has failed. Not thread-safe.
get_last_socket_error()	Returns the error code of the latest failed socket operation.
get_socket_error_str(int)	Returns an error description of the failed socket operation. Not thread-safe.

init_sockets()	Initializes the socket library.
listen_socket(int, int)	Begins listening the port of a bound socket.
receive_socket(int, void*, int, int)	Receives data from a connected socket.
send_socket(int, void*, int, int)	Sends data to a connected socket.
shutdown_socket(int, ShutdownMode)	Shuts down receiving from a socket, sending to a socket or both.

2.1.1 accept_socket(int) Function

Accepts a connection to a TCP socket.

Syntax

```
public cdecl int accept_socket(int socket);
```

Parameters

Name	Type	Description
socket	int	The handle of the socket where to accept a connection.

Returns

int

Returns the handle of a connected socket (positive integer) if the call succeeds, or -1 otherwise.

2.1.2 begin_connect() Function

Preliminary operation for connect_socket(const char*, const char*, int*, int*) operation. Locks a mutex, because getting error after connect is not thread-safe.

Syntax

```
public cdecl void begin_connect();
```

2.1.3 begin_get_socket_error_str() Function

Preliminary operation for get_socket_error_str(int) operation. Locks a mutex, because retrieving socket error string is not thread-safe.

Syntax

```
public cdecl void begin_get_socket_error_str();
```

2.1.4 bind_socket(int, int) Function

Binds a socket to a port.

Syntax

```
public cdecl int bind_socket(int socket, int port);
```

Parameters

Name	Type	Description
socket	int	The handle of the socket to bind.
port	int	Port number to bind.

Returns

int

Returns 0 if the call succeeds, -1 otherwise.

2.1.5 close_socket(int) Function

Closes a socket.

Syntax

```
public cdecl int close_socket(int socket);
```

Parameters

Name	Type	Description
socket	int	The handle of the socket to close.

Returns

int

Returns 0 if the call succeeds, -1 otherwise.

2.1.6 connect_socket(const char*, const char*, int*, int*) Function

Creates a new TCP connection.

Syntax

```
public cdecl int connect_socket(const char* node, const char* service, int* scktm,
int* getaddrinfofailed);
```

Parameters

Name	Type	Description
node	const char*	The name of the host to connect.
service	const char*	The protocol name or port number to connect.

scktm	int*	Receives the handle of the connected socket.
getaddrinfofailed	int*	Set to 1, if getaddrinfo call failed, 0 otherwise.

Returns

int

Returns 0 if the call succeeds or a nonzero error code otherwise.

2.1.7 create_tcp_socket() Function

Creates a TCP socket.

Syntax

```
public cdecl int create_tcp_socket();
```

Returns

int

Returns the handle of the created socket (positive integer) if the call succeeds, or -1 otherwise.

2.1.8 done_sockets() Function

Uninitializes the socket library.

Syntax

```
public cdecl void done_sockets();
```

2.1.9 end_connect() Function

Closing operation for connect_socket(const char*, const char*, int*, int*) operation. Unlocks a mutex.

Syntax

```
public cdecl void end_connect();
```

2.1.10 end_get_socket_error_str() Function

Closing operation for get_socket_error_str(int) operation. Unlocks a mutex.

Syntax

```
public cdecl void end_get_socket_error_str();
```

2.1.11 get_addrinfo_error(int) Function

Returns error description when getaddrinfo call has failed. Not thread-safe.

Syntax

```
public cdecl const char* get_addrinfo_error(int errorCode);
```

Parameters

Name	Type	Description
errorCode	int	Error code returned by getaddrinfo call.

Returns

const char*

Returns error description.

2.1.12 get_last_socket_error() Function

Returns the error code of the latest failed socket operation.

Syntax

```
public cdecl int get_last_socket_error();
```

Returns

int

Returns the error code of the latest failed socket operation.

2.1.13 get_socket_error_str(int) Function

Returns an error description of the failed socket operation. Not thread-safe.

Syntax

```
public cdecl const char* get_socket_error_str(int errorCode);
```

Parameters

Name	Type	Description

errorCode	int	Error code of the failed socket operation.

Returns

const char*

Returns an error description.

2.1.14 init_sockets() Function

Initializes the socket library.

Syntax

```
public cdecl int init_sockets();
```

Returns

int

Returns 0 if the call succeeds, or -1 otherwise.

2.1.15 listen_socket(int, int) Function

Begins listening the port of a bound socket.

Syntax

```
public cdecl int listen_socket(int socket, int backlog);
```

Parameters

Name	Type	Description
socket	int	The handle of a bound socket.
backlog	int	Number of pending connection.

Returns

int

Returns 0 if the call succeeds, or -1 otherwise.

2.1.16 receive_socket(int, void*, int, int) Function

Receives data from a connected socket.

Syntax

```
public cdecl int receive_socket(int socket, void* buf, int len, int flags);
```

Parameters

Name	Type	Description
socket	int	The handle of a connected socket.
buf	void*	A buffer for the data.
len	int	Maximum number of bytes to receive.
flags	int	Options for the operation.

Returns

int

Returns the number of bytes received if the call succeeds, or -1 otherwise. The number of bytes received might be less than the number of bytes requested.

2.1.17 send_socket(int, void*, int, int) Function

Sends data to a connected socket.

Syntax

```
public cdecl int send_socket(int socket, void* buf, int len, int flags);
```

Parameters

Name	Type	Description
socket	int	The handle of a connected socket.
buf	void*	A buffer of data.
len	int	Maximum number of bytes to send.
flags	int	Options for the operation.

Returns

int

Returns the number of bytes sent if the call succeeds, or -1 otherwise. The number of bytes sent might be less than the number of bytes requested.

2.1.18 shutdown_socket(int, ShutdownMode) Function

Shuts down receiving from a socket, sending to a socket or both.

Syntax

```
public cdecl int shutdown_socket(int socket, ShutdownMode mode);
```

Parameters

Name	Type	Description

socket	int	The handle of the socket to shut down.
mode	ShutdownMode	Mode for shut down operation.

Returns

int

Returns 0 if the call succeeds, or -1 otherwise.

2.2 Enumerations

Enumeration	**Description**
ShutdownMode	Mode for the shut down operation.

2.2.18.1 ShutdownMode Enumeration

Mode for the shut down operation.

Enumeration Constants

Constant	Value	Description
receive	0	Shuts down receiving from a socket.
send	1	Shuts down sending to a socket.
both	2	Shuts down both receiving and sending.

3 System.Net.Sockets Namespace

Provides support for TCP sockets.

3.3 Classes

Class	Description
NetworkBuffer	A handle to a dynamically allocated memory.
SocketError	An exception class throw when a socket operation fails.
SocketLibrary	Represents the socket library initializer implemented as a singleton.
SocketLibraryException	Exception class thrown when the initialization of the socket library fails.
TcpSocket	Represents a TCP socket.

3.3.1 NetworkBuffer Class

A handle to a dynamically allocated memory.

Syntax

```
public class NetworkBuffer;
```

3.3.1.1 Member Functions

Member Function	Description
NetworkBuffer(NetworkBuffer&&)	Move constructor.
NetworkBuffer(int)	Constructor. Allocates specified number of bytes from the system.
operator=(NetworkBuffer&&)	Move assignment.
~NetworkBuffer()	Destructor. Free the allocated memory back to the system.
Mem() const	Returns a pointer to the allocated memory block.
Size() const	Returns the size of the allocated memory block.

3.3.1.1.1 NetworkBuffer(NetworkBuffer&&) Member Function

Move constructor.

Syntax

```
public nothrow NetworkBuffer(NetworkBuffer&& that);
```

Parameters

Name	Type	Description
that	NetworkBuffer&&	A network buffer to move.

3.3.1.1.2 NetworkBuffer(int) Member Function

Constructor. Allocates specified number of bytes from the system.

Syntax

```
public nothrow NetworkBuffer(int size_);
```

Parameters

Name	Type	Description
size_	int	The number of bytes to allocate.

3.3.1.1.3 operator=(NetworkBuffer&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(NetworkBuffer&& that);
```

Parameters

Name	Type	Description
that	NetworkBuffer&&	A network buffer to move.

3.3.1.1.4 ~NetworkBuffer() Member Function

Destructor. Free the allocated memory back to the system.

Syntax

```
public nothrow ~NetworkBuffer();
```

3.3.1.1.5 Mem() const Member Function

Returns a pointer to the allocated memory block.

Syntax

```
public inline nothrow void* Mem() const;
```

Returns

void*

Returns a pointer to the allocated memory block.

3.3.1.1.6 Size() const Member Function

Returns the size of the allocated memory block.

Syntax

```
public inline nothrow int Size() const;
```

Returns

int

Returns the size of the allocated memory block.

3.3.2 SocketError Class

An exception class throw when a socket operation fails.

Syntax

```
public class SocketError;
```

Base Class

Exception

3.3.2.1 Member Functions

Member Function	Description
SocketError(const SocketError&)	Copy constructor.
SocketError(SocketError&&)	Move constructor.
SocketError(const String&, const String&, int)	Constructor. Initializes the socket error with the specified operation text, error description text and error code.
SocketError(const String&, int)	Constructor. Initializes the socket error with the specified operation text, retrieved error description and the specified error code.
operator=(SocketError&&)	Move assignment.
operator=(const SocketError&)	Copy assignment.
~SocketError()	Destructor.
ErrorCode() const	Returns the error code.

3.3.2.1.1 SocketError(const SocketError&) Member Function

Copy constructor.

Syntax

```
public nothrow SocketError(const SocketError& that);
```

Parameters

Name	Type	Description

that	const SocketError&	A socket error to copy.

3.3.2.1.2 SocketError(SocketError&&) Member Function

Move constructor.

Syntax

```
public nothrow SocketError(SocketError&& that);
```

Parameters

Name	Type	Description
that	SocketError&&	A socket error to move.

3.3.2.1.3 SocketError(const String&, const String&, int) Member Function

Constructor. Initializes the socket error with the specified operation text, error description text and error code.

Syntax

```
public SocketError(const String& operation, const String& errorMessage, int errorCode_);
```

Parameters

Name	Type	Description
operation	const String&	Description of the failed operation.
errorMessage	const String&	Description of the error.
errorCode_	int	Error code.

3.3.2.1.4 SocketError(const String&, int) Member Function

Constructor. Initializes the socket error with the specified operation text, retrieved error description and the specified error code.

Syntax

```
public SocketError(const String& operation, int errorCode_);
```

Parameters

Name	Type	Description
operation	const String&	Description of failed operation.
errorCode_	int	Error code.

3.3.2.1.5 operator=(SocketError&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(SocketError&& that);
```

Parameters

Name	Type	Description
that	SocketError&&	A socket error to move.

3.3.2.1.6 operator=(const SocketError&) Member Function

Copy assignment.

Syntax

```
public nothrow void operator=(const SocketError& that);
```

Parameters

Name	Type	Description
that	const SocketError&	A socket error to assign.

3.3.2.1.7 ~SocketError() Member Function

Destructor.

Syntax

```
public override nothrow ~SocketError();
```

3.3.2.1.8 ErrorCode() const Member Function

Returns the error code.

Syntax

```
public nothrow int ErrorCode() const;
```

Returns

int

Returns the error code.

3.3.3 SocketLibrary Class

Represents the socket library initializer implemented as a singleton.

Syntax

```
public class SocketLibrary;
```

3.3.3.1 Member Functions

Member Function	Description
~SocketLibrary()	Uninitializes the socket library.
Init()	Initializes the socket library.
Instance()	Returns a reference to the socket library singleton instance.

3.3.3.1.1 ~SocketLibrary() Member Function

Uninitializes the socket library.

Syntax

```
public nothrow ~SocketLibrary();
```

3.3.3.1.2 Init() Member Function

Initializes the socket library.

Syntax

```
public void Init();
```

3.3.3.1.3 Instance() Member Function

Returns a reference to the socket library singleton instance.

Syntax

```
public static nothrow SocketLibrary& Instance();
```

Returns

SocketLibrary&

Returns a reference to the socket library singleton instance.

3.3.4 SocketLibraryException Class

Exception class thrown when the initialization of the socket library fails.

Syntax

```
public class SocketLibraryException;
```

Base Class

Exception

3.3.4.1 Member Functions

Member Function	Description
SocketLibraryException(SocketLibraryException&&)	Move constructor.
SocketLibraryException(const String&)	Constructor. Initializes the socket library exception with the specified error message.
SocketLibraryException(const SocketLibraryException&)	Copy constructor.
operator=(SocketLibraryException&&)	Move assignment.
operator=(const SocketLibraryException&)	Copy assignment.
~SocketLibraryException()	Destructor.

3.3.4.1.1 SocketLibraryException(SocketLibraryException&&) Member Function

Move constructor.

Syntax

```
public nothrow SocketLibraryException(SocketLibraryException&& that);
```

Parameters

Name	Type	Description
that	SocketLibraryException&&	A socket library exception to move.

3.3.4.1.2 SocketLibraryException(const String&) Member Function

Constructor. Initializes the socket library exception with the specified error message.

Syntax

```
public SocketLibraryException(const String& message_);
```

Parameters

Name	Type	Description
message_	const String&	An error message.

3.3.4.1.3 SocketLibraryException(const SocketLibraryException&) Member Function

Copy constructor.

Syntax

```
public nothrow SocketLibraryException(const SocketLibraryException& that);
```

Parameters

Name	Type	Description
that	const SocketLibraryException&	A socket library exception to copy.

3.3.4.1.4 operator=(SocketLibraryException&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(SocketLibraryException&& that);
```

Parameters

Name	Type	Description
that	SocketLibraryException&&	A socket library exception to move.

3.3.4.1.5 operator=(const SocketLibraryException&) Member Function

Copy assignment.

Syntax

```
public nothrow void operator=(const SocketLibraryException& that);
```

Parameters

Name	**Type**	**Description**
that	const SocketLibraryException&	A socket library exception to assign.

3.3.4.1.6 ∼SocketLibraryException() Member Function

Destructor.

Syntax

```
public override nothrow ~SocketLibraryException();
```

3.3.5 TcpSocket Class

Represents a TCP socket.

Syntax

```
public class TcpSocket;
```

3.3.5.1 Member Functions

Member Function	Description
TcpSocket()	Default constructor. Creates an unbound TCP socket.
TcpSocket(TcpSocket&&)	Move constructor.
TcpSocket(const String&, const String&)	Constructor. Creates a TCP socket and connects it to the specified node and service.
TcpSocket(int)	Constructor. Initializes a TCP socket with an existing socket handle.
operator=(TcpSocket&&)	Move assignment.
~TcpSocket()	Destructor. Closes the socket if it is bound or connected.
Accept()	Accepts a connection to a bound socket and returns a new connected TCP socket that represents the connection.
Bind(int)	Binds the socket to a port.
Close()	Closes the socket.
GetSocketHandle() const	Returns the socket handle.
Listen(int)	Begins listening connections to a bound TCP socket.
Receive(void*, int)	Receives data from a connected socket.
ReceiveAll()	Receives rest of data from a connected socket. That is: receives data until the peer shuts down its sending side of the connection.
Send(const String&)	Sends a string of data to a connected socket.
Send(void*, int)	Sends data to a connected socket.
Shutdown(ShutdownMode)	Shuts down a connected socket.

3.3.5.1.1 TcpSocket() Member Function

Default constructor. Creates an unbound TCP socket.

Syntax

```
public TcpSocket();
```

3.3.5.1.2 TcpSocket(TcpSocket&&) Member Function

Move constructor.

Syntax

```
public nothrow TcpSocket(TcpSocket&& that);
```

Parameters

Name	Type	Description
that	TcpSocket&&	A TCP socket to move.

3.3.5.1.3 TcpSocket(const String&, const String&) Member Function

Constructor. Creates a TCP socket and connects it to the specified node and service.

Syntax

```
public TcpSocket(const String& node, const String& service);
```

Parameters

Name	Type	Description
node	const String&	A host name or an IP address to connect.
service	const String&	A protocol name or port number to connect.

3.3.5.1.4 TcpSocket(int) Member Function

Constructor. Initializes a TCP socket with an existing socket handle.

Syntax

```
public nothrow TcpSocket(int socket_);
```

Parameters

Name	Type	Description
socket_	int	A handle of an existing TCP socket.

3.3.5.1.5 operator=(TcpSocket&&) Member Function

Move assignment.

Syntax

```
public nothrow void operator=(TcpSocket&& that);
```

Parameters

Name	Type	Description
that	TcpSocket&&	A TCP socket to move.

3.3.5.1.6 ~TcpSocket() Member Function

Destructor. Closes the socket if it is bound or connected.

Syntax

```
public nothrow ~TcpSocket();
```

3.3.5.1.7 Accept() Member Function

Accepts a connection to a bound socket and returns a new connected TCP socket that represents the connection.

Syntax

```
public TcpSocket Accept();
```

Returns

TcpSocket

Returns a connected TCP socket that represents the connection.

3.3.5.1.8 Bind(int) Member Function

Binds the socket to a port.

Syntax

```
public void Bind(int port);
```

Parameters

Name	Type	Description

port	int	A port number to which to bind.

3.3.5.1.9 Close() Member Function

Closes the socket.

Syntax

```
public void Close();
```

3.3.5.1.10 GetSocketHandle() const Member Function

Returns the socket handle.

Syntax

```
public inline nothrow int GetSocketHandle() const;
```

Returns

int

Returns the socket handle.

3.3.5.1.11 Listen(int) Member Function

Begins listening connections to a bound TCP socket.

Syntax

```
public void Listen(int backlog);
```

Parameters

Name	Type	Description
backlog	int	The number of pending connections.

3.3.5.1.12 Receive(void*, int) Member Function

Receives data from a connected socket.

Syntax

```
public int Receive(void* buf, int len);
```

Parameters

Name	Type	Description
buf	void*	A buffer.

len	int	Maximum number of bytes to receive.

Returns

int

Returns the number of bytes received. This might be less than the number of bytes requested.

3.3.5.1.13 ReceiveAll() Member Function

Receives rest of data from a connected socket. That is: receives data until the peer shuts down its sending side of the connection.

Syntax

```
public String ReceiveAll();
```

Returns

String

Returns the received data as a string.

3.3.5.1.14 Send(const String&) Member Function

Sends a string of data to a connected socket.

Syntax

```
public void Send(const String& s);
```

Parameters

Name	Type	Description
s	const String&	A string to send.

3.3.5.1.15 Send(void*, int) Member Function

Sends data to a connected socket.

Syntax

```
public int Send(void* buf, int len);
```

Parameters

Name	Type	Description
buf	void*	A buffer of data to send.
len	int	Maximum number of bytes to send.

Returns

int

Returns the number of bytes sent. This might be less than the number of bytes requested.

3.3.5.1.16 Shutdown(ShutdownMode) Member Function

Shuts down a connected socket.

Syntax

```
public void Shutdown(ShutdownMode mode);
```

Parameters

Name	Type	Description
mode	ShutdownMode	Shut down mode.

3.4 Constants

Constant	Type	Value	Description
invalidSocketHandle	int	-1	Represents invalid socket handle.