Table of Contents

Namespace SuperNet.Compress

Classes

CompressorDeflate

Compression based on the DEFLATE algorithm.

CompressorLZF

Compression based on the LZF algorithm.

Interfaces

ICompressor

Defines methods for compressing and decompressing network packets.

Class CompressorDeflate

Compression based on the DEFLATE algorithm.

Inheritance

System.Object
CompressorDeflate

Implements

ICompressor
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Compress

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class CompressorDeflate : ICompressor, IDisposable
```

Constructors

CompressorDeflate(Allocator)

Create a new DEFLATE compressor.

Declaration

```
public CompressorDeflate(Allocator allocator)
```

Parameters

TYPE	NAME	DESCRIPTION
Allocator	allocator	Allocator to use for resizing buffers.

Methods

Compress(ArraySegment<Byte>, Byte[], Int32)

Compress data.

Declaration

```
public int Compress(ArraySegment<byte> input, byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to compress.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

Decompress(ArraySegment<Byte>, ref Byte[], Int32)

Decompress data and resize output if needed.

Declaration

```
public int Decompress(ArraySegment<byte> input, ref byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to decompress.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

Dispose()

Instantly dispose of all resources.

Declaration

```
public void Dispose()
```

MaxCompressedLength(Int32)

Compute the maximum compressed length before compressing.

Compute the maximum compressed length before compressing.

Declaration

```
public int MaxCompressedLength(int inputLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputLength	Length of the uncompressed input.

Returns

TYPE	DESCRIPTION
System.Int32	Maximum possible compressed length.

Implements

ICompressor
System.IDisposable

Class CompressorLZF

Compression based on the LZF algorithm.

Inheritance

System.Object
CompressorLZF

Implements

ICompressor
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Compress

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class CompressorLZF : ICompressor, IDisposable
```

Constructors

CompressorLZF(Allocator)

Create a new LZF compressor.

Declaration

```
public CompressorLZF(Allocator allocator)
```

Parameters

TYPE	NAME	DESCRIPTION
Allocator	allocator	Allocator to use for resizing buffers.

Methods

Compress(ArraySegment<Byte>, Byte[], Int32)

Compress data.

Declaration

```
public int Compress(ArraySegment<byte> input, byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to compress.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

Decompress(ArraySegment<Byte>, ref Byte[], Int32)

Decompress data and resize output if needed.

Declaration

```
public int Decompress(ArraySegment<byte> input, ref byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to decompress.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

Dispose()

Instantly dispose of all resources.

Declaration

```
public void Dispose()
```

MaxCompressedLength(Int32)

Compute the maximum compressed length before compressing.

Compute the maximum compressed length before compressing.

Declaration

```
public int MaxCompressedLength(int inputLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputLength	Length of the uncompressed input.

Returns

TYPE	DESCRIPTION
System.Int32	Maximum possible compressed length.

Implements

ICompressor
System.IDisposable

Interface ICompressor

Defines methods for compressing and decompressing network packets.

Inherited Members

System.IDisposable.Dispose()

Namespace: SuperNet.Compress

Assembly: cs.temp.dll.dll

Syntax

```
public interface ICompressor : IDisposable
```

Methods

Compress(ArraySegment<Byte>, Byte[], Int32)

Compress data.

Declaration

```
int Compress(ArraySegment<byte> input, byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to compress.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

Decompress(ArraySegment<Byte>, ref Byte[], Int32)

Decompress data and resize output if needed.

Declaration

```
int Decompress(ArraySegment<byte> input, ref byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to decompress.

TYPE	NAME	DESCRIPTION
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

MaxCompressedLength(Int32)

Compute the maximum compressed length before compressing.

Declaration

```
int MaxCompressedLength(int inputLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputLength	Length of the uncompressed input.

Returns

TYPE	DESCRIPTION
System.Int32	Maximum possible compressed length.

Namespace SuperNet.Crypto

Classes

CryptoAES

Encryptor based on 256-bit Advanced Encryption Standard (AES).

CryptoECDH

Implements Elliptic Curve Diffie Hellman key exchange.

CryptoRandom

Cryptographically secure random number generator.

CryptoRSA

Authenticator based on 2048-bit RSA (Rivest–Shamir–Adleman).

Curve25519

Elliptic Curve methods used for Diffie Hellman key exchange.

Interfaces

ICryptoAuthenticator

Defines methods used for authenticating secure hosts.

ICryptoEncryptor

Defines methods for encrypting and decrypting network packets.

ICryptoExchanger

Defines methods used for a key exchange that is able to derive a shared encryptor.

ICryptoRandom

Defines methods for generating random data.

Class CryptoAES

Encryptor based on 256-bit Advanced Encryption Standard (AES).

Inheritance

System.Object
CryptoAES

Implements

ICryptoEncryptor
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class CryptoAES : ICryptoEncryptor, IDisposable
```

Constructors

CryptoAES(Byte[], Allocator)

Create a new AES encryptor with the provided key.

Declaration

```
public CryptoAES(byte[] key, Allocator allocator)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	key	Encryption key to use.
Allocator	allocator	Allocator to use for allocating keys.

Methods

Decrypt(ArraySegment<Byte>, Byte[], Int32)

Decrypt data.

Declaration

```
public int Decrypt(ArraySegment<byte> input, byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to decrypt.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

Dispose()

Instantly dispose of all resources.

Declaration

```
public void Dispose()
```

Encrypt(ArraySegment<Byte>, Byte[], Int32)

Encrypt data.

Declaration

```
public int Encrypt(ArraySegment<byte> input, byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to encrypt.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

MaxDecryptedLength(Int32)

Compute the maximum decrypted length before decrypting

Compute the maximum decrypted length before decrypting.

Declaration

```
public int MaxDecryptedLength(int inputLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputLength	Length of the input that is about to be decrypted.

Returns

TYPE	DESCRIPTION
System.Int32	Maximum possible decrypted length.

MaxEncryptedLength(Int32)

Compute the maximum encrypted length before encrypting.

Declaration

```
public int MaxEncryptedLength(int inputLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputLength	Length of the input that is about to be encrypted.

Returns

TYPE	DESCRIPTION
System.Int32	Maximum possible encrypted length.

Implements

ICryptoEncryptor
System.IDisposable

Class CryptoECDH

Implements Elliptic Curve Diffie Hellman key exchange.

Inheritance

System.Object
CryptoECDH

Implements

ICryptoExchanger
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class CryptoECDH : ICryptoExchanger, IDisposable
```

Constructors

CryptoECDH(ICryptoRandom, Allocator)

Create a new ECDH key pair for key exchange.

Declaration

```
public CryptoECDH(ICryptoRandom random, Allocator allocator)
```

Parameters

TYPE	NAME	DESCRIPTION
ICryptoRandom	random	Random number generator to use for generating keys.
Allocator	allocator	Allocator to use for allocating keys.

Fields

KeyLength

Size of exchange key in bytes. For ECDH this is 32.

Declaration

```
public const int KeyLength = 32
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Methods

DeriveEncryptor(ArraySegment<Byte>)

Generate a shared encryptor.

Declaration

```
public ICryptoEncryptor DeriveEncryptor(ArraySegment<byte> remoteKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	remoteKey	Received remote exchange key.

Returns

TYPE	DESCRIPTION
ICryptoEncryptor	Shared encryptor that is guaranteed to be the same on both peers.

Dispose()

Returns key pair back to the allocator.

Declaration

```
public void Dispose()
```

ExportKey(ArraySegment<Byte>)

Copy exchange key to the output.

Declaration

```
public void ExportKey(ArraySegment<byte> output)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	output	Output to write to.

Explicit Interface Implementations

ICryptoExchanger.KeyLength

Size of exchange key in bytes. For ECDH this is 32.

Declaration

```
int ICryptoExchanger.KeyLength { get; }
```

Returns

TYPE	DESCRIPTION
System.Int32	

Implements

ICryptoExchanger
System.IDisposable

Class CryptoRandom

Cryptographically secure random number generator.

Inheritance

System.Object
CryptoRandom

Implements

ICryptoRandom
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public class CryptoRandom : ICryptoRandom, IDisposable
```

Constructors

CryptoRandom()

Create random number generator.

Declaration

```
public CryptoRandom()
```

Methods

Dispose()

Instantly dispose of all resources.

Declaration

```
public void Dispose()
```

GetBytes(Byte[], Int32, Int32)

Generate cryptographically secure random data.

This method is thread safe.

Declaration

```
public void GetBytes(byte[] output, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.
System.Int32	count	Number of bytes to write.

Implements

ICryptoRandom
System.IDisposable

Class CryptoRSA

Authenticator based on 2048-bit RSA (Rivest–Shamir–Adleman).

Inheritance

System.Object
CryptoRSA

Implements

ICryptoAuthenticator
System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class CryptoRSA : ICryptoAuthenticator, IDisposable
```

Constructors

CryptoRSA(Allocator)

Create a new RSA authenticator.

Declaration

```
public CryptoRSA(Allocator allocator = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Allocator	allocator	Allocator to use for keys or null for none.

Properties

SignatureLength

Number of bytes in the signature. For RSA this is 256.

Declaration

```
public int SignatureLength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Dispose()

Instantly dispose of all resources.

Declaration

```
public void Dispose()
```

ExportPrivateKey()

Export all RSA parameters as a Base64 string.

Declaration

```
public string ExportPrivateKey()
```

Returns

TYPE	DESCRIPTION
System.String	Base64 encoded parameters.

ExportPublicKey()

Export RSA modulus as a Base64 string.

Declaration

```
public string ExportPublicKey()
```

Returns

TYPE	DESCRIPTION
System.String	Base64 encoded RSA modulus.

ImportPrivateKey(String)

Import previously exported private key.

Declaration

```
public void ImportPrivateKey(string privateKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	privateKey	Private key to import.

Sign(ArraySegment<Byte>, ArraySegment<Byte>)

Generate a signature of that can then be verified by the public key.

Declaration

```
public void Sign(ArraySegment<byte> data, ArraySegment<byte> output)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	data	Data to sign.
System.ArraySegment<System.Byte>	output	Output to write signature to.

Verify(ArraySegment<Byte>, ArraySegment<Byte>, String)

Verify that signature has been signed by someone with the private key.

Declaration

```
public bool Verify(ArraySegment<byte> data, ArraySegment<byte> signature, string remotePublicKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	data	Data that has been signed.
System.ArraySegment<System.Byte>	signature	Signature that has been generated.
System.String	remotePublicKey	Public key corresponding to the private key that signed the data.

Returns

TYPE	DESCRIPTION
System.Boolean	True if verified, false if not.

Implements

ICryptoAuthenticator
System.IDisposable

Class Curve25519

Elliptic Curve methods used for Diffie Hellman key exchange.

Inheritance

System.Object
Curve25519

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public static class Curve25519
```

Fields

KeySize

Key size in bytes.

Declaration

```
public const int KeySize = 32
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Methods

ClampPrivateKey(Byte[])

Private key clamping.

Declaration

```
public static byte[] ClampPrivateKey(byte[] rawKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	rawKey	[in] Random 32 bytes

Returns

TYPE	DESCRIPTION
System.Byte[]	Clamped private key.

ClampPrivateKeyInline(Byte[])

Private key clamping (inline, for performance).

Declaration

```
public static void ClampPrivateKeyInline(byte[] key)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	key	[out] Random 32 bytes.

CreateRandomPrivateKey()

Create a random clamped private key.

Declaration

```
public static byte[] CreateRandomPrivateKey()
```

Returns

TYPE	DESCRIPTION
System.Byte[]	Random 32 bytes that are clamped to a suitable private key.

GetPublicKey(Byte[])

Generate the public key out of the clamped private key.

Declaration

```
public static byte[] GetPublicKey(byte[] privateKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	privateKey	[in] Private key (must be clamped).

Returns

TYPE	DESCRIPTION
System.Byte[]	Public key.

GetPublicKeyInline(Byte[], Byte[])

Generate the public key out of the clamped private key (inline, for performance).

Declaration

```
public static void GetPublicKeyInline(byte[] privateKey, byte[] publicKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	privateKey	[in] Private key (must be clamped).
System.Byte[]	publicKey	[out] Public key.

GetSharedSecret(Byte[], Byte[])

Key agreement.

Declaration

```
public static byte[] GetSharedSecret(byte[] privateKey, byte[] peerPublicKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	privateKey	[in] Your private key for key agreement.
System.Byte[]	peerPublicKey	[in] Peer's public key.

Returns

TYPE	DESCRIPTION
System.Byte[]	Shared secret (needs hashing before use).

GetSharedSecretInline(Byte[], Byte[], Byte[])

Key agreement.

Declaration

```
public static void GetSharedSecretInline(byte[] privateKey, byte[] peerPublicKey, byte[] sharedSecret)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	privateKey	[in] Your private key for key agreement.

TYPE	NAME	DESCRIPTION
System.Byte[]	peerPublicKey	[in] Peer's public key.
System.Byte[]	sharedSecret	[out] Shared secret (needs hashing before use).

GetSigningKey(Byte[])

Generate signing key out of the clamped private key.

Declaration

```
public static byte[] GetSigningKey(byte[] privateKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	privateKey	[in] Private key (must be clamped).

Returns

TYPE	DESCRIPTION
System.Byte[]	Signing key.

KeyGenInline(Byte[], Byte[], Byte[])

Generate key-pair (inline, for performance).

Declaration

```
public static void KeyGenInline(byte[] publicKey, byte[] signingKey, byte[] privateKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	publicKey	[out] Public key.
System.Byte[]	signingKey	[out] Signing key (ignored if NULL).
System.Byte[]	privateKey	[out] Private key.

Remarks

WARNING: if signingKey is not NULL, this function has data-dependent timing.

Interface ICryptoAuthenticator

Defines methods used for authenticating secure hosts.

Inherited Members

System.IDisposable.Dispose()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public interface ICryptoAuthenticator : IDisposable
```

Properties

SignatureLength

Number of bytes in the signature.

Declaration

```
int SignatureLength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

ExportPrivateKey()

Export the current private key to a human readable format.

Declaration

```
string ExportPrivateKey()
```

Returns

TYPE	DESCRIPTION
System.String	Human readable private key

ExportPublicKey()

Export the current public key to a human readable format.

Declaration

```
string ExportPublicKey()
```

Returns

TYPE	DESCRIPTION
System.String	Human readable public key.

ImportPrivateKey(String)

Import private key.

Declaration

```
void ImportPrivateKey(string privateKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	privateKey	Private key to import.

Sign(ArraySegment<Byte>, ArraySegment<Byte>)

Generate a signature that can be verified by the public key.

Declaration

```
void Sign(ArraySegment<byte> data, ArraySegment<byte> output)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	data	Data to sign.
System.ArraySegment<System.Byte>	output	Output to write signature to.

Verify(ArraySegment<Byte>, ArraySegment<Byte>, String)

Verify signature.

Declaration

```
bool Verify(ArraySegment<byte> data, ArraySegment<byte> signature, string remotePublicKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	data	Data that has been signed.
System.ArraySegment<System.Byte>	signature	Signature that has been generated.
System.String	remotePublicKey	Public key corresponding to the private key that signed the data.

Returns

TYPE	DESCRIPTION
System.Boolean	True if verified, false if not.

Interface ICryptoEncryptor

Defines methods for encrypting and decrypting network packets.

Inherited Members

System.IDisposable.Dispose()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public interface ICryptoEncryptor : IDisposable
```

Methods

Decrypt(ArraySegment<Byte>, Byte[], Int32)

Decrypt data.

Declaration

```
int Decrypt(ArraySegment<byte> input, byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to decrypt.
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

Encrypt(ArraySegment<Byte>, Byte[], Int32)

Encrypt data.

Declaration

```
int Encrypt(ArraySegment<byte> input, byte[] output, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	input	Array segment to encrypt.

TYPE	NAME	DESCRIPTION
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.

Returns

TYPE	DESCRIPTION
System.Int32	Total number of bytes written to the output.

MaxDecryptedLength(Int32)

Compute the maximum decrypted length before decrypting.

Declaration

```
int MaxDecryptedLength(int inputLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputLength	Length of the input that is about to be decrypted.

Returns

TYPE	DESCRIPTION
System.Int32	Maximum possible decrypted length.

MaxEncryptedLength(Int32)

Compute the maximum encrypted length before encrypting.

Declaration

```
int MaxEncryptedLength(int inputLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	inputLength	Length of the input that is about to be encrypted.

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Int32	Maximum possible encrypted length.

Interface ICryptoExchanger

Defines methods used for a key exchange that is able to derive a shared encryptor.

Inherited Members

System.IDisposable.Dispose()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public interface ICryptoExchanger : IDisposable
```

Properties

KeyLength

Size of exchange key in bytes.

Declaration

```
int KeyLength { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

DeriveEncryptor(ArraySegment<Byte>)

Generate a shared encryptor.

Declaration

```
ICryptoEncryptor DeriveEncryptor(ArraySegment<byte> remoteKey)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	remoteKey	Received remote exchange key.

Returns

TYPE	DESCRIPTION
ICryptoEncryptor	Shared encryptor that is guaranteed to be the same on both peers.

ExportKey(ArraySegment<Byte>)

Copy exchange key to the output.

Declaration

```
void ExportKey(ArraySegment<byte> output)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	output	Output to write to.

Interface ICryptoRandom

Defines methods for generating random data.

Inherited Members

System.IDisposable.Dispose()

Namespace: SuperNet.Crypto

Assembly: cs.temp.dll.dll

Syntax

```
public interface ICryptoRandom : IDisposable
```

Methods

GetBytes(Byte[], Int32, Int32)

Generate cryptographically secure random data.

This method is thread safe.

Declaration

```
void GetBytes(byte[] output, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	output	Output buffer to write to.
System.Int32	offset	Output offset to write to.
System.Int32	count	Number of bytes to write.

Namespace SuperNet.Transport

Classes

ConnectionRequest

A connection request received by an active host.

Host

Manages a network socket and all network communication between peers.

HostConfig

Holds configuration values for hosts.

HostEvents

Event based implementation of a host listener.

HostStatistics

Stores packet statistics for hosts.

MessageEvents

Event based implementation of a message listener.

MessageReceived

Extra information for a network message that has been received by a connected peer.

MessageSent

Network message that has been sent to a connected peer.

Peer

Manages an active network connection.

PeerConfig

Holds configuration values for peers.

PeerEvents

Event based implementation of a peer listener.

PeerStatistics

Stores packet statistics for peers.

Structs

HostTimestamp

Stores a local timestamp of an event accurate down to a millisecond.

Interfaces

IHostListener

Implements a host listener.

IMessage

Implements a message that can be sent by the netcode to a connected peer.

IMessageListener

Implements a sent message listener.

IPeerListener

Implements a peer listener.

Enums

DisconnectReason

Reason provided when a connection ends.

Class ConnectionRequest

A connection request received by an active host.

Inheritance

System.Object
ConnectionRequest

Implements

System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class ConnectionRequest : IDisposable
```

Fields

Host

The host that received this request.

Declaration

```
public readonly Host Host
```

Field Value

TYPE	DESCRIPTION
Host	

Remote

Remote address that the request was received from.

Declaration

```
public readonly IPEndPoint Remote
```

Field Value

TYPE	DESCRIPTION
IPEndPoint	

Properties

Authenticate

True if remote peer requires us to authenticate.

Declaration

```
public bool Authenticate { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Disposed

True if the underlying buffers for the request have been repurposed for something else.

Declaration

```
public bool Disposed { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Encrypted

True if remote peer requires encryption.

Declaration

```
public bool Encrypted { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

Accept(PeerConfig, IPeerListener)

Accept the request, create a new peer and establish a connection.

Declaration

```
public Peer Accept(PeerConfig config, IPeerListener listener)
```

Parameters

TYPE	NAME	DESCRIPTION
PeerConfig	config	Peer configuration values. If null, default is used.
IPeerListener	listener	Peer listener. If null, event based listener is created.

Returns

TYPE	DESCRIPTION
Peer	The created peer.

Dispose()

Used internally by the netcode to invalidate the request, making it unable to be accepted.

This is called when the underlying buffers have been repurposed for something else.

Declaration

```
public void Dispose()
```

Reject(IWritable)

Reject the request by sending a reject message.

Declaration

```
public void Reject(IWritable message = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IWritable	message	Message to reject with.

Implements

System.IDisposable

Enum DisconnectReason

Reason provided when a connection ends.

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public enum DisconnectReason : byte
```

Fields

NAME	DESCRIPTION
BadSignature	Remote host failed to authenticate.
Disconnected	Graceful disconnect after requested.
Disposed	Peer or host has been disposed.
Exception	An exception has caused the peer to be disconnected. Always includes the actual exception.
Rejected	Connection request has been rejected by the remote peer.
Terminated	Graceful disconnect after the remote peer has requested it.
Timeout	Remote host has stopped responding to messages.

Class Host

Manages a network socket and all network communication between peers.

Inheritance

System.Object
Host

Implements

System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class Host : IDisposable
```

Constructors

Host(HostConfig, IHostListener)

Create a new UDP socket and start listening for packets.

Declaration

```
public Host(HostConfig config, IHostListener listener)
```

Parameters

TYPE	NAME	DESCRIPTION
HostConfig	config	Host configuration values. If null, defaults are used.
IHostListener	listener	Host listener to use. If null, event based listener is created.

Fields

Config

Configuration values for this host.

Declaration

```
public readonly HostConfig Config
```

Field Value

TYPE	DESCRIPTION
HostConfig	

Listener

Listener used by this host.

Declaration

```
public readonly IHostListener Listener
```

Field Value

TYPE	DESCRIPTION
IHostListener	

Statistics

Packet statistics.

Declaration

```
public readonly HostStatistics Statistics
```

Field Value

TYPE	DESCRIPTION
HostStatistics	

Properties

Disposed

True if host is disposed and cannot be used anymore.

Declaration

```
public bool Disposed { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

LocalEndPoint

Address this host is listening on.

Declaration

```
public IPEndPoint LocalEndPoint { get; }
```

Property Value

TYPE	DESCRIPTION
IPEndPoint	

SupportsIPv6

Platform dependant IPv6 support check. True if IPv6 is supported, false if not.

Declaration

```
public static bool SupportsIPv6 { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Ticks

Number of milliseconds that have elapsed since the host was created.

Declaration

```
public long Ticks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

Timestamp

Create a new timestamp at the current host time.

Declaration

```
public HostTimestamp Timestamp { get; }
```

Property Value

TYPE	DESCRIPTION
HostTimestamp	

Methods

Accept(ConnectionRequest, PeerConfig, IPeerListener)

Accept a connection request and return a connected local peer.

Declaration

```
public Peer Accept(ConnectionRequest request, PeerConfig config, IPeerListener listener)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	Connection request to accept.
PeerConfig	config	Peer configuration values. If null, default is used.
IPeerListener	listener	Peer listener to use. If null, event based listener is created.

Returns

TYPE	DESCRIPTION
Peer	Connected local peer.

Connect(IPEndPoint, PeerConfig, IPeerListener, IWritable)

Create a local peer and start connecting to an active remote host.

Declaration

```
public Peer Connect(IPEndPoint remote, PeerConfig config, IPeerListener listener, IWritable message = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address to connect to.
PeerConfig	config	Peer configuration values. If null, default is used.
IPeerListener	listener	Peer listener to use. If null, PeerEvents is used.
IWritable	message	Connect message to use.

Returns

TYPE	DESCRIPTION
Peer	Local peer that attempts to connect.

Dispose()

Instantly dispose all resources held by this host and connected peers.

Declaration

```
public void Dispose()
```

FindPeer(IPEndPoint)

Attempt to find an existing peer based on remote address.

Declaration

```
public Peer FindPeer(IPEndPoint remote)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address of the peer.

Returns

TYPE	DESCRIPTION
Peer	An existing peer or null if not found.

Reject(ConnectionRequest, IWritable)

Reject a connection request.

Declaration

```
public void Reject(ConnectionRequest request, IWritable message = null)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	Connection request to reject.
IWritable	message	Rejection message.

SendAll(IMessage, Peer[])

Send a message to all connected peers.

Declaration

```
public void SendAll(IMessage message, params Peer[] exclude)
```

Parameters

TYPE	NAME	DESCRIPTION
IMessage	message	Message to send.

TYPE	NAME	DESCRIPTION
Peer[]	exclude	Peers to exclude.

SendBroadcast(Int32, IWritable)

Send an unconnected message to all machines on the local network.

Declaration

```
public void SendBroadcast(int port, IWritable message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	port	Network port to send to.
IWritable	message	Message to send.

SendBroadcastAsync(Int32, IWritable)

Send an unconnected message to all machines on the local network.

Declaration

```
public Task<int> SendBroadcastAsync(int port, IWritable message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	port	Network port to send to.
IWritable	message	Message to send.

Returns

TYPE	DESCRIPTION
Task<System.Int32>	Task that returns number of bytes sent.

SendUnconnected(IPEndPoint, IWritable)

Send an unconnected message to a remote host.

Declaration

```
public void SendUnconnected(IPEndPoint remote, IWritable message)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address to send to.
IWritable	message	Message to send.

SendUnconnectedAsync(IPEndPoint, IWritable)

Send an unconnected message to a remote host.

Declaration

```
public Task<int> SendUnconnectedAsync(IPEndPoint remote, IWritable message)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address to send to.
IWritable	message	Message to send.

Returns

TYPE	DESCRIPTION
Task<System.Int32>	Task that returns number of bytes sent.

Shutdown()

Gracefully disconnect all peers and perform a shutdown.

Declaration

```
public void Shutdown()
```

ShutdownAsync()

Gracefully disconnect all peers and perform a shutdown.

Declaration

```
public Task ShutdownAsync()
```

Returns

TYPE	DESCRIPTION
Task	Task that completes when shutdown is completed.

Implements

System.IDisposable

Class HostConfig

Holds configuration values for hosts.

Inheritance

System.Object
HostConfig

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class HostConfig
```

Fields

AllocatorCount

Number of pooled arrays.

Declaration

```
public int AllocatorCount
```

Field Value

TYPE	DESCRIPTION
System.Int32	

AllocatorExpandLength

Number of bytes to add when a non-pooled array becomes too small

Declaration

```
public int AllocatorExpandLength
```

Field Value

TYPE	DESCRIPTION
System.Int32	

AllocatorMaxLength

Maximum length of allocated arrays.

Declaration

```
public int AllocatorMaxLength
```

Field Value

TYPE	DESCRIPTION
System.Int32	

AllocatorPooledExpandLength

Number of bytes to add when a pooled array becomes too small.

Declaration

```
public int AllocatorPooledExpandLength
```

Field Value

TYPE	DESCRIPTION
System.Int32	

AllocatorPooledLength

Maximum length an array can still be to be pooled.

Declaration

```
public int AllocatorPooledLength
```

Field Value

TYPE	DESCRIPTION
System.Int32	

BindAddress

Bind socket to a specific address or null for any.

Declaration

```
public IPAddress BindAddress
```

Field Value

TYPE	DESCRIPTION
IPAddress	

Broadcast

Set `Socket.EnableBroadcast` when creating a socket.

If true, allow broadcast messages to be sent.

Declaration

```
public bool Broadcast
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Compression

Enable compression of outgoing network packets.

If false and a compressed packet is received, it is still decompressed.

Declaration

```
public bool Compression
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

CRC32

Enable CRC32 error checking to make sure packets don't get corrupted in transit.

If false and a CRC32 code is received, it is ignored.

Declaration

```
public bool CRC32
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

DualMode

Set `Socket.DualMode` when creating a socket.

If true, accept both IPv6 and IPv4 connections.

Declaration

```
public bool DualMode
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Encryption

Enable end to end encryption between peers.

A connection request without encryption can still be accepted.

Declaration

```
public bool Encryption
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Port

UDP port to listen on or zero for random.

Must be between 0 and 65536.

Declaration

```
public int Port
```

Field Value

TYPE	DESCRIPTION
System.Int32	

PrivateKey

Private key to use when authenticating this host.

If null, this host cannot be authenticated.

Declaration

```
public string PrivateKey
```

Field Value

TYPE	DESCRIPTION
System.String	

ReceiveBufferSize

Set `Socket.ReceiveBufferSize` when creating a socket.

Maximum socket receive buffer in bytes.

Declaration

```
public int ReceiveBufferSize
```

Field Value

TYPE	DESCRIPTION
System.Int32	

ReceiveCount

Maximum number of possible concurrent read operations.

Declaration

```
public int ReceiveCount
```

Field Value

TYPE	DESCRIPTION
System.Int32	

ReceiveMTU

Maximum number of bytes in a single received UDP packet.

This is used to allocate appropriately sized receive buffers.

Declaration

```
public int ReceiveMTU
```

Field Value

TYPE	DESCRIPTION
System.Int32	

SendBufferSize

Set `Socket.SendBufferSize` when creating a socket.

Maximum socket send buffer in bytes.

Declaration

```
public int SendBufferSize
```

Field Value

TYPE	DESCRIPTION
System.Int32	

TTL

Set `Socket.Ttl` when creating a socket.

Maximum number of hops packets can take before being dropped.

Declaration

```
public short TTL
```

Field Value

TYPE	DESCRIPTION
System.Int16	

Class HostEvents

Event based implementation of a host listener.

Inheritance

System.Object
HostEvents

Implements

IHostListener

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class HostEvents : IHostListener
```

Events

OnException

Called when an exception occurs internally. Can be ignored.

Declaration

```
public event HostEvents.OnExceptionHandler OnException
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.HostEvents.OnExceptionHandler	

OnReceiveBroadcast

Called for every broadcast message the host receives.

Declaration

```
public event HostEvents.OnReceiveBroadcastHandler OnReceiveBroadcast
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.HostEvents.OnReceiveBroadcastHandler	

OnReceiveRequest

Called when a connection request is received.

Declaration

```
public event HostEvents.OnReceiveRequestHandler OnReceiveRequest
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.HostEvents.OnReceiveRequestHandler	

OnReceiveSocket

Called for every raw packet the host receives.

Declaration

```
public event HostEvents.OnReceiveSocketHandler OnReceiveSocket
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.HostEvents.OnReceiveSocketHandler	

OnReceiveUnconnected

Called for every unconnected message the host receives.

Declaration

```
public event HostEvents.OnReceiveUnconnectedHandler OnReceiveUnconnected
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.HostEvents.OnReceiveUnconnectedHandler	

OnShutdown

Called when the host shuts down.

Declaration

```
public event HostEvents.OnShutdownHandler OnShutdown
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.HostEvents.OnShutdownHandler	

Explicit Interface Implementations

IHostListener.OnHostException(IPEndPoint, Exception)

Declaration

```
void IHostListener.OnHostException(IPEndPoint remote, Exception exception)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	
System.Exception	exception	

IHostListener.OnHostReceiveBroadcast(IPEndPoint, Reader)

Declaration

```
void IHostListener.OnHostReceiveBroadcast(IPEndPoint remote, Reader message)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	
Reader	message	

IHostListener.OnHostReceiveRequest(ConnectionRequest, Reader)

Declaration

```
void IHostListener.OnHostReceiveRequest(ConnectionRequest request, Reader message)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	
Reader	message	

IHostListener.OnHostReceiveSocket(IPEndPoint, Byte[], Int32)

Declaration

```
void IHostListener.OnHostReceiveSocket(IPEndPoint remote, byte[] buffer, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	
System.Byte[]	buffer	
System.Int32	length	

IHostListener.OnHostReceiveUnconnected(IPEndPoint, Reader)

Declaration

```
void IHostListener.OnHostReceiveUnconnected(IPEndPoint remote, Reader message)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	
Reader	message	

IHostListener.OnHostShutdown()

Declaration

```
void IHostListener.OnHostShutdown()
```

Implements

IHostListener

Class HostStatistics

Stores packet statistics for hosts.

Inheritance

System.Object
HostStatistics

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class HostStatistics
```

Properties

SocketReceiveBytes

Total number of bytes received.

Declaration

```
public long SocketReceiveBytes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

SocketReceiveCount

Total number of packets received.

Declaration

```
public long SocketReceiveCount { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

SocketReceiveTicks

Host ticks at the moment of the last socket receive operation.

Declaration

```
public long SocketReceiveTicks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

SocketSendBytes

Total number of bytes sent.

Declaration

```
public long SocketSendBytes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

SocketSendCount

Total number of packets sent.

Declaration

```
public long SocketSendCount { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

SocketSendTicks

Host ticks at the moment of the last socket send operation.

Declaration

```
public long SocketSendTicks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

Methods

Reset()

Reset all statistics back to zero.

Declaration

```
public void Reset()
```

Struct HostTimestamp

Stores a local timestamp of an event accurate down to a millisecond.

Inherited Members

System.ValueType.Equals(System.Object)
System.ValueType.GetHashCode()
System.ValueType.ToString()
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetType()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public struct HostTimestamp
```

Fields

Host

Host that created this timestamp.

Declaration

```
public readonly Host Host
```

Field Value

TYPE	DESCRIPTION
Host	

Ticks

Raw host ticks.

Declaration

```
public readonly long Ticks
```

Field Value

TYPE	DESCRIPTION
System.Int64	

Properties

ElapsedDays

Number of days since the creation of this timestamp.

Declaration

```
public double ElapsedDays { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

ElapsedHours

Number of hours since the creation of this timestamp.

Declaration

```
public double ElapsedHours { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

ElapsedMilliseconds

Number of milliseconds since the creation of this timestamp.

Declaration

```
public long ElapsedMilliseconds { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

ElapsedMinutes

Number of minutes since the creation of this timestamp.

Declaration

```
public double ElapsedMinutes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

ElapsedSeconds

Number of seconds since the creation of this timestamp.

Declaration

```
public double ElapsedSeconds { get; }
```

Property Value

TYPE	DESCRIPTION
System.Double	

Interface IHostListener

Implements a host listener.

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public interface IHostListener
```

Methods

OnHostException(IPEndPoint, Exception)

Called when an exception occurs internally.

This does not usually indicate any fatal errors and can be ignored.

Declaration

```
void OnHostException(IPEndPoint remote, Exception exception)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address associated with this exception or null.
System.Exception	exception	Exception that was thrown.

OnHostReceiveBroadcast(IPEndPoint, Reader)

Called for every broadcast message the host receives.

Declaration

```
void OnHostReceiveBroadcast(IPEndPoint remote, Reader message)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address message was received from.
Reader	message	Message that was received.

OnHostReceiveRequest(ConnectionRequest, Reader)

Called when a connection request is received.

The request can only be accepted during this call.

Declaration

```
void OnHostReceiveRequest(ConnectionRequest request, Reader message)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	Connection request received.
Reader	message	Message sent with the connection request.

OnHostReceiveSocket(IPEndPoint, Byte[], Int32)

Called for every raw packet the host receives.

Declaration

```
void OnHostReceiveSocket(IPEndPoint remote, byte[] buffer, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address packet was received from.
System.Byte[]	buffer	Receive buffer the packet is written on.
System.Int32	length	Number of bytes in the packet.

OnHostReceiveUnconnected(IPEndPoint, Reader)

Called for every unconnected message the host receives.

Declaration

```
void OnHostReceiveUnconnected(IPEndPoint remote, Reader message)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	remote	Remote address message was received from.
Reader	message	Message that was received.

OnHostShutdown()

Called when the host shuts down.

Declaration

```
void OnHostShutdown()
```

Interface IMessage

Implements a message that can be sent by the netcode to a connected peer.

Inherited Members

IWritable.Write(Writer)

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public interface IMessage : IWritable
```

Properties

Channel

Which data channel to send the message on.

Declaration

```
byte Channel { get; }
```

Property Value

TYPE	DESCRIPTION
System.Byte	

Ordered

Message must be delivered in order within the channel.

Any unreliable messages that arrive out of order are dropped.

Any reliable messages that arrive out of order are reordered automatically.

Declaration

```
bool Ordered { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Reliable

Message requires an acknowledgment and needs to be resent until acknowledged.

This makes sure the message will never be lost.

Declaration

```
bool Reliable { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Timed

Message includes a timestamp at the moment of creation.

If this is false, received timestamp might be innacurate due to message delays.

Declaration

```
bool Timed { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Unique

Message is guaranteed not to be duplicated.

Declaration

```
bool Unique { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Interface IMessageListener

Implements a sent message listener.

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public interface IMessageListener
```

Methods

OnMessageAcknowledge(Peer, MessageSent)

Called when a reliable message gets acknowledged.

Declaration

```
void OnMessageAcknowledge(Peer peer, MessageSent message)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	Peer that received the acknowledgment.
MessageSent	message	Message that was acknowledged.

OnMessageSend(Peer, MessageSent)

Called after the message gets sent to the socket.

Declaration

```
void OnMessageSend(Peer peer, MessageSent message)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	Peer that sent the message.
MessageSent	message	Message that was sent.

Interface IPeerListener

Implements a peer listener.

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public interface IPeerListener
```

Methods

OnPeerConnect(Peer)

Called when a peer successfully connects.

Declaration

```
void OnPeerConnect(Peer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	Connected peer.

OnPeerDisconnect(Peer, Reader, DisconnectReason, Exception)

Called when a peer disconnects.

Declaration

```
void OnPeerDisconnect(Peer peer, Reader message, DisconnectReason reason, Exception exception)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	Disconnected peer.
Reader	message	Disconnect message or null if not included.
DisconnectReason	reason	Disconnect reason.
System.Exception	exception	Exception associated with the disconnect or null if none.

OnPeerException(Peer, Exception)

Called when an exception occurs internally. Can be ignored.

Declaration

```
void OnPeerException(Peer peer, Exception exception)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	Peer involved.
System.Exception	exception	Exception that was thrown.

OnPeerReceive(Peer, Reader, MessageReceived)

Called when a peer receives a connected message.

Declaration

```
void OnPeerReceive(Peer peer, Reader message, MessageReceived info)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	Receiver of the message.
Reader	message	Message that was received.
MessageReceived	info	Extra message information.

OnPeerUpdateRTT(Peer, UInt16)

Called when round trip time (ping) is updated.

Declaration

```
void OnPeerUpdateRTT(Peer peer, ushort rtt)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	Updated peer.
System.UInt16	rtt	New RTT value.

Class MessageEvents

Event based implementation of a message listener.

Inheritance

System.Object
MessageEvents

Implements

IMessageListener

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class MessageEvents : IMessageListener
```

Events

OnAcknowledge

Called when a reliable message gets acknowledged.

Declaration

```
public event MessageEvents.OnAcknowledgeHandler OnAcknowledge
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.MessageEvents.OnAcknowledgeHandler	

OnSend

Called after the message gets sent to the socket.

Declaration

```
public event MessageEvents.OnSendHandler OnSend
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.MessageEvents.OnSendHandler	

Explicit Interface Implementations

IMessageListener.OnMessageAcknowledge(Peer, MessageSent)

Declaration

```
void IMessageListener.OnMessageAcknowledge(Peer peer, MessageSent message)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	
MessageSent	message	

IMessageListener.OnMessageSend(Peer, MessageSent)

Declaration

```
void IMessageListener.OnMessageSend(Peer peer, MessageSent message)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	
MessageSent	message	

Implements

IMessageListener

Class MessageReceived

Extra information for a network message that has been received by a connected peer.

Inheritance

System.Object
MessageReceived

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class MessageReceived
```

Fields

Attempt

How many times the message was previously sent before.

Declaration

```
public readonly byte Attempt
```

Field Value

TYPE	DESCRIPTION
System.Byte	

Channel

Data channel the message was sent over.

Declaration

```
public readonly byte Channel
```

Field Value

TYPE	DESCRIPTION
System.Byte	

Peer

Peer that the message was received by.

Declaration

```
public readonly Peer Peer
```

Field Value

TYPE	DESCRIPTION
Peer	

Timestamp

Timestamp in local host time at the moment of creation of the message.

If message was not timed, this is approximated using round trip time.

Declaration

```
public readonly HostTimestamp Timestamp
```

Field Value

TYPE	DESCRIPTION
HostTimestamp	

Class MessageSent

Network message that has been sent to a connected peer.

Inheritance

System.Object
MessageSent

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class MessageSent
```

Fields

Channel

Data channel this message is sent over.

Declaration

```
public readonly byte Channel
```

Field Value

TYPE	DESCRIPTION
System.Byte	

Listener

Listener used for this message or null if not provided.

Declaration

```
public readonly IMessageListener Listener
```

Field Value

TYPE	DESCRIPTION
IMessageListener	

Payload

Message payload that is used to write to internal buffers.

Declaration

```
public readonly IWritable Payload
```

Field Value

TYPE	DESCRIPTION
IWritable	

Peer

Peer that the message was sent through.

Declaration

```
public readonly Peer Peer
```

Field Value

TYPE	DESCRIPTION
Peer	

Sequence

Internal sequence number of the message.

Declaration

```
public readonly ushort Sequence
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Properties

Acknowledged

True if message is reliable and has been acknowledged.

Declaration

```
public bool Acknowledged { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Attempts

Number of times this message has been sent.

Declaration

```
public int Attempts { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

TimeCreated

Host timestamp at the moment of creation of this message.

Declaration

```
public HostTimestamp TimeCreated { get; }
```

Property Value

TYPE	DESCRIPTION
HostTimestamp	

TimeSent

Host timestamp at the moment the message was sent to the network socket.

Declaration

```
public HostTimestamp TimeSent { get; }
```

Property Value

TYPE	DESCRIPTION
HostTimestamp	

Methods

StopResending()

Stop resending this message if reliable. May cause the message to be lost.

Declaration

```
public void StopResending()
```

Class Peer

Manages an active network connection.

Inheritance

System.Object
Peer

Implements

System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class Peer : IDisposable
```

Fields

Config

Configuration values for this peer.

Declaration

```
public readonly PeerConfig Config
```

Field Value

TYPE	DESCRIPTION
PeerConfig	

Host

Host used to manage this peer.

Declaration

```
public readonly Host Host
```

Field Value

TYPE	DESCRIPTION
Host	

Listener

Listener used by this peer.

Declaration

```
public readonly IPeerListener Listener
```

Field Value

TYPE	DESCRIPTION
IPeerListener	

Remote

Address this peer is connected to.

Declaration

```
public readonly IPEndPoint Remote
```

Field Value

TYPE	DESCRIPTION
IPEndPoint	

Statistics

Packet statistics.

Declaration

```
public readonly PeerStatistics Statistics
```

Field Value

TYPE	DESCRIPTION
PeerStatistics	

Properties

Connected

True if messages can be sent.

Declaration

```
public bool Connected { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Connecting

True if peer is in the process of connecting.

Declaration

```
public bool Connecting { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Disposed

True if peer has been disposed.

Declaration

```
public bool Disposed { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

RTT

Current round trip time (ping) in milliseconds.

Declaration

```
public ushort RTT { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Disconnect(IWritable)

Disconnect by sending a disconnect message.

Declaration

```
public void Disconnect(IWritable message = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IWritable	message	Disconnect message to include or null if none.

DisconnectAsync(IWritable)

Disconnect by sending a disconnect message.

Declaration

```
public Task DisconnectAsync(IWritable message = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IWritable	message	Disconnect message to include or null if none.

Returns

TYPE	DESCRIPTION
Task	

Dispose()

Instantly dispose of all resources held by this peer.

Declaration

```
public void Dispose()
```

Send(IMessage, IMessageListener)

Queue a message for sending and return a sent message handle.

Declaration

```
public MessageSent Send(IMessage message, IMessageListener listener = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IMessage	message	Message to send.
IMessageListener	listener	Message listener to use or null if not used.

Returns

TYPE	DESCRIPTION
MessageSent	Sent message handle.

Implements

System.IDisposable

Class PeerConfig

Holds configuration values for peers.

Inheritance

System.Object
PeerConfig

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class PeerConfig
```

Fields

ConnectAttempts

Number of connection requests to send before giving up. This is a high number to allow enough time for UDP hole punching.

Declaration

```
public int ConnectAttempts
```

Field Value

TYPE	DESCRIPTION
System.Int32	

ConnectDelay

Delay in milliseconds between connection requests.

Declaration

```
public int ConnectDelay
```

Field Value

TYPE	DESCRIPTION
System.Int32	

DisconnectDelay

Number of milliseconds to delay closing the connection when a disconnect request is received.

This is useful in cases where both peers disconnect at the same time. It is also useful for when a disconnect acknowledge gets lost. Set to zero to disable this delay.

Declaration

```
public int DisconnectDelay
```

Field Value

TYPE	DESCRIPTION
System.Int32	

DuplicateTimeout

How long in milliseconds to keep received reliable messages for. If the same reliable message is received during this timeout, it is ignored.

Declaration

```
public int DuplicateTimeout
```

Field Value

TYPE	DESCRIPTION
System.Int32	

FragmentTimeout

Timeout in milliseconds until a received incompleted fragmented packet times out.

Declaration

```
public int FragmentTimeout
```

Field Value

TYPE	DESCRIPTION
System.Int32	

MTU

Maximum bytes to send in one UDP packet.

MTU on ethernet is 1500 bytes - 20 bytes for IP header - 8 bytes for UDP header.

Declaration

```
public int MTU
```

Field Value

TYPE	DESCRIPTION
System.Int32	

OrderedDelayMax

Maximum number of messages to wait for before processing a reliable ordered message that came out of order.

If an ordered reliable message comes late, it is delayed until all missing messages are received. This value controls maximum number of missing messages to wait for. If this is zero, delaying is disabled.

Declaration

```
public int OrderedDelayMax
```

Field Value

TYPE	DESCRIPTION
System.Int32	

OrderedDelayTimeout

Maximum number of milliseconds to wait for before processing a reliable ordered message that came out of order.

If an ordered reliable message comes late, it is delayed until all missing messages are received. This controls maximum number of milliseconds to wait for. If this is zero, delaying is disabled.

Declaration

```
public int OrderedDelayTimeout
```

Field Value

TYPE	DESCRIPTION
System.Int32	

PingDelay

Delay in milliseconds between ping messages.

Declaration

```
public int PingDelay
```

Field Value

TYPE	DESCRIPTION
System.Int32	

RemotePublicKey

Remote public key to verify authentication signature against.

If provided and remote peer has authentication disabled, they will ignore connection requests.

Declaration

```
public string RemotePublicKey
```

Field Value

TYPE	DESCRIPTION
System.String	

ResendCount

Maximum number of times a reliable message is resent without being acknowledged before the connection times out.

Declaration

```
public byte ResendCount
```

Field Value

TYPE	DESCRIPTION
System.Byte	

ResendDelayJitter

Maximum number of milliseconds to wait before declaring a reliable message as lost.

When a reliable message is sent, peer waits RTT + ResendDelayJitter milliseconds for an acknowledgment. If no acknowledgment is received within that time, the message is resent. A small value can result in unnecessary duplicated messages wasting networking bandwidth.

Declaration

```
public int ResendDelayJitter
```

Field Value

TYPE	DESCRIPTION
System.Int32	

ResendDelayMax

Maximum delay in milliseconds before resending unacknowledged reliable messages.

Declaration

```
public int ResendDelayMax
```

Field Value

TYPE	DESCRIPTION
System.Int32	

ResendDelayMin

Minimum delay in milliseconds before resending unacknowledged reliable messages.

Declaration

```
public int ResendDelayMin
```

Field Value

TYPE	DESCRIPTION
System.Int32	

SendDelay

Delay in milliseconds before combining and sending messages to the socket.

Declaration

```
public int SendDelay
```

Field Value

TYPE	DESCRIPTION
System.Int32	

UnsequencedMax

Maximum number of consecutive unsequenced messages to send.

All reliable messages include a sequence number. Unreliable messages don't need a sequence number but can include it. This value controls how often to include a sequence number. Sending a sequence number every so often is important to check for lost messages.

Declaration

```
public int UnsequencedMax
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Class PeerEvents

Event based implementation of a peer listener.

Inheritance

System.Object
PeerEvents

Implements

IPeerListener

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class PeerEvents : IPeerListener
```

Events

OnConnect

Called when a peer successfully connects.

Declaration

```
public event PeerEvents.OnConnectHandler OnConnect
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.PeerEvents.OnConnectHandler	

OnDisconnect

Called when a peer disconnects.

Declaration

```
public event PeerEvents.OnDisconnectHandler OnDisconnect
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.PeerEvents.OnDisconnectHandler	

OnException

Called when an exception occurs internally. Can be ignored.

Declaration

```
public event PeerEvents.OnExceptionHandler OnException
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.PeerEvents.OnExceptionHandler	

OnReceive

Called when a peer receives a connected message.

Declaration

```
public event PeerEvents.OnReceiveHandler OnReceive
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.PeerEvents.OnReceiveHandler	

OnUpdateRTT

Called when round trip time (ping) is updated.

Declaration

```
public event PeerEvents.OnUpdateRTTHandler OnUpdateRTT
```

Event Type

TYPE	DESCRIPTION
SuperNet.Transport.PeerEvents.OnUpdateRTTHandler	

Explicit Interface Implementations

IPeerListener.OnPeerConnect(Peer)

Declaration

```
void IPeerListener.OnPeerConnect(Peer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	

IPeerListener.OnPeerDisconnect(Peer, Reader, DisconnectReason, Exception)

Declaration

```
void IPeerListener.OnPeerDisconnect(Peer peer, Reader message, DisconnectReason reason, Exception exception)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	
Reader	message	
DisconnectReason	reason	
System.Exception	exception	

IPeerListener.OnPeerException(Peer, Exception)

Declaration

```
void IPeerListener.OnPeerException(Peer peer, Exception exception)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	
System.Exception	exception	

IPeerListener.OnPeerReceive(Peer, Reader, MessageReceived)

Declaration

```
void IPeerListener.OnPeerReceive(Peer peer, Reader message, MessageReceived info)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	
Reader	message	
MessageReceived	info	

IPeerListener.OnPeerUpdateRTT(Peer, UInt16)

Declaration

```
void IPeerListener.OnPeerUpdateRTT(Peer peer, ushort rtt)
```

Parameters

TYPE	NAME	DESCRIPTION
Peer	peer	
System.UInt16	rtt	

Implements

IPeerListener

Class PeerStatistics

Stores packet statistics for peers.

Inheritance

System.Object
PeerStatistics

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Transport

Assembly: cs.temp.dll.dll

Syntax

```
public class PeerStatistics
```

Properties

MessageReceiveAcknowledge

Total number of received acknowledgements.

Declaration

```
public long MessageReceiveAcknowledge { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageReceiveBytes

Total number of bytes recieved in messages after decryption and decompression.

Declaration

```
public long MessageReceiveBytes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageReceiveDuplicated

Total number of received duplicated messages.

Declaration

```
public long MessageReceiveDuplicated { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageReceiveLost

Total number of lost messages.

Declaration

```
public long MessageReceiveLost { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageReceivePing

Total number of received pings.

Declaration

```
public long MessageReceivePing { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageReceiveReliable

Total number of received reliable messages.

Declaration

```
public long MessageReceiveReliable { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageReceiveTotal

Total number of received messages.

Declaration

```
public long MessageReceiveTotal { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageReceiveUnreliable

Total number of received unreliable messages.

Declaration

```
public long MessageReceiveUnreliable { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageSendAcknowledge

Total number of sent acknowledgements.

Declaration

```
public long MessageSendAcknowledge { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageSendBytes

Total number of sent bytes in messages before compression and encryption.

Declaration

```
public long MessageSendBytes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageSendDuplicated

Total number of sent duplicated messages.

Declaration

```
public long MessageSendDuplicated { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageSendPing

Total number of sent pings.

Declaration

```
public long MessageSendPing { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageSendReliable

Total number of sent reliable messages.

Declaration

```
public long MessageSendReliable { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageSendTotal

Total number of sent messages.

Declaration

```
public long MessageSendTotal { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

MessageSendUnreliable

Total number of sent unreliable messages.

Declaration

```
public long MessageSendUnreliable { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

PacketReceiveBytes

Total number of bytes received.

Declaration

```
public long PacketReceiveBytes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

PacketReceiveCount

Total number of packets received.

Declaration

```
public long PacketReceiveCount { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

PacketReceiveTicks

Host ticks at the moment of the last receive operation.

Declaration

```
public long PacketReceiveTicks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

PacketSendBytes

Total number of bytes sent.

Declaration

```
public long PacketSendBytes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

PacketSendCount

Total number of sent packets.

Declaration

```
public long PacketSendCount { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

PacketSendTicks

Host ticks at the moment of the last send operation.

Declaration

```
public long PacketSendTicks { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

Methods

Reset()

Reset all statistics to zero.

Declaration

```
public void Reset()
```

Namespace SuperNet.Util

Classes

Allocator

ArrayPool<T>

Array pool for reusing arrays to avoid too many allocations.

CRC32

Fast CRC32 error-checking code calculation for network packets.

IPComparer

Equality comparer used by the netcode to distinguish between peers.

IPResolver

Helper methods that convert a connection string to an `IPEndPoint` used by the netcode.

ObjectPool<T>

Object pool for reusing objects to avoid too many allocations.

Reader

Fast deserializer for network messages.

Serializer

Platform independent serialization of values in Big-endian (network byte order).

Writer

Fast serializer for network messages.

Interfaces

IWritable

Defines a serializable network payload.

Class Allocator

Inheritance

System.Object
Allocator

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class Allocator
```

Constructors

Allocator()

Create a new allocator without any pooling.

Declaration

```
public Allocator()
```

Allocator(HostConfig)

Create a new allocator for a host.

Declaration

```
public Allocator(HostConfig config)
```

Parameters

TYPE	NAME	DESCRIPTION
HostConfig	config	Configuration to use.

Methods

CreateIV(Int32)

Allocate a new IV array for crypto.

Declaration

```
public byte[] CreateIV(int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	length	Length of the array.

Returns

TYPE	DESCRIPTION
System.Byte[]	A new unused array.

CreateKey(Int32)

Allocate a new key array for crypto.

Declaration

```
public byte[] CreateKey(int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	length	Length of the array.

Returns

TYPE	DESCRIPTION
System.Byte[]	A new unused array.

CreateMessage(Int32)

Allocate a new resizable array to store a single message.

Declaration

```
public byte[] CreateMessage(int minimumLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	minimumLength	Minimum length of the returned array.

Returns

TYPE	DESCRIPTION
System.Byte[]	A new unused array.

CreatePacket(Int32)

Allocate a new short array to store a single packet.

Declaration

```
public byte[] CreatePacket(int minimumLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	minimumLength	Minimum length of the returned array.

Returns

TYPE	DESCRIPTION
System.Byte[]	A new unused array.

CreateSent()

Allocate a new sent message storage used by peers.

Declaration

```
public Dictionary<Tuple<byte, ushort>, MessageSent> CreateSent()
```

Returns

TYPE	DESCRIPTION
Dictionary<System.Tuple<System.Byte, System.UInt16>, MessageSent>	A new unused sent message storage.

CreateSet()

Allocate a new HashSet used by peers.

Declaration

```
public HashSet<Tuple<byte, ushort>> CreateSet()
```

Returns

TYPE	DESCRIPTION
HashSet<System.Tuple<System.Byte, System.UInt16>>	A new unused HashSet.

ExpandMessage(Byte[], Int32, Int32)

Resize a message array to a larger size.

Declaration

```
public byte[] ExpandMessage(byte[] array, int offset, int length = 1)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to resize.
System.Int32	offset	Current array offset.
System.Int32	length	Length beyond the array offset to add.

Returns

TYPE	DESCRIPTION
System.Byte[]	A new resized array with copied data.

HashTableCreate(Int32)

Allocate a new hash table array for the LZF compressor.

Declaration

```
public long[] HashTableCreate(int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	length	Length of the array.

Returns

TYPE	DESCRIPTION
System.Int64[]	A new unused array.

HashTableReturn(Int64[])

Return a hash table array back to the pool.

Declaration

```
public void HashTableReturn(long[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64[]	array	Array to return.

ReturnIV(ref Byte[])

Return an IV array back to the pool.

Declaration

```
public void ReturnIV(ref byte[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to return.

ReturnKey(ref Byte[])

Return a key array back to the pool.

Declaration

```
public void ReturnKey(ref byte[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to return.

ReturnMessage(ref Byte[])

Return a message array back to the pool.

Declaration

```
public void ReturnMessage(ref byte[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to return.

ReturnPacket(ref Byte[])

Return a packet array back to the pool.

Declaration

```
public void ReturnPacket(ref byte[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to return.

ReturnSent(ref Dictionary<Tuple<Byte, UInt16>, MessageSent>)

Return a sent message storage back to the pool.

Declaration

```
public void ReturnSent(ref Dictionary<Tuple<byte, ushort>, MessageSent> set)
```

Parameters

TYPE	NAME	DESCRIPTION
Dictionary<System.Tuple<System.Byte, System.UInt16>, MessageSent>	set	Sent message storage to return.

ReturnSet(ref HashSet<Tuple<Byte, UInt16>>)

Return a HashSet back to the pool.

Declaration

```
public void ReturnSet(ref HashSet<Tuple<byte, ushort>> set)
```

Parameters

TYPE	NAME	DESCRIPTION
HashSet<System.Tuple<System.Byte, System.UInt16>>	set	HashSet to return.

SequenceNew(Int32)

Allocate a new array to store message sequence for each channel.

Declaration

```
public int[] SequenceNew(int channels)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channels	Number of channels.

Returns

TYPE	DESCRIPTION
System.Int32[]	A new unused array.

SequenceReturn(ref Int32[])

Return a sequence array back to the pool.

Declaration

```
public void SequenceReturn(ref int[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32[]	array	Array to return.

TokensNew(Int32)

Allocate a new cancellation token array for each channel.

Declaration

```
public CancellationTokenSource[] TokensNew(int channels)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	channels	Number of channels.

Returns

TYPE	DESCRIPTION
CancellationTokenSource[]	A new unused array.

TokensReturn(ref CancellationTokenSource[])

Return a cancellation token array back to the pool.

Declaration

```
public void TokensReturn(ref CancellationTokenSource[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationTokenSource[]	array	Array to return.

Class ArrayPool<T>

Array pool for reusing arrays to avoid too many allocations.

Inheritance

System.Object
ArrayPool<T>

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class ArrayPool<T>
```

Type Parameters

NAME	DESCRIPTION
T	Underlying array type.

Constructors

ArrayPool(Int32, Int32)

Create a new array pool.

Declaration

```
public ArrayPool(int count, int maxLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	count	Number of arrays this pool can hold.
System.Int32	maxLength	Maximum length arrays can be saved at.

Methods

Expand(T[], Int32, Int32, Int32)

Resize an array created by this pool.

Declaration

```
public T[] Expand(T[] array, int copyLength, int addLength, int expandLength)
```

Parameters

TYPE	NAME	DESCRIPTION
T[]	array	Array to resize.
System.Int32	copyLength	Number of bytes to copy to the new array.
System.Int32	addLength	Number of bytes to add after the copy length.
System.Int32	expandLength	Array length multiplier.

Returns

TYPE	DESCRIPTION
T[]	A new resized array.

Rent(Int32)

Extract an array from this pool or allocate a new one.

Declaration

```
public T[] Rent(int minimumLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	minimumLength	Minimum length that the returned array has to be.

Returns

TYPE	DESCRIPTION
T[]	An unused array.

Return(T[])

Return an array back to this pool.

Declaration

```
public void Return(T[] array)
```

Parameters

TYPE	NAME	DESCRIPTION
T[]	array	Array to return.

Class CRC32

Fast CRC32 error-checking code calculation for network packets.

Inheritance

System.Object
CRC32

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public static class CRC32
```

Methods

Compute(Byte[], Int32, Int32)

Compute a CRC32 code for the input array segment.

Declaration

```
public static uint Compute(byte[] array, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to read from.
System.Int32	offset	Offset in the array to start reading from.
System.Int32	count	Number of bytes to read.

Returns

TYPE	DESCRIPTION
System.UInt32	CRC32 code of the input.

Class IPComparer

Equality comparer used by the netcode to distinguish between peers.

Inheritance

System.Object
IPComparer

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class IPComparer : IEqualityComparer<IPEndPoint>
```

Methods

Equals(IPEndPoint, IPEndPoint)

Check if both address and port match.

Declaration

```
public bool Equals(IPEndPoint x, IPEndPoint y)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	x	First IP
IPEndPoint	y	Second IP

Returns

TYPE	DESCRIPTION
System.Boolean	True if they match, false if not.

GetHashCode(IPEndPoint)

Construct a hash code based on address and port.

Declaration

```
public int GetHashCode(IPEndPoint obj)
```

Parameters

TYPE	NAME	DESCRIPTION
IPEndPoint	obj	Object to construct the hash code from.

Returns

TYPE	DESCRIPTION
System.Int32	Constructed hash code.

Class IPResolver

Helper methods that convert a connection string to an `IPEndPoint` used by the netcode.

Inheritance

System.Object
IPResolver

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public static class IPResolver
```

Methods

GetLocalAddress(Int32)

Get local IPv4 address other machines on the same network can use to connect to us. This can be used to create LAN connections.

Declaration

```
public static IPEndPoint GetLocalAddress(int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	port	Port to use.

Returns

TYPE	DESCRIPTION
IPEndPoint	Local IPv4 address or 127.0.0.1 if none found.

Resolve(String, Action<IPEndPoint, Exception>)

Perform an asynchronous DNS lookup if needed and create an `IPEndPoint`. All exceptions are thrown via the callback.

Host must be a valid IP address, followed by a colon and a port such as `192.168.12.43:80` or `127.0.0.1:44015`.

Declaration

```
public static void Resolve(string host, Action<IPEndPoint, Exception> callback)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	Hostname with port to resolve.
System.Action<IPEndPoint, System.Exception>	callback	Callback to invoke after DNS lookup completes.

Resolve(String, Int32, Action<IPEndPoint, Exception>)

Perform an asynchronous DNS lookup if needed and create an `IPEndPoint`. All exceptions are thrown via the callback.

Host must be a valid hostname without a port such as `192.168.12.43` or `superversus.com`.

Declaration

```
public static void Resolve(string host, int port, Action<IPEndPoint, Exception> callback)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	Hostname to resolve.
System.Int32	port	Port to use.
System.Action<IPEndPoint, System.Exception>	callback	Callback to invoke after DNS lookup completes.

ResolveAsync(String)

Perform an asynchronous DNS lookup if needed and create an `IPEndPoint`.

Host must be a valid hostname, followed by a colon and a port such as `192.168.12.43:80` or `superversus.com:44015`.

Declaration

```
public static Task<IPEndPoint> ResolveAsync(string host)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	Hostname with port to resolve.

Returns

TYPE	DESCRIPTION
Task<IPEndPoint>	A valid `IPEndPoint` with the provided IP address and port.

ResolveAsync(String, CancellationToken)

Perform an asynchronous DNS lookup if needed and create an `IPEndPoint`.

Host must be a valid hostname, followed by a colon and a port such as `192.168.12.43:80` or `superversus.com:44015`.

Declaration

```
public static Task<IPEndPoint> ResolveAsync(string host, CancellationToken token)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	Hostname with port to resolve.
CancellationToken	token	Cancellation token that can stop the DNS lookup before it is completed.

Returns

TYPE	DESCRIPTION
Task<IPEndPoint>	A valid `IPEndPoint` with the provided IP address and port.

ResolveAsync(String, Int32)

Perform an asynchronous DNS lookup if needed and create an `IPEndPoint`.

Host must be a valid hostname without a port such as `192.168.12.43` or `superversus.com`.

Declaration

```
public static Task<IPEndPoint> ResolveAsync(string host, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	Hostname to resolve.
System.Int32	port	Port to use.

Returns

TYPE	DESCRIPTION
Task<IPEndPoint>	A valid `IPEndPoint` with the provided IP address and port.

ResolveAsync(String, Int32, CancellationToken)

Perform an asynchronous DNS lookup if needed and create an `IPEndPoint`.

Host must be a valid hostname without a port such as `192.168.12.43` or `superversus.com`.

Declaration

```
public static Task<IPEndPoint> ResolveAsync(string host, int port, CancellationToken token)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	Hostname to resolve.
System.Int32	port	Port to use.
CancellationToken	token	Cancellation token that can stop the DNS lookup before it is completed.

Returns

TYPE	DESCRIPTION
Task<IPEndPoint>	A valid `IPEndPoint` with the provided IP address and port.

TryParse(String)

Try to parse the host as an IP address followed by a colon and a part. This method never throws any exceptions and returns immediately.

Host must be a valid IP address, followed by a colon and a port such as `192.168.12.43:80` or `127.0.0.1:44015`.

Declaration

```
public static IPEndPoint TryParse(string host)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	IP address with port to parse.

Returns

TYPE	DESCRIPTION
IPEndPoint	Parsed `IPEndPoint` or null if invalid.

TryParse(String, Int32)

Try to parse the host as an IP address. This method never throws any exceptions and returns immediately.

Host must contain a valid IP address such as `192.168.12.43` or `127.0.0.1`.

Declaration

```
public static IPEndPoint TryParse(string host, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	host	IP address to parse.
System.Int32	port	Port to use.

Returns

TYPE	DESCRIPTION
IPEndPoint	Parsed `IPEndPoint` or null if invalid.

Interface IWritable

Defines a serializable network payload.

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public interface IWritable
```

Methods

Write(Writer)

Serialize payload into the provided writer.

Declaration

```
void Write(Writer writer)
```

Parameters

TYPE	NAME	DESCRIPTION
Writer	writer	Writer to write to.

Class ObjectPool<T>

Object pool for reusing objects to avoid too many allocations.

Inheritance

System.Object
ObjectPool<T>

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public sealed class ObjectPool<T>
    where T : class
```

Type Parameters

NAME	DESCRIPTION
T	Object type.

Constructors

ObjectPool(Int32)

Create a new object pool.

Declaration

```
public ObjectPool(int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	count	Number of objects this pool can hold.

Methods

Rent()

Extract an object from this pool or return null.

Declaration

```
public T Rent()
```

Returns

TYPE	DESCRIPTION
T	Extracted object or null if none available.

Return(T)

Return an object back to this pool.

Declaration

```
public void Return(T obj)
```

Parameters

TYPE	NAME	DESCRIPTION
T	obj	Object to return.

Class Reader

Fast deserializer for network messages.

Inheritance

System.Object
Reader

Implements

System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public class Reader : IDisposable
```

Constructors

Reader(ArraySegment<Byte>)

Create a new reader from the provided array segment.

Declaration

```
public Reader(ArraySegment<byte> segment)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	segment	Array segment to read from.

Reader(Byte[], Int32, Int32)

Create a new reader from the provided array segment.

Declaration

```
public Reader(byte[] array, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to read from.

TYPE	NAME	DESCRIPTION
System.Int32	offset	Offset in the array to start reading from.
System.Int32	count	Number of bytes to read.

Properties

Available

Number of bytes still available to be read or 0 if the reader has been disposed.

Declaration

```
public int Available { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Disposed

True if reader has been disposed.

Declaration

```
public bool Disposed { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Last

Index of the last byte that is not included in the message.

Declaration

```
public int Last { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Position

Current position in the internal buffer or 0 if the reader has been disposed.

Declaration

```
public int Position { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Dispose()

Invalidate the underlying buffer when it gets used for something else. Calling this causes all future read operation to fail.

Declaration

```
public void Dispose()
```

ReadBoolean()

Read a single boolean (1 byte).

Declaration

```
public bool ReadBoolean()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Boolean value.

ReadBoolean(out Boolean, out Boolean, out Boolean, out Boolean, out Boolean, out Boolean, out Boolean, out Boolean)

Read 8 booleans (1 byte).

Declaration

```
public void ReadBoolean(out bool v0, out bool v1, out bool v2, out bool v3, out bool v4, out bool v5, out bool
v6, out bool v7)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	v0	First boolean value.
System.Boolean	v1	Second boolean value.
System.Boolean	v2	Third boolean value.
System.Boolean	v3	Fourth boolean value.

TYPE	NAME	DESCRIPTION
System.Boolean	v4	Fifth boolean value.
System.Boolean	v5	Sixth boolean value.
System.Boolean	v6	Seventh boolean value.
System.Boolean	v7	Eighth boolean value.

ReadByte()

Read byte (1 byte).

Declaration

```
public byte ReadByte()
```

Returns

TYPE	DESCRIPTION
System.Byte	Byte value.

ReadBytes(Byte[], Int32, Int32)

Read into an array segment.

Declaration

```
public void ReadBytes(byte[] array, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	array	Array to read to.
System.Int32	offset	Array offset to read to.
System.Int32	count	Number of bytes to read.

ReadChar()

Read a single character (2 bytes).

Declaration

```
public char ReadChar()
```

Returns

TYPE	DESCRIPTION
System.Char	Character value.

ReadDecimal()

Read decimal (16 bytes).

Declaration

```
public decimal ReadDecimal()
```

Returns

TYPE	DESCRIPTION
System.Decimal	Decimal value.

ReadDouble()

Read double (8 bytes).

Declaration

```
public double ReadDouble()
```

Returns

TYPE	DESCRIPTION
System.Double	Double value.

ReadEnum<T>()

Read enum (1, 2 or 4 bytes).

Number of bytes read is dependant on the underlying type the enum is backed by.

Declaration

```
public T ReadEnum<T>()
    where T : struct, IConvertible
```

Returns

TYPE	DESCRIPTION
T	Enum value.

Type Parameters

NAME	DESCRIPTION
T	Enum type.

ReadInt16()

Read short (2 bytes).

Declaration

```
public short ReadInt16()
```

Returns

TYPE	DESCRIPTION
System.Int16	Short value.

ReadInt32()

Read integer (4 bytes).

Declaration

```
public int ReadInt32()
```

Returns

TYPE	DESCRIPTION
System.Int32	Integer value.

ReadInt64()

Read long integer (8 bytes).

Declaration

```
public long ReadInt64()
```

Returns

TYPE	DESCRIPTION
System.Int64	Long integer value.

ReadSByte()

Read signed byte (1 byte).

Declaration

```
public sbyte ReadSByte()
```

Returns

TYPE	DESCRIPTION
System.SByte	Signed byte value.

ReadSingle()

Read float (4 bytes).

Declaration

```
public float ReadSingle()
```

Returns

TYPE	DESCRIPTION
System.Single	Float value.

ReadString()

Read 4 bytes length, then UTF8 encoded string.

Declaration

```
public string ReadString()
```

Returns

TYPE	DESCRIPTION
System.String	String value or null if length is negative.

ReadUInt16()

Read unsigned short (2 bytes).

Declaration

```
public ushort ReadUInt16()
```

Returns

TYPE	DESCRIPTION
System.UInt16	Unsigned short value.

ReadUint32()

Read unsigned integer (4 bytes).

Declaration

```
public uint ReadUint32()
```

Returns

TYPE	DESCRIPTION
System.UInt32	Unsigned integer value.

ReadUInt64()

Read unsigned long integer (8 bytes).

Declaration

```
public ulong ReadUInt64()
```

Returns

TYPE	DESCRIPTION
System.UInt64	Unsigned long integer value.

Skip(Int32)

Advance the read position without reading anything.

Declaration

```
public void Skip(int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	length	Number of bytes to skip for.

Implements

System.IDisposable

Class Serializer

Platform independent serialization of values in Big-endian (network byte order).

Inheritance

System.Object
Serializer

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public static class Serializer
```

Properties

Encoding

Character encoding to use when serializing strings.

Declaration

```
public static Encoding Encoding { get; }
```

Property Value

TYPE	DESCRIPTION
Encoding	

Methods

ReadDouble(Byte[], Int32)

Deserialize double (8 bytes) from the buffer.

Declaration

```
public static double ReadDouble(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.Double	Deserialized value.

ReadInt16(Byte[], Int32)

Deserialize short (2 bytes) from the buffer.

Declaration

```
public static short ReadInt16(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.Int16	Deserialized value.

ReadInt32(Byte[], Int32)

Deserialize int (4 bytes) from the buffer.

Declaration

```
public static int ReadInt32(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.Int32	Deserialized value.

ReadInt64(Byte[], Int32)

Deserialize long (8 bytes) from the buffer.

Declaration

```
public static long ReadInt64(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.Int64	Deserialized value.

ReadSingle(Byte[], Int32)

Deserialize float (4 bytes) from the buffer.

Declaration

```
public static float ReadSingle(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.Single	Deserialized value.

ReadUInt16(Byte[], Int32)

Deserialize ushort (2 bytes) from the buffer.

Declaration

```
public static ushort ReadUInt16(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.UInt16	Deserialized value.

ReadUInt32(Byte[], Int32)

Deserialize uint (4 bytes) from the buffer.

Declaration

```
public static uint ReadUInt32(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.UInt32	Deserialized value.

ReadUInt64(Byte[], Int32)

Deserialize ulong (8 bytes) from the buffer.

Declaration

```
public static ulong ReadUInt64(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to read from.

TYPE	NAME	DESCRIPTION
System.Int32	offset	Buffer offset to read from.

Returns

TYPE	DESCRIPTION
System.UInt64	Deserialized value.

Write16(Byte[], Int32, Int16)

Serialize short (2 bytes) to the buffer.

Declaration

```
public static void Write16(byte[] buffer, int offset, short value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.
System.Int32	offset	Buffer offset to write to.
System.Int16	value	Value to write.

Write16(Byte[], Int32, UInt16)

Serialize ushort (2 bytes) to the buffer.

Declaration

```
public static void Write16(byte[] buffer, int offset, ushort value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.
System.Int32	offset	Buffer offset to write to.
System.UInt16	value	Value to write.

Write32(Byte[], Int32, Int32)

Serialize int (4 bytes) to the buffer.

Declaration

```
public static void Write32(byte[] buffer, int offset, int value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.
System.Int32	offset	Buffer offset to write to.
System.Int32	value	Value to write.

Write32(Byte[], Int32, UInt32)

Serialize uint (4 bytes) to the buffer.

Declaration

```
public static void Write32(byte[] buffer, int offset, uint value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.
System.Int32	offset	Buffer offset to write to.
System.UInt32	value	Value to write.

Write64(Byte[], Int32, Int64)

Serialize long (8 bytes) to the buffer.

Declaration

```
public static void Write64(byte[] buffer, int offset, long value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.

TYPE	NAME	DESCRIPTION
System.Int32	offset	Buffer offset to write to.
System.Int64	value	Value to write.

Write64(Byte[], Int32, UInt64)

Serialize ulong (8 bytes) to the buffer.

Declaration

```
public static void Write64(byte[] buffer, int offset, ulong value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.
System.Int32	offset	Buffer offset to write to.
System.UInt64	value	Value to write.

WriteDouble(Byte[], Int32, Double)

Serialize double (8 bytes) to the buffer.

Declaration

```
public static void WriteDouble(byte[] buffer, int offset, double value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.
System.Int32	offset	Buffer offset to write to.
System.Double	value	Value to write.

WriteSingle(Byte[], Int32, Single)

Serialize float (4 bytes) to the buffer.

Declaration

```
public static void WriteSingle(byte[] buffer, int offset, float value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to write to.
System.Int32	offset	Buffer offset to write to.
System.Single	value	Value to write.

Class Writer

Fast serializer for network messages.

Inheritance

System.Object
Writer

Implements

System.IDisposable

Inherited Members

System.Object.ToString()
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.ReferenceEquals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()

Namespace: SuperNet.Util

Assembly: cs.temp.dll.dll

Syntax

```
public class Writer : IDisposable
```

Properties

Disposed

True if writer has been disposed.

Declaration

```
public bool Disposed { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Position

Current write position within the internal buffer or 0 if the writer has been disposed.

Declaration

```
public int Position { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Dispose()

Invalidate the underlying buffer when it gets used for something else. Calling this causes all future write operation to fail.

[illegible]

Declaration

```
public void Dispose()
```

Reset(Int32)

Manually set the write position.

Declaration

```
public void Reset(int position = 0)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	position	Position to set.

Skip(Int32)

Advance the write position without writing anything.

Declaration

```
public void Skip(int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	length	Number of bytes to skip for.

Write(Boolean)

Write a single boolean value (1 byte) to the writer.

Declaration

```
public void Write(bool value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	value	Boolean to write.

Write(Boolean, Boolean, Boolean, Boolean, Boolean, Boolean, Boolean, Boolean)

Write 8 boolean values (1 byte) to the writer.

Declaration

```
public void Write(bool v0, bool v1, bool v2, bool v3, bool v4, bool v5, bool v6, bool v7)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	v0	First boolean value.
System.Boolean	v1	Second boolean value.
System.Boolean	v2	Third boolean value.
System.Boolean	v3	Fourth boolean value.
System.Boolean	v4	Fifth boolean value.
System.Boolean	v5	Sixth boolean value.
System.Boolean	v6	Seventh boolean value.
System.Boolean	v7	Eighth boolean value.

Write(Byte)

Write a single byte (1 byte) to the writer.

Declaration

```
public void Write(byte value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte	value	Byte to write.

Write(Char)

Write a single character (2 bytes) to the writer.

Declaration

```
public void Write(char value)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
System.Char	value	Character to write.

Write(Decimal)

Write a decimal (16 bytes) to the writer.

Declaration

```
public void Write(decimal value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Decimal	value	Decimal to write.

Write(Double)

Write a double (8 bytes) to the writer.

Declaration

```
public void Write(double value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	value	Double to write.

Write(Int16)

Write a short (2 bytes) to the writer.

Declaration

```
public void Write(short value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	value	Short to write.

Write(Int32)

Write an integer (4 bytes) to the writer.

Declaration

```
public void Write(int value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	value	Integer to write.

Write(Int64)

Write a long (8 bytes) to the writer.

Declaration

```
public void Write(long value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64	value	Long to write.

Write(SByte)

Write a signed byte (1 byte) to the writer.

Declaration

```
public void Write(sbyte value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.SByte	value	Signed byte to write.

Write(Single)

Write a float (4 bytes) to the writer.

Declaration

```
public void Write(float value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	value	Float to write.

Write(String)

Write 4 bytes for length, then UTF8 encoded string.

Declaration

```
public void Write(string value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	value	String to write.

Write(UInt16)

Write an unsigned short (2 bytes) to the writer.

Declaration

```
public void Write(ushort value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	value	Unsigned short to write.

Write(UInt32)

Write an unsigned integer (4 bytes) to the writer.

Declaration

```
public void Write(uint value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt32	value	Unsigned integer to write.

Write(UInt64)

Write an unsigned long (8 bytes) to the writer.

Declaration

```
public void Write(ulong value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt64	value	Unsigned long to write.

WriteBytes(ArraySegment<Byte>)

Copy a segment of bytes to the writer.

Declaration

```
public void WriteBytes(ArraySegment<byte> segment)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ArraySegment<System.Byte>	segment	Segment to copy

WriteBytes(Byte[])

Copy an entire buffer to the writer.

Declaration

```
public void WriteBytes(byte[] buffer)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to copy.

WriteBytes(Byte[], Int32, Int32)

Copy a segment of bytes to the writer.

Declaration

```
public void WriteBytes(byte[] buffer, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	Buffer to copy from.
System.Int32	offset	Offset within the provided buffer.
System.Int32	count	Number of bytes to copy.

WriteEnum<T>(T)

Write an enum (1, 2 or 4 bytes) to the writer.

Number of bytes written is dependant on the underlying type enum is backed by.

Declaration

```
public void WriteEnum<T>(T value)
    where T : struct, IConvertible
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
T	value	Enum value.

Type Parameters

NAME	DESCRIPTION
T	Enum type.

Implements

System.IDisposable