Table of Contents

- Articles
  - Introduction
  - Multiplayer Rookie
  - Demo Tutorial: ExamplePawn
- Api Documentation
  - Assets.TinyBirdNet.Utils
    - PropertyInfoExtensions
  - FastMember
    - Member
    - MemberSet
    - ObjectAccessor
    - ObjectReader
    - TypeAccessor
    - TypeAccessor.RuntimeTypeAccessor
  - LiteNetLib
    - ConnectionRequest
    - ConnectionRequestResult
    - ConnectionState
    - DeliveryMethod
    - DisconnectInfo
    - DisconnectReason
    - EventBasedNatPunchListener
    - EventBasedNatPunchListener.OnNatIntroductionRequest
    - EventBasedNatPunchListener.OnNatIntroductionSuccess
    - EventBasedNetListener
    - EventBasedNetListener.OnConnectionRequest
    - EventBasedNetListener.OnNetworkError
    - EventBasedNetListener.OnNetworkLatencyUpdate
    - EventBasedNetListener.OnNetworkReceive
    - EventBasedNetListener.OnNetworkReceiveUnconnected
    - EventBasedNetListener.OnPeerConnected
    - EventBasedNetListener.OnPeerDisconnected
    - INatPunchListener

- INetEventListener
- INetLogger
- InvalidPacketException
- LocalAddrType
- NatPunchModule
- NetConstants
- NetDebug
- NetEndPoint
- NetManager
- NetPeer
- NetStatistics
- NetUtils
- TooBigPacketException
- UnconnectedMessageType

LiteNetLib.Utils
- FastBitConverter
- INetSerializable
- InvalidTypeException
- NetDataReader
- NetDataWriter
- NetPacketProcessor
- NetPacketProcessor.SubscrieDelegate
- NetSerializer
- ParseException

TinyBirdNet
- ITinyNetInstanceID
- ITinyNetObject
- LogFilter
- NetworkScenePostProcess
- RPCCallers
- RPCDelegate
- RPCMethodInfo
- RPCTarget
- SpawnDelegate
- TinyNetBehaviour
- TinyNetClient
- TinyNetConnection

- TinyNetGameManager
- TinyNetGameManagerEditor
- TinyNetIdentity
- TinyNetLogLevel
- TinyNetMessageHandlers
- TinyNetPlayerController
- TinyNetPropertyAccessor<T>
- TinyNetReflector
- TinyNetRPC
- TinyNetScene
- TinyNetServer
- TinyNetSimpleMenu
- TinyNetStateSyncer
- TinyNetSyncVar
- UnSpawnDelegate

TinyBirdNet.Messaging

- ITinyNetMessage
- TinyNetAddPlayerMessage
- TinyNetBoolMessage
- TinyNetByteMessage
- TinyNetClientAuthorityMessage
- TinyNetEmptyMessage
- TinyNetFloatMessage
- TinyNetInputMessage
- TinyNetIntegerMessage
- TinyNetMessageDelegate
- TinyNetMessageReader
- TinyNetMsgType
- TinyNetNotReadyMessage
- TinyNetObjectDestroyMessage
- TinyNetObjectHideMessage
- TinyNetObjectSpawnFinishedMessage
- TinyNetObjectSpawnMessage
- TinyNetObjectSpawnSceneMessage
- TinyNetObjectStateUpdate
- TinyNetOwnerMessage
- TinyNetReadyMessage

- TinyNetRemovePlayerMessage
- TinyNetRequestAddPlayerMessage
- TinyNetRequestRemovePlayerMessage
- TinyNetRPCMessage
- TinyNetShortMessage
- TinyNetStringMessage

TinyBirdUtils

- TinyLogger

Welcome to the TinyBirdNet-Unity wiki!

Table of Contests

- [[Home|Home]]
  - What is TinyBirdNet?
  - Why use TinyBirdNet?

- [[Multiplayer Rookie|Multiplayer-Rookie]]
  - [[Types of Networked Games|Multiplayer-Rookie#types-of-networked-games]]
  - [[How do Network?|Multiplayer-Rookie#how-do-network]]
  - [[The Tricks|Multiplayer-Rookie#the-tricks]]
  - [[The Data|Multiplayer-Rookie#the-data]]
  - [[Client/Server Architecture|Multiplayer-Rookie#clientserver-architecture]]

- [[Demo Tutorial: ExamplePawn|Demo-Tutorial:-ExamplePawn]]

What is TinyBirdNet?

It's a high-level api built on top of LiteNetLib, this mean you can add network multiplayer to your games without having to directly touch difficult stuff like serialization, conversion to bits, message handling and others.

Why use TinyBirdNet?

Made as a substitute for UNet, TinyBirdNet offers a lot of familiar stuff like attributes for variable syncing, rpc/command calls and similar workflow, easing the migration of projects from Unet to it.

In relation to other similar products, TinyBirdNet offers full access to code, nothing is hidden from you, and you are free to modify the core functionality to better fit your project.

Besides being free for noncommercial use, there is no additional fee in the form of CCU, Bandwidth or others.

This page is directed to people who have never done networking multiplayer or those who are very new to it.
Basic concepts will be explained, if you feel you already have a good grasp of it, please proceed to the next tutorials.

Table of Contests

- Types of Networked Games
- How do Network?
- The Tricks
- The Data
- Client/Server Architecture

Types of Networked Games

TinyBirdNet uses a **Client/Server** archetype, this mean that each client is connected to a single server. So if a client wants to send information to another client it will first need to send it to the server, who will redirect the information.

While this may seem cumbersome, having a connection to each and every other client is in most cases not what you need and extremely difficult.
For imagine that you need to make sure all clients are able to connect to each other, have their ports open, handle any connection error, etc...

A good read about different types of networked games can be found here:
https://gafferongames.com/post/what_every_programmer_needs_to_know_about_game_networking/

How do Network?

So how exactly is networking multiplayer implemented in games? Is the video streamed to the players? Do we just keep sending every possible information to everyone? Is it the cloud magic?

At a basic level most networked games is mirror and smokes, you normally have huge amounts of data needed to be synced really fast between computers with different speeds. So unless you have a game where time is irrelevant, like a turn based game with timers, you need to up on some tricks.

So, the first issue when making a game work across the network is Latency. Latency, sometimes mentioned as Ping, is the time interval between the stimulation and response. In this case, this means the time it takes for a message to be sent from one computer and it being received in another.
It might also be referred to the time it takes for a message to be sent, plus the time it takes for another message to be sent back. Although this is often referred to as "round-trip delay time", RTD or RTT (round-trip time) to avoid confusion.

Some of you may think, why is Latency more important than Bandwidth? If you don't know yet, Network Bandwidth normally defined as the amount of data able to be sent across the network in a given period of time.
As said before networked games uses a lot of tricks to work, if you are too young you might not known that before we all had internet speed ranging around 56kbps (Around 7 kilobyte per second), while today it's common to have internet speed surpassing 1 megabyte per second (Around 142 times more information per second).

Games then had to deal with sending extremely small packets of data, not much information could be exchanged. Although you might see later that 7kilobytes is a lot of info, this was in excellent conditions, and you couldn't just hug all the resources for yourself.

The Tricks

In a nutshell: **Sending the minimum amount of data necessary to interpret the current game state.**

There are a variety of ways of doing this, from most simply not sending data that the client don't need to known about (Like not sending data of objects the client can't see nor interact, If it's irrelevant to known, why known it?), packing all data together (We will see in the next part about it), to making predictions about stuff (If someone has been walking forward last update, it is more

likely he will keep walking forward until we are certain he has stopped or changed directions).
This can get complicated really fast, but you can take your time learning it.

Some common tricks may be found at this article: https://en.wikipedia.org/wiki/Lag#Solutions_and_lag_compensation

The Data

TinyBirdNet uses UDP (User Datagram Protocol) to communicate, UDP uses a little less data than TCP (Transmission Control Protocol) to send information, but it has less safe handling mechanisms, don't worry tho as the LiteNetLib handles that most graciously!

If you were to send an empty message in UDP, meaning no data is actually sent, the packet would contain at minimum 28bytes(IPv4) or 48bytes(IPv6). This is because the packet needs to contain data about the destination and content, we call those the Header of the packet.

Remember that Bandwidth is measured by amount of data per second, but one second would mean you have a 1000ms delay, everything happening one second late for clients. Data is constantly sent across the network instead of being sent every second, this is made by fragmenting packets and this fragmentation may have a huge impact on lower Bandwidth connections.

You can read more about how Bandwidth and Packet size may affect Latency here: https://books.google.com.br/books?id=MLxfy6W8bxgC&lpg=PA193&ots=9FaVhqdM2m&dq=56kbps%20packet%20size&pg=PA194#v=onepage&q&f=false

Client/Server Architecture

TinyBirdNet operates on the basis that there is one Server that every Client connects to. On the included demo any Host will also be a Listen Server, meaning they are a Client and a Server at the same time. (A Client connected to their own computer, which is also a Host)

In most cases one would adopt the Server Authority architecture, meaning the Server is the correct version of the game and any important information is validated by it, while Clients try to keep a simulation as close as possible to the Server.
This Server Authority is not provided by TinyBirdNet, it's upon you to implement it or not on your game.

Welcome to the first part of the tutorial series where you will learn how to make a very basic game using TinyBirdNet.
For that I will be using the Demo game included, and will teach you what every part does and why it is necessary.

What we gonna need?

We will be using our own derived classes of:

- [[TinyNetGameManager|TinyNetGameManager]]
- [[TinyNetPlayerController|TinyNetPlayerController]]
- [[TinyNetBehaviour|TinyNetBehaviour]]

In addition to a GameManager and SpawnPointManager, not related to networking tho.

You might be wondering if that is enough, but worry not as TinyBirdNet will take care of most things for you. From syncing scene changing, serialization/deserialization, authentication, you just worry about your game logic!

Making our character

I would like to start with the cubes the players will move around, since the `ExamplePawn` is the script with less direct references to others.

```
    public class ExamplePawn : TinyNetBehaviour {
```

A `TinyNetBehaviour` is a `MonoBehaviour` who implements the interface `ITinyNetObject`, in addition, TinyBirdNet handles it's spawning, serialization, rpc, and mostly anything you need to create a new instance of it in a multiplayer game and have it automatically synced.

Most of the important network variables are declared here:

```
    string _playerName;
    [TinyNetSyncVar]
    public string PlayerName { get { return _playerName; } set { _playerName = value; } }

    Vector3 _networkPosition;

    [TinyNetSyncVar]
    float xPos { get { return _networkPosition.x; } set { _networkPosition.x = value; } }
    [TinyNetSyncVar]
    float zPos { get { return _networkPosition.z; } set { _networkPosition.z = value; } }
    [TinyNetSyncVar]
    byte netDir { get; set; }
```

Why so many properties you ask? Well, the `[TinyNetSyncVar]` attribute only works with properties.
Why again? Mostly a whim I guess, but manly because that way you are assured to receive an event every time it is changed, and you get both the old and the new value assigned to it.

The `[TinyNetSyncVar]` attribute means that property will be automatically sent from the server to all clients whenever it is detected to be dirty (has changed) between network frames.
This mean that by just setting it, all clients will automatically sync their values.

`PlayerName` is used to display that text on top of the players, each client is responsible for designating their own name, but since replication can't be done client to client we first send it to the server and it is synced to all clients.

'xPos' and 'zPos' is just the current player's position, we route the property to a vector3 just to get some sugar from Unity.

Finally, `netDir` is a numerical representation of the player's facing direction. 1 is top and 4 is left, goes clockwise. (0 means there was an error)

Next about the methods, and again, I will be skipping ones not directly related to network.

```
    private void Start() {
        xPos = transform.position.x;
        zPos = transform.position.z;
    }
```

This one just makes sure that as soon as this object is fully created, it's network position will be set.

```
    public override void OnStartServer() {
        base.OnStartServer();

        timeForNextShoot = Time.time + 0.3f;
    }
```

OnStartServer() Is called on a Server when an object is network created, it is triggered after OnNetworkCreate and before OnStartClient() if the Server is also a Client.

```
    public override void OnStartAuthority() {
        base.OnStartAuthority();

        controller = TinyNetClient.instance.connToHost.GetPlayerController<ExamplePlayerController>
(ownerPlayerControllerId);
        controller.GetPawn(this);

        cameraTransform = GameObject.FindGameObjectWithTag("MainCamera").transform;
    }
```

This one is called when someone acquires Authority of this object. Authority is a fairly abstract concept tho, it don't really do anything besides being a marker for special privileges you might want to give to something.
In this case, it represents the client that controls that player.

Here we get our Player Controller, which I will explain later, by means of using our connection to the host and our `ownerPlayerControllerId` field that is given to us by the server. Since we are using Authority here to mean the player you control, we also take the opportunity to grab the Main Camera and makes it follow us.

```
    public override void OnStartClient() {
        base.OnStartClient();

        playerText.text = PlayerName;
    }
```

OnStartClient(), as you may have correctly guessed, is called on all Clients that receive this Object from the network. It is also called after all variables have been synced.
Here we are only displaying the player's name on our text mesh.

```
    public override void OnNetworkDestroy() {
        base.OnNetworkDestroy();

        if (hasAuthority) {
            controller.LosePawn();
            controller = null;
        }
    }
```

Called when an object is removed from the network simulation, at this one we just make the Player Controller known the player has died.

```
    private void FixedUpdate() {
        if (!hasAuthority) {
            Vector3 pos = transform.position;
            Vector3 result = Vector3.MoveTowards(pos, _networkPosition, movementSpeed * Time.fixedDeltaTime);

            float dist = (result - _networkPosition).sqrMagnitude;
            if (dist <= 0.1f || dist >= movespeedPow) {
                result = _networkPosition;
            }

            FaceDir(netDir);

            transform.position = result;
        } else {
            cameraTransform.position = new Vector3(transform.position.x, 10.0f, transform.position.z - 6f);
        }
    }
```

This one have two modes, if we are not the owner of that player we take the information we received from the server and try to update our simulation of it.
Frankly this was really done poorly here as this was just the minimum example needed to work. You can see how the bullets don't really align when someone else is moving and shooting.
I recommend reading about interpolation or any other lag compensation methods.

If we are the owner tho, we just update the camera position, as everything else is controlled by the Player Controller.

```
    public void Killed() {
        TinyNetServer.instance.DestroyObject(gameObject);
    }
```

Called when our player is hit by an enemy bullet, it asks the Server to remove it from the network.
There are no safety checks here to see we are indeed the erver, but in our case we want errors to be thrown cos this is not to be called by any Client!

Next I want to explain the RPC (Remote Procedure Call) for the shooting mechanics. Firstly, RPC are methods which are called but not resolved on the same machine. This mean we can call methods at the Client which are executed on the Server and other combinations.

Sadly, this part ended up being a little bothersome since I tried to stay away from Weaving the Unet used, and reflection could only get me so far on it.
First we will declare our shoot method:

```
    [TinyNetRPC(RPCTarget.Server, RPCCallers.ClientOwner)]
    void ServerShoot(float xPos, float zPos, byte dir) {
        if (!isServer) {
            rpcRecycleWriter.Reset();
            rpcRecycleWriter.Put(xPos);
            rpcRecycleWriter.Put(zPos);
            rpcRecycleWriter.Put(dir);

            SendRPC(rpcRecycleWriter, "ServerShoot");
            return;
        }

        ExampleBullet bullet = Instantiate(bulletPrefab, bulletSpawnPosition.position, 
transform.rotation).GetComponent<ExampleBullet>();
        bullet.ownerNetworkId = NetIdentity.NetworkID;
        bullet.direction = dir;
        switch (dir) {
            case 1:
                bullet.transform.rotation = Quaternion.Euler(new Vector3(0f, 0f, 0f));
                break;
            //Right
            case 2:
                bullet.transform.rotation = Quaternion.Euler(new Vector3(0f, 90f, 0f));
                break;
            //Down
            case 3:
                bullet.transform.rotation = Quaternion.Euler(new Vector3(0f, 180f, 0f));
                break;
            //Left
            case 4:
                bullet.transform.rotation = Quaternion.Euler(new Vector3(0f, 270f, 0f));
                break;
        }

        TinyNetServer.instance.SpawnObject(bullet.gameObject);
    }
```

The `[TinyNetRPC(RPCTarget.Server, RPCCallers.ClientOwner)]` declares that this method can only be called by the Client that owns (has Authority on) this object, and it will be executed at the Server.

Then we do a manual check to see if we are not the Server already, since in this game the Server is always a Client too, it could mean the owner of this object is already the target Server.
If we are not the Server, gather the parameters of this method at a reusable `NetDataWritter` that all `TinyNetBehaviour` have access to, and send an RPC with it, ending the method.

If we are the Server, by means of receiving the RPC or just initially being the owner, we proceed with the normal shooting. Create a new bullet, spawn it, done.

Now, going back to one network method we skipped...

```
    public override void OnNetworkCreate() {
        base.OnNetworkCreate();

        RegisterRPCDelegate(ServerShootReceive, "ServerShoot");
    }
```

At this one, called when the object is added to the network, but before it have received any data, we register the `ServerShootReceive` method with the string id "ServerShoot".
This mean that if we ever receive an RPC with the "ServerShoot" id, we will call the method registered for it.

And finally:

```
    void ServerShootReceive(NetDataReader reader) {
        ServerShoot(reader.GetFloat(), reader.GetFloat(), reader.GetByte());
    }
```

The method registered, basically receives the data from the RPC and routes it to the original `ServerShoot` method.

A little cumbersome, if there is the opportunity to improve upon it I will do so, but for now it shall suffice.

At the next part we will take a look at the `ExamplePlayerController` script!

Namespace Assets.TinyBirdNet.Utils

Classes

PropertyInfoExtensions

Class PropertyInfoExtensions

Inheritance

System.Object
PropertyInfoExtensions

Inherited Members

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

Namespace: Assets.TinyBirdNet.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public static class PropertyInfoExtensions
```

Methods

GetValueGetter<T>(PropertyInfo)

Declaration

```
public static Func<T, object> GetValueGetter<T>(this PropertyInfo propertyInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Reflection.PropertyInfo	propertyInfo	

Returns

TYPE	DESCRIPTION
System.Func<T, System.Object>	

Type Parameters

NAME	DESCRIPTION
T	

GetValueSetter<T>(PropertyInfo)

Declaration

```
public static Action<T, object> GetValueSetter<T>(this PropertyInfo propertyInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Reflection.PropertyInfo	propertyInfo	

Returns

TYPE	DESCRIPTION
System.Action<T, System.Object>	

Type Parameters

NAME	DESCRIPTION
T	

Namespace FastMember

Classes

Member

Represents an abstracted view of an individual member defined for a type

MemberSet

Represents an abstracted view of the members defined for a type

ObjectAccessor

Represents an individual object, allowing access to members by-name

ObjectReader

Provides a means of reading a sequence of objects as a data-reader, for example for use with SqlBulkCopy or other data-base oriented code

TypeAccessor

Provides by-name member-access to objects of a given type

TypeAccessor.RuntimeTypeAccessor

A TypeAccessor based on a Type implementation, with available member metadata

Class Member

Represents an abstracted view of an individual member defined for a type

Inheritance

System.Object
Member

Inherited Members

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

Namespace: FastMember

Assembly: FastMember.dll

Syntax

```
public sealed class Member
```

Properties

Name

The name of this member

Declaration

```
public string Name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Type

The type of value stored in this member

Declaration

```
public Type Type { get; }
```

Property Value

TYPE	DESCRIPTION
System.Type	

Methods

IsDefined(Type)

Is the attribute specified defined on this type

Declaration

```
public bool IsDefined(Type attributeType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	attributeType	

Returns

TYPE	DESCRIPTION
System.Boolean	

Class MemberSet

Represents an abstracted view of the members defined for a type

Inheritance

System.Object
MemberSet

Implements

System.Collections.Generic.IList<Member>
System.Collections.Generic.ICollection<Member>
System.Collections.Generic.IEnumerable<Member>
System.Collections.IEnumerable

Inherited Members

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

Namespace: FastMember

Assembly: FastMember.dll

Syntax

```
public sealed class MemberSet : IList<Member>, ICollection<Member>, IEnumerable<Member>, IEnumerable
```

Properties

Count

The number of members defined for this type

Declaration

```
public int Count { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Item[Int32]

Get a member by index

Declaration

```
public Member this[int index] { get; }
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	

Property Value

TYPE	DESCRIPTION
Member	

Methods

GetEnumerator()

Return a sequence of all defined members

Declaration

```
public IEnumerator<Member> GetEnumerator()
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.IEnumerator<Member>	

Explicit Interface Implementations

ICollection<Member>.Add(Member)

Declaration

```
void ICollection<Member>.Add(Member item)
```

Parameters

TYPE	NAME	DESCRIPTION
Member	item	

ICollection<Member>.Clear()

Declaration

```
void ICollection<Member>.Clear()
```

ICollection<Member>.Contains(Member)

Declaration

```
bool ICollection<Member>.Contains(Member item)
```

Parameters

TYPE	NAME	DESCRIPTION
Member	item	

Returns

TYPE	DESCRIPTION
System.Boolean	

ICollection<Member>.CopyTo(Member[], Int32)

Declaration

```
void ICollection<Member>.CopyTo(Member[] array, int arrayIndex)
```

Parameters

TYPE	NAME	DESCRIPTION
Member[]	array	
System.Int32	arrayIndex	

ICollection<Member>.IsReadOnly

Declaration

```
bool ICollection<Member>.IsReadOnly { get; }
```

Returns

TYPE	DESCRIPTION
System.Boolean	

ICollection<Member>.Remove(Member)

Declaration

```
bool ICollection<Member>.Remove(Member item)
```

Parameters

TYPE	NAME	DESCRIPTION
Member	item	

Returns

TYPE	DESCRIPTION
System.Boolean	

IList<Member>.IndexOf(Member)

Declaration

```
int IList<Member>.IndexOf(Member member)
```

Parameters

TYPE	NAME	DESCRIPTION
Member	member	

Returns

TYPE	DESCRIPTION
System.Int32	

IList<Member>.Insert(Int32, Member)

Declaration

```
void IList<Member>.Insert(int index, Member item)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	
Member	item	

IList<Member>.Item[Int32]

Declaration

```
Member IList<Member>.this[int index] { get; set; }
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	

Returns

TYPE	DESCRIPTION
Member	

IList<Member>.RemoveAt(Int32)

Declaration

```
void IList<Member>.RemoveAt(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	

IEnumerable.GetEnumerator()

Declaration

```
IEnumerator IEnumerable.GetEnumerator()
```

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	

Implements

System.Collections.Generic.IList<T>
System.Collections.Generic.ICollection<T>
System.Collections.Generic.IEnumerable<T>
System.Collections.IEnumerable

Class ObjectAccessor

Represents an individual object, allowing access to members by-name

Inheritance

System.Object
ObjectAccessor

Inherited Members

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

Namespace: FastMember

Assembly: FastMember.dll

Syntax

```
public abstract class ObjectAccessor
```

Properties

Item[String]

Get or Set the value of a named member for the underlying object

Declaration

```
public abstract object this[string name] { get; set; }
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	

Property Value

TYPE	DESCRIPTION
System.Object	

Target

The object represented by this instance

Declaration

```
public abstract object Target { get; }
```

Property Value

TYPE	DESCRIPTION
System.Object	

Methods

Create(Object)

Wraps an individual object, allowing by-name access to that instance

Declaration

```
public static ObjectAccessor Create(object target)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	target	

Returns

TYPE	DESCRIPTION
ObjectAccessor	

Create(Object, Boolean)

Wraps an individual object, allowing by-name access to that instance

Declaration

```
public static ObjectAccessor Create(object target, bool allowNonPublicAccessors)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	target	
System.Boolean	allowNonPublicAccessors	

Returns

TYPE	DESCRIPTION
ObjectAccessor	

Equals(Object)

Use the target types definition of equality

Declaration

```
public override bool Equals(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Object.Equals(System.Object)

GetHashCode()

Obtain the hash of the target object

Declaration

```
public override int GetHashCode()
```

Returns

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Object.GetHashCode()

ToString()

Use the target's definition of a string representation

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Object.ToString()

Class ObjectReader

Provides a means of reading a sequence of objects as a data-reader, for example for use with SqlBulkCopy or other data-base oriented code

Inheritance

System.Object
System.MarshalByRefObject
System.Data.Common.DbDataReader
ObjectReader

Implements

System.Collections.IEnumerable
System.Data.IDataReader
System.IDisposable
System.Data.IDataRecord

Inherited Members

System.Data.Common.DbDataReader.System.Data.IDataRecord.GetData(System.Int32)
System.Data.Common.DbDataReader.VisibleFieldCount
System.MarshalByRefObject.CreateObjRef(System.Type)
System.MarshalByRefObject.GetLifetimeService()
System.MarshalByRefObject.InitializeLifetimeService()
System.MarshalByRefObject.MemberwiseClone(System.Boolean)
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ToString()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: FastMember
Assembly: FastMember.dll

Syntax

```
public class ObjectReader : DbDataReader, IEnumerable, IDataReader, IDisposable, IDataRecord
```

Constructors

ObjectReader(Type, IEnumerable, String[])

Creates a new ObjectReader instance for reading the supplied data

Declaration

```
public ObjectReader(Type type, IEnumerable source, params string[] members)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	The expected Type of the information to be read
System.Collections.IEnumerable	source	The sequence of objects to represent

TYPE	NAME	DESCRIPTION
System.String[]	members	The members that should be exposed to the reader

Properties

Depth

Declaration

```
public override int Depth { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Data.Common.DbDataReader.Depth

FieldCount

Declaration

```
public override int FieldCount { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Data.Common.DbDataReader.FieldCount

HasRows

Declaration

```
public override bool HasRows { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Data.Common.DbDataReader.HasRows

IsClosed

Declaration

```
public override bool IsClosed { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Data.Common.DbDataReader.IsClosed

Item[Int32]

Gets the value of the current object in the member specified

Declaration

```
public override object this[int i] { get; }
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Property Value

TYPE	DESCRIPTION
System.Object	

Overrides

System.Data.Common.DbDataReader.Item[System.Int32]

Item[String]

Declaration

```
public override object this[string name] { get; }
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	

Property Value

TYPE	DESCRIPTION
System.Object	

Overrides

System.Data.Common.DbDataReader.Item[System.String]

RecordsAffected

Declaration

```
public override int RecordsAffected { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Data.Common.DbDataReader.RecordsAffected

Methods

Close()

Declaration

```
public override void Close()
```

Overrides

System.Data.Common.DbDataReader.Close()

Create<T>(IEnumerable<T>, String[])

Creates a new ObjectReader instance for reading the supplied data

Declaration

```
public static ObjectReader Create<T>(IEnumerable<T> source, params string[] members)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.IEnumerable<T>	source	The sequence of objects to represent
System.String[]	members	The members that should be exposed to the reader

Returns

TYPE	DESCRIPTION
ObjectReader	

Type Parameters

NAME	DESCRIPTION
T	

Dispose(Boolean)

Declaration

```
protected override void Dispose(bool disposing)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	disposing	

Overrides

System.Data.Common.DbDataReader.Dispose(System.Boolean)

GetBoolean(Int32)

Declaration

```
public override bool GetBoolean(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Data.Common.DbDataReader.GetBoolean(System.Int32)

GetByte(Int32)

Declaration

```
public override byte GetByte(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Byte	

Overrides

System.Data.Common.DbDataReader.GetByte(System.Int32)

GetBytes(Int32, Int64, Byte[], Int32, Int32)

Declaration

```
public override long GetBytes(int i, long fieldOffset, byte[] buffer, int bufferoffset, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	
System.Int64	fieldOffset	
System.Byte[]	buffer	

TYPE	NAME	DESCRIPTION
System.Int32	bufferoffset	
System.Int32	length	

Returns

TYPE	DESCRIPTION
System.Int64	

Overrides

System.Data.Common.DbDataReader.GetBytes(System.Int32, System.Int64, System.Byte[], System.Int32, System.Int32)

GetChar(Int32)

Declaration

```
public override char GetChar(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Char	

Overrides

System.Data.Common.DbDataReader.GetChar(System.Int32)

GetChars(Int32, Int64, Char[], Int32, Int32)

Declaration

```
public override long GetChars(int i, long fieldoffset, char[] buffer, int bufferoffset, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	
System.Int64	fieldoffset	
System.Char[]	buffer	
System.Int32	bufferoffset	
System.Int32	length	

Returns

TYPE	DESCRIPTION
System.Int64	

Overrides

System.Data.Common.DbDataReader.GetChars(System.Int32, System.Int64, System.Char[], System.Int32, System.Int32)

GetDataTypeName(Int32)

Declaration

```
public override string GetDataTypeName(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Data.Common.DbDataReader.GetDataTypeName(System.Int32)

GetDateTime(Int32)

Declaration

```
public override DateTime GetDateTime(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.DateTime	

Overrides

System.Data.Common.DbDataReader.GetDateTime(System.Int32)

GetDbDataReader(Int32)

Declaration

```
protected override DbDataReader GetDbDataReader(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Data.Common.DbDataReader	

Overrides

System.Data.Common.DbDataReader.GetDbDataReader(System.Int32)

GetDecimal(Int32)

Declaration

```
public override decimal GetDecimal(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Decimal	

Overrides

System.Data.Common.DbDataReader.GetDecimal(System.Int32)

GetDouble(Int32)

Declaration

```
public override double GetDouble(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Double	

Overrides

System.Data.Common.DbDataReader.GetDouble(System.Int32)

GetEnumerator()

Declaration

```
public override IEnumerator GetEnumerator()
```

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	

Overrides

System.Data.Common.DbDataReader.GetEnumerator()

GetFieldType(Int32)

Declaration

```
public override Type GetFieldType(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Type	

Overrides

System.Data.Common.DbDataReader.GetFieldType(System.Int32)

GetFloat(Int32)

Declaration

```
public override float GetFloat(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Single	

Overrides

System.Data.Common.DbDataReader.GetFloat(System.Int32)

GetGuid(Int32)

Declaration

```
public override Guid GetGuid(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Guid	

Overrides

System.Data.Common.DbDataReader.GetGuid(System.Int32)

GetInt16(Int32)

Declaration

```
public override short GetInt16(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Int16	

Overrides

System.Data.Common.DbDataReader.GetInt16(System.Int32)

GetInt32(Int32)

Declaration

```
public override int GetInt32(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Data.Common.DbDataReader.GetInt32(System.Int32)

GetInt64(Int32)

Declaration

```
public override long GetInt64(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Int64	

Overrides

System.Data.Common.DbDataReader.GetInt64(System.Int32)

GetName(Int32)

Declaration

```
public override string GetName(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Data.Common.DbDataReader.GetName(System.Int32)

GetOrdinal(String)

Declaration

```
public override int GetOrdinal(string name)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	name	

Returns

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Data.Common.DbDataReader.GetOrdinal(System.String)

GetSchemaTable()

Declaration

```
public override DataTable GetSchemaTable()
```

Returns

TYPE	DESCRIPTION
System.Data.DataTable	

Overrides

System.Data.Common.DbDataReader.GetSchemaTable()

GetString(Int32)

Declaration

```
public override string GetString(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Data.Common.DbDataReader.GetString(System.Int32)

GetValue(Int32)

Declaration

```
public override object GetValue(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Object	

Overrides

System.Data.Common.DbDataReader.GetValue(System.Int32)

GetValues(Object[])

Declaration

```
public override int GetValues(object[] values)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object[]	values	

Returns

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Data.Common.DbDataReader.GetValues(System.Object[])

IsDBNull(Int32)

Declaration

```
public override bool IsDBNull(int i)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	i	

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Data.Common.DbDataReader.IsDBNull(System.Int32)

NextResult()

Declaration

```
public override bool NextResult()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Data.Common.DbDataReader.NextResult()

Read()

Declaration

```
public override bool Read()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Data.Common.DbDataReader.Read()

Implements

System.Collections.IEnumerable
System.Data.IDataReader
System.IDisposable
System.Data.IDataRecord

Class TypeAccessor

Provides by-name member-access to objects of a given type

Inheritance

System.Object
TypeAccessor
TypeAccessor.RuntimeTypeAccessor

Inherited Members

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

Namespace: FastMember

Assembly: FastMember.dll

Syntax

```
public abstract class TypeAccessor
```

Properties

CreateNewSupported

Does this type support new instances via a parameterless constructor?

Declaration

```
public virtual bool CreateNewSupported { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

GetMembersSupported

Can this type be queried for member availability?

Declaration

```
public virtual bool GetMembersSupported { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Item[Object, String]

Get or set the value of a named member on the target instance

Declaration

```
public abstract object this[object target, string name] { get; set; }
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	target	
System.String	name	

Property Value

TYPE	DESCRIPTION
System.Object	

Methods

Create(Type)

Provides a type-specific accessor, allowing by-name access for all objects of that type

Declaration

```
public static TypeAccessor Create(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	

Returns

TYPE	DESCRIPTION
TypeAccessor	

Remarks

The accessor is cached internally; a pre-existing accessor may be returned

Create(Type, Boolean)

Provides a type-specific accessor, allowing by-name access for all objects of that type

Declaration

```
public static TypeAccessor Create(Type type, bool allowNonPublicAccessors)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	
System.Boolean	allowNonPublicAccessors	

Returns

TYPE	DESCRIPTION
TypeAccessor	

Remarks

The accessor is cached internally; a pre-existing accessor may be returned

CreateNew()

Create a new instance of this type

Declaration

```
public virtual object CreateNew()
```

Returns

TYPE	DESCRIPTION
System.Object	

GetMembers()

Query the members available for this type

Declaration

```
public virtual MemberSet GetMembers()
```

Returns

TYPE	DESCRIPTION
MemberSet	

Class TypeAccessor.RuntimeTypeAccessor

A TypeAccessor based on a Type implementation, with available member metadata

Inheritance

System.Object
TypeAccessor
TypeAccessor.RuntimeTypeAccessor

Inherited Members

TypeAccessor.CreateNewSupported
TypeAccessor.CreateNew()
TypeAccessor.Create(Type)
TypeAccessor.Create(Type, Boolean)
TypeAccessor.Item[Object, String]
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ToString()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: FastMember

Assembly: FastMember.dll

Syntax

```
protected abstract class RuntimeTypeAccessor : TypeAccessor
```

Properties

GetMembersSupported

Can this type be queried for member availability?

Declaration

```
public override bool GetMembersSupported { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Overrides

TypeAccessor.GetMembersSupported

Type

Returns the Type represented by this accessor

Declaration

```
protected abstract Type Type { get; }
```

Property Value

TYPE	DESCRIPTION
System.Type	

Methods

GetMembers()

Query the members available for this type

Declaration

```
public override MemberSet GetMembers()
```

Returns

TYPE	DESCRIPTION
MemberSet	

Overrides

TypeAccessor.GetMembers()

Namespace LiteNetLib

Classes

ConnectionRequest

EventBasedNatPunchListener

EventBasedNetListener

InvalidPacketException

NatPunchModule

Module for UDP NAT Hole punching operations. Can be accessed from NetManager

NetConstants

Network constants. Can be tuned from sources for your purposes.

NetDebug

Static class for defining your own LiteNetLib logger instead of Console.WriteLine or Debug.Log if compiled with UNITY flag

NetEndPoint

Network End Point. Contains ip address and port

NetManager

Main class for all network operations. Can be used as client and/or server.

NetPeer

Network peer. Main purpose is sending messages to specific peer.

NetStatistics

NetUtils

Some specific network utilities

TooBigPacketException

Structs

DisconnectInfo

Additional information about disconnection

Interfaces

INatPunchListener

INetEventListener

INetLogger

Interface to implement for your own logger

Enums

ConnectionRequestResult

ConnectionState

Peer connection state

DeliveryMethod

Sending method type

DisconnectReason

Disconnect reason that you receive in OnPeerDisconnected event

LocalAddrType

Address type that you want to receive from NetUtils.GetLocalIp method

UnconnectedMessageType

Type of message that you receive in OnNetworkReceiveUnconnected event

Delegates

EventBasedNatPunchListener.OnNatIntroductionRequest

EventBasedNatPunchListener.OnNatIntroductionSuccess

EventBasedNetListener.OnConnectionRequest

EventBasedNetListener.OnNetworkError

EventBasedNetListener.OnNetworkLatencyUpdate

EventBasedNetListener.OnNetworkReceive

EventBasedNetListener.OnNetworkReceiveUnconnected

EventBasedNetListener.OnPeerConnected

EventBasedNetListener.OnPeerDisconnected

Class ConnectionRequest

Inheritance

System.Object
ConnectionRequest

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public class ConnectionRequest
```

Fields

ConnectionId

Declaration

```
public readonly long ConnectionId
```

Field Value

TYPE	DESCRIPTION
System.Int64	

Data

Declaration

```
public readonly NetDataReader Data
```

Field Value

TYPE	DESCRIPTION
NetDataReader	

RemoteEndPoint

Declaration

```
public readonly NetEndPoint RemoteEndPoint
```

Field Value

TYPE	DESCRIPTION
NetEndPoint	

Properties

Result

Declaration

```
public ConnectionRequestResult Result { get; }
```

Property Value

TYPE	DESCRIPTION
ConnectionRequestResult	

Methods

Accept()

Accept connection and get new NetPeer as result

Declaration

```
public NetPeer Accept()
```

Returns

TYPE	DESCRIPTION
NetPeer	Connected NetPeer

AcceptIfKey(String)

Declaration

```
public bool AcceptIfKey(string key)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	key	

Returns

TYPE	DESCRIPTION
System.Boolean	

Reject()

Declaration

```
public void Reject()
```

Enum ConnectionRequestResult

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public enum ConnectionRequestResult
```

Fields

NAME	DESCRIPTION
Accept	
None	
Reject	

Enum ConnectionState

Peer connection state

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
[Flags]
public enum ConnectionState
```

Fields

NAME	DESCRIPTION
Any	
Connected	
Disconnected	
InProgress	
ShutdownRequested	

Enum DeliveryMethod

Sending method type

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public enum DeliveryMethod
```

Fields

NAME	DESCRIPTION
ReliableOrdered	Reliable and ordered. All packets will be sent and received in order
ReliableUnordered	Reliable. All packets will be sent and received, but without order
Sequenced	Unreliable. Packets can be dropped, but never duplicated and arrive in order
Unreliable	Unreliable. Packets can be dropped, duplicated or arrive without order

Struct DisconnectInfo

Additional information about disconnection

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public struct DisconnectInfo
```

Fields

AdditionalData

Additional data that can be accessed (only if reason is RemoteConnectionClose)

Declaration

```
public NetDataReader AdditionalData
```

Field Value

TYPE	DESCRIPTION
NetDataReader	

Reason

Additional info why peer disconnected

Declaration

```
public DisconnectReason Reason
```

Field Value

TYPE	DESCRIPTION
DisconnectReason	

SocketErrorCode

Error code (if reason is SocketSendError or SocketReceiveError)

Declaration

```
public int SocketErrorCode
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Enum DisconnectReason

Disconnect reason that you receive in OnPeerDisconnected event

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public enum DisconnectReason
```

Fields

NAME	DESCRIPTION
ConnectionFailed	
DisconnectPeerCalled	
RemoteConnectionClose	
SocketReceiveError	
SocketSendError	
Timeout	

Class EventBasedNatPunchListener

Inheritance

System.Object
EventBasedNatPunchListener

Implements

INatPunchListener

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public class EventBasedNatPunchListener : INatPunchListener
```

Events

NatIntroductionRequest

Declaration

```
public event EventBasedNatPunchListener.OnNatIntroductionRequest NatIntroductionRequest
```

Event Type

TYPE	DESCRIPTION
EventBasedNatPunchListener.OnNatIntroductionRequest	

NatIntroductionSuccess

Declaration

```
public event EventBasedNatPunchListener.OnNatIntroductionSuccess NatIntroductionSuccess
```

Event Type

TYPE	DESCRIPTION
EventBasedNatPunchListener.OnNatIntroductionSuccess	

Explicit Interface Implementations

INatPunchListener.OnNatIntroductionRequest(NetEndPoint, NetEndPoint, String)

Declaration

```
void INatPunchListener.OnNatIntroductionRequest(NetEndPoint localEndPoint, NetEndPoint remoteEndPoint, string
token)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	localEndPoint	
NetEndPoint	remoteEndPoint	
System.String	token	

INatPunchListener.OnNatIntroductionSuccess(NetEndPoint, String)

Declaration

```
void INatPunchListener.OnNatIntroductionSuccess(NetEndPoint targetEndPoint, string token)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	targetEndPoint	
System.String	token	

Implements

INatPunchListener

Delegate EventBasedNatPunchListener.OnNatIntroductionRequest

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnNatIntroductionRequest(NetEndPoint localEndPoint, NetEndPoint remoteEndPoint, string
token);
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	localEndPoint	
NetEndPoint	remoteEndPoint	
System.String	token	

Delegate EventBasedNatPunchListener.OnNatIntroductionSuccess

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnNatIntroductionSuccess(NetEndPoint targetEndPoint, string token);
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	targetEndPoint	
System.String	token	

Class EventBasedNetListener

Inheritance

System.Object
EventBasedNetListener

Implements

INetEventListener

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public class EventBasedNetListener : INetEventListener
```

Events

ConnectionRequestEvent

Declaration

```
public event EventBasedNetListener.OnConnectionRequest ConnectionRequestEvent
```

Event Type

TYPE	DESCRIPTION
EventBasedNetListener.OnConnectionRequest	

NetworkErrorEvent

Declaration

```
public event EventBasedNetListener.OnNetworkError NetworkErrorEvent
```

Event Type

TYPE	DESCRIPTION
EventBasedNetListener.OnNetworkError	

NetworkLatencyUpdateEvent

Declaration

```
public event EventBasedNetListener.OnNetworkLatencyUpdate NetworkLatencyUpdateEvent
```

Event Type

TYPE	DESCRIPTION
EventBasedNetListener.OnNetworkLatencyUpdate	

NetworkReceiveEvent

Declaration

```
public event EventBasedNetListener.OnNetworkReceive NetworkReceiveEvent
```

Event Type

TYPE	DESCRIPTION
EventBasedNetListener.OnNetworkReceive	

NetworkReceiveUnconnectedEvent

Declaration

```
public event EventBasedNetListener.OnNetworkReceiveUnconnected NetworkReceiveUnconnectedEvent
```

Event Type

TYPE	DESCRIPTION
EventBasedNetListener.OnNetworkReceiveUnconnected	

PeerConnectedEvent

Declaration

```
public event EventBasedNetListener.OnPeerConnected PeerConnectedEvent
```

Event Type

TYPE	DESCRIPTION
EventBasedNetListener.OnPeerConnected	

PeerDisconnectedEvent

Declaration

```
public event EventBasedNetListener.OnPeerDisconnected PeerDisconnectedEvent
```

Event Type

TYPE	DESCRIPTION
EventBasedNetListener.OnPeerDisconnected	

Explicit Interface Implementations

INetEventListener.OnConnectionRequest(ConnectionRequest)

Declaration

```
void INetEventListener.OnConnectionRequest(ConnectionRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	

INetEventListener.OnNetworkError(NetEndPoint, Int32)

Declaration

```
void INetEventListener.OnNetworkError(NetEndPoint endPoint, int socketErrorCode)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	endPoint	
System.Int32	socketErrorCode	

INetEventListener.OnNetworkLatencyUpdate(NetPeer, Int32)

Declaration

```
void INetEventListener.OnNetworkLatencyUpdate(NetPeer peer, int latency)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
System.Int32	latency	

INetEventListener.OnNetworkReceive(NetPeer, NetDataReader, DeliveryMethod)

Declaration

```
void INetEventListener.OnNetworkReceive(NetPeer peer, NetDataReader reader, DeliveryMethod deliveryMethod)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
NetDataReader	reader	
DeliveryMethod	deliveryMethod	

INetEventListener.OnNetworkReceiveUnconnected(NetEndPoint, NetDataReader, UnconnectedMessageType)

Declaration

```
void INetEventListener.OnNetworkReceiveUnconnected(NetEndPoint remoteEndPoint, NetDataReader reader,
UnconnectedMessageType messageType)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	remoteEndPoint	
NetDataReader	reader	
UnconnectedMessageType	messageType	

INetEventListener.OnPeerConnected(NetPeer)

Declaration

```
void INetEventListener.OnPeerConnected(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	

INetEventListener.OnPeerDisconnected(NetPeer, DisconnectInfo)

Declaration

```
void INetEventListener.OnPeerDisconnected(NetPeer peer, DisconnectInfo disconnectInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
DisconnectInfo	disconnectInfo	

Implements

INetEventListener

Delegate EventBasedNetListener.OnConnectionRequest

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnConnectionRequest(ConnectionRequest request);
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	

Delegate EventBasedNetListener.OnNetworkError

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnNetworkError(NetEndPoint endPoint, int socketErrorCode);
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	endPoint	
System.Int32	socketErrorCode	

Delegate EventBasedNetListener.OnNetworkLatencyUpdate

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnNetworkLatencyUpdate(NetPeer peer, int latency);
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
System.Int32	latency	

Delegate EventBasedNetListener.OnNetworkReceive

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnNetworkReceive(NetPeer peer, NetDataReader reader, DeliveryMethod deliveryMethod);
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
NetDataReader	reader	
DeliveryMethod	deliveryMethod	

Delegate EventBasedNetListener.OnNetworkReceiveUnconnected

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnNetworkReceiveUnconnected(NetEndPoint remoteEndPoint, NetDataReader reader,
UnconnectedMessageType messageType);
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	remoteEndPoint	
NetDataReader	reader	
UnconnectedMessageType	messageType	

Delegate EventBasedNetListener.OnPeerConnected

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnPeerConnected(NetPeer peer);
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	

Delegate EventBasedNetListener.OnPeerDisconnected

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void OnPeerDisconnected(NetPeer peer, DisconnectInfo disconnectInfo);
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
DisconnectInfo	disconnectInfo	

Interface INatPunchListener

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public interface INatPunchListener
```

Methods

OnNatIntroductionRequest(NetEndPoint, NetEndPoint, String)

Declaration

```
void OnNatIntroductionRequest(NetEndPoint localEndPoint, NetEndPoint remoteEndPoint, string token)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	localEndPoint	
NetEndPoint	remoteEndPoint	
System.String	token	

OnNatIntroductionSuccess(NetEndPoint, String)

Declaration

```
void OnNatIntroductionSuccess(NetEndPoint targetEndPoint, string token)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	targetEndPoint	
System.String	token	

Interface INetEventListener

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public interface INetEventListener
```

Methods

OnConnectionRequest(ConnectionRequest)

On peer connection requested

Declaration

```
void OnConnectionRequest(ConnectionRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	Request information (EndPoint, internal id, additional data)

OnNetworkError(NetEndPoint, Int32)

Network error (on send or receive)

Declaration

```
void OnNetworkError(NetEndPoint endPoint, int socketErrorCode)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	endPoint	From endPoint (can be null)
System.Int32	socketErrorCode	Socket error code

OnNetworkLatencyUpdate(NetPeer, Int32)

Latency information updated

Declaration

```
void OnNetworkLatencyUpdate(NetPeer peer, int latency)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	Peer with updated latency

TYPE	NAME	DESCRIPTION
System.Int32	latency	latency value in milliseconds

OnNetworkReceive(NetPeer, NetDataReader, DeliveryMethod)

Received some data

Declaration

```
void OnNetworkReceive(NetPeer peer, NetDataReader reader, DeliveryMethod deliveryMethod)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	From peer
NetDataReader	reader	DataReader containing all received data
DeliveryMethod	deliveryMethod	Type of received packet

OnNetworkReceiveUnconnected(NetEndPoint, NetDataReader, UnconnectedMessageType)

Received unconnected message

Declaration

```
void OnNetworkReceiveUnconnected(NetEndPoint remoteEndPoint, NetDataReader reader, UnconnectedMessageType
messageType)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	remoteEndPoint	From address (IP and Port)
NetDataReader	reader	Message data
UnconnectedMessageType	messageType	Message type (simple, discovery request or responce)

OnPeerConnected(NetPeer)

New remote peer connected to host, or client connected to remote host

Declaration

```
void OnPeerConnected(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	Connected peer object

OnPeerDisconnected(NetPeer, DisconnectInfo)

Peer disconnected

Declaration

```
void OnPeerDisconnected(NetPeer peer, DisconnectInfo disconnectInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	disconnected peer
DisconnectInfo	disconnectInfo	additional info about reason, errorCode or data received with disconnect message

Interface INetLogger

Interface to implement for your own logger

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public interface INetLogger
```

Methods

WriteNet(ConsoleColor, String, Object[])

Declaration

```
void WriteNet(ConsoleColor color, string str, params object[] args)
```

Parameters

TYPE	NAME	DESCRIPTION
System.ConsoleColor	color	
System.String	str	
System.Object[]	args	

Class InvalidPacketException

Inheritance

System.Object

System.Exception

System.SystemException

System.ArgumentException

InvalidPacketException

TooBigPacketException

Implements

System.Runtime.Serialization.ISerializable

System.Runtime.InteropServices._Exception

Inherited Members

System.ArgumentException.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)

System.ArgumentException.ParamName

System.ArgumentException.Message

System.Exception.GetBaseException()

System.Exception.ToString()

System.Exception.GetType()

System.Exception.InnerException

System.Exception.HelpLink

System.Exception.HResult

System.Exception.Source

System.Exception.StackTrace

System.Exception.TargetSite

System.Exception.Data

System.Object.Equals(System.Object)

System.Object.Equals(System.Object, System.Object)

System.Object.GetHashCode()

System.Object.MemberwiseClone()

System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public class InvalidPacketException : ArgumentException, ISerializable, _Exception
```

Constructors

InvalidPacketException()

Declaration

```
public InvalidPacketException()
```

InvalidPacketException(String)

Declaration

```
public InvalidPacketException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

InvalidPacketException(String, Exception)

Declaration

```
public InvalidPacketException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Enum LocalAddrType

Address type that you want to receive from NetUtils.GetLocalIp method

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
[Flags]
public enum LocalAddrType
```

Fields

NAME	DESCRIPTION
All	
IPv4	
IPv6	

Class NatPunchModule

Module for UDP NAT Hole punching operations. Can be accessed from NetManager

Inheritance

System.Object
NatPunchModule

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public sealed class NatPunchModule
```

Fields

MaxTokenLength

Declaration

```
public const int MaxTokenLength = 256
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Methods

Init(INatPunchListener)

Declaration

```
public void Init(INatPunchListener listener)
```

Parameters

TYPE	NAME	DESCRIPTION
INatPunchListener	listener	

NatIntroduce(NetEndPoint, NetEndPoint, NetEndPoint, NetEndPoint, String)

Declaration

```
public void NatIntroduce(NetEndPoint hostInternal, NetEndPoint hostExternal, NetEndPoint clientInternal,
NetEndPoint clientExternal, string additionalInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	hostInternal	
NetEndPoint	hostExternal	
NetEndPoint	clientInternal	
NetEndPoint	clientExternal	
System.String	additionalInfo	

PollEvents()

Declaration

```
public void PollEvents()
```

SendNatIntroduceRequest(NetEndPoint, String)

Declaration

```
public void SendNatIntroduceRequest(NetEndPoint masterServerEndPoint, string additionalInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	masterServerEndPoint	
System.String	additionalInfo	

Class NetConstants

Network constants. Can be tuned from sources for your purposes.

Inheritance

System.Object
NetConstants

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public static class NetConstants
```

Fields

DefaultWindowSize

Declaration

```
public const int DefaultWindowSize = 64
```

Field Value

TYPE	DESCRIPTION
System.Int32	

FlowIncreaseThreshold

Declaration

```
public const int FlowIncreaseThreshold = 4
```

Field Value

TYPE	DESCRIPTION
System.Int32	

FlowUpdateTime

Declaration

```
public const int FlowUpdateTime = 1000
```

Field Value

TYPE	DESCRIPTION
System.Int32	

FragmentHeaderSize

Declaration

```
public const int FragmentHeaderSize = 6
```

Field Value

TYPE	DESCRIPTION
System.Int32	

HalfMaxSequence

Declaration

```
public const ushort HalfMaxSequence = 16384
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

HeaderSize

Declaration

```
public const int HeaderSize = 1
```

Field Value

TYPE	DESCRIPTION
System.Int32	

MaxSequence

Declaration

```
public const ushort MaxSequence = 32768
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

MinPacketDataSize

Declaration

```
public const int MinPacketDataSize = 507
```

Field Value

TYPE	DESCRIPTION
System.Int32	

MinPacketSize

Declaration

```
public const int MinPacketSize = 508
```

Field Value

TYPE	DESCRIPTION
System.Int32	

MinSequencedPacketDataSize

Declaration

```
public const int MinSequencedPacketDataSize = 505
```

Field Value

TYPE	DESCRIPTION
System.Int32	

SequencedHeaderSize

Declaration

```
public const int SequencedHeaderSize = 3
```

Field Value

TYPE	DESCRIPTION
System.Int32	

SocketBufferSize

Declaration

```
public const int SocketBufferSize = 4194304
```

Field Value

TYPE	DESCRIPTION
System.Int32	

SocketTTL

Declaration

```
public const int SocketTTL = 255
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Class NetDebug

Static class for defining your own LiteNetLib logger instead of Console.WriteLine or Debug.Log if compiled with UNITY flag

Inheritance

System.Object
NetDebug

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public static class NetDebug
```

Fields

Logger

Declaration

```
public static INetLogger Logger
```

Field Value

TYPE	DESCRIPTION
INetLogger	

Class NetEndPoint

Network End Point. Contains ip address and port

Inheritance

System.Object
NetEndPoint

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public sealed class NetEndPoint
```

Constructors

NetEndPoint(String, Int32)

Declaration

```
public NetEndPoint(string hostStr, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostStr	A valid host string that can be resolved by DNS or parsed as an IP address
System.Int32	port	Port of the end point

Exceptions

TYPE	CONDITION
System.ArgumentException	`hostStr` contains an invalid IP address
System.ArgumentOutOfRangeException	`port` is less than IPEndPoint.MinPort or port is greater than IPEndPoint.MaxPort

Fields

IPv4Any

Declaration

```
public static readonly string IPv4Any
```

Field Value

TYPE	DESCRIPTION
System.String	

IPv6Any

Declaration

```
public static readonly string IPv6Any
```

Field Value

TYPE	DESCRIPTION
System.String	

Properties

Host

Declaration

```
public string Host { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Port

Declaration

```
public int Port { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Equals(Object)

Declaration

```
public override bool Equals(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	

Returns

TYPE	DESCRIPTION
System.Boolean	

Overrides

System.Object.Equals(System.Object)

GetHashCode()

Declaration

```
public override int GetHashCode()
```

Returns

TYPE	DESCRIPTION
System.Int32	

Overrides

System.Object.GetHashCode()

ToString()

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Object.ToString()

Class NetManager

Main class for all network operations. Can be used as client and/or server.

Inheritance

System.Object
NetManager

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public sealed class NetManager
```

Constructors

NetManager(INetEventListener)

NetManager constructor with maxConnections = 1 (usable for client)

Declaration

```
public NetManager(INetEventListener listener)
```

Parameters

TYPE	NAME	DESCRIPTION
INetEventListener	listener	Network events listener

NetManager(INetEventListener, Int32)

NetManager constructor

Declaration

```
public NetManager(INetEventListener listener, int maxConnections)
```

Parameters

TYPE	NAME	DESCRIPTION
INetEventListener	listener	Network events listener
System.Int32	maxConnections	Maximum connections (incoming and outcoming)

Fields

DisconnectTimeout

If NetManager doesn't receive any packet from remote peer during this time then connection will be closed (including library internal keepalive packets)

Declaration

```
public int DisconnectTimeout
```

Field Value

TYPE	DESCRIPTION
System.Int32	

DiscoveryEnabled

Allows receive DiscoveryRequests

Declaration

```
public bool DiscoveryEnabled
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

MaxConnectAttempts

Maximum connection attempts before client stops and call disconnect event.

Declaration

```
public int MaxConnectAttempts
```

Field Value

TYPE	DESCRIPTION
System.Int32	

MergeEnabled

Merge small packets into one before sending to reduce outgoing packets count. (May increase a bit outgoing data size)

Declaration

```
public bool MergeEnabled
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

NatPunchEnabled

Enable nat punch messages

Declaration

```
public bool NatPunchEnabled
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

NatPunchModule

NatPunchModule for NAT hole punching operations

Declaration

```
public readonly NatPunchModule NatPunchModule
```

Field Value

TYPE	DESCRIPTION
NatPunchModule	

PingInterval

Interval for latency detection and checking connection

Declaration

```
public int PingInterval
```

Field Value

TYPE	DESCRIPTION
System.Int32	

ReconnectDelay

Delay betwen initial connection attempts

Declaration

```
public int ReconnectDelay
```

Field Value

TYPE	DESCRIPTION
System.Int32	

ReuseAddress

Enables socket option "ReuseAddress" for specific purposes

Declaration

```
public bool ReuseAddress
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

SimulateLatency

Simulate latency by holding packets for random time. (Works only in DEBUG mode)

Declaration

```
public bool SimulateLatency
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

SimulatePacketLoss

Simulate packet loss by dropping random amout of packets. (Works only in DEBUG mode)

Declaration

```
public bool SimulatePacketLoss
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

SimulationMaxLatency

Maximum simulated latency

Declaration

```
public int SimulationMaxLatency
```

Field Value

TYPE	DESCRIPTION
System.Int32	

SimulationMinLatency

Minimum simulated latency

Declaration

```
public int SimulationMinLatency
```

Field Value

TYPE	DESCRIPTION
System.Int32	

SimulationPacketLossChance

Chance of packet loss when simulation enabled. value in percents (1 - 100).

Declaration

```
public int SimulationPacketLossChance
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Statistics

Statistics of all connections

Declaration

```
public readonly NetStatistics Statistics
```

Field Value

TYPE	DESCRIPTION
NetStatistics	

UnconnectedMessagesEnabled

Enable messages receiving without connection. (with SendUnconnectedMessage method)

Declaration

```
public bool UnconnectedMessagesEnabled
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

UnsyncedEvents

Experimental feature. Events automatically will be called without PollEvents method from another thread

Declaration

```
public bool UnsyncedEvents
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

UpdateTime

Library logic update and send period in milliseconds

Declaration

```
public int UpdateTime
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

IsRunning

Returns true if socket listening and update thread is running

Declaration

```
public bool IsRunning { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

LocalPort

Local EndPoint (host and port)

Declaration

```
public int LocalPort { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PeersCount

Declaration

```
public int PeersCount { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Connect(NetEndPoint, NetDataWriter)

Connect to remote host

Declaration

```
public NetPeer Connect(NetEndPoint target, NetDataWriter connectionData)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	target	Server end point (ip and port)
NetDataWriter	connectionData	Additional data for remote peer

Returns

TYPE	DESCRIPTION
NetPeer	Null if connections limit reached, New NetPeer if new connection, Old NetPeer if already connected

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Manager is not running. Call Start()

Connect(NetEndPoint, String)

Connect to remote host

Declaration

```
public NetPeer Connect(NetEndPoint target, string key)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	target	Server end point (ip and port)
System.String	key	Connection key

Returns

TYPE	DESCRIPTION
NetPeer	Null if connections limit reached, New NetPeer if new connection, Old NetPeer if already connected

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Manager is not running. Call Start()

Connect(String, Int32, NetDataWriter)

Connect to remote host

Declaration

```
public NetPeer Connect(string address, int port, NetDataWriter connectionData)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	address	Server IP or hostname
System.Int32	port	Server Port
NetDataWriter	connectionData	Additional data for remote peer

Returns

TYPE	DESCRIPTION
NetPeer	Null if connections limit reached, New NetPeer if new connection, Old NetPeer if already connected

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Manager is not running. Call Start()

Connect(String, Int32, String)

Connect to remote host

Declaration

```
public NetPeer Connect(string address, int port, string key)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	address	Server IP or hostname
System.Int32	port	Server Port
System.String	key	Connection key

Returns

TYPE	DESCRIPTION
NetPeer	Null if connections limit reached, New NetPeer if new connection, Old NetPeer if already connected

Exceptions

TYPE	CONDITION
System.InvalidOperationException	Manager is not running. Call Start()

DisconnectAll()

Declaration

```
public void DisconnectAll()
```

DisconnectAll(Byte[], Int32, Int32)

Declaration

```
public void DisconnectAll(byte[] data, int start, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
System.Int32	start	
System.Int32	count	

DisconnectPeer(NetPeer)

Disconnect peer from server

Declaration

```
public void DisconnectPeer(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	peer to disconnect

DisconnectPeer(NetPeer, NetDataWriter)

Disconnect peer from server and send additional data (Size must be less or equal MTU - 8)

Declaration

```
public void DisconnectPeer(NetPeer peer, NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	peer to disconnect
NetDataWriter	writer	additional data

DisconnectPeer(NetPeer, Byte[])

Disconnect peer from server and send additional data (Size must be less or equal MTU - 8)

Declaration

```
public void DisconnectPeer(NetPeer peer, byte[] data)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	peer to disconnect
System.Byte[]	data	additional data

DisconnectPeer(NetPeer, Byte[], Int32, Int32)

Disconnect peer from server and send additional data (Size must be less or equal MTU - 8)

Declaration

```
public void DisconnectPeer(NetPeer peer, byte[] data, int start, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	peer to disconnect
System.Byte[]	data	additional data
System.Int32	start	data start
System.Int32	count	data length

DisconnectPeerForce(NetPeer)

Immediately disconnect peer from server without additional data

Declaration

```
public void DisconnectPeerForce(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	peer to disconnect

Flush()

Flush all queued packets of all peers

Declaration

```
public void Flush()
```

GetFirstPeer()

Get first peer. Usefull for Client mode

Declaration

```
public NetPeer GetFirstPeer()
```

Returns

TYPE	DESCRIPTION
NetPeer	

GetPeers()

Get copy of current connected peers (slow! use GetPeersNonAlloc for best performance)

Declaration

```
[Obsolete("Use GetPeers(ConnectionState peerState)")]
public NetPeer[] GetPeers()
```

Returns

TYPE	DESCRIPTION
NetPeer[]	Array with connected peers

GetPeers(ConnectionState)

Get copy of current connected peers (slow! use GetPeersNonAlloc for best performance)

Declaration

```
public NetPeer[] GetPeers(ConnectionState peerState)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
ConnectionState	peerState	

Returns

TYPE	DESCRIPTION
NetPeer[]	Array with connected peers

GetPeersCount(ConnectionState)

Declaration

```
public int GetPeersCount(ConnectionState peerState)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionState	peerState	

Returns

TYPE	DESCRIPTION
System.Int32	

GetPeersNonAlloc(List<NetPeer>, ConnectionState)

Get copy of peers (without allocations)

Declaration

```
public void GetPeersNonAlloc(List<NetPeer> peers, ConnectionState peerState)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.List<NetPeer>	peers	List that will contain result
ConnectionState	peerState	State of peers

PollEvents()

Receive all pending events. Call this in game update code

Declaration

```
public void PollEvents()
```

SendDiscoveryRequest(NetDataWriter, Int32)

Declaration

```
public bool SendDiscoveryRequest(NetDataWriter writer, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	
System.Int32	port	

Returns

TYPE	DESCRIPTION
System.Boolean	

SendDiscoveryRequest(Byte[], Int32)

Declaration

```
public bool SendDiscoveryRequest(byte[] data, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
System.Int32	port	

Returns

TYPE	DESCRIPTION
System.Boolean	

SendDiscoveryRequest(Byte[], Int32, Int32, Int32)

Declaration

```
public bool SendDiscoveryRequest(byte[] data, int start, int length, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
System.Int32	start	
System.Int32	length	
System.Int32	port	

Returns

TYPE	DESCRIPTION
System.Boolean	

SendDiscoveryResponse(NetDataWriter, NetEndPoint)

Declaration

```
public bool SendDiscoveryResponse(NetDataWriter writer, NetEndPoint remoteEndPoint)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	
NetEndPoint	remoteEndPoint	

Returns

TYPE	DESCRIPTION
System.Boolean	

SendDiscoveryResponse(Byte[], NetEndPoint)

Declaration

```
public bool SendDiscoveryResponse(byte[] data, NetEndPoint remoteEndPoint)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
NetEndPoint	remoteEndPoint	

Returns

TYPE	DESCRIPTION
System.Boolean	

SendDiscoveryResponse(Byte[], Int32, Int32, NetEndPoint)

Declaration

```
public bool SendDiscoveryResponse(byte[] data, int start, int length, NetEndPoint remoteEndPoint)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
System.Int32	start	

TYPE	NAME	DESCRIPTION
System.Int32	length	
NetEndPoint	remoteEndPoint	

Returns

TYPE	DESCRIPTION
System.Boolean	

SendToAll(NetDataWriter, DeliveryMethod)

Send data to all connected peers

Declaration

```
public void SendToAll(NetDataWriter writer, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	DataWriter with data
DeliveryMethod	options	Send options (reliable, unreliable, etc.)

SendToAll(NetDataWriter, DeliveryMethod, NetPeer)

Send data to all connected peers

Declaration

```
public void SendToAll(NetDataWriter writer, DeliveryMethod options, NetPeer excludePeer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	DataWriter with data
DeliveryMethod	options	Send options (reliable, unreliable, etc.)
NetPeer	excludePeer	Excluded peer

SendToAll(Byte[], DeliveryMethod)

Send data to all connected peers

Declaration

```
public void SendToAll(byte[] data, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	Data
DeliveryMethod	options	Send options (reliable, unreliable, etc.)

SendToAll(Byte[], DeliveryMethod, NetPeer)

Send data to all connected peers

Declaration

```
public void SendToAll(byte[] data, DeliveryMethod options, NetPeer excludePeer)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	Data
DeliveryMethod	options	Send options (reliable, unreliable, etc.)
NetPeer	excludePeer	Excluded peer

SendToAll(Byte[], Int32, Int32, DeliveryMethod)

Send data to all connected peers

Declaration

```
public void SendToAll(byte[] data, int start, int length, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	Data
System.Int32	start	Start of data
System.Int32	length	Length of data

TYPE	NAME	DESCRIPTION
DeliveryMethod	options	Send options (reliable, unreliable, etc.)

SendToAll(Byte[], Int32, Int32, DeliveryMethod, NetPeer)

Send data to all connected peers

Declaration

```
public void SendToAll(byte[] data, int start, int length, DeliveryMethod options, NetPeer excludePeer)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	Data
System.Int32	start	Start of data
System.Int32	length	Length of data
DeliveryMethod	options	Send options (reliable, unreliable, etc.)
NetPeer	excludePeer	Excluded peer

SendUnconnectedMessage(NetDataWriter, NetEndPoint)

Send message without connection

Declaration

```
public bool SendUnconnectedMessage(NetDataWriter writer, NetEndPoint remoteEndPoint)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	Data serializer
NetEndPoint	remoteEndPoint	Packet destination

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
System.Boolean	Operation result

SendUnconnectedMessage(Byte[], NetEndPoint)

Send message without connection

Declaration

```
public bool SendUnconnectedMessage(byte[] message, NetEndPoint remoteEndPoint)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	message	Raw data
NetEndPoint	remoteEndPoint	Packet destination

Returns

TYPE	DESCRIPTION
System.Boolean	Operation result

SendUnconnectedMessage(Byte[], Int32, Int32, NetEndPoint)

Send message without connection

Declaration

```
public bool SendUnconnectedMessage(byte[] message, int start, int length, NetEndPoint remoteEndPoint)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	message	Raw data
System.Int32	start	data start
System.Int32	length	data length
NetEndPoint	remoteEndPoint	Packet destination

Returns

TYPE	DESCRIPTION
System.Boolean	Operation result

Start()

Start logic thread and listening on available port

Declaration

```
public bool Start()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

Start(Int32)

Start logic thread and listening on selected port

Declaration

```
public bool Start(int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	port	port to listen

Returns

TYPE	DESCRIPTION
System.Boolean	

Start(String, String, Int32)

Start logic thread and listening on selected port

Declaration

```
public bool Start(string addressIPv4, string addressIPv6, int port)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	addressIPv4	bind to specific ipv4 address
System.String	addressIPv6	bind to specific ipv6 address

TYPE	NAME	DESCRIPTION
System.Int32	port	port to listen

Returns

TYPE	DESCRIPTION
System.Boolean	

Stop()

Force closes connection and stop all threads.

Declaration

```
public void Stop()
```

Class NetPeer

Network peer. Main purpose is sending messages to specific peer.

Inheritance

System.Object
NetPeer

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public sealed class NetPeer
```

Fields

Statistics

Statistics of peer connection

Declaration

```
public readonly NetStatistics Statistics
```

Field Value

TYPE	DESCRIPTION
NetStatistics	

Tag

Application defined object containing data about the connection

Declaration

```
public object Tag
```

Field Value

TYPE	DESCRIPTION
System.Object	

Properties

ConnectId

Connection id for internal purposes, but can be used as key in your dictionary of peers

Declaration

```
public long ConnectId { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	

ConnectionState

Current connection state

Declaration

```
public ConnectionState ConnectionState { get; }
```

Property Value

TYPE	DESCRIPTION
ConnectionState	

EndPoint

Peer ip address and port

Declaration

```
public NetEndPoint EndPoint { get; }
```

Property Value

TYPE	DESCRIPTION
NetEndPoint	

Mtu

Current MTU - Maximum Transfer Unit (maximum udp packet size without fragmentation)

Declaration

```
public int Mtu { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

NetManager

Peer parent NetManager

Declaration

```
public NetManager NetManager { get; }
```

Property Value

TYPE	DESCRIPTION
NetManager	

PacketsCountInReliableOrderedQueue

Declaration

```
public int PacketsCountInReliableOrderedQueue { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

PacketsCountInReliableQueue

Declaration

```
public int PacketsCountInReliableQueue { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Ping

Current ping in milliseconds

Declaration

```
public int Ping { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

TimeSinceLastPacket

Time since last packet received (including internal library packets)

Declaration

```
public int TimeSinceLastPacket { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Disconnect()

Declaration

```
public void Disconnect()
```

Disconnect(NetDataWriter)

Declaration

```
public void Disconnect(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Disconnect(Byte[])

Declaration

```
public void Disconnect(byte[] data)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	

Disconnect(Byte[], Int32, Int32)

Declaration

```
public void Disconnect(byte[] data, int start, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
System.Int32	start	
System.Int32	count	

Flush()

Flush all queued packets

Declaration

```
public void Flush()
```

GetMaxSinglePacketSize(DeliveryMethod)

Gets maximum size of packet that will be not fragmented.

Declaration

```
public int GetMaxSinglePacketSize(DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
DeliveryMethod	options	Type of packet that you want send

Returns

TYPE	DESCRIPTION
System.Int32	size in bytes

Send(NetDataWriter, DeliveryMethod)

Send data to peer

Declaration

```
public void Send(NetDataWriter dataWriter, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	dataWriter	DataWriter with data
DeliveryMethod	options	Send options (reliable, unreliable, etc.)

Exceptions

TYPE	CONDITION
TooBigPacketException	If size exceeds maximum limit: MTU - headerSize bytes for Unreliable Fragment count exceeded ushort.MaxValue

Send(Byte[], DeliveryMethod)

Send data to peer

Declaration

```
public void Send(byte[] data, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	Data

TYPE	NAME	DESCRIPTION
DeliveryMethod	options	Send options (reliable, unreliable, etc.)

Exceptions

TYPE	CONDITION
TooBigPacketException	If size exceeds maximum limit: MTU - headerSize bytes for Unreliable Fragment count exceeded ushort.MaxValue

Send(Byte[], Int32, Int32, DeliveryMethod)

Send data to peer

Declaration

```
public void Send(byte[] data, int start, int length, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	Data
System.Int32	start	Start of data
System.Int32	length	Length of data
DeliveryMethod	options	Send options (reliable, unreliable, etc.)

Exceptions

TYPE	CONDITION
TooBigPacketException	If size exceeds maximum limit: MTU - headerSize bytes for Unreliable Fragment count exceeded ushort.MaxValue

Class NetStatistics

Inheritance

System.Object
NetStatistics

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public sealed class NetStatistics
```

Fields

BytesReceived

Declaration

```
public ulong BytesReceived
```

Field Value

TYPE	DESCRIPTION
System.UInt64	

BytesSent

Declaration

```
public ulong BytesSent
```

Field Value

TYPE	DESCRIPTION
System.UInt64	

PacketLoss

Declaration

```
public ulong PacketLoss
```

Field Value

TYPE	DESCRIPTION
System.UInt64	

PacketsReceived

Declaration

```
public ulong PacketsReceived
```

Field Value

TYPE	DESCRIPTION
System.UInt64	

PacketsSent

Declaration

```
public ulong PacketsSent
```

Field Value

TYPE	DESCRIPTION
System.UInt64	

Properties

PacketLossPercent

Declaration

```
public ulong PacketLossPercent { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt64	

Methods

ToString()

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	

Overrides

System.Object.ToString()

Class NetUtils

Some specific network utilities

Inheritance

System.Object
NetUtils

Inherited Members

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

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public static class NetUtils
```

Methods

GetLocalIp(LocalAddrType)

Get first detected local ip address

Declaration

```
public static string GetLocalIp(LocalAddrType addrType)
```

Parameters

TYPE	NAME	DESCRIPTION
LocalAddrType	addrType	type of address (IPv4, IPv6 or both)

Returns

TYPE	DESCRIPTION
System.String	IP address if available. Else - string.Empty

GetLocalIpList(LocalAddrType)

Get all local ip addresses

Declaration

```
public static List<string> GetLocalIpList(LocalAddrType addrType)
```

Parameters

TYPE	NAME	DESCRIPTION
LocalAddrType	addrType	type of address (IPv4, IPv6 or both)

Returns

TYPE	DESCRIPTION
System.Collections.Generic.List<System.String>	List with all local ip adresses

GetLocalIpList(List<String>, LocalAddrType)

Get all local ip addresses (non alloc version)

Declaration

```
public static void GetLocalIpList(List<string> targetList, LocalAddrType addrType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.Generic.List<System.String>	targetList	result list
LocalAddrType	addrType	type of address (IPv4, IPv6 or both)

RequestTimeFromNTP(String, Int32, Action<Nullable<DateTime>>)

Request time from NTP server and calls callback (if success)

Declaration

```
public static void RequestTimeFromNTP(string ntpServerAddress, int port, Action<DateTime? > onRequestComplete)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	ntpServerAddress	NTP Server address
System.Int32	port	port
System.Action<System.Nullable<System.DateTime>>	onRequestComplete	callback (called from other thread!)

Class TooBigPacketException

Inheritance

System.Object
System.Exception
System.SystemException
System.ArgumentException
InvalidPacketException
TooBigPacketException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.ArgumentException.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.ArgumentException.ParamName
System.ArgumentException.Message
System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetType()
System.Exception.InnerException
System.Exception.HelpLink
System.Exception.HResult
System.Exception.Source
System.Exception.StackTrace
System.Exception.TargetSite
System.Exception.Data
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public class TooBigPacketException : InvalidPacketException, ISerializable, _Exception
```

Constructors

TooBigPacketException()

Declaration

```
public TooBigPacketException()
```

TooBigPacketException(String)

Declaration

```
public TooBigPacketException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

TooBigPacketException(String, Exception)

Declaration

```
public TooBigPacketException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Enum UnconnectedMessageType

Type of message that you receive in OnNetworkReceiveUnconnected event

Namespace: LiteNetLib

Assembly: Assembly-CSharp.dll

Syntax

```
public enum UnconnectedMessageType
```

Fields

NAME	DESCRIPTION
BasicMessage	
DiscoveryRequest	
DiscoveryResponse	

Namespace LiteNetLib.Utils

Classes

FastBitConverter

InvalidTypeException

NetDataReader

NetDataWriter

NetPacketProcessor

NetSerializer

ParseException

Interfaces

INetSerializable

Delegates

NetPacketProcessor.SubscrieDelegate

Class FastBitConverter

Inheritance

System.Object
FastBitConverter

Inherited Members

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

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public static class FastBitConverter
```

Methods

GetBytes(Byte[], Int32, Double)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, double value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	
System.Double	value	

GetBytes(Byte[], Int32, Int16)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, short value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	
System.Int16	value	

GetBytes(Byte[], Int32, Int32)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, int value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	
System.Int32	value	

GetBytes(Byte[], Int32, Int64)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, long value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	
System.Int64	value	

GetBytes(Byte[], Int32, Single)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, float value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	
System.Single	value	

GetBytes(Byte[], Int32, UInt16)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, ushort value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	

TYPE	NAME	DESCRIPTION
System.UInt16	value	

GetBytes(Byte[], Int32, UInt32)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, uint value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	
System.UInt32	value	

GetBytes(Byte[], Int32, UInt64)

Declaration

```
public static void GetBytes(byte[] bytes, int startIndex, ulong value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	
System.Int32	startIndex	
System.UInt64	value	

WriteLittleEndian(Byte[], Int32, Int16)

Declaration

```
public static void WriteLittleEndian(byte[] buffer, int offset, short data)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	buffer	
System.Int32	offset	
System.Int16	data	

Interface INetSerializable

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public interface INetSerializable
```

Methods

Deserialize(NetDataReader)

Declaration

```
void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Class InvalidTypeException

Inheritance

System.Object
System.Exception
System.SystemException
System.ArgumentException
InvalidTypeException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.ArgumentException.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.ArgumentException.ParamName
System.ArgumentException.Message
System.Exception.GetBaseException()
System.Exception.ToString()
System.Exception.GetType()
System.Exception.InnerException
System.Exception.HelpLink
System.Exception.HResult
System.Exception.Source
System.Exception.StackTrace
System.Exception.TargetSite
System.Exception.Data
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public class InvalidTypeException : ArgumentException, ISerializable, _Exception
```

Constructors

InvalidTypeException()

Declaration

```
public InvalidTypeException()
```

InvalidTypeException(String)

Declaration

```
public InvalidTypeException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

InvalidTypeException(String, Exception)

Declaration

```
public InvalidTypeException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

InvalidTypeException(String, String)

Declaration

```
public InvalidTypeException(string message, string paramName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.String	paramName	

InvalidTypeException(String, String, Exception)

Declaration

```
public InvalidTypeException(string message, string paramName, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.String	paramName	
System.Exception	innerException	

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Class NetDataReader

Inheritance

System.Object
NetDataReader

Inherited Members

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

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public class NetDataReader
```

Constructors

NetDataReader()

Declaration

```
public NetDataReader()
```

NetDataReader(Byte[])

Declaration

```
public NetDataReader(byte[] source)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	source	

NetDataReader(Byte[], Int32)

Declaration

```
public NetDataReader(byte[] source, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	source	
System.Int32	offset	

NetDataReader(Byte[], Int32, Int32)

Declaration

```
public NetDataReader(byte[] source, int offset, int maxSize)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	source	
System.Int32	offset	
System.Int32	maxSize	

Fields

_data

Declaration

```
protected byte[] _data
```

Field Value

TYPE	DESCRIPTION
System.Byte[]	

_dataSize

Declaration

```
protected int _dataSize
```

Field Value

TYPE	DESCRIPTION
System.Int32	

_position

Declaration

```
protected int _position
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

AvailableBytes

Declaration

```
public int AvailableBytes { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Data

Declaration

```
public byte[] Data { get; }
```

Property Value

TYPE	DESCRIPTION
System.Byte[]	

EndOfData

Declaration

```
public bool EndOfData { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Position

Declaration

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

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

Clear()

Declaration

```
public void Clear()
```

Clone()

Clone NetDataReader without data copy (usable for OnReceive)

Declaration

```
public NetDataReader Clone()
```

Returns

TYPE	DESCRIPTION
NetDataReader	new NetDataReader instance

GetBool()

Declaration

```
public bool GetBool()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

GetBoolArray()

Declaration

```
public bool[] GetBoolArray()
```

Returns

TYPE	DESCRIPTION
System.Boolean[]	

GetByte()

Declaration

```
public byte GetByte()
```

Returns

TYPE	DESCRIPTION
System.Byte	

GetBytes(Byte[], Int32)

Declaration

```
public void GetBytes(byte[] destination, int lenght)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	destination	
System.Int32	lenght	

GetBytesWithLength()

Declaration

```
public byte[] GetBytesWithLength()
```

Returns

TYPE	DESCRIPTION
System.Byte[]	

GetChar()

Declaration

```
public char GetChar()
```

Returns

TYPE	DESCRIPTION
System.Char	

GetDouble()

Declaration

```
public double GetDouble()
```

Returns

TYPE	DESCRIPTION
System.Double	

GetDoubleArray()

Declaration

```
public double[] GetDoubleArray()
```

Returns

TYPE	DESCRIPTION
System.Double[]	

GetFloat()

Declaration

```
public float GetFloat()
```

Returns

TYPE	DESCRIPTION
System.Single	

GetFloatArray()

Declaration

```
public float[] GetFloatArray()
```

Returns

TYPE	DESCRIPTION
System.Single[]	

GetInt()

Declaration

```
public int GetInt()
```

Returns

TYPE	DESCRIPTION
System.Int32	

GetIntArray()

Declaration

```
public int[] GetIntArray()
```

Returns

TYPE	DESCRIPTION
System.Int32[]	

GetLong()

Declaration

```
public long GetLong()
```

Returns

TYPE	DESCRIPTION
System.Int64	

GetLongArray()

Declaration

```
public long[] GetLongArray()
```

Returns

TYPE	DESCRIPTION
System.Int64[]	

GetNetEndPoint()

Declaration

```
public NetEndPoint GetNetEndPoint()
```

Returns

TYPE	DESCRIPTION
NetEndPoint	

GetRemainingBytes()

Declaration

```
public byte[] GetRemainingBytes()
```

Returns

TYPE	DESCRIPTION
System.Byte[]	

GetRemainingBytes(Byte[])

Declaration

```
public void GetRemainingBytes(byte[] destination)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	destination	

GetSByte()

Declaration

```
public sbyte GetSByte()
```

Returns

TYPE	DESCRIPTION
System.SByte	

GetShort()

Declaration

```
public short GetShort()
```

Returns

TYPE	DESCRIPTION
System.Int16	

GetShortArray()

Declaration

```
public short[] GetShortArray()
```

Returns

TYPE	DESCRIPTION
System.Int16[]	

GetString()

Declaration

```
public string GetString()
```

Returns

TYPE	DESCRIPTION
System.String	

GetString(Int32)

Declaration

```
public string GetString(int maxLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	maxLength	

Returns

TYPE	DESCRIPTION
System.String	

GetStringArray()

Declaration

```
public string[] GetStringArray()
```

Returns

TYPE	DESCRIPTION
System.String[]	

GetStringArray(Int32)

Declaration

```
public string[] GetStringArray(int maxStringLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	maxStringLength	

Returns

TYPE	DESCRIPTION
System.String[]	

GetUInt()

Declaration

```
public uint GetUInt()
```

Returns

TYPE	DESCRIPTION
System.UInt32	

GetUIntArray()

Declaration

```
public uint[] GetUIntArray()
```

Returns

TYPE	DESCRIPTION
System.UInt32[]	

GetULong()

Declaration

```
public ulong GetULong()
```

Returns

TYPE	DESCRIPTION
System.UInt64	

GetULongArray()

Declaration

```
public ulong[] GetULongArray()
```

Returns

TYPE	DESCRIPTION
System.UInt64[]	

GetUShort()

Declaration

```
public ushort GetUShort()
```

Returns

TYPE	DESCRIPTION
System.UInt16	

GetUShortArray()

Declaration

```
public ushort[] GetUShortArray()
```

Returns

TYPE	DESCRIPTION
System.UInt16[]	

PeekBool()

Declaration

```
public bool PeekBool()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

PeekByte()

Declaration

```
public byte PeekByte()
```

Returns

TYPE	DESCRIPTION
System.Byte	

PeekChar()

Declaration

```
public char PeekChar()
```

Returns

TYPE	DESCRIPTION
System.Char	

PeekDouble()

Declaration

```
public double PeekDouble()
```

Returns

TYPE	DESCRIPTION
System.Double	

PeekFloat()

Declaration

```
public float PeekFloat()
```

Returns

TYPE	DESCRIPTION
System.Single	

PeekInt()

Declaration

```
public int PeekInt()
```

Returns

TYPE	DESCRIPTION
System.Int32	

PeekLong()

Declaration

```
public long PeekLong()
```

Returns

TYPE	DESCRIPTION
System.Int64	

PeekSByte()

Declaration

```
public sbyte PeekSByte()
```

Returns

TYPE	DESCRIPTION
System.SByte	

PeekShort()

Declaration

```
public short PeekShort()
```

Returns

TYPE	DESCRIPTION
System.Int16	

PeekString()

Declaration

```
public string PeekString()
```

Returns

TYPE	DESCRIPTION
System.String	

PeekString(Int32)

Declaration

```
public string PeekString(int maxLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	maxLength	

Returns

TYPE	DESCRIPTION
System.String	

PeekUInt()

Declaration

```
public uint PeekUInt()
```

Returns

TYPE	DESCRIPTION
System.UInt32	

PeekULong()

Declaration

```
public ulong PeekULong()
```

Returns

TYPE	DESCRIPTION
System.UInt64	

PeekUShort()

Declaration

```
public ushort PeekUShort()
```

Returns

TYPE	DESCRIPTION
System.UInt16	

SetSource(NetDataWriter)

Declaration

```
public void SetSource(NetDataWriter dataWriter)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	dataWriter	

SetSource(Byte[])

Declaration

```
public void SetSource(byte[] source)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	source	

SetSource(Byte[], Int32)

Declaration

```
public void SetSource(byte[] source, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	source	
System.Int32	offset	

SetSource(Byte[], Int32, Int32)

Declaration

```
public void SetSource(byte[] source, int offset, int maxSize)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	source	
System.Int32	offset	
System.Int32	maxSize	

Class NetDataWriter

Inheritance

System.Object
NetDataWriter

Inherited Members

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

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public class NetDataWriter
```

Constructors

NetDataWriter()

Declaration

```
public NetDataWriter()
```

NetDataWriter(Boolean)

Declaration

```
public NetDataWriter(bool autoResize)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	autoResize	

NetDataWriter(Boolean, Int32)

Declaration

```
public NetDataWriter(bool autoResize, int initialSize)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	autoResize	
System.Int32	initialSize	

Fields

_data

Declaration

```
protected byte[] _data
```

Field Value

TYPE	DESCRIPTION
System.Byte[]	

_position

Declaration

```
protected int _position
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

Capacity

Declaration

```
public int Capacity { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Data

Declaration

```
public byte[] Data { get; }
```

Property Value

TYPE	DESCRIPTION
System.Byte[]	

Length

Declaration

```
public int Length { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

CopyData()

Declaration

```
public byte[] CopyData()
```

Returns

TYPE	DESCRIPTION
System.Byte[]	

FromBytes(Byte[], Boolean)

Creates NetDataWriter from existing ByteArray

Declaration

```
public static NetDataWriter FromBytes(byte[] bytes, bool copy)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	Source byte array
System.Boolean	copy	Copy array to new location or use existing

Returns

TYPE	DESCRIPTION
NetDataWriter	

FromBytes(Byte[], Int32, Int32)

Creates NetDataWriter from existing ByteArray (always copied data)

Declaration

```
public static NetDataWriter FromBytes(byte[] bytes, int offset, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	bytes	Source byte array
System.Int32	offset	Offset of array
System.Int32	length	Length of array

Returns

TYPE	DESCRIPTION
NetDataWriter	

FromString(String)

Declaration

```
public static NetDataWriter FromString(string value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	value	

Returns

TYPE	DESCRIPTION
NetDataWriter	

Put(NetEndPoint)

Declaration

```
public void Put(NetEndPoint endPoint)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	endPoint	

Put(Boolean)

Declaration

```
public void Put(bool value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	value	

Put(Byte)

Declaration

```
public void Put(byte value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte	value	

Put(Byte[])

Declaration

```
public void Put(byte[] data)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	

Put(Byte[], Int32, Int32)

Declaration

```
public void Put(byte[] data, int offset, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
System.Int32	offset	
System.Int32	length	

Put(Char)

Declaration

```
public void Put(char value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Char	value	

Put(Double)

Declaration

```
public void Put(double value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double	value	

Put(Int16)

Declaration

```
public void Put(short value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	value	

Put(Int32)

Declaration

```
public void Put(int value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	value	

Put(Int64)

Declaration

```
public void Put(long value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64	value	

Put(SByte)

Declaration

```
public void Put(sbyte value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.SByte	value	

Put(Single)

Declaration

```
public void Put(float value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	value	

Put(String)

Declaration

```
public void Put(string value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	value	

Put(String, Int32)

Declaration

```
public void Put(string value, int maxLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	value	
System.Int32	maxLength	

Put(UInt16)

Declaration

```
public void Put(ushort value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	value	

Put(UInt32)

Declaration

```
public void Put(uint value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt32	value	

Put(UInt64)

Declaration

```
public void Put(ulong value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt64	value	

PutArray(Boolean[])

Declaration

```
public void PutArray(bool[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean[]	value	

PutArray(Double[])

Declaration

```
public void PutArray(double[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Double[]	value	

PutArray(Int16[])

Declaration

```
public void PutArray(short[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16[]	value	

PutArray(Int32[])

Declaration

```
public void PutArray(int[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32[]	value	

PutArray(Int64[])

Declaration

```
public void PutArray(long[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64[]	value	

PutArray(Single[])

Declaration

```
public void PutArray(float[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single[]	value	

PutArray(String[])

Declaration

```
public void PutArray(string[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String[]	value	

PutArray(String[], Int32)

Declaration

```
public void PutArray(string[] value, int maxLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String[]	value	
System.Int32	maxLength	

PutArray(UInt16[])

Declaration

```
public void PutArray(ushort[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16[]	value	

PutArray(UInt32[])

Declaration

```
public void PutArray(uint[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt32[]	value	

PutArray(UInt64[])

Declaration

```
public void PutArray(ulong[] value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt64[]	value	

PutBytesWithLength(Byte[])

Declaration

```
public void PutBytesWithLength(byte[] data)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	

PutBytesWithLength(Byte[], Int32, Int32)

Declaration

```
public void PutBytesWithLength(byte[] data, int offset, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	
System.Int32	offset	
System.Int32	length	

Reset()

Declaration

```
public void Reset()
```

Reset(Int32)

Declaration

```
public void Reset(int size)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	size	

ResizeIfNeed(Int32)

Declaration

```
public void ResizeIfNeed(int newSize)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newSize	

Class NetPacketProcessor

Inheritance

System.Object
NetPacketProcessor

Inherited Members

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

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public class NetPacketProcessor
```

Methods

GetCallbackFromData(NetDataReader)

Declaration

```
protected virtual NetPacketProcessor.SubscrieDelegate GetCallbackFromData(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Returns

TYPE	DESCRIPTION
NetPacketProcessor.SubscrieDelegate	

GetHash(Type)

Declaration

```
protected virtual ulong GetHash(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	

Returns

TYPE	DESCRIPTION
System.UInt64	

ReadAllPackets(NetDataReader)

Reads all available data from NetDataReader and calls OnReceive delegates

Declaration

```
public void ReadAllPackets(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	NetDataReader with packets data

ReadAllPackets(NetDataReader, Object)

Reads all available data from NetDataReader and calls OnReceive delegates

Declaration

```
public void ReadAllPackets(NetDataReader reader, object userData)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	NetDataReader with packets data
System.Object	userData	Argument that passed to OnReceivedEvent

Exceptions

TYPE	CONDITION
ParseException	Malformed packet

ReadPacket(NetDataReader)

Reads one packet from NetDataReader and calls OnReceive delegate

Declaration

```
public void ReadPacket(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	NetDataReader with packet

Exceptions

TYPE	CONDITION
ParseException	Malformed packet

ReadPacket(NetDataReader, Object)

Reads one packet from NetDataReader and calls OnReceive delegate

Declaration

```
public void ReadPacket(NetDataReader reader, object userData)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	NetDataReader with packet
System.Object	userData	Argument that passed to OnReceivedEvent

Exceptions

TYPE	CONDITION
ParseException	Malformed packet

RegisterNestedType<T>()

Register nested property type

Declaration

```
public bool RegisterNestedType<T>()where T : struct, INetSerializable
```

Returns

TYPE	DESCRIPTION
System.Boolean	True - if register successful, false - if type already registered

Type Parameters

NAME	DESCRIPTION
T	INetSerializable structure

RegisterNestedType<T>(Action<NetDataWriter, T>, Func<NetDataReader, T>)

Register nested property type

Declaration

```
public bool RegisterNestedType<T>(Action<NetDataWriter, T> writeDelegate, Func<NetDataReader, T> readDelegate)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<NetDataWriter, T>	writeDelegate	
System.Func<NetDataReader, T>	readDelegate	

Returns

TYPE	DESCRIPTION
System.Boolean	True - if register successful, false - if type already registered

Type Parameters

NAME	DESCRIPTION
T	

RegisterNestedType<T>(Func<T>)

Register nested property type

Declaration

```
public bool RegisterNestedType<T>(Func<T> constructor)where T : class, INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
System.Func<T>	constructor	

Returns

TYPE	DESCRIPTION
System.Boolean	True - if register successful, false - if type already registered

Type Parameters

NAME	DESCRIPTION
T	INetSerializable class

Send<T>(NetManager, T, DeliveryMethod)

Declaration

```
public void Send<T>(NetManager manager, T packet, DeliveryMethod options)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
NetManager	manager	
T	packet	
DeliveryMethod	options	

Type Parameters

NAME	DESCRIPTION
T	

Send<T>(NetPeer, T, DeliveryMethod)

Declaration

```
public void Send<T>(NetPeer peer, T packet, DeliveryMethod options)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
T	packet	
DeliveryMethod	options	

Type Parameters

NAME	DESCRIPTION
T	

SendNetSerializable<T>(NetManager, T, DeliveryMethod)

Declaration

```
public void SendNetSerializable<T>(NetManager manager, T packet, DeliveryMethod options)where T :
INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
NetManager	manager	
T	packet	
DeliveryMethod	options	

Type Parameters

NAME	DESCRIPTION
T	

SendNetSerializable<T>(NetPeer, T, DeliveryMethod)

Declaration

```
public void SendNetSerializable<T>(NetPeer peer, T packet, DeliveryMethod options)where T : INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	
T	packet	
DeliveryMethod	options	

Type Parameters

NAME	DESCRIPTION
T	

Subscribe<T>(Action<T>, Func<T>)

Register and subscribe to packet receive event

Declaration

```
public void Subscribe<T>(Action<T> onReceive, Func<T> packetConstructor)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T>	onReceive	event that will be called when packet deserialized with ReadPacket method
System.Func<T>	packetConstructor	Method that constructs packet intead of slow Activator.CreateInstance

Type Parameters

NAME	DESCRIPTION
T	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

Subscribe<T, TUserData>(Action<T, TUserData>, Func<T>)

Register and subscribe to packet receive event (with userData)

Declaration

```
public void Subscribe<T, TUserData>(Action<T, TUserData> onReceive, Func<T> packetConstructor)where T : class,
new ()
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T, TUserData>	onReceive	event that will be called when packet deserialized with ReadPacket method
System.Func<T>	packetConstructor	Method that constructs packet intead of slow Activator.CreateInstance

Type Parameters

NAME	DESCRIPTION
T	
TUserData	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

SubscribeNetSerializable<T>(Action<T>)

Declaration

```
public void SubscribeNetSerializable<T>(Action<T> onReceive)where T : INetSerializable, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T>	onReceive	

Type Parameters

NAME	DESCRIPTION
T	

SubscribeNetSerializable<T>(Action<T>, Func<T>)

Declaration

```
public void SubscribeNetSerializable<T>(Action<T> onReceive, Func<T> packetConstructor)where T :
INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T>	onReceive	
System.Func<T>	packetConstructor	

Type Parameters

NAME	DESCRIPTION
T	

SubscribeNetSerializable<T, TUserData>(Action<T, TUserData>)

Declaration

```
public void SubscribeNetSerializable<T, TUserData>(Action<T, TUserData> onReceive)where T : INetSerializable,
new ()
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T, TUserData>	onReceive	

Type Parameters

NAME	DESCRIPTION
T	
TUserData	

SubscribeNetSerializable<T, TUserData>(Action<T, TUserData>, Func<T>)

Declaration

```
public void SubscribeNetSerializable<T, TUserData>(Action<T, TUserData> onReceive, Func<T>
packetConstructor)where T : INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T, TUserData>	onReceive	
System.Func<T>	packetConstructor	

Type Parameters

NAME	DESCRIPTION
T	
TUserData	

SubscribeReusable<T>(Action<T>)

Register and subscribe to packet receive event This metod will overwrite last received packet class on receive (less garbage)

Declaration

```
public void SubscribeReusable<T>(Action<T> onReceive)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T>	onReceive	event that will be called when packet deserialized with ReadPacket method

Type Parameters

NAME	DESCRIPTION
T	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

SubscribeReusable<T, TUserData>(Action<T, TUserData>)

Register and subscribe to packet receive event This metod will overwrite last received packet class on receive (less garbage)

Declaration

```
public void SubscribeReusable<T, TUserData>(Action<T, TUserData> onReceive)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<T, TUserData>	onReceive	event that will be called when packet deserialized with ReadPacket method

Type Parameters

NAME	DESCRIPTION
T	
TUserData	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

Write<T>(T)

Declaration

```
public byte[] Write<T>(T packet)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
T	packet	

Returns

TYPE	DESCRIPTION
System.Byte[]	

Type Parameters

NAME	DESCRIPTION
T	

Write<T>(NetDataWriter, T)

Declaration

```
public void Write<T>(NetDataWriter writer, T packet)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	
T	packet	

Type Parameters

NAME	DESCRIPTION
T	

WriteHash(Type, NetDataWriter)

Declaration

```
protected virtual void WriteHash(Type type, NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	
NetDataWriter	writer	

WriteNetSerializable<T>(T)

Declaration

```
public byte[] WriteNetSerializable<T>(T packet)where T : INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
T	packet	

Returns

TYPE	DESCRIPTION
System.Byte[]	

Type Parameters

NAME	DESCRIPTION
T	

WriteNetSerializable<T>(NetDataWriter, T)

Declaration

```
public void WriteNetSerializable<T>(NetDataWriter writer, T packet)where T : INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	
T	packet	

Type Parameters

NAME	DESCRIPTION
T	

Delegate NetPacketProcessor.SubscrieDelegate

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
protected delegate void SubscrieDelegate(NetDataReader reader, object userData);
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	
System.Object	userData	

Class NetSerializer

Inheritance

System.Object
NetSerializer

Inherited Members

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

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public sealed class NetSerializer
```

Constructors

NetSerializer()

Declaration

```
public NetSerializer()
```

NetSerializer(Int32)

Declaration

```
public NetSerializer(int maxStringLength)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	maxStringLength	

Methods

Deserialize<T>(NetDataReader)

Reads packet with known type

Declaration

```
public T Deserialize<T>(NetDataReader reader)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	NetDataReader with packet

Returns

TYPE	DESCRIPTION
T	Returns packet if packet in reader is matched type

Type Parameters

NAME	DESCRIPTION
T	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

Deserialize<T>(NetDataReader, T)

Reads packet with known type (non alloc variant)

Declaration

```
public bool Deserialize<T>(NetDataReader reader, T target)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	NetDataReader with packet
T	target	Deserialization target

Returns

TYPE	DESCRIPTION
System.Boolean	Returns true if packet in reader is matched type

Type Parameters

NAME	DESCRIPTION
T	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

Register<T>()

Declaration

```
public void Register<T>()
```

Type Parameters

NAME	DESCRIPTION
T	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

RegisterNestedType<T>()

Register nested property type

Declaration

```
public bool RegisterNestedType<T>()where T : struct, INetSerializable
```

Returns

TYPE	DESCRIPTION
System.Boolean	True - if register successful, false - if type already registered

Type Parameters

NAME	DESCRIPTION
T	INetSerializable structure

RegisterNestedType<T>(Action<NetDataWriter, T>, Func<NetDataReader, T>)

Register nested property type

Declaration

```
public bool RegisterNestedType<T>(Action<NetDataWriter, T> writeDelegate, Func<NetDataReader, T> readDelegate)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Action<NetDataWriter, T>	writeDelegate	
System.Func<NetDataReader, T>	readDelegate	

Returns

TYPE	DESCRIPTION
System.Boolean	True - if register successful, false - if type already registered

Type Parameters

NAME	DESCRIPTION
T	

RegisterNestedType<T>(Func<T>)

Register nested property type

Declaration

```
public bool RegisterNestedType<T>(Func<T> constructor)where T : class, INetSerializable
```

Parameters

TYPE	NAME	DESCRIPTION
System.Func<T>	constructor	

Returns

TYPE	DESCRIPTION
System.Boolean	True - if register successful, false - if type already registered

Type Parameters

NAME	DESCRIPTION
T	INetSerializable class

Serialize<T>(T)

Serialize struct to byte array

Declaration

```
public byte[] Serialize<T>(T obj)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
T	obj	Object to serialize

Returns

TYPE	DESCRIPTION
System.Byte[]	byte array with serialized data

Type Parameters

NAME	DESCRIPTION
T	

Serialize<T>(NetDataWriter, T)

Serialize struct to NetDataWriter (fast)

Declaration

```
public void Serialize<T>(NetDataWriter writer, T obj)where T : class, new ()
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	Serialization target NetDataWriter
T	obj	Object to serialize

Type Parameters

NAME	DESCRIPTION
T	

Exceptions

TYPE	CONDITION
InvalidTypeException	`T` 's fields are not supported, or it has no fields

Class ParseException

Inheritance

System.Object
System.Exception
ParseException

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Inherited Members

System.Exception.GetBaseException()
System.Exception.GetObjectData(System.Runtime.Serialization.SerializationInfo, System.Runtime.Serialization.StreamingContext)
System.Exception.ToString()
System.Exception.GetType()
System.Exception.InnerException
System.Exception.HelpLink
System.Exception.HResult
System.Exception.Message
System.Exception.Source
System.Exception.StackTrace
System.Exception.TargetSite
System.Exception.Data
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: LiteNetLib.Utils

Assembly: Assembly-CSharp.dll

Syntax

```
public class ParseException : Exception, ISerializable, _Exception
```

Constructors

ParseException()

Declaration

```
public ParseException()
```

ParseException(String)

Declaration

```
public ParseException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

ParseException(String, Exception)

Declaration

```
public ParseException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

Implements

System.Runtime.Serialization.ISerializable
System.Runtime.InteropServices._Exception

Namespace TinyBirdNet

Classes

NetworkScenePostProcess

Used to run a method on the PostProcessScene.

RPCMethodInfo

A data storage class for RPC methods information.

TinyNetBehaviour

A TinyNetBehaviour is a MonoBehaviour who implements the interface ITinyNetObject.

In addition, TinyBirdNet handles it's spawning, serialization, RPC, and mostly anything you need to create a new instance of it in a multiplayer game and have it automatically synced.

TinyNetClient

Represents the Scene of a Client.

TinyNetConnection

A container for a connection to a NetPeer.

TinyNetGameManager

This class manages and communicates with

TinyNetGameManagerEditor

Custom inspector for the TinyNetGameManager class.

TinyNetIdentity

Any UnityEngine.GameObject that contains this component, can be spawned accross the network.

This is basically a container for an "universal id" accross the network.

TinyNetLogLevel

A simple log filter level to use in debug logs.

TinyNetMessageHandlers

A class that represents a container for TinyNetMessageDelegate.

TinyNetPlayerController

This class represents the player entity in a network game, there can be multiple players per client, when there are multiple people playing on one machine.

The server has one TinyNetConnection per NetPeer.

TinyNetPropertyAccessor<T>

Creates an acessor for a property, used for TinyNetSyncVar.

TinyNetReflector

This class is used to get all TinyNetSyncVar properties and TinyNetRPC methods and store their info.

TinyNetRPC

When used on a method, allows it to be executed remotely on another machine when called.

TinyNetScene

Represents a Scene, which is all data required to reproduce the game state.

TinyNetServer

Represents the Scene of a server.

TinyNetSimpleMenu

TinyNetStateSyncer

This class stores all SyncVar allowed properties and is used to sync the game state.

TinyNetSyncVar

When used on a compatible property type, it will send it's value to all clients if they are changed.

byte, sbyte, short, ushort, int, uint, long, ulong, float, double, bool, string.

Interfaces

ITinyNetInstanceID

Implement this interface to allow your custom class to receive a NetworkID.

ITinyNetObject

Implements basic functionality to allow network syncing.

Enums

LogFilter

The available levels of filter.

RPCCallers

Identifies the caller of a RPC.

RPCTarget

Identifies the target of a RPC.

Delegates

RPCDelegate

Handles RPC calls

SpawnDelegate

Handles requests to spawn objects on the client

UnSpawnDelegate

Handles requests to unspawn objects on the client

Interface ITinyNetInstanceID

Implement this interface to allow your custom class to receive a NetworkID.

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public interface ITinyNetInstanceID
```

Properties

NetworkID

The ID of an instance in the network, given by the server on spawn.

Declaration

```
int NetworkID { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

ReceiveNetworkID(Int32)

Receives the network identifier.

Declaration

```
void ReceiveNetworkID(int newID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newID	The new identifier.

Interface ITinyNetObject

Implements basic functionality to allow network syncing.

Inherited Members

ITinyNetInstanceID.NetworkID
ITinyNetInstanceID.ReceiveNetworkID(Int32)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public interface ITinyNetObject : ITinyNetInstanceID
```

Properties

isClient

Gets a value indicating whether this instance is client.

Declaration

```
bool isClient { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is client; otherwise, `false`.

isServer

Gets a value indicating whether this instance is server.

Declaration

```
bool isServer { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is server; otherwise, `false`.

NetIdentity

Gets the net identity.

Declaration

```
TinyNetIdentity NetIdentity { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
TinyNetIdentity	The net identity.

Methods

GetNetworkChannel()

Declaration

```
DeliveryMethod GetNetworkChannel()
```

Returns

TYPE	DESCRIPTION
DeliveryMethod	

InvokeRPC(Int32, NetDataReader)

Invokes the RPC.

Declaration

```
bool InvokeRPC(int rpcMethodIndex, NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	rpcMethodIndex	Index of the RPC method.
NetDataReader	reader	The reader.

Returns

TYPE	DESCRIPTION
System.Boolean	

OnGiveAuthority()

Called on the server when giving authority of this object to a client.

Declaration

```
void OnGiveAuthority()
```

OnNetworkCreate()

Always called, regardless of being a client or server. Called before variables are synced. (Order: 0)

Declaration

```
void OnNetworkCreate()
```

OnNetworkDestroy()

Called when the object receives an order to be destroyed from the network, in a listen server the object could just be unspawned without being actually destroyed.

Declaration

```
void OnNetworkDestroy()
```

OnRemoveAuthority()

Called on the server when removing authority of a client to this object.

Declaration

```
void OnRemoveAuthority()
```

OnSetLocalVisibility(Boolean)

This is only called on a listen server, for spawn and hide messages. Objects being destroyed will trigger OnNetworkDestroy as normal.

Declaration

```
void OnSetLocalVisibility(bool vis)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	vis	

OnStartAuthority()

Called on the client that receives authority of this object.

Declaration

```
void OnStartAuthority()
```

OnStartClient()

Called on the client when the object is spawned. Called after variables are synced. (Order: 2)

Declaration

```
void OnStartClient()
```

OnStartLocalPlayer()

Not implemented yet.

Declaration

```
void OnStartLocalPlayer()
```

OnStartServer()

Called on the server when Spawn is called for this object. (Order: 1)

Declaration

```
void OnStartServer()
```

OnStopAuthority()

Called on the client that loses authorithy of this object.

Declaration

```
void OnStopAuthority()
```

SendRPC(NetDataWriter, String)

Sends the RPC.

Declaration

```
void SendRPC(NetDataWriter stream, string rpcName)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	stream	The stream.
System.String	rpcName	Name of the RPC.

TinyDeserialize(NetDataReader, Boolean)

Deserializations the data received.

Declaration

```
void TinyDeserialize(NetDataReader reader, bool firstStateUpdate)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	The reader.
System.Boolean	firstStateUpdate	if set to `true` it's the first state update.

TinyNetUpdate()

Called after all FixedUpdates and physics but before any Update.

It is used by TinyNetServer to check if it is time to send the current state to clients.

Declaration

```
void TinyNetUpdate()
```

TinySerialize(NetDataWriter, Boolean)

Serializates the data.

Declaration

```
void TinySerialize(NetDataWriter writer, bool firstStateUpdate)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	The writer.
System.Boolean	firstStateUpdate	if set to `true` it's the first state update.

Enum LogFilter

The available levels of filter.

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public enum LogFilter
```

Fields

NAME	DESCRIPTION
Debug	
Dev	
Error	
Info	
Warn	

Class NetworkScenePostProcess

Used to run a method on the PostProcessScene.

Inheritance

System.Object
NetworkScenePostProcess

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
public class NetworkScenePostProcess
```

Methods

OnPostProcessScene()

Called when [PostProcessScene].

Checks all scene objects.

Declaration

```
[PostProcessScene]
public static void OnPostProcessScene()
```

Enum RPCCallers

Identifies the caller of a RPC.

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public enum RPCCallers
```

Fields

NAME	DESCRIPTION
Anyone	
ClientOwner	
Server	

Delegate RPCDelegate

Handles RPC calls

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void RPCDelegate(NetDataReader reader);
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	The reader.

Class RPCMethodInfo

A data storage class for RPC methods information.

Inheritance

System.Object
RPCMethodInfo

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class RPCMethodInfo
```

Constructors

RPCMethodInfo(String, RPCTarget, RPCCallers)

Declaration

```
public RPCMethodInfo(string rpcName, RPCTarget nTarget, RPCCallers nCaller)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	rpcName	
RPCTarget	nTarget	
RPCCallers	nCaller	

Properties

caller

Declaration

```
public RPCCallers caller { get; }
```

Property Value

TYPE	DESCRIPTION
RPCCallers	

name

Declaration

```
public string name { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

target

Declaration

```
public RPCTarget target { get; }
```

Property Value

TYPE	DESCRIPTION
RPCTarget	

Enum RPCTarget

Identifies the target of a RPC.

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public enum RPCTarget
```

Fields

NAME	DESCRIPTION
ClientOwner	
Everyone	
Server	

Delegate SpawnDelegate

Handles requests to spawn objects on the client

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate GameObject SpawnDelegate(Vector3 position, int assetIndex);
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.Vector3	position	The position.
System.Int32	assetIndex	Index of the asset.

Returns

TYPE	DESCRIPTION
UnityEngine.GameObject	

Class TinyNetBehaviour

A TinyNetBehaviour is a MonoBehaviour who implements the interface ITinyNetObject.

In addition, TinyBirdNet handles it's spawning, serialization, RPC, and mostly anything you need to create a new instance of it in a multiplayer game and have it automatically synced.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
TinyNetBehaviour

Implements

ITinyNetObject
ITinyNetInstanceID

Inherited Members

UnityEngine.MonoBehaviour.Invoke(System.String, System.Single)
UnityEngine.MonoBehaviour.InvokeRepeating(System.String, System.Single, System.Single)
UnityEngine.MonoBehaviour.CancelInvoke()
UnityEngine.MonoBehaviour.CancelInvoke(System.String)
UnityEngine.MonoBehaviour.IsInvoking(System.String)
UnityEngine.MonoBehaviour.IsInvoking()
UnityEngine.MonoBehaviour.StartCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine_Auto(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine(System.String, System.Object)
UnityEngine.MonoBehaviour.StartCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StopCoroutine(UnityEngine.Coroutine)
UnityEngine.MonoBehaviour.StopAllCoroutines()
UnityEngine.MonoBehaviour.print(System.Object)
UnityEngine.MonoBehaviour.useGUILayout
UnityEngine.MonoBehaviour.runInEditMode
UnityEngine.Behaviour.enabled
UnityEngine.Behaviour.isActiveAndEnabled
UnityEngine.Component.GetComponent(System.Type)
UnityEngine.Component.GetComponent<T>()
UnityEngine.Component.GetComponent(System.String)
UnityEngine.Component.GetComponentInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentInChildren(System.Type)
UnityEngine.Component.GetComponentInChildren<T>()
UnityEngine.Component.GetComponentInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren(System.Type)
UnityEngine.Component.GetComponentsInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInChildren<T>()
UnityEngine.Component.GetComponentsInChildren<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentInParent(System.Type)
UnityEngine.Component.GetComponentInParent<T>()

UnityEngine.Component.GetComponentsInParent(System.Type)
UnityEngine.Component.GetComponentsInParent(System.Type, System.Boolean)
UnityEngine.Component.GetComponentsInParent<T>(System.Boolean)
UnityEngine.Component.GetComponentsInParent<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInParent<T>()
UnityEngine.Component.GetComponents(System.Type)
UnityEngine.Component.GetComponents(System.Type, System.Collections.Generic.List<UnityEngine.Component>)
UnityEngine.Component.GetComponents<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponents<T>()
UnityEngine.Component.CompareTag(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object)
UnityEngine.Component.SendMessageUpwards(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object)
UnityEngine.Component.SendMessage(System.String)
UnityEngine.Component.SendMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object)
UnityEngine.Component.BroadcastMessage(System.String)
UnityEngine.Component.BroadcastMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.transform
UnityEngine.Component.gameObject
UnityEngine.Component.tag
UnityEngine.Component.rigidbody
UnityEngine.Component.rigidbody2D
UnityEngine.Component.camera
UnityEngine.Component.light
UnityEngine.Component.animation
UnityEngine.Component.constantForce
UnityEngine.Component.renderer
UnityEngine.Component.audio
UnityEngine.Component.guiText
UnityEngine.Component.networkView
UnityEngine.Component.guiElement
UnityEngine.Component.guiTexture
UnityEngine.Component.collider
UnityEngine.Component.collider2D
UnityEngine.Component.hingeJoint
UnityEngine.Component.particleEmitter
UnityEngine.Component.particleSystem
UnityEngine.Object.Destroy(UnityEngine.Object, System.Single)
UnityEngine.Object.Destroy(UnityEngine.Object)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object, System.Boolean)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object)
UnityEngine.Object.FindObjectsOfType(System.Type)
UnityEngine.Object.DontDestroyOnLoad(UnityEngine.Object)
UnityEngine.Object.DestroyObject(UnityEngine.Object, System.Single)
UnityEngine.Object.DestroyObject(UnityEngine.Object)
UnityEngine.Object.FindSceneObjectsOfType(System.Type)
UnityEngine.Object.FindObjectsOfTypeIncludingAssets(System.Type)

UnityEngine.Object.FindObjectsOfTypeAll(System.Type)
UnityEngine.Object.ToString()
UnityEngine.Object.GetInstanceID()
UnityEngine.Object.GetHashCode()
UnityEngine.Object.Equals(System.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Instantiate<T>(T)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.FindObjectsOfType<T>()
UnityEngine.Object.FindObjectOfType<T>()
UnityEngine.Object.FindObjectOfType(System.Type)
UnityEngine.Object.name
UnityEngine.Object.hideFlags
System.Object.Equals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
[RequireComponent(typeof (TinyNetIdentity))]
public class TinyNetBehaviour : MonoBehaviour, ITinyNetObject, ITinyNetInstanceID
```

Fields

_lastSendTime

[Server Only] The last Time.time registered at an UpdateDirtyFlag call.

Declaration

```
protected float _lastSendTime
```

Field Value

TYPE	DESCRIPTION
System.Single	

rpcRecycleWriter

A static NetDataWriter that can be used to convert most Objects to bytes.

Declaration

```
protected static NetDataWriter rpcRecycleWriter
```

Field Value

TYPE	DESCRIPTION
NetDataWriter	

Properties

bIsDirty

Gets or sets a value indicating whether this instance is dirty.

Declaration

```
public bool bIsDirty { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if instance is dirty; otherwise, `false`.

DirtyFlag

Gets the dirty flag.

Declaration

```
public BitArray DirtyFlag { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.BitArray	The dirty flag.

hasAuthority

Declaration

```
public bool hasAuthority { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

isClient

Gets a value indicating whether this instance is client.

Declaration

```
public bool isClient { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is client; otherwise, `false`.

isServer

Gets a value indicating whether this instance is server.

Declaration

```
public bool isServer { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is server; otherwise, `false`.

NetIdentity

Gets the net identity.

Declaration

```
public TinyNetIdentity NetIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
TinyNetIdentity	The net identity.

NetworkID

The ID of an instance in the network, given by the server on spawn.

Declaration

```
public int NetworkID { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

CheckIfPropertyUpdated(String, Type)

Checks if a TinyNetSyncVar property updated.

Declaration

```
public bool CheckIfPropertyUpdated(string propName, Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	propName	Name of the property.
System.Type	type	The type.

Returns

TYPE	DESCRIPTION
System.Boolean	

CreateAccessors()

Creates the TinyNetSyncVar property accessors.

Declaration

```
public void CreateAccessors()
```

GetNetworkChannel()

Declaration

```
public virtual DeliveryMethod GetNetworkChannel()
```

Returns

TYPE	DESCRIPTION
DeliveryMethod	

GetNetworkSendInterval()

Not implemented yet.

Declaration

```
public virtual float GetNetworkSendInterval()
```

Returns

TYPE	DESCRIPTION
System.Single	

InvokeRPC(Int32, NetDataReader)

Invokes the RPC.

Declaration

```
public virtual bool InvokeRPC(int rpcMethodIndex, NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	rpcMethodIndex	Index of the RPC method.
NetDataReader	reader	The reader.

Returns

TYPE	DESCRIPTION
System.Boolean	

OnGiveAuthority()

Called on the server when giving authority of this object to a client.

Declaration

```
public virtual void OnGiveAuthority()
```

OnNetworkCreate()

Remember that this is called first and before variables are synced.

Declaration

```
public virtual void OnNetworkCreate()
```

OnNetworkDestroy()

Called when the object receives an order to be destroyed from the network, in a listen server the object could just be unspawned without being actually destroyed.

Declaration

```
public virtual void OnNetworkDestroy()
```

OnRemoveAuthority()

Called on the server when removing authority of a client to this object.

Declaration

```
public virtual void OnRemoveAuthority()
```

OnSetLocalVisibility(Boolean)

This is only called on a listen server, for spawn and hide messages. Objects being destroyed will trigger OnNetworkDestroy as normal.

Declaration

```
public virtual void OnSetLocalVisibility(bool vis)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	vis	

OnStartAuthority()

Called on the client that receives authority of this object.

Declaration

```
public virtual void OnStartAuthority()
```

OnStartClient()

Called on the client when the object is spawned. Called after variables are synced. (Order: 2)

Declaration

```
public virtual void OnStartClient()
```

OnStartLocalPlayer()

Declaration

```
public virtual void OnStartLocalPlayer()
```

OnStartServer()

Called on the server when Spawn is called for this object. (Order: 1)

Declaration

```
public virtual void OnStartServer()
```

OnStopAuthority()

Called on the client that loses authorithy of this object.

Declaration

```
public virtual void OnStopAuthority()
```

ReceiveNetworkID(Int32)

Receives the network identifier.

Declaration

```
public void ReceiveNetworkID(int newID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newID	The new identifier.

RegisterRPCDelegate(RPCDelegate, String)

Registers the RPC delegate.

Declaration

```
protected void RegisterRPCDelegate(RPCDelegate rpcDel, string methodName)
```

Parameters

TYPE	NAME	DESCRIPTION
RPCDelegate	rpcDel	The RPC delegate.
System.String	methodName	Name of the method.

SendRPC(NetDataWriter, String)

Sends the RPC.

Declaration

```
public virtual void SendRPC(NetDataWriter stream, string rpcName)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	stream	The stream.
System.String	rpcName	Name of the RPC.

SendRPC(NetDataWriter, RPCTarget, RPCCallers, Int32)

Sends the RPC.

Declaration

```
public virtual void SendRPC(NetDataWriter stream, RPCTarget target, RPCCallers caller, int rpcMethodIndex)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	stream	The stream.
RPCTarget	target	The target.
RPCCallers	caller	The caller.
System.Int32	rpcMethodIndex	Index of the RPC method.

SetDirtyFlag(Int32, Boolean)

Sets the bit value on the dirty flag at the given index

Sets the bit value on the dirty flag at the given index.

Declaration

```
protected void SetDirtyFlag(int index, bool bValue)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	index	The index.
System.Boolean	bValue	The new bool value.

TinyDeserialize(NetDataReader, Boolean)

Deserializations the data received.

Declaration

```
public virtual void TinyDeserialize(NetDataReader reader, bool firstStateUpdate)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	The reader.
System.Boolean	firstStateUpdate	if set to `true` it's the first state update.

TinyNetUpdate()

Called after all FixedUpdates and physics but before any Update.

It is used by TinyNetServer to check if it is time to send the current state to clients.

Declaration

```
public void TinyNetUpdate()
```

TinySerialize(NetDataWriter, Boolean)

Serializates the data.

Declaration

```
public virtual void TinySerialize(NetDataWriter writer, bool firstStateUpdate)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	The writer.

TYPE	NAME	DESCRIPTION
System.Boolean	firstStateUpdate	if set to `true` it's the first state update.

Implements

ITinyNetObject
ITinyNetInstanceID

See Also

UnityEngine.MonoBehaviour
ITinyNetObject
ITinyNetInstanceID

Class TinyNetClient

Represents the Scene of a Client.

Inheritance

System.Object

TinyNetScene

TinyNetClient

Implements

INetEventListener

Inherited Members

TinyNetScene.createPlayerAction

TinyNetScene._localIdentityObjects

TinyNetScene._localNetObjects

TinyNetScene.recycleWriter

TinyNetScene.recycleMessageReader

TinyNetScene.s_TinyNetRPCMessage

TinyNetScene.s_TinyNetObjectHideMessage

TinyNetScene.s_TinyNetObjectDestroyMessage

TinyNetScene.s_TinyNetObjectSpawnMessage

TinyNetScene.s_TinyNetObjectSpawnSceneMessage

TinyNetScene.s_TineNetObjectSpawnFinishedMessage

TinyNetScene.s_TinyNetAddPlayerMessage

TinyNetScene.s_TinyNetRemovePlayerMessage

TinyNetScene.s_TinyNetRequestAddPlayerMessage

TinyNetScene.s_TinyNetRequestRemovePlayerMessage

TinyNetScene.s_TinyNetClientAuthorityMessage

TinyNetScene._tinyMessageHandlers

TinyNetScene._tinyNetConns

TinyNetScene.tinyNetConns

TinyNetScene.connToHost

TinyNetScene._netManager

TinyNetScene.currentFixedFrame

TinyNetScene.isRunning

TinyNetScene.isConnected

TinyNetScene.RegisterHandler(UInt16, TinyNetMessageDelegate)

TinyNetScene.RegisterHandlerSafe(UInt16, TinyNetMessageDelegate)

TinyNetScene.InternalUpdate()

TinyNetScene.TinyNetUpdate()

TinyNetScene.ClearNetManager()

TinyNetScene.ConfigureNetManager(Boolean)

TinyNetScene.ToggleNatPunching(Boolean)

TinyNetScene.SetPingInterval(Int32)

TinyNetScene.GetTinyNetConnection(Int64)

TinyNetScene.GetTinyNetConnection(NetPeer)

TinyNetScene.RemoveTinyNetConnection(TinyNetConnection)

TinyNetScene.RemoveTinyNetConnection(NetPeer)

TinyNetScene.RemoveTinyNetConnection(Int64)

TinyNetScene.AddTinyNetIdentityToList(TinyNetIdentity)

TinyNetScene.AddTinyNetObjectToList(ITinyNetObject)

TinyNetScene.RemoveTinyNetIdentityFromList(TinyNetIdentity)

TinyNetScene.RemoveTinyNetObjectFromList(ITinyNetObject)
TinyNetScene.GetTinyNetIdentityByNetworkID(Int32)
TinyNetScene.GetTinyNetObjectByNetworkID(Int32)
TinyNetScene.SendMessageByChannelToHost(ITinyNetMessage, DeliveryMethod)
TinyNetScene.SendMessageByChannelToTargetConnection(ITinyNetMessage, DeliveryMethod, TinyNetConnection)
TinyNetScene.SendMessageByChannelToAllConnections(ITinyNetMessage, DeliveryMethod)
TinyNetScene.SendMessageByChannelToAllReadyConnections(ITinyNetMessage, DeliveryMethod)
TinyNetScene.SendMessageByChannelToAllObserversOf(TinyNetIdentity, ITinyNetMessage, DeliveryMethod)
TinyNetScene.OnConnectionRequest(ConnectionRequest)
TinyNetScene.OnPeerConnected(NetPeer)
TinyNetScene.OnPeerDisconnected(NetPeer, DisconnectInfo)
TinyNetScene.OnNetworkError(NetEndPoint, Int32)
TinyNetScene.OnNetworkReceive(NetPeer, NetDataReader, DeliveryMethod)
TinyNetScene.OnNetworkReceiveUnconnected(NetEndPoint, NetDataReader, UnconnectedMessageType)
TinyNetScene.OnNetworkLatencyUpdate(NetPeer, Int32)
TinyNetScene.OnDiscoveryRequestReceived(NetEndPoint, NetDataReader)
TinyNetScene.OnDisconnect(TinyNetConnection)
TinyNetScene.OnRPCMessage(TinyNetMessageReader)
TinyNetScene.AddPlayerControllerToConnection(TinyNetConnection, Int32)
TinyNetScene.RemovePlayerControllerFromConnection(TinyNetConnection, Int16)
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ToString()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetClient : TinyNetScene, INetEventListener
```

Constructors

TinyNetClient()

Initializes a new instance of the TinyNetClient class.

Declaration

```
public TinyNetClient()
```

Fields

_localPlayers

The local players

Declaration

```
protected List<TinyNetPlayerController> _localPlayers
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<TinyNetPlayerController>	

instance

The singleton instance.

Declaration

```
public static TinyNetClient instance
```

Field Value

TYPE	DESCRIPTION
TinyNetClient	

OnClientReadyEvent

The client ready event.

Declaration

```
public static Action OnClientReadyEvent
```

Field Value

TYPE	DESCRIPTION
System.Action	

Properties

bLoadedScene

Gets or sets a value indicating whether the scene has been loaded.

Declaration

```
public bool bLoadedScene { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if scene has been loaded; otherwise, `false`.

localPlayers

Gets the local players.

Declaration

```
public List<TinyNetPlayerController> localPlayers { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<TinyNetPlayerController>	The local players.

TYPE

Sugar for generating debug logs.

Declaration

```
public override string TYPE { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

TinyNetScene.TYPE

Methods

ClientConnectTo(String, Int32)

Attempts to connect the client to the given server.

Declaration

```
public virtual void ClientConnectTo(string hostAddress, int hostPort)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostAddress	The host address.
System.Int32	hostPort	The host port.

ClientFinishLoadScene()

Called from the TinyNetGameManager when a scene finishes loading.

Declaration

```
public virtual void ClientFinishLoadScene()
```

CreatePlayerAndAdd(TinyNetConnection, Int32)

Creates a player controller and adds it to the connection.

Declaration

```
protected override void CreatePlayerAndAdd(TinyNetConnection conn, int playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection.
System.Int32	playerControllerId	The player controller identifier.

Overrides

TinyNetScene.CreatePlayerAndAdd(TinyNetConnection, Int32)

CreateTinyNetConnection(NetPeer)

Creates a TinyNetConnection

Declaration

```
protected override TinyNetConnection CreateTinyNetConnection(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	The NetPeer.

Returns

TYPE	DESCRIPTION
TinyNetConnection	

Overrides

TinyNetScene.CreateTinyNetConnection(NetPeer)

OnAddPlayerMessage(TinyNetMessageReader)

Called when an AddPlayerMessage is received.

Declaration

```
protected virtual void OnAddPlayerMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	A wrapper for a TinyNetAddPlayerMessage.

OnClientChangeSceneMessage(TinyNetMessageReader)

Handler for a scene change message.

Declaration

```
protected virtual void OnClientChangeSceneMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	A wrapper for a TinyNetStringMessage containing the scene name.

OnClientSceneChanged()

Called when a scene change finishes.

Declaration

```
public virtual void OnClientSceneChanged()
```

OnConnectionCreated(TinyNetConnection)

Called after a peer has connected and a TinyNetConnection was created for it.

Declaration

```
protected override void OnConnectionCreated(TinyNetConnection nConn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	nConn	The connection created.

Overrides

TinyNetScene.OnConnectionCreated(TinyNetConnection)

OnLocalAddPlayerMessage(TinyNetMessageReader)

Called when an AddPlayerMessage is received and we are a Listen Server.

Declaration

```
protected virtual void OnLocalAddPlayerMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	A wrapper for a TinyNetAddPlayerMessage.

OnRemovePlayerMessage(TinyNetMessageReader)

Called when a TinyNetRemovePlayerMessage is received.

Declaration

```
protected virtual void OnRemovePlayerMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	A wrapper for a TinyNetRemovePlayerMessage.

Ready()

Readies this instance.

Declaration

```
public virtual bool Ready()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

RegisterMessageHandlers()

Registers the message handlers.

Declaration

```
protected override void RegisterMessageHandlers()
```

Overrides

TinyNetScene.RegisterMessageHandlers()

RequestAddPlayerControllerToServer(Int32)

Requests a new TinyNetPlayerController to the server.

Declaration

```
public void RequestAddPlayerControllerToServer(int amountPlayers = 1)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	amountPlayers	The amount of players to create.

SendRPCToServer(NetDataWriter, Int32, ITinyNetObject)

Sends the RPC to server.

Declaration

```
public void SendRPCToServer(NetDataWriter stream, int rpcMethodIndex, ITinyNetObject iObj)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	stream	The stream.

TYPE	NAME	DESCRIPTION
System.Int32	rpcMethodIndex	Index of the RPC method.
ITinyNetObject	iObj	The ITinyNetObject instance.

StartClient()

Starts the client.

Declaration

```
public virtual bool StartClient()
```

Returns

TYPE	DESCRIPTION
System.Boolean	

Implements

INetEventListener

See Also

TinyNetScene

Class TinyNetConnection

A container for a connection to a NetPeer.

Inheritance

System.Object
TinyNetConnection

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetConnection
```

Constructors

TinyNetConnection(NetPeer)

Initializes a new instance of the TinyNetConnection class.

Declaration

```
public TinyNetConnection(NetPeer newPeer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	newPeer	The NetPeer.

Fields

_observingNetObjects

This is a list of objects the connection is able to observe, aka, are spawned and synced.

Declaration

```
protected HashSet<TinyNetIdentity> _observingNetObjects
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.HashSet<TinyNetIdentity>	

_ownedObjectsId

A hash containing the NetworkIDs of objects owned by this connection.

Declaration

```
protected HashSet<int> _ownedObjectsId
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.HashSet<System.Int32>	

_peer

The NetPeer of this connection.

Declaration

```
protected NetPeer _peer
```

Field Value

TYPE	DESCRIPTION
NetPeer	

isReady

If this instance is ready

Declaration

```
public bool isReady
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

recycleWriter

If using this, always Reset before use!

Declaration

```
protected static NetDataWriter recycleWriter
```

Field Value

TYPE	DESCRIPTION
NetDataWriter	

Properties

ConnectId

Gets the connect identifier.

Declaration

```
public long ConnectId { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int64	The connect identifier.

netPeer

Gets the NetPeer.

Declaration

```
public NetPeer netPeer { get; }
```

Property Value

TYPE	DESCRIPTION
NetPeer	The NetPeer.

playerControllers

Gets the TinyNetPlayerController.

Declaration

```
public List<TinyNetPlayerController> playerControllers { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<TinyNetPlayerController>	The TinyNetPlayerController.

Methods

AddOwnedObject(TinyNetIdentity)

Adds an object to the list of owned objects.

Declaration

```
public void AddOwnedObject(TinyNetIdentity obj)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	obj	The TinyNetIdentity of the object to own.

GetFirstPlayerController()

Gets the first player controller.

Useful if your game only have one player per connection.

Declaration

```
public TinyNetPlayerController GetFirstPlayerController()
```

Returns

TYPE	DESCRIPTION
TinyNetPlayerController	

GetPlayerController(Int16)

Returns a TinyNetPlayerController, given an identifier.

Declaration

```
public TinyNetPlayerController GetPlayerController(short playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	playerControllerId	The player controller identifier.

Returns

TYPE	DESCRIPTION
TinyNetPlayerController	

GetPlayerController(Int16, out TinyNetPlayerController)

Outs a player controller, given an identifier. Returns true if one was found.

Declaration

```
public bool GetPlayerController(short playerControllerId, out TinyNetPlayerController playerController)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	playerControllerId	The player controller identifier.
TinyNetPlayerController	playerController	The player controller found.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if a player controller was found; otherwise, `false`.

GetPlayerController<T>(Int16)

Returns a player controller cast to the type given.

Declaration

```
public T GetPlayerController<T>(short playerControllerId)where T : TinyNetPlayerController
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	playerControllerId	The player controller identifier.

Returns

TYPE	DESCRIPTION
T	A player controller cast to T.

Type Parameters

NAME	DESCRIPTION
T	A type derived from TinyNetPlayerController.

GetPlayerInputMessage(TinyNetMessageReader)

Redirects an TinyNetInputMessage to the correct player controller.

Declaration

```
public void GetPlayerInputMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	The TinyNetInputMessage.

HideObjectToConnection(TinyNetIdentity, Boolean)

Always call this to hide an object from a client, or you will have sync issues.

Declaration

```
public void HideObjectToConnection(TinyNetIdentity tni, bool isDestroyed)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tni	The TinyNetIdentity of the object to hide.
System.Boolean	isDestroyed	

IsObservingNetIdentity(TinyNetIdentity)

Determines whether this instance is observing the specified TinyNetIdentity.

Declaration

```
public bool IsObservingNetIdentity(TinyNetIdentity tni)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tni	The TinyNetIdentity.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if is observing the specified TinyNetIdentity; otherwise, `false`.

RemoveOwnedObject(TinyNetIdentity)

Removes the owned object from the list.

Declaration

```
public void RemoveOwnedObject(TinyNetIdentity obj)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	obj	The TinyNetIdentity of the object to remove.

RemovePlayerController(Int16)

Removes the player controller from this connection.

Declaration

```
public void RemovePlayerController(short playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	playerControllerId	The player controller identifier.

Send(NetDataWriter, DeliveryMethod)

Sends the specified data.

Declaration

```
public void Send(NetDataWriter dataWriter, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	dataWriter	The data writer.
DeliveryMethod	options	The options.

Send(Byte[], DeliveryMethod)

Sends the specified data.

Declaration

```
public void Send(byte[] data, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	data	The data.
DeliveryMethod	options	The options.

Send(ITinyNetMessage, DeliveryMethod)

Sends the specified ITinyNetMessage.

Declaration

```
public void Send(ITinyNetMessage msg, DeliveryMethod options)
```

Parameters

TYPE	NAME	DESCRIPTION
ITinyNetMessage	msg	The message.
DeliveryMethod	options	The options.

SetPlayerController<T>(TinyNetPlayerController)

Adds a TinyNetPlayerController to the list of player controllers of this connection.

Declaration

```
public void SetPlayerController<T>(TinyNetPlayerController player)where T : TinyNetPlayerController, new ()
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
TinyNetPlayerController	player	The player controller to add.

Type Parameters

NAME	DESCRIPTION
T	A type derived from TinyNetPlayerController.

ShowObjectToConnection(TinyNetIdentity)

Always call this to spawn an object to a client, or you will have sync issues.

Declaration

```
public void ShowObjectToConnection(TinyNetIdentity tni)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tni	The TinyNetIdentity of the object to spawn.

ToString()

Returns a System.String that represents this instance.

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	A System.String that represents this instance.

Overrides

System.Object.ToString()

Class TinyNetGameManager

This class manages and communicates with

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
TinyNetGameManager

Inherited Members

UnityEngine.MonoBehaviour.Invoke(System.String, System.Single)
UnityEngine.MonoBehaviour.InvokeRepeating(System.String, System.Single, System.Single)
UnityEngine.MonoBehaviour.CancelInvoke()
UnityEngine.MonoBehaviour.CancelInvoke(System.String)
UnityEngine.MonoBehaviour.IsInvoking(System.String)
UnityEngine.MonoBehaviour.IsInvoking()
UnityEngine.MonoBehaviour.StartCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine_Auto(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine(System.String, System.Object)
UnityEngine.MonoBehaviour.StartCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StopCoroutine(UnityEngine.Coroutine)
UnityEngine.MonoBehaviour.StopAllCoroutines()
UnityEngine.MonoBehaviour.print(System.Object)
UnityEngine.MonoBehaviour.useGUILayout
UnityEngine.MonoBehaviour.runInEditMode
UnityEngine.Behaviour.enabled
UnityEngine.Behaviour.isActiveAndEnabled
UnityEngine.Component.GetComponent(System.Type)
UnityEngine.Component.GetComponent<T>()
UnityEngine.Component.GetComponent(System.String)
UnityEngine.Component.GetComponentInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentInChildren(System.Type)
UnityEngine.Component.GetComponentInChildren<T>()
UnityEngine.Component.GetComponentInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren(System.Type)
UnityEngine.Component.GetComponentsInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInChildren<T>()
UnityEngine.Component.GetComponentsInChildren<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentInParent(System.Type)
UnityEngine.Component.GetComponentInParent<T>()
UnityEngine.Component.GetComponentsInParent(System.Type)
UnityEngine.Component.GetComponentsInParent(System.Type, System.Boolean)
UnityEngine.Component.GetComponentsInParent<T>(System.Boolean)
UnityEngine.Component.GetComponentsInParent<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInParent<T>()
UnityEngine.Component.GetComponents(System.Type)

UnityEngine.Component.GetComponents(System.Type, System.Collections.Generic.List<UnityEngine.Component>)
UnityEngine.Component.GetComponents<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponents<T>()
UnityEngine.Component.CompareTag(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object)
UnityEngine.Component.SendMessageUpwards(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object)
UnityEngine.Component.SendMessage(System.String)
UnityEngine.Component.SendMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object)
UnityEngine.Component.BroadcastMessage(System.String)
UnityEngine.Component.BroadcastMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.transform
UnityEngine.Component.gameObject
UnityEngine.Component.tag
UnityEngine.Component.rigidbody
UnityEngine.Component.rigidbody2D
UnityEngine.Component.camera
UnityEngine.Component.light
UnityEngine.Component.animation
UnityEngine.Component.constantForce
UnityEngine.Component.renderer
UnityEngine.Component.audio
UnityEngine.Component.guiText
UnityEngine.Component.networkView
UnityEngine.Component.guiElement
UnityEngine.Component.guiTexture
UnityEngine.Component.collider
UnityEngine.Component.collider2D
UnityEngine.Component.hingeJoint
UnityEngine.Component.particleEmitter
UnityEngine.Component.particleSystem
UnityEngine.Object.Destroy(UnityEngine.Object, System.Single)
UnityEngine.Object.Destroy(UnityEngine.Object)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object, System.Boolean)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object)
UnityEngine.Object.FindObjectsOfType(System.Type)
UnityEngine.Object.DontDestroyOnLoad(UnityEngine.Object)
UnityEngine.Object.DestroyObject(UnityEngine.Object, System.Single)
UnityEngine.Object.DestroyObject(UnityEngine.Object)
UnityEngine.Object.FindSceneObjectsOfType(System.Type)
UnityEngine.Object.FindObjectsOfTypeIncludingAssets(System.Type)
UnityEngine.Object.FindObjectsOfTypeAll(System.Type)
UnityEngine.Object.ToString()
UnityEngine.Object.GetInstanceID()
UnityEngine.Object.GetHashCode()
UnityEngine.Object.Equals(System.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion)

UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Instantiate<T>(T)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.FindObjectsOfType<T>()
UnityEngine.Object.FindObjectOfType<T>()
UnityEngine.Object.FindObjectOfType(System.Type)
UnityEngine.Object.name
UnityEngine.Object.hideFlags
System.Object.Equals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetGameManager : MonoBehaviour
```

Fields

_spawnHandlers

The spawn handlers.

`int` is the asset index in TinyBirdNet.TinyNetGameManager.registeredPrefabs.

Declaration

```
protected Dictionary<int, SpawnDelegate> _spawnHandlers
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.Int32, SpawnDelegate>	

_unspawnHandlers

The unspawn handlers.

`int` is the asset index in TinyBirdNet.TinyNetGameManager.registeredPrefabs.

Declaration

```
protected Dictionary<int, UnSpawnDelegate> _unspawnHandlers
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.Int32, UnSpawnDelegate>	

ApplicationGUID

Declaration

```
public static readonly Guid ApplicationGUID
```

Field Value

TYPE	DESCRIPTION
System.Guid	

ApplicationGUIDString

Declaration

```
public static readonly string ApplicationGUIDString
```

Field Value

TYPE	DESCRIPTION
System.String	

clientManager

The client scene manager.

Declaration

```
protected TinyNetClient clientManager
```

Field Value

TYPE	DESCRIPTION
TinyNetClient	

currentLogFilter

The current log filter for TinyLogger.

Declaration

```
public LogFilter currentLogFilter
```

Field Value

TYPE	DESCRIPTION
LogFilter	

instance

The singleton instance.

Declaration

```
public static TinyNetGameManager instance
```

Field Value

TYPE	DESCRIPTION
TinyNetGameManager	

maxNumberOfPlayers

The maximum number of players allowed in the network.

Declaration

```
[SerializeField]
protected int maxNumberOfPlayers
```

Field Value

TYPE	DESCRIPTION
System.Int32	

multiplayerConnectKey

Insert here a unique key per version of your game, if the key mismatches the player will be denied connection.

Declaration

```
public string multiplayerConnectKey
```

Field Value

TYPE	DESCRIPTION
System.String	

NetworkEveryXFixedFrames

The network state update will happen every x fixed frames.

Declaration

```
[Range(1F, 60F)]
public int NetworkEveryXFixedFrames
```

Field Value

TYPE	DESCRIPTION
System.Int32	

networkSceneName

Current scene name at runtime.

Declaration

```
public static string networkSceneName
```

Field Value

TYPE	DESCRIPTION
System.String	

pingInterval

The ping interval in ms.

Declaration

```
protected int pingInterval
```

Field Value

TYPE	DESCRIPTION
System.Int32	

port

The port

Declaration

```
protected int port
```

Field Value

TYPE	DESCRIPTION
System.Int32	

serverManager

The server scene manager.

Declaration

```
protected TinyNetServer serverManager
```

Field Value

TYPE	DESCRIPTION
TinyNetServer	

Properties

bNatPunchEnabled

Gets or sets a value indicating whether nat punch is enabled.

Needs custom implementation to work.

Declaration

```
public bool bNatPunchEnabled { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if nat punch is enabled; otherwise, `false`.

isClient

Gets a value indicating whether this instance is client.

Declaration

```
public bool isClient { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is client; otherwise, `false`.

isListenServer

Gets a value indicating whether this instance is a listen server.

Declaration

```
public bool isListenServer { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is a listen server; otherwise, `false`.

isServer

Gets a value indicating whether this instance is server.

Declaration

```
public bool isServer { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is server; otherwise, `false`.

MaxNumberOfPlayers

Gets the maximum number of players.

Declaration

```
public int MaxNumberOfPlayers { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The maximum number of players.

NextNetworkID

Gets the next network identifier.

Declaration

```
public int NextNetworkID { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The next network identifier.

NextPlayerID

Gets the next player identifier.

Declaration

```
public int NextPlayerID { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The next player identifier.

PingInterval

Gets or sets the ping interval.

Declaration

```
public int PingInterval { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The ping interval.

Port

Gets the port.

Declaration

```
public int Port { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The port.

Methods

AwakeVirtual()

Provides a function to be overrrided in case you need to add something in the Awake call.

Declaration

```
protected virtual void AwakeVirtual()
```

CheckForSceneLoad()

Checks if a scene load was requested and if it finished.

Declaration

```
protected virtual void CheckForSceneLoad()
```

ClearNetManager()

Clears the net manager.

Declaration

```
protected virtual void ClearNetManager()
```

ClientChangeScene(String, Boolean)

Orders the client to change to the given scene.

Declaration

```
public virtual void ClientChangeScene(string newSceneName, bool forceReload)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	newSceneName	Name of the new scene.
System.Boolean	forceReload	if set to `true`, force reload.

ClientConnectTo(String, Int32)

Attempts to connect to the target server, StartClient() must have been called before.

Declaration

```
public virtual void ClientConnectTo(string hostAddress, int hostPort)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	hostAddress	An IPv4 or IPv6 string containing the address of the server.
System.Int32	hostPort	An int representing the port to use for the connection.

FinishLoadScene()

Called when a scene has finished loading.

Declaration

```
public virtual void FinishLoadScene()
```

GetAmountOfRegisteredAssets()

Gets the amount of registered assets.

Declaration

```
public int GetAmountOfRegisteredAssets()
```

Returns

TYPE	DESCRIPTION
System.Int32	

GetAssetGUIDFromAssetId(Int32)

Gets the asset unique identifier from an asset identifier.

Declaration

```
public string GetAssetGUIDFromAssetId(int assetId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	assetId	The asset identifier.

Returns

TYPE	DESCRIPTION
System.String	

GetAssetIdFromAssetGUID(String)

Gets the asset identifier from an asset unique identifier.

The GUID is provided by Unity, the id is generated by TinyBirdNet for easier network handling.

Declaration

```
public int GetAssetIdFromAssetGUID(string assetGUID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	assetGUID	The asset unique identifier.

Returns

TYPE	DESCRIPTION
System.Int32	

GetAssetIdFromPrefab(GameObject)

Gets the asset identifier from a prefab.

Declaration

```
public int GetAssetIdFromPrefab(GameObject prefab)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject	prefab	The prefab.

Returns

TYPE	DESCRIPTION
System.Int32	

GetPrefabFromAssetGUID(String)

Gets the prefab from an asset unique identifier.

Declaration

```
public GameObject GetPrefabFromAssetGUID(string assetGUID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	assetGUID	The asset unique identifier.

Returns

TYPE	DESCRIPTION
UnityEngine.GameObject	

GetPrefabFromAssetId(Int32)

Gets the prefab from an asset identifier.

Declaration

```
public GameObject GetPrefabFromAssetId(int assetId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	assetId	The asset identifier.

Returns

TYPE	DESCRIPTION
UnityEngine.GameObject	

GetSpawnHandler(Int32, out SpawnDelegate)

Gets the spawn handler of an asset.

Declaration

```
public bool GetSpawnHandler(int assetIndex, out SpawnDelegate handler)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	assetIndex	Index of the asset.
SpawnDelegate	handler	The handler.

Returns

TYPE	DESCRIPTION
System.Boolean	

GetSpawnHandler(String, out SpawnDelegate)

Gets the spawn handler of an asset.

Declaration

```
public bool GetSpawnHandler(string assetGUID, out SpawnDelegate handler)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	assetGUID	The asset unique identifier.

TYPE	NAME	DESCRIPTION
SpawnDelegate	handler	The handler.

Returns

TYPE	DESCRIPTION
System.Boolean	

InvokeUnSpawnHandler(Int32, GameObject)

Invokes the unspawn handler of an asset.

Declaration

```
public bool InvokeUnSpawnHandler(int assetIndex, GameObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	assetIndex	Index of the asset.
UnityEngine.GameObject	obj	The object.

Returns

TYPE	DESCRIPTION
System.Boolean	

InvokeUnSpawnHandler(String, GameObject)

Invokes the unspawn handler of an asset.

Declaration

```
public bool InvokeUnSpawnHandler(string assetGUID, GameObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	assetGUID	The asset unique identifier.
UnityEngine.GameObject	obj	The object.

Returns

TYPE	DESCRIPTION
System.Boolean	

OnClientConnectToServer(TinyNetConnection)

Called when a client connect to the server.

Currently not implemented!

Declaration

```
public void OnClientConnectToServer(TinyNetConnection conn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection.

RebuildAllRegisteredPrefabs(GameObject[])

Receives a new list of registered prefabs from the custom Editor.

Declaration

```
public void RebuildAllRegisteredPrefabs(GameObject[] newArray)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject[]	newArray	

RegisterMessageHandlersClient()

Registers message handlers for the client.

Declaration

```
public virtual void RegisterMessageHandlersClient()
```

RegisterMessageHandlersServer()

Registers message handlers for the server.

Declaration

```
public virtual void RegisterMessageHandlersServer()
```

RegisterSpawnHandler(Int32, SpawnDelegate, UnSpawnDelegate)

Registers a spawn handler.

Declaration

```
public void RegisterSpawnHandler(int assetIndex, SpawnDelegate spawnHandler, UnSpawnDelegate unspawnHandler)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	assetIndex	Id of the asset.
SpawnDelegate	spawnHandler	The spawn handler.
UnSpawnDelegate	unspawnHandler	The unspawn handler.

ServerChangeScene(String)

Orders the server to change to the given scene.

Declaration

```
public virtual void ServerChangeScene(string newSceneName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	newSceneName	The name of the scene to change to.

SetMaxNumberOfPlayers(Int32)

> Changes the current max amount of players, this only has an effect before starting a Server.

Declaration

```
public virtual void SetMaxNumberOfPlayers(int newNumber)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newNumber	The new number.

SetPort(Int32)

Changes the port that will be used for hosting, this only has an effect before starting a Server.

Declaration

```
public virtual void SetPort(int newPort)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newPort	The new port.

StartClient()

Prepares this game to work as a client.

Declaration

```
public virtual void StartClient()
```

StartServer()

Prepares this game to work as a server.

Declaration

```
public virtual void StartServer()
```

StartVirtual()

Provides a function to be overrrided in case you need to add something in the Start call.

Declaration

```
protected virtual void StartVirtual()
```

ToggleNatPunching(Boolean)

Toggles the nat punching.

Declaration

```
public virtual void ToggleNatPunching(bool bNewState)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	bNewState	if set to `true` [b new state].

UnregisterSpawnHandler(Int32)

Unregisters a spawn handler.

Declaration

```
public void UnregisterSpawnHandler(int assetIndex)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	assetIndex	Id of the asset.

UpdateVirtual()

Provides a function to be overrrided in case you need to add something in the Update call.

Declaration

```
protected virtual void UpdateVirtual()
```

See Also

UnityEngine.MonoBehaviour

Class TinyNetGameManagerEditor

Custom inspector for the TinyNetGameManager class.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.ScriptableObject
UnityEditor.Editor
TinyNetGameManagerEditor

Inherited Members

UnityEditor.Editor.CreateEditorWithContext(UnityEngine.Object[], UnityEngine.Object, System.Type)
UnityEditor.Editor.CreateEditorWithContext(UnityEngine.Object[], UnityEngine.Object)
UnityEditor.Editor.CreateCachedEditorWithContext(UnityEngine.Object, UnityEngine.Object, System.Type, UnityEditor.Editor)
UnityEditor.Editor.CreateCachedEditorWithContext(UnityEngine.Object[], UnityEngine.Object, System.Type, UnityEditor.Editor)
UnityEditor.Editor.CreateCachedEditor(UnityEngine.Object, System.Type, UnityEditor.Editor)
UnityEditor.Editor.CreateCachedEditor(UnityEngine.Object[], System.Type, UnityEditor.Editor)
UnityEditor.Editor.CreateEditor(UnityEngine.Object)
UnityEditor.Editor.CreateEditor(UnityEngine.Object, System.Type)
UnityEditor.Editor.CreateEditor(UnityEngine.Object[])
UnityEditor.Editor.CreateEditor(UnityEngine.Object[], System.Type)
UnityEditor.Editor.DrawPropertiesExcluding(UnityEditor.SerializedObject, System.String[])
UnityEditor.Editor.DrawDefaultInspector()
UnityEditor.Editor.RequiresConstantRepaint()
UnityEditor.Editor.Repaint()
UnityEditor.Editor.HasPreviewGUI()
UnityEditor.Editor.GetPreviewTitle()
UnityEditor.Editor.RenderStaticPreview(System.String, UnityEngine.Object[], System.Int32, System.Int32)
UnityEditor.Editor.OnPreviewGUI(UnityEngine.Rect, UnityEngine.GUIStyle)
UnityEditor.Editor.OnInteractivePreviewGUI(UnityEngine.Rect, UnityEngine.GUIStyle)
UnityEditor.Editor.OnPreviewSettings()
UnityEditor.Editor.GetInfoString()
UnityEditor.Editor.ReloadPreviewInstances()
UnityEditor.Editor.DrawHeader()
UnityEditor.Editor.OnHeaderGUI()
UnityEditor.Editor.DrawPreview(UnityEngine.Rect)
UnityEditor.Editor.UseDefaultMargins()
UnityEditor.Editor.Initialize(UnityEngine.Object[])
UnityEditor.Editor.MoveNextTarget()
UnityEditor.Editor.ResetTarget()
UnityEditor.Editor.target
UnityEditor.Editor.targets
UnityEditor.Editor.serializedObject
UnityEngine.ScriptableObject.SetDirty()
UnityEngine.ScriptableObject.CreateInstance(System.String)
UnityEngine.ScriptableObject.CreateInstance(System.Type)
UnityEngine.ScriptableObject.CreateInstance<T>()
UnityEngine.Object.Destroy(UnityEngine.Object, System.Single)
UnityEngine.Object.Destroy(UnityEngine.Object)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object, System.Boolean)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object)
UnityEngine.Object.FindObjectsOfType(System.Type)

UnityEngine.Object.DontDestroyOnLoad(UnityEngine.Object)
UnityEngine.Object.DestroyObject(UnityEngine.Object, System.Single)
UnityEngine.Object.DestroyObject(UnityEngine.Object)
UnityEngine.Object.FindSceneObjectsOfType(System.Type)
UnityEngine.Object.FindObjectsOfTypeIncludingAssets(System.Type)
UnityEngine.Object.FindObjectsOfTypeAll(System.Type)
UnityEngine.Object.ToString()
UnityEngine.Object.GetInstanceID()
UnityEngine.Object.GetHashCode()
UnityEngine.Object.Equals(System.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Instantiate<T>(T)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.FindObjectsOfType<T>()
UnityEngine.Object.FindObjectOfType<T>()
UnityEngine.Object.FindObjectOfType(System.Type)
UnityEngine.Object.name
UnityEngine.Object.hideFlags
System.Object.Equals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
[CustomEditor(typeof (TinyNetGameManager), true)]
public class TinyNetGameManagerEditor : Editor, IPreviewable, IToolModeOwner
```

Methods

OnInspectorGUI()

Declaration

```
public override void OnInspectorGUI()
```

Overrides

UnityEditor.Editor.OnInspectorGUI()

See Also

UnityEditor.Editor

Class TinyNetIdentity

Any UnityEngine.GameObject that contains this component, can be spawned accross the network.

This is basically a container for an "universal id" accross the network.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
TinyNetIdentity

Implements

ITinyNetInstanceID

Inherited Members

UnityEngine.MonoBehaviour.Invoke(System.String, System.Single)
UnityEngine.MonoBehaviour.InvokeRepeating(System.String, System.Single, System.Single)
UnityEngine.MonoBehaviour.CancelInvoke()
UnityEngine.MonoBehaviour.CancelInvoke(System.String)
UnityEngine.MonoBehaviour.IsInvoking(System.String)
UnityEngine.MonoBehaviour.IsInvoking()
UnityEngine.MonoBehaviour.StartCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine_Auto(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine(System.String, System.Object)
UnityEngine.MonoBehaviour.StartCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StopCoroutine(UnityEngine.Coroutine)
UnityEngine.MonoBehaviour.StopAllCoroutines()
UnityEngine.MonoBehaviour.print(System.Object)
UnityEngine.MonoBehaviour.useGUILayout
UnityEngine.MonoBehaviour.runInEditMode
UnityEngine.Behaviour.enabled
UnityEngine.Behaviour.isActiveAndEnabled
UnityEngine.Component.GetComponent(System.Type)
UnityEngine.Component.GetComponent<T>()
UnityEngine.Component.GetComponent(System.String)
UnityEngine.Component.GetComponentInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentInChildren(System.Type)
UnityEngine.Component.GetComponentInChildren<T>()
UnityEngine.Component.GetComponentInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren(System.Type)
UnityEngine.Component.GetComponentsInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInChildren<T>()
UnityEngine.Component.GetComponentsInChildren<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentInParent(System.Type)
UnityEngine.Component.GetComponentInParent<T>()
UnityEngine.Component.GetComponentsInParent(System.Type)
UnityEngine.Component.GetComponentsInParent(System.Type, System.Boolean)

UnityEngine.Component.GetComponentsInParent<T>(System.Boolean)
UnityEngine.Component.GetComponentsInParent<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInParent<T>()
UnityEngine.Component.GetComponents(System.Type)
UnityEngine.Component.GetComponents(System.Type, System.Collections.Generic.List<UnityEngine.Component>)
UnityEngine.Component.GetComponents<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponents<T>()
UnityEngine.Component.CompareTag(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object)
UnityEngine.Component.SendMessageUpwards(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object)
UnityEngine.Component.SendMessage(System.String)
UnityEngine.Component.SendMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object)
UnityEngine.Component.BroadcastMessage(System.String)
UnityEngine.Component.BroadcastMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.transform
UnityEngine.Component.gameObject
UnityEngine.Component.tag
UnityEngine.Component.rigidbody
UnityEngine.Component.rigidbody2D
UnityEngine.Component.camera
UnityEngine.Component.light
UnityEngine.Component.animation
UnityEngine.Component.constantForce
UnityEngine.Component.renderer
UnityEngine.Component.audio
UnityEngine.Component.guiText
UnityEngine.Component.networkView
UnityEngine.Component.guiElement
UnityEngine.Component.guiTexture
UnityEngine.Component.collider
UnityEngine.Component.collider2D
UnityEngine.Component.hingeJoint
UnityEngine.Component.particleEmitter
UnityEngine.Component.particleSystem
UnityEngine.Object.Destroy(UnityEngine.Object, System.Single)
UnityEngine.Object.Destroy(UnityEngine.Object)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object, System.Boolean)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object)
UnityEngine.Object.FindObjectsOfType(System.Type)
UnityEngine.Object.DontDestroyOnLoad(UnityEngine.Object)
UnityEngine.Object.DestroyObject(UnityEngine.Object, System.Single)
UnityEngine.Object.DestroyObject(UnityEngine.Object)
UnityEngine.Object.FindSceneObjectsOfType(System.Type)
UnityEngine.Object.FindObjectsOfTypeIncludingAssets(System.Type)
UnityEngine.Object.FindObjectsOfTypeAll(System.Type)
UnityEngine.Object.ToString()

UnityEngine.Object.GetInstanceID()
UnityEngine.Object.GetHashCode()
UnityEngine.Object.Equals(System.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Instantiate<T>(T)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.FindObjectsOfType<T>()
UnityEngine.Object.FindObjectOfType<T>()
UnityEngine.Object.FindObjectOfType(System.Type)
UnityEngine.Object.name
UnityEngine.Object.hideFlags
System.Object.Equals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
[ExecuteInEditMode]
[DisallowMultipleComponent]
[AddComponentMenu("TinyBirdNet/TinyNetIdentity")]
public class TinyNetIdentity : MonoBehaviour, ITinyNetInstanceID
```

Fields

bStartClientTwiceTest

Used as a stopgag in case this object has tried to be initialized twice.

Declaration

```
protected bool bStartClientTwiceTest
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Properties

assetGUID

Gets the asset unique identifier.

Declaration

```
public string assetGUID { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	The asset unique identifier.

connectionToOwnerClient

[Server Only] Gets the connection to owner client.

Declaration

```
public TinyNetConnection connectionToOwnerClient { get; }
```

Property Value

TYPE	DESCRIPTION
TinyNetConnection	The connection to owner client.

hasAuthority

Gets a value indicating whether this instance has authority.

Declaration

```
public bool hasAuthority { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance has authority; otherwise, `false`.

isClient

Declaration

```
public bool isClient { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

isServer

Declaration

```
public bool isServer { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

NetworkID

The ID of an instance in the network, given by the server on spawn.

Declaration

```
public int NetworkID { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

sceneID

Gets the object scene identifier.

Declaration

```
public int sceneID { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	The object scene identifier.

ServerOnly

Gets a value indicating whether this object only exists on the server.

Declaration

```
public bool ServerOnly { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if server only; otherwise, `false`.

Methods

AssignClientAuthority(TinyNetConnection)

[Server only] Assigns the client authority.

Declaration

```
public bool AssignClientAuthority(TinyNetConnection conn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection.

Returns

TYPE	DESCRIPTION
System.Boolean	

DeserializeAllTinyNetObjects(NetDataReader, Boolean)

Deserializes all ITinyNetObject data.

Declaration

```
public void DeserializeAllTinyNetObjects(NetDataReader reader, bool bInitialState)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	The reader.
System.Boolean	bInitialState	if set to `true` [b initial state].

ForceAuthority(Boolean)

Forces the authority setting.

Declaration

```
public void ForceAuthority(bool authority)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	authority	if set to `true` it will have authority.

ForceSceneId(Int32)

Forces the scene identifier. Only used when fixing duplicate scene IDs duing post-processing

Declaration

```
public void ForceSceneId(int newSceneId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newSceneId	The new scene identifier.

OnGiveAuthority()

Called when [give authority].

Declaration

```
public virtual void OnGiveAuthority()
```

OnNetworkCreate()

Called when this object is created.

Declaration

```
public virtual void OnNetworkCreate()
```

OnNetworkDestroy()

Called when destroyed by the network.

Declaration

```
public virtual void OnNetworkDestroy()
```

OnRemoveAuthority()

Called when [remove authority].

Declaration

```
public virtual void OnRemoveAuthority()
```

OnSetLocalVisibility(Boolean)

Called when [set local visibility].

Declaration

```
public virtual void OnSetLocalVisibility(bool vis)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	vis	if set to `true` [vis].

OnStartAuthority()

Called when [start authority].

Declaration

```
public virtual void OnStartAuthority()
```

OnStartClient()

Called when an object is spawned on the client.

Declaration

```
public void OnStartClient()
```

OnStartLocalPlayer()

Called when [start local player].

Declaration

```
public virtual void OnStartLocalPlayer()
```

OnStartServer(Boolean)

Called when an object is spawned on the server.

Declaration

```
public void OnStartServer(bool allowNonZeroNetId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	allowNonZeroNetId	If the object already have a NetworkId, it was probably recycled.

OnStopAuthority()

Called when [stop authority].

Declaration

```
public virtual void OnStopAuthority()
```

ReceiveNetworkID(Int32)

Receives the network identifier.

Declaration

```
public void ReceiveNetworkID(int newID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newID	The new identifier.

RemoveClientAuthority(TinyNetConnection)

[Server only] Removes the client authority.

Not implemmented yet.

Declaration

```
public bool RemoveClientAuthority(TinyNetConnection conn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection.

Returns

TYPE	DESCRIPTION
System.Boolean	

SerializeAllTinyNetObjects(NetDataWriter)

Called on the server to serialize all ITinyNetObject attached to this prefab.

Declaration

```
public void SerializeAllTinyNetObjects(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

SetConnectionToClient(TinyNetConnection, Int16)

Used by the server to have a shortcut in the case a client owns this object.

Not implemmented yet.

Declaration

```
public void SetConnectionToClient(TinyNetConnection conn, short newPlayerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection that owns this object.
System.Int16	newPlayerControllerId	The player controller identifier that owns this object.

SetDynamicAssetGUID(String)

Sets the asset unique identifier during play.

Declaration

```
public void SetDynamicAssetGUID(string newAssetGUID)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	newAssetGUID	The new asset unique identifier.

Implements

ITinyNetInstanceID

See Also

UnityEngine.MonoBehaviour

ITinyNetInstanceID

Class TinyNetLogLevel

A simple log filter level to use in debug logs.

Inheritance

System.Object
TinyNetLogLevel

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetLogLevel
```

Fields

currentLevel

Declaration

```
public static LogFilter currentLevel
```

Field Value

TYPE	DESCRIPTION
LogFilter	

Properties

logDebug

Declaration

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

Property Value

TYPE	DESCRIPTION
System.Boolean	

logDev

Declaration

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

Property Value

TYPE	DESCRIPTION
System.Boolean	

logError

Declaration

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

Property Value

TYPE	DESCRIPTION
System.Boolean	

logInfo

Declaration

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

Property Value

TYPE	DESCRIPTION
System.Boolean	

logWarn

Declaration

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

Property Value

TYPE	DESCRIPTION
System.Boolean	

Class TinyNetMessageHandlers

A class that represents a container for TinyNetMessageDelegate.

Inheritance

System.Object
TinyNetMessageHandlers

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetMessageHandlers
```

Methods

Contains(UInt16)

Determines whether this instance contains a handler for the specified ITinyNetMessage type.

Declaration

```
public bool Contains(ushort msgType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	msgType	Type of the ITinyNetMessage.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if it contains a handler for it; otherwise, `false`.

RegisterHandler(UInt16, TinyNetMessageDelegate)

Registers a handler for a message, it will not check for conflicts, but cannot be used for system messages.

Declaration

```
public void RegisterHandler(ushort msgType, TinyNetMessageDelegate handler)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	msgType	Type of the ITinyNetMessage.
TinyNetMessageDelegate	handler	The delegate.

UnregisterHandler(UInt16)

Unregisters a handler.

Declaration

```
public void UnregisterHandler(ushort msgType)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	msgType	Type of the ITinyNetMessage.

Class TinyNetPlayerController

This class represents the player entity in a network game, there can be multiple players per client, when there are multiple people playing on one machine.

The server has one TinyNetConnection per NetPeer.

Inheritance

System.Object
TinyNetPlayerController

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetPlayerController
```

Constructors

TinyNetPlayerController()

Initializes a new instance of the TinyNetPlayerController class.

Declaration

```
public TinyNetPlayerController()
```

TinyNetPlayerController(Int16, TinyNetConnection)

Initializes a new instance of the TinyNetPlayerController class.

Declaration

```
public TinyNetPlayerController(short playerControllerId, TinyNetConnection nConn)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	playerControllerId	The player controller identifier.
TinyNetConnection	nConn	The TinyNetConnection.

Fields

conn

Holds a reference to the client connection on the server, and to the server connection on the client.

In a Listen Server this will only hold a reference to the client connection.

Declaration

```
protected TinyNetConnection conn
```

Field Value

TYPE	DESCRIPTION
TinyNetConnection	

inputWriter

A stream used for input.

Declaration

```
protected static NetDataWriter inputWriter
```

Field Value

TYPE	DESCRIPTION
NetDataWriter	

playerControllerId

The player controller identifier

Declaration

```
public short playerControllerId
```

Field Value

TYPE	DESCRIPTION
System.Int16	

Properties

Conn

Gets or sets the connection.

Declaration

```
public virtual TinyNetConnection Conn { get; set; }
```

Property Value

TYPE	DESCRIPTION
TinyNetConnection	The connection.

IsValid

Returns true if this instance is valid.

Declaration

```
public bool IsValid { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	`true` if this instance is valid; otherwise, `false`.

Methods

GetInputMessage(TinyNetMessageReader)

Receives an input message

Declaration

```
public virtual void GetInputMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	The message reader.

ToString()

Returns a System.String that represents this instance.

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
System.String	A System.String that represents this instance.

Overrides

System.Object.ToString()

Class TinyNetPropertyAccessor<T>

Creates an acessor for a property, used for TinyNetSyncVar.

Inheritance

System.Object
TinyNetPropertyAccessor<T>

Inherited Members

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

Namespace: TinyBirdNet
Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetPropertyAccessor<T>
```

Type Parameters

NAME	DESCRIPTION
T	The System.Type of this property.

Constructors

TinyNetPropertyAccessor(Object, String)

Initializes a new instance of the TinyNetPropertyAccessor<T> class.

Declaration

```
public TinyNetPropertyAccessor(object obj, string newPropName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	The object that owns the property.
System.String	newPropName	New name of the property.

Methods

CheckIfChangedAndUpdate(Object)

Checks if the value has changed and then updates the TinyBirdNet.TinyNetPropertyAccessor`1.previousValue.

Declaration

```
public bool CheckIfChangedAndUpdate(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	The object.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if this property value has changed since the last time it was checked; otherwise, `false`.

Get(Object)

Gets the property value.

Declaration

```
public T Get(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	The object.

Returns

TYPE	DESCRIPTION
T	The property value.

Set(Object, T)

Sets the property value.

Declaration

```
public void Set(object obj, T value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	The object.
T	value	The value.

UpdateValue(Object)

Updates the TinyBirdNet.TinyNetPropertyAccessor`1.previousValue.

Declaration

```
public void UpdateValue(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	The object.

WasChanged(Object)

Checks if the property value has changed.

Declaration

```
public bool WasChanged(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	obj	The object.

Returns

TYPE	DESCRIPTION
System.Boolean	`true` if this property value has changed since the last time it was checked; otherwise, `false`.

Class TinyNetReflector

This class is used to get all TinyNetSyncVar properties and TinyNetRPC methods and store their info.

Inheritance

System.Object
TinyNetReflector

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public static class TinyNetReflector
```

Methods

GetAllClassesAndChildsOf<T>()

Gets all classes and childs of the given class.

Declaration

```
public static List<Type> GetAllClassesAndChildsOf<T>()where T : class
```

Returns

TYPE	DESCRIPTION
System.Collections.Generic.List<System.Type>	

Type Parameters

NAME	DESCRIPTION
T	

GetAllSyncVarProps()

Does the reflection process.

Declaration

```
public static void GetAllSyncVarProps()
```

Class TinyNetRPC

When used on a method, allows it to be executed remotely on another machine when called.

Inheritance

System.Object
System.Attribute
TinyNetRPC

Implements

System.Runtime.InteropServices._Attribute

Inherited Members

System.Attribute.System.Runtime.InteropServices._Attribute.GetIDsOfNames(System.Guid, System.IntPtr, System.UInt32, System.UInt32, System.IntPtr)
System.Attribute.System.Runtime.InteropServices._Attribute.GetTypeInfo(System.UInt32, System.UInt32, System.IntPtr)
System.Attribute.System.Runtime.InteropServices._Attribute.GetTypeInfoCount(System.UInt32)
System.Attribute.System.Runtime.InteropServices._Attribute.Invoke(System.UInt32, System.Guid, System.UInt32, System.Int16, System.IntPtr, System.IntPtr, System.IntPtr, System.IntPtr)
System.Attribute.GetCustomAttribute(System.Reflection.ParameterInfo, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.MemberInfo, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.Assembly, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.Module, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.Module, System.Type, System.Boolean)
System.Attribute.GetCustomAttribute(System.Reflection.Assembly, System.Type, System.Boolean)
System.Attribute.GetCustomAttribute(System.Reflection.ParameterInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttribute(System.Reflection.MemberInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo)
System.Attribute.GetCustomAttributes(System.Reflection.Module)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.Module, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Module, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Module, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo, System.Boolean)
System.Attribute.GetHashCode()
System.Attribute.IsDefaultAttribute()
System.Attribute.IsDefined(System.Reflection.Module, System.Type)
System.Attribute.IsDefined(System.Reflection.ParameterInfo, System.Type)
System.Attribute.IsDefined(System.Reflection.MemberInfo, System.Type)
System.Attribute.IsDefined(System.Reflection.Assembly, System.Type)
System.Attribute.IsDefined(System.Reflection.MemberInfo, System.Type, System.Boolean)
System.Attribute.IsDefined(System.Reflection.Assembly, System.Type, System.Boolean)
System.Attribute.IsDefined(System.Reflection.Module, System.Type, System.Boolean)
System.Attribute.IsDefined(System.Reflection.ParameterInfo, System.Type, System.Boolean)

System.Attribute.Match(System.Object)
System.Attribute.Equals(System.Object)
System.Attribute.TypeId
System.Object.Equals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ToString()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = false, Inherited = true)]
public class TinyNetRPC : Attribute, _Attribute
```

Constructors

TinyNetRPC(RPCTarget, RPCCallers)

Declaration

```
public TinyNetRPC(RPCTarget newTargets, RPCCallers newCallers)
```

Parameters

TYPE	NAME	DESCRIPTION
RPCTarget	newTargets	
RPCCallers	newCallers	

Properties

Callers

Declaration

```
public RPCCallers Callers { get; }
```

Property Value

TYPE	DESCRIPTION
RPCCallers	

SendOption

Declaration

```
public DeliveryMethod SendOption { get; set; }
```

Property Value

TYPE	DESCRIPTION
DeliveryMethod	

Targets

Declaration

```
public RPCTarget Targets { get; }
```

Property Value

TYPE	DESCRIPTION
RPCTarget	

Implements

System.Runtime.InteropServices._Attribute

See Also

System.Attribute

Class TinyNetScene

Represents a Scene, which is all data required to reproduce the game state.

Inheritance

System.Object
TinyNetScene
TinyNetClient
TinyNetServer

Implements

INetEventListener

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class TinyNetScene : INetEventListener
```

Constructors

TinyNetScene()

Initializes a new instance of the TinyNetScene class.

Declaration

```
public TinyNetScene()
```

Fields

_localIdentityObjects

int is the NetworkID of the TinyNetIdentity object.

Declaration

```
protected static Dictionary<int, TinyNetIdentity> _localIdentityObjects
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.Int32, TinyNetIdentity>	

_localNetObjects

int is the NetworkID of the ITinyNetObject object.

Declaration

```
protected static Dictionary<int, ITinyNetObject> _localNetObjects
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.Int32, ITinyNetObject>	

_netManager

The NetManager.

Declaration

```
protected NetManager _netManager
```

Field Value

TYPE	DESCRIPTION
NetManager	

_tinyMessageHandlers

The ITinyNetMessage handlers.

Declaration

```
protected TinyNetMessageHandlers _tinyMessageHandlers
```

Field Value

TYPE	DESCRIPTION
TinyNetMessageHandlers	

_tinyNetConns

All connections to this scene.

Declaration

```
protected List<TinyNetConnection> _tinyNetConns
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<TinyNetConnection>	

createPlayerAction

If set, overrides the CreatePlayerAndAdd(TinyNetConnection, Int32) implementation.

Declaration

```
public static Action<TinyNetConnection, int> createPlayerAction
```

Field Value

TYPE	DESCRIPTION
System.Action<TinyNetConnection, System.Int32>	

currentFixedFrame

The current fixed frame, used for calculation the network state update frequency.

Declaration

```
protected int currentFixedFrame
```

Field Value

TYPE	DESCRIPTION
System.Int32	

recycleMessageReader

A message reader used to prevent garbage collection.

Declaration

```
protected static TinyNetMessageReader recycleMessageReader
```

Field Value

TYPE	DESCRIPTION
TinyNetMessageReader	

recycleWriter

If using this, always Reset before use!

Declaration

```
protected static NetDataWriter recycleWriter
```

Field Value

TYPE	DESCRIPTION
NetDataWriter	

s_TineNetObjectSpawnFinishedMessage

Declaration

```
protected static TinyNetObjectSpawnFinishedMessage s_TineNetObjectSpawnFinishedMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetObjectSpawnFinishedMessage	

s_TinyNetAddPlayerMessage

Declaration

```
protected static TinyNetAddPlayerMessage s_TinyNetAddPlayerMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetAddPlayerMessage	

s_TinyNetClientAuthorityMessage

Declaration

```
protected static TinyNetClientAuthorityMessage s_TinyNetClientAuthorityMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetClientAuthorityMessage	

s_TinyNetObjectDestroyMessage

Declaration

```
protected static TinyNetObjectDestroyMessage s_TinyNetObjectDestroyMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetObjectDestroyMessage	

s_TinyNetObjectHideMessage

Declaration

```
protected static TinyNetObjectHideMessage s_TinyNetObjectHideMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetObjectHideMessage	

s_TinyNetObjectSpawnMessage

Declaration

```
protected static TinyNetObjectSpawnMessage s_TinyNetObjectSpawnMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetObjectSpawnMessage	

s_TinyNetObjectSpawnSceneMessage

Declaration

```
protected static TinyNetObjectSpawnSceneMessage s_TinyNetObjectSpawnSceneMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetObjectSpawnSceneMessage	

s_TinyNetRemovePlayerMessage

Declaration

```
protected static TinyNetRemovePlayerMessage s_TinyNetRemovePlayerMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetRemovePlayerMessage	

s_TinyNetRequestAddPlayerMessage

Declaration

```
protected static TinyNetRequestAddPlayerMessage s_TinyNetRequestAddPlayerMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetRequestAddPlayerMessage	

s_TinyNetRequestRemovePlayerMessage

Declaration

```
protected static TinyNetRequestRemovePlayerMessage s_TinyNetRequestRemovePlayerMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetRequestRemovePlayerMessage	

s_TinyNetRPCMessage

Declaration

```
protected static TinyNetRPCMessage s_TinyNetRPCMessage
```

Field Value

TYPE	DESCRIPTION
TinyNetRPCMessage	

Properties

connToHost

Gets or sets the connection to host.

Declaration

```
public TinyNetConnection connToHost { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
TinyNetConnection	The connection to host.

isConnected

Returns true if it's connected to at least one peer.

Declaration

```
public bool isConnected { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

isRunning

Returns true if socket is listening and update thread is running.

Declaration

```
public bool isRunning { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

tinyNetConns

Gets the connections to this scene.

Declaration

```
public List<TinyNetConnection> tinyNetConns { get; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<TinyNetConnection>	The connection list.

TYPE

Sugar for generating debug logs.

Declaration

```
public virtual string TYPE { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Methods

AddPlayerControllerToConnection(TinyNetConnection, Int32)

Attempts to add a player controller to the connection.

Declaration

```
protected virtual void AddPlayerControllerToConnection(TinyNetConnection conn, int playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection.
System.Int32	playerControllerId	The player controller identifier.

AddTinyNetIdentityToList(TinyNetIdentity)

Adds the TinyNetIdentity to list.

Declaration

```
public static void AddTinyNetIdentityToList(TinyNetIdentity netIdentity)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	netIdentity	The net identity.

AddTinyNetObjectToList(ITinyNetObject)

Adds the ITinyNetObject to list.

Declaration

```
public static void AddTinyNetObjectToList(ITinyNetObject netObj)
```

Parameters

TYPE	NAME	DESCRIPTION
ITinyNetObject	netObj	The net object.

ClearNetManager()

Clears the net manager.

Declaration

```
public virtual void ClearNetManager()
```

ConfigureNetManager(Boolean)

Configures the net manager.

Declaration

```
protected virtual void ConfigureNetManager(bool bUseFixedTime)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	bUseFixedTime	if set to `true` use fixed update time.

CreatePlayerAndAdd(TinyNetConnection, Int32)

Creates a player controller and adds it to the connection.

Declaration

```
protected virtual void CreatePlayerAndAdd(TinyNetConnection conn, int playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection.
System.Int32	playerControllerId	The player controller identifier.

CreateTinyNetConnection(NetPeer)

Creates a TinyNetConnection for the given NetPeer.

Declaration

```
protected virtual TinyNetConnection CreateTinyNetConnection(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	The peer.

Returns

TYPE	DESCRIPTION
TinyNetConnection	

GetTinyNetConnection(NetPeer)

Returns the TinyNetConnection with the given NetPeer.

Declaration

```
protected TinyNetConnection GetTinyNetConnection(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	The peer.

Returns

TYPE	DESCRIPTION
TinyNetConnection	

GetTinyNetConnection(Int64)

Returns the TinyNetConnection with the given connection identifier.

Declaration

```
protected TinyNetConnection GetTinyNetConnection(long connId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64	connId	The connection identifier.

Returns

TYPE	DESCRIPTION
TinyNetConnection	

GetTinyNetIdentityByNetworkID(Int32)

Gets a TinyNetIdentity by it's network identifier.

Declaration

```
public static TinyNetIdentity GetTinyNetIdentityByNetworkID(int nId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	nId	The NetworkID.

Returns

TYPE	DESCRIPTION
TinyNetIdentity	

GetTinyNetObjectByNetworkID(Int32)

Gets a ITinyNetObject by it's network identifier.

Declaration

```
public static ITinyNetObject GetTinyNetObjectByNetworkID(int nId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	nId	The NetworkID.

Returns

TYPE	DESCRIPTION
ITinyNetObject	

InternalUpdate()

It is called from TinyNetGameManager Update(), handles PollEvents().

Declaration

```
public virtual void InternalUpdate()
```

OnConnectionCreated(TinyNetConnection)

Called after a peer has connected and a TinyNetConnection was created for it.

Declaration

```
protected virtual void OnConnectionCreated(TinyNetConnection nConn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	nConn	The connection created.

OnConnectionRequest(ConnectionRequest)

On peer connection requested

Declaration

```
public virtual void OnConnectionRequest(ConnectionRequest request)
```

Parameters

TYPE	NAME	DESCRIPTION
ConnectionRequest	request	Request information (EndPoint, internal id, additional data)

OnDisconnect(TinyNetConnection)

Called after a peer has been disconnected but before the TinyNetConnection has been removed from the list.

Declaration

```
protected virtual void OnDisconnect(TinyNetConnection nConn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	nConn	The connection that disconnected.

OnDiscoveryRequestReceived(NetEndPoint, NetDataReader)

Called when a discovery request is received.

Declaration

```
protected virtual void OnDiscoveryRequestReceived(NetEndPoint remoteEndPoint, NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	remoteEndPoint	The remote end point.
NetDataReader	reader	The reader.

OnNetworkError(NetEndPoint, Int32)

Network error (on send or receive)

Declaration

```
public virtual void OnNetworkError(NetEndPoint endPoint, int socketErrorCode)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	endPoint	From endPoint (can be null)
System.Int32	socketErrorCode	Socket error code

OnNetworkLatencyUpdate(NetPeer, Int32)

Latency information updated

Declaration

```
public virtual void OnNetworkLatencyUpdate(NetPeer peer, int latency)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	Peer with updated latency
System.Int32	latency	latency value in milliseconds

OnNetworkReceive(NetPeer, NetDataReader, DeliveryMethod)

Received some data

Declaration

```
public virtual void OnNetworkReceive(NetPeer peer, NetDataReader reader, DeliveryMethod deliveryMethod)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	From peer
NetDataReader	reader	DataReader containing all received data
DeliveryMethod	deliveryMethod	Type of received packet

OnNetworkReceiveUnconnected(NetEndPoint, NetDataReader, UnconnectedMessageType)

Received unconnected message

Declaration

```
public virtual void OnNetworkReceiveUnconnected(NetEndPoint remoteEndPoint, NetDataReader reader,
UnconnectedMessageType messageType)
```

Parameters

TYPE	NAME	DESCRIPTION
NetEndPoint	remoteEndPoint	From address (IP and Port)
NetDataReader	reader	Message data

TYPE	NAME	DESCRIPTION
UnconnectedMessageType	messageType	Message type (simple, discovery request or responce)

OnPeerConnected(NetPeer)

New remote peer connected to host, or client connected to remote host

Declaration

```
public virtual void OnPeerConnected(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	Connected peer object

OnPeerDisconnected(NetPeer, DisconnectInfo)

Peer disconnected

Declaration

```
public virtual void OnPeerDisconnected(NetPeer peer, DisconnectInfo disconnectInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	disconnected peer
DisconnectInfo	disconnectInfo	additional info about reason, errorCode or data received with disconnect message

OnRPCMessage(TinyNetMessageReader)

Called when an RPC message is received.

Declaration

```
protected virtual void OnRPCMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	The net message.

RegisterHandler(UInt16, TinyNetMessageDelegate)

Registers a message handler.

Declaration

```
public void RegisterHandler(ushort msgType, TinyNetMessageDelegate handler)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	msgType	Type of the message.
TinyNetMessageDelegate	handler	The handler.

RegisterHandlerSafe(UInt16, TinyNetMessageDelegate)

Registers a message handler safely.

Declaration

```
public void RegisterHandlerSafe(ushort msgType, TinyNetMessageDelegate handler)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	msgType	Type of the message.
TinyNetMessageDelegate	handler	The handler.

RegisterMessageHandlers()

Registers the message handlers.

Declaration

```
protected virtual void RegisterMessageHandlers()
```

RemovePlayerControllerFromConnection(TinyNetConnection, Int16)

Removes a player controller from connection.

Declaration

```
protected virtual void RemovePlayerControllerFromConnection(TinyNetConnection conn, short playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	conn	The connection.
System.Int16	playerControllerId	The player controller identifier.

RemoveTinyNetConnection(NetPeer)

Removes the connection.

Declaration

```
protected virtual bool RemoveTinyNetConnection(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	The peer.

Returns

TYPE	DESCRIPTION
System.Boolean	

RemoveTinyNetConnection(Int64)

Removes the connection.

Declaration

```
protected virtual bool RemoveTinyNetConnection(long connectId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64	connectId	The connection identifier.

Returns

TYPE	DESCRIPTION
System.Boolean	

RemoveTinyNetConnection(TinyNetConnection)

Removes the connection.

Declaration

```
protected virtual bool RemoveTinyNetConnection(TinyNetConnection nConn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetConnection	nConn	The connection.

Returns

TYPE	DESCRIPTION
System.Boolean	

RemoveTinyNetIdentityFromList(TinyNetIdentity)

Removes the TinyNetIdentity from the list.

Declaration

```
public static void RemoveTinyNetIdentityFromList(TinyNetIdentity netIdentity)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	netIdentity	The net identity.

RemoveTinyNetObjectFromList(ITinyNetObject)

Removes the ITinyNetObject from the list.

Declaration

```
public static void RemoveTinyNetObjectFromList(ITinyNetObject netObj)
```

Parameters

TYPE	NAME	DESCRIPTION
ITinyNetObject	netObj	The net object.

SendMessageByChannelToAllConnections(ITinyNetMessage, DeliveryMethod)

Sends the message by a specific channel to all connections.

Declaration

```
public virtual void SendMessageByChannelToAllConnections(ITinyNetMessage msg, DeliveryMethod sendOptions)
```

Parameters

TYPE	NAME	DESCRIPTION
ITinyNetMessage	msg	The message.
DeliveryMethod	sendOptions	The send options.

SendMessageByChannelToAllObserversOf(TinyNetIdentity, ITinyNetMessage, DeliveryMethod)

Sends the message by a specific channel to all observers of a TinyNetIdentity.

Declaration

```
public virtual void SendMessageByChannelToAllObserversOf(TinyNetIdentity tni, ITinyNetMessage msg,
DeliveryMethod sendOptions)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tni	The TinyNetIdentity.
ITinyNetMessage	msg	The message.
DeliveryMethod	sendOptions	The send options.

SendMessageByChannelToAllReadyConnections(ITinyNetMessage, DeliveryMethod)

Sends the message by a specific channel to all ready connections.

Declaration

```
public virtual void SendMessageByChannelToAllReadyConnections(ITinyNetMessage msg, DeliveryMethod sendOptions)
```

Parameters

TYPE	NAME	DESCRIPTION
ITinyNetMessage	msg	The message.
DeliveryMethod	sendOptions	The send options.

SendMessageByChannelToHost(ITinyNetMessage, DeliveryMethod)

Sends the message by a specific channel to host.

Declaration

```
public virtual void SendMessageByChannelToHost(ITinyNetMessage msg, DeliveryMethod sendOptions)
```

Parameters

TYPE	NAME	DESCRIPTION
ITinyNetMessage	msg	The message.
DeliveryMethod	sendOptions	The send options.

SendMessageByChannelToTargetConnection(ITinyNetMessage, DeliveryMethod, TinyNetConnection)

Sends the message by a specific channel to target connection.

Declaration

```
public virtual void SendMessageByChannelToTargetConnection(ITinyNetMessage msg, DeliveryMethod sendOptions,
TinyNetConnection tinyNetConn)
```

Parameters

TYPE	NAME	DESCRIPTION
ITinyNetMessage	msg	The message.
DeliveryMethod	sendOptions	The send options.
TinyNetConnection	tinyNetConn	The connection.

SetPingInterval(Int32)

Sets the ping interval.

Declaration

```
public virtual void SetPingInterval(int newPingInterval)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	newPingInterval	The new ping interval.

TinyNetUpdate()

Run every frame, called from TinyNetGameManager.

Declaration

```
public virtual void TinyNetUpdate()
```

ToggleNatPunching(Boolean)

Toggles the nat punching.

Declaration

```
public virtual void ToggleNatPunching(bool bNewState)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	bNewState	The new nat punching state.

Implements

INetEventListener

See Also

INetEventListener

Class TinyNetServer

Represents the Scene of a server.

Inheritance

System.Object
TinyNetScene
TinyNetServer

Implements

INetEventListener

Inherited Members

TinyNetScene.createPlayerAction
TinyNetScene._localIdentityObjects
TinyNetScene._localNetObjects
TinyNetScene.recycleWriter
TinyNetScene.recycleMessageReader
TinyNetScene.s_TinyNetRPCMessage
TinyNetScene.s_TinyNetObjectHideMessage
TinyNetScene.s_TinyNetObjectDestroyMessage
TinyNetScene.s_TinyNetObjectSpawnMessage
TinyNetScene.s_TinyNetObjectSpawnSceneMessage
TinyNetScene.s_TineNetObjectSpawnFinishedMessage
TinyNetScene.s_TinyNetAddPlayerMessage
TinyNetScene.s_TinyNetRemovePlayerMessage
TinyNetScene.s_TinyNetRequestAddPlayerMessage
TinyNetScene.s_TinyNetRequestRemovePlayerMessage
TinyNetScene.s_TinyNetClientAuthorityMessage
TinyNetScene._tinyMessageHandlers
TinyNetScene._tinyNetConns
TinyNetScene.tinyNetConns
TinyNetScene.connToHost
TinyNetScene._netManager
TinyNetScene.currentFixedFrame
TinyNetScene.isRunning
TinyNetScene.isConnected
TinyNetScene.RegisterHandler(UInt16, TinyNetMessageDelegate)
TinyNetScene.RegisterHandlerSafe(UInt16, TinyNetMessageDelegate)
TinyNetScene.InternalUpdate()
TinyNetScene.ClearNetManager()
TinyNetScene.ConfigureNetManager(Boolean)
TinyNetScene.ToggleNatPunching(Boolean)
TinyNetScene.SetPingInterval(Int32)
TinyNetScene.GetTinyNetConnection(Int64)
TinyNetScene.GetTinyNetConnection(NetPeer)
TinyNetScene.RemoveTinyNetConnection(TinyNetConnection)
TinyNetScene.RemoveTinyNetConnection(NetPeer)
TinyNetScene.RemoveTinyNetConnection(Int64)
TinyNetScene.AddTinyNetIdentityToList(TinyNetIdentity)
TinyNetScene.AddTinyNetObjectToList(ITinyNetObject)
TinyNetScene.RemoveTinyNetIdentityFromList(TinyNetIdentity)
TinyNetScene.RemoveTinyNetObjectFromList(ITinyNetObject)

TinyNetScene.GetTinyNetIdentityByNetworkID(Int32)
TinyNetScene.GetTinyNetObjectByNetworkID(Int32)
TinyNetScene.SendMessageByChannelToHost(ITinyNetMessage, DeliveryMethod)
TinyNetScene.SendMessageByChannelToTargetConnection(ITinyNetMessage, DeliveryMethod, TinyNetConnection)
TinyNetScene.SendMessageByChannelToAllConnections(ITinyNetMessage, DeliveryMethod)
TinyNetScene.SendMessageByChannelToAllReadyConnections(ITinyNetMessage, DeliveryMethod)
TinyNetScene.SendMessageByChannelToAllObserversOf(TinyNetIdentity, ITinyNetMessage, DeliveryMethod)
TinyNetScene.OnConnectionRequest(ConnectionRequest)
TinyNetScene.OnPeerConnected(NetPeer)
TinyNetScene.OnPeerDisconnected(NetPeer, DisconnectInfo)
TinyNetScene.OnNetworkError(NetEndPoint, Int32)
TinyNetScene.OnNetworkReceive(NetPeer, NetDataReader, DeliveryMethod)
TinyNetScene.OnNetworkReceiveUnconnected(NetEndPoint, NetDataReader, UnconnectedMessageType)
TinyNetScene.OnNetworkLatencyUpdate(NetPeer, Int32)
TinyNetScene.OnDiscoveryRequestReceived(NetEndPoint, NetDataReader)
TinyNetScene.OnConnectionCreated(TinyNetConnection)
TinyNetScene.OnDisconnect(TinyNetConnection)
TinyNetScene.OnRPCMessage(TinyNetMessageReader)
TinyNetScene.AddPlayerControllerToConnection(TinyNetConnection, Int32)
TinyNetScene.RemovePlayerControllerFromConnection(TinyNetConnection, Int16)
TinyNetScene.CreatePlayerAndAdd(TinyNetConnection, Int32)
System.Object.Equals(System.Object)
System.Object.Equals(System.Object, System.Object)
System.Object.GetHashCode()
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ToString()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetServer : TinyNetScene, INetEventListener
```

Constructors

TinyNetServer()

Initializes a new instance of the TinyNetServer class.

Declaration

```
public TinyNetServer()
```

Fields

instance

The singleton instance.

Declaration

```
public static TinyNetServer instance
```

Field Value

TYPE	DESCRIPTION
TinyNetServer	

Properties

TYPE

Sugar for generating debug logs.

Declaration

```
public override string TYPE { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Overrides

TinyNetScene.TYPE

Methods

CreateTinyNetConnection(NetPeer)

Creates a TinyNetConnection for the given NetPeer.

Declaration

```
protected override TinyNetConnection CreateTinyNetConnection(NetPeer peer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetPeer	peer	The peer.

Returns

TYPE	DESCRIPTION
TinyNetConnection	

Overrides

TinyNetScene.CreateTinyNetConnection(NetPeer)

DestroyObject(TinyNetIdentity, Boolean)

Destroys the object.

Declaration

```
public void DestroyObject(TinyNetIdentity tni, bool destroyServerObject)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tni	The TinyNetIdentity of the object.
System.Boolean	destroyServerObject	if set to `true` destroy the object on server too.

DestroyObject(GameObject)

Destroys the object.

Declaration

```
public void DestroyObject(GameObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject	obj	The object to destroy.

GetPlayerController(Int32)

Gets the player controller that have the given identifier.

Declaration

```
public TinyNetPlayerController GetPlayerController(int playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	playerControllerId	The player controller identifier.

Returns

TYPE	DESCRIPTION
TinyNetPlayerController	

GetPlayerControllerFromConnection(Int64, Int32)

Gets the player controller from a specific connection.

Declaration

```
public TinyNetPlayerController GetPlayerControllerFromConnection(long connId, int playerControllerId)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int64	connId	The connection identifier.

TYPE	NAME	DESCRIPTION
System.Int32	playerControllerId	The player controller identifier.

Returns

TYPE	DESCRIPTION
TinyNetPlayerController	

HideForConnection(TinyNetIdentity, TinyNetConnection)

Always call this from a TinyNetConnection RemoveFromVisList, or you will have sync issues.

Declaration

```
public void HideForConnection(TinyNetIdentity tinyNetId, TinyNetConnection conn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tinyNetId	The tiny net identifier.
TinyNetConnection	conn	The connection.

OnConnectMessage(TinyNetMessageReader)

Called when a connection message is received.

Declaration

```
protected virtual void OnConnectMessage(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	The net message.

OnServerSceneChanged(String)

Called when the server scene is changed.

Declaration

```
public virtual void OnServerSceneChanged(string sceneName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	sceneName	Name of the scene.

RegisterMessageHandlers()

Registers the message handlers.

Declaration

```
protected override void RegisterMessageHandlers()
```

Overrides

TinyNetScene.RegisterMessageHandlers()

SendRPCToAllClients(NetDataWriter, Int32, ITinyNetObject)

Sends the RPC to all clients.

Declaration

```
public void SendRPCToAllClients(NetDataWriter stream, int rpcMethodIndex, ITinyNetObject iObj)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	stream	The stream.
System.Int32	rpcMethodIndex	Index of the RPC method.
ITinyNetObject	iObj	The object.

SendRPCToClientOwner(NetDataWriter, Int32, ITinyNetObject)

Sends the RPC to the client owner of an object.

Declaration

```
public void SendRPCToClientOwner(NetDataWriter stream, int rpcMethodIndex, ITinyNetObject iObj)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	stream	The stream.
System.Int32	rpcMethodIndex	Index of the RPC method.
ITinyNetObject	iObj	The object.

SendSpawnMessage(TinyNetIdentity, TinyNetConnection)

Send a spawn message.

Declaration

```
public void SendSpawnMessage(TinyNetIdentity netIdentity, TinyNetConnection targetConn = null)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	netIdentity	The TinyNetIdentity of the object to spawn.
TinyNetConnection	targetConn	

SendStateUpdateToAllObservers(TinyNetBehaviour, DeliveryMethod)

Sends the state update to all observers of an object.

Declaration

```
public virtual void SendStateUpdateToAllObservers(TinyNetBehaviour netBehaviour, DeliveryMethod sendOptions)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetBehaviour	netBehaviour	The TinyNetBehaviour.
DeliveryMethod	sendOptions	The send options.

SetAllClientsNotReady()

Sets all clients as not ready.

Declaration

```
public void SetAllClientsNotReady()
```

ShowForConnection(TinyNetIdentity, TinyNetConnection)

Always call this from a TinyNetConnection ShowObjectToConnection, or you will have sync issues.

Declaration

```
public void ShowForConnection(TinyNetIdentity tinyNetId, TinyNetConnection conn)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tinyNetId	The tiny net identifier.
TinyNetConnection	conn	The connection.

Spawn(GameObject)

Just a shortcut to SpawnObject(obj)

Declaration

```
public static void Spawn(GameObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject	obj	The object to spawn.

SpawnAllObjects()

Spawns all TinyNetIdentity objects in the scene.

Declaration

```
public bool SpawnAllObjects()
```

Returns

TYPE	DESCRIPTION
System.Boolean	This actually always return true?

SpawnObject(GameObject)

Spawns the object.

Declaration

```
public void SpawnObject(GameObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject	obj	The object to spawn.

SpawnWithClientAuthority(GameObject, TinyNetConnection)

Spawns the object with client authority.

Declaration

```
public bool SpawnWithClientAuthority(GameObject obj, TinyNetConnection conn)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject	obj	The object to spawn.
TinyNetConnection	conn	The connection that will own it.

Returns

TYPE	DESCRIPTION
System.Boolean	

StartServer(Int32, Int32)

Starts the server.

Declaration

```
public virtual bool StartServer(int port, int maxNumberOfPlayers)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	port	The port.
System.Int32	maxNumberOfPlayers	The maximum number of players.

Returns

TYPE	DESCRIPTION
System.Boolean	

TinyNetUpdate()

Run every frame, called from TinyNetGameManager.

Declaration

```
public override void TinyNetUpdate()
```

Overrides

TinyNetScene.TinyNetUpdate()

UnSpawnObject(TinyNetIdentity)

Unspawn an object.

Declaration

```
public void UnSpawnObject(TinyNetIdentity tni)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetIdentity	tni	

UnSpawnObject(GameObject)

Unspawn an object.

Declaration

```
public void UnSpawnObject(GameObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject	obj	The object to unspawn.

Implements

INetEventListener

See Also

TinyNetScene

Class TinyNetSimpleMenu

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
TinyNetSimpleMenu

Inherited Members

UnityEngine.MonoBehaviour.Invoke(System.String, System.Single)
UnityEngine.MonoBehaviour.InvokeRepeating(System.String, System.Single, System.Single)
UnityEngine.MonoBehaviour.CancelInvoke()
UnityEngine.MonoBehaviour.CancelInvoke(System.String)
UnityEngine.MonoBehaviour.IsInvoking(System.String)
UnityEngine.MonoBehaviour.IsInvoking()
UnityEngine.MonoBehaviour.StartCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine_Auto(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StartCoroutine(System.String, System.Object)
UnityEngine.MonoBehaviour.StartCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.String)
UnityEngine.MonoBehaviour.StopCoroutine(System.Collections.IEnumerator)
UnityEngine.MonoBehaviour.StopCoroutine(UnityEngine.Coroutine)
UnityEngine.MonoBehaviour.StopAllCoroutines()
UnityEngine.MonoBehaviour.print(System.Object)
UnityEngine.MonoBehaviour.useGUILayout
UnityEngine.MonoBehaviour.runInEditMode
UnityEngine.Behaviour.enabled
UnityEngine.Behaviour.isActiveAndEnabled
UnityEngine.Component.GetComponent(System.Type)
UnityEngine.Component.GetComponent<T>()
UnityEngine.Component.GetComponent(System.String)
UnityEngine.Component.GetComponentInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentInChildren(System.Type)
UnityEngine.Component.GetComponentInChildren<T>()
UnityEngine.Component.GetComponentInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren(System.Type)
UnityEngine.Component.GetComponentsInChildren(System.Type, System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean)
UnityEngine.Component.GetComponentsInChildren<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInChildren<T>()
UnityEngine.Component.GetComponentsInChildren<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentInParent(System.Type)
UnityEngine.Component.GetComponentInParent<T>()
UnityEngine.Component.GetComponentsInParent(System.Type)
UnityEngine.Component.GetComponentsInParent(System.Type, System.Boolean)
UnityEngine.Component.GetComponentsInParent<T>(System.Boolean)
UnityEngine.Component.GetComponentsInParent<T>(System.Boolean, System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponentsInParent<T>()
UnityEngine.Component.GetComponents(System.Type)
UnityEngine.Component.GetComponents(System.Type, System.Collections.Generic.List<UnityEngine.Component>)

UnityEngine.Component.GetComponents<T>(System.Collections.Generic.List<T>)
UnityEngine.Component.GetComponents<T>()
UnityEngine.Component.CompareTag(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessageUpwards(System.String, System.Object)
UnityEngine.Component.SendMessageUpwards(System.String)
UnityEngine.Component.SendMessageUpwards(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.SendMessage(System.String, System.Object)
UnityEngine.Component.SendMessage(System.String)
UnityEngine.Component.SendMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object, UnityEngine.SendMessageOptions)
UnityEngine.Component.BroadcastMessage(System.String, System.Object)
UnityEngine.Component.BroadcastMessage(System.String)
UnityEngine.Component.BroadcastMessage(System.String, UnityEngine.SendMessageOptions)
UnityEngine.Component.transform
UnityEngine.Component.gameObject
UnityEngine.Component.tag
UnityEngine.Component.rigidbody
UnityEngine.Component.rigidbody2D
UnityEngine.Component.camera
UnityEngine.Component.light
UnityEngine.Component.animation
UnityEngine.Component.constantForce
UnityEngine.Component.renderer
UnityEngine.Component.audio
UnityEngine.Component.guiText
UnityEngine.Component.networkView
UnityEngine.Component.guiElement
UnityEngine.Component.guiTexture
UnityEngine.Component.collider
UnityEngine.Component.collider2D
UnityEngine.Component.hingeJoint
UnityEngine.Component.particleEmitter
UnityEngine.Component.particleSystem
UnityEngine.Object.Destroy(UnityEngine.Object, System.Single)
UnityEngine.Object.Destroy(UnityEngine.Object)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object, System.Boolean)
UnityEngine.Object.DestroyImmediate(UnityEngine.Object)
UnityEngine.Object.FindObjectsOfType(System.Type)
UnityEngine.Object.DontDestroyOnLoad(UnityEngine.Object)
UnityEngine.Object.DestroyObject(UnityEngine.Object, System.Single)
UnityEngine.Object.DestroyObject(UnityEngine.Object)
UnityEngine.Object.FindSceneObjectsOfType(System.Type)
UnityEngine.Object.FindObjectsOfTypeIncludingAssets(System.Type)
UnityEngine.Object.FindObjectsOfTypeAll(System.Type)
UnityEngine.Object.ToString()
UnityEngine.Object.GetInstanceID()
UnityEngine.Object.GetHashCode()
UnityEngine.Object.Equals(System.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)

UnityEngine.Object.Instantiate(UnityEngine.Object)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform)
UnityEngine.Object.Instantiate(UnityEngine.Object, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.Instantiate<T>(T)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Vector3, UnityEngine.Quaternion, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform)
UnityEngine.Object.Instantiate<T>(T, UnityEngine.Transform, System.Boolean)
UnityEngine.Object.FindObjectsOfType<T>()
UnityEngine.Object.FindObjectOfType<T>()
UnityEngine.Object.FindObjectOfType(System.Type)
UnityEngine.Object.name
UnityEngine.Object.hideFlags
System.Object.Equals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetSimpleMenu : MonoBehaviour
```

Fields

hostPortField

Declaration

```
public InputField hostPortField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.InputField	

ipToConnectField

Declaration

```
public InputField ipToConnectField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.InputField	

portToConnectField

Declaration

```
public InputField portToConnectField
```

Field Value

TYPE	DESCRIPTION
UnityEngine.UI.InputField	

Methods

PressedConnectButton()

Declaration

```
public void PressedConnectButton()
```

PressedHostButton()

Declaration

```
public void PressedHostButton()
```

ToggleNatPunching(Boolean)

Declaration

```
public void ToggleNatPunching(bool bNewValue)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	bNewValue	

Class TinyNetStateSyncer

This class stores all SyncVar allowed properties and is used to sync the game state.

Inheritance

System.Object
TinyNetStateSyncer

Inherited Members

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

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class TinyNetStateSyncer
```

Fields

rpcMethods

Declaration

```
protected static Dictionary<Type, List<RPCMethodInfo>> rpcMethods
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.Type, System.Collections.Generic.List<RPCMethodInfo>>	

syncVarProps

Declaration

```
protected static Dictionary<Type, List<PropertyInfo>> syncVarProps
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.Dictionary<System.Type, System.Collections.Generic.List<System.Reflection.PropertyInfo>>	

tempIntArray

Declaration

```
protected static int[] tempIntArray
```

Field Value

TYPE	DESCRIPTION
System.Int32[]	

Methods

AddPropertyToType(PropertyInfo, Type)

Declaration

```
public static void AddPropertyToType(PropertyInfo prop, Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Reflection.PropertyInfo	prop	
System.Type	type	

AddRPCMethodNameToType(String, RPCTarget, RPCCallers, Type)

Declaration

```
public static void AddRPCMethodNameToType(string rpcName, RPCTarget nTarget, RPCCallers nCaller, Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	rpcName	
RPCTarget	nTarget	
RPCCallers	nCaller	
System.Type	type	

DirtyFlagToInt(BitArray)

Declaration

```
public static int DirtyFlagToInt(BitArray bitArray)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Collections.BitArray	bitArray	

Returns

TYPE	DESCRIPTION
System.Int32	

Display(Int32, Boolean)

Declaration

```
public static string Display(int value, bool cull = false)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	value	
System.Boolean	cull	

Returns

TYPE	DESCRIPTION
System.String	

GetNumberOfRPCMethods(Type)

Declaration

```
public static int GetNumberOfRPCMethods(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	

Returns

TYPE	DESCRIPTION
System.Int32	

GetNumberOfSyncedProperties(Type)

Declaration

```
public static int GetNumberOfSyncedProperties(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	

Returns

TYPE	DESCRIPTION
System.Int32	

GetPropertyInfoFromType(Type, String)

Declaration

```
public static PropertyInfo GetPropertyInfoFromType(Type type, string propName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	

TYPE	NAME	DESCRIPTION
System.String	propName	

Returns

TYPE	DESCRIPTION
System.Reflection.PropertyInfo	

GetRPCMethodIndexFromType(Type, String)

Declaration

```
public static int GetRPCMethodIndexFromType(Type type, string rpcName)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	
System.String	rpcName	

Returns

TYPE	DESCRIPTION
System.Int32	

GetRPCMethodInfoFromType(Type, Int32, ref RPCMethodInfo)

Declaration

```
public static void GetRPCMethodInfoFromType(Type type, int rpcMethodIndex, ref RPCMethodInfo rpcMethodInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	
System.Int32	rpcMethodIndex	
RPCMethodInfo	rpcMethodInfo	

GetRPCMethodInfoFromType(Type, String, ref RPCMethodInfo)

Declaration

```
public static int GetRPCMethodInfoFromType(Type type, string rpcName, ref RPCMethodInfo rpcMethodInfo)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	

TYPE	NAME	DESCRIPTION
System.String	rpcName	
RPCMethodInfo	rpcMethodInfo	

Returns

TYPE	DESCRIPTION
System.Int32	

InitializePropertyInfoListOfType(Int32, Type)

Declaration

```
public static void InitializePropertyInfoListOfType(int size, Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	size	
System.Type	type	

InitializeRPCMethodsOfType(Int32, Type)

Declaration

```
public static void InitializeRPCMethodsOfType(int size, Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	size	
System.Type	type	

IntToDirtyFlag(Int32, BitArray)

Declaration

```
public static void IntToDirtyFlag(int input, BitArray bitArray)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	input	
System.Collections.BitArray	bitArray	

OutPropertyNamesFromType(Type, out String[])

Declaration

```
public static void OutPropertyNamesFromType(Type type, out string[] propNames)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	
System.String[]	propNames	

OutPropertyTypesFromType(Type, out Type[])

Declaration

```
public static void OutPropertyTypesFromType(Type type, out Type[] propTypes)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	
System.Type[]	propTypes	

OutRPCMethodNamesFromType(Type, out String[])

Declaration

```
public static void OutRPCMethodNamesFromType(Type type, out string[] rpcNames)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Type	type	
System.String[]	rpcNames	

UpdateDirtyFlagOf(TinyNetBehaviour, BitArray)

Declaration

```
public static void UpdateDirtyFlagOf(TinyNetBehaviour instance, BitArray bitArray)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetBehaviour	instance	
System.Collections.BitArray	bitArray	

Class TinyNetSyncVar

When used on a compatible property type, it will send it's value to all clients if they are changed.

byte, sbyte, short, ushort, int, uint, long, ulong, float, double, bool, string.

Inheritance

System.Object
System.Attribute
TinyNetSyncVar

Implements

System.Runtime.InteropServices._Attribute

Inherited Members

System.Attribute.System.Runtime.InteropServices._Attribute.GetIDsOfNames(System.Guid, System.IntPtr, System.UInt32, System.UInt32, System.IntPtr)
System.Attribute.System.Runtime.InteropServices._Attribute.GetTypeInfo(System.UInt32, System.UInt32, System.IntPtr)
System.Attribute.System.Runtime.InteropServices._Attribute.GetTypeInfoCount(System.UInt32)
System.Attribute.System.Runtime.InteropServices._Attribute.Invoke(System.UInt32, System.Guid, System.UInt32, System.Int16, System.IntPtr, System.IntPtr, System.IntPtr, System.IntPtr)
System.Attribute.GetCustomAttribute(System.Reflection.ParameterInfo, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.MemberInfo, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.Assembly, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.Module, System.Type)
System.Attribute.GetCustomAttribute(System.Reflection.Module, System.Type, System.Boolean)
System.Attribute.GetCustomAttribute(System.Reflection.Assembly, System.Type, System.Boolean)
System.Attribute.GetCustomAttribute(System.Reflection.ParameterInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttribute(System.Reflection.MemberInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo)
System.Attribute.GetCustomAttributes(System.Reflection.Module)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.Module, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo, System.Type)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Module, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo, System.Type, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Module, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.Assembly, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.MemberInfo, System.Boolean)
System.Attribute.GetCustomAttributes(System.Reflection.ParameterInfo, System.Boolean)
System.Attribute.GetHashCode()
System.Attribute.IsDefaultAttribute()
System.Attribute.IsDefined(System.Reflection.Module, System.Type)
System.Attribute.IsDefined(System.Reflection.ParameterInfo, System.Type)
System.Attribute.IsDefined(System.Reflection.MemberInfo, System.Type)
System.Attribute.IsDefined(System.Reflection.Assembly, System.Type)
System.Attribute.IsDefined(System.Reflection.MemberInfo, System.Type, System.Boolean)
System.Attribute.IsDefined(System.Reflection.Assembly, System.Type, System.Boolean)
System.Attribute.IsDefined(System.Reflection.Module, System.Type, System.Boolean)

System.Attribute.IsDefined(System.Reflection.ParameterInfo, System.Type, System.Boolean)
System.Attribute.Match(System.Object)
System.Attribute.Equals(System.Object)
System.Attribute.TypeId
System.Object.Equals(System.Object, System.Object)
System.Object.GetType()
System.Object.MemberwiseClone()
System.Object.ToString()
System.Object.ReferenceEquals(System.Object, System.Object)

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
[AttributeUsage(AttributeTargets.Property)]
public class TinyNetSyncVar : Attribute, _Attribute
```

Fields

allowedTypes

The types allowed for this attribute.

Declaration

```
public static HashSet<Type> allowedTypes
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.HashSet<System.Type>	

Implements

System.Runtime.InteropServices._Attribute

Delegate UnSpawnDelegate

Handles requests to unspawn objects on the client

Namespace: TinyBirdNet

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void UnSpawnDelegate(GameObject gObj);
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.GameObject	gObj	The GameObject.

Namespace TinyBirdNet.Messaging

Classes

TinyNetAddPlayerMessage

TinyNetBoolMessage

TinyNetByteMessage

TinyNetClientAuthorityMessage

TinyNetEmptyMessage

TinyNetFloatMessage

TinyNetInputMessage

This is basically a message that gets delivered directly to a TinyNetPlayerController.

TinyNetIntegerMessage

TinyNetMessageReader

Used to provide an easy way to read different messages.

TinyNetMsgType

built-in system network messages

TinyNetNotReadyMessage

TinyNetObjectDestroyMessage

TinyNetObjectHideMessage

TinyNetObjectSpawnFinishedMessage

TinyNetObjectSpawnMessage

TinyNetObjectSpawnSceneMessage

TinyNetObjectStateUpdate

TinyNetOwnerMessage

Something about player controllers objects, but since they are not gameobjects in TinyBirdNet this message is useless.

TinyNetReadyMessage

TinyNetRemovePlayerMessage

TinyNetRequestAddPlayerMessage

TinyNetRequestRemovePlayerMessage

TinyNetRPCMessage

TinyNetShortMessage

TinyNetStringMessage

Interfaces

ITinyNetMessage

An interface used by all messages.

Delegates

TinyNetMessageDelegate

The delegate used for message handlers.

Interface ITinyNetMessage

An interface used by all messages.

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public interface ITinyNetMessage
```

Properties

msgType

Declaration

```
ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Deserializes the contents of the NetDataReader into this message.

Declaration

```
void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	The NetDataReader.

Serialize(NetDataWriter)

Serializes the contents of this message into the NetDataWriter.

Declaration

```
void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	The NetDataWriter.

Class TinyNetAddPlayerMessage

Inheritance

System.Object
TinyNetAddPlayerMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetAddPlayerMessage : ITinyNetMessage
```

Fields

msgData

Declaration

```
public byte[] msgData
```

Field Value

TYPE	DESCRIPTION
System.Byte[]	

msgSize

Declaration

```
public int msgSize
```

Field Value

TYPE	DESCRIPTION
System.Int32	

playerControllerId

Declaration

```
public short playerControllerId
```

Field Value

TYPE	DESCRIPTION
System.Int16	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetBoolMessage

Inheritance

System.Object
TinyNetBoolMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetBoolMessage : ITinyNetMessage
```

Constructors

TinyNetBoolMessage()

Declaration

```
public TinyNetBoolMessage()
```

TinyNetBoolMessage(Boolean)

Declaration

```
public TinyNetBoolMessage(bool v)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Boolean	v	

Fields

value

Declaration

```
public bool value
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Properties

msgType

Declaration

```
public ushort msgType { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetByteMessage

Inheritance

System.Object
TinyNetByteMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetByteMessage : ITinyNetMessage
```

Constructors

TinyNetByteMessage()

Declaration

```
public TinyNetByteMessage()
```

TinyNetByteMessage(Byte)

Declaration

```
public TinyNetByteMessage(byte v)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte	v	

Fields

value

Declaration

```
public byte value
```

Field Value

TYPE	DESCRIPTION
System.Byte	

Properties

msgType

Declaration

```
public ushort msgType { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetClientAuthorityMessage

Inheritance

System.Object
TinyNetClientAuthorityMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetClientAuthorityMessage : ITinyNetMessage
```

Fields

authority

Declaration

```
public bool authority
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetEmptyMessage

Inheritance

System.Object
TinyNetEmptyMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetEmptyMessage : ITinyNetMessage
```

Properties

msgType

Declaration

```
public ushort msgType { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetFloatMessage

Inheritance

System.Object
TinyNetFloatMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetFloatMessage : ITinyNetMessage
```

Constructors

TinyNetFloatMessage()

Declaration

```
public TinyNetFloatMessage()
```

TinyNetFloatMessage(Single)

Declaration

```
public TinyNetFloatMessage(float v)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single	v	

Fields

value

Declaration

```
public float value
```

Field Value

TYPE	DESCRIPTION
System.Single	

Properties

msgType

Declaration

```
public ushort msgType { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetInputMessage

This is basically a message that gets delivered directly to a TinyNetPlayerController.

Inheritance

System.Object
TinyNetInputMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public abstract class TinyNetInputMessage : ITinyNetMessage
```

Fields

playerControllerId

Declaration

```
public short playerControllerId
```

Field Value

TYPE	DESCRIPTION
System.Int16	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public virtual void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

PeekAtPlayerControllerId(TinyNetMessageReader)

Declaration

```
public static short PeekAtPlayerControllerId(TinyNetMessageReader netMsg)
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	

Returns

TYPE	DESCRIPTION
System.Int16	

Serialize(NetDataWriter)

Declaration

```
public virtual void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

See Also

ITinyNetMessage

Class TinyNetIntegerMessage

Inheritance

System.Object
TinyNetIntegerMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetIntegerMessage : ITinyNetMessage
```

Constructors

TinyNetIntegerMessage()

Declaration

```
public TinyNetIntegerMessage()
```

TinyNetIntegerMessage(Int32)

Declaration

```
public TinyNetIntegerMessage(int v)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int32	v	

Fields

value

Declaration

```
public int value
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

msgType

Declaration

```
public ushort msgType { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Delegate TinyNetMessageDelegate

The delegate used for message handlers.

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public delegate void TinyNetMessageDelegate(TinyNetMessageReader netMsg);
```

Parameters

TYPE	NAME	DESCRIPTION
TinyNetMessageReader	netMsg	The TinyNetMessageReader.

Class TinyNetMessageReader

Used to provide an easy way to read different messages.

Inheritance

System.Object
TinyNetMessageReader

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetMessageReader
```

Fields

channelId

The delivery method of this message.

Not implemmented yet.

Declaration

```
public DeliveryMethod channelId
```

Field Value

TYPE	DESCRIPTION
DeliveryMethod	

MaxMessageSize

The maximum message size allowed.

Declaration

```
public const int MaxMessageSize = 65535
```

Field Value

TYPE	DESCRIPTION
System.Int32	

msgType

The message type id

Declaration

```
public ushort msgType
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

reader

A reader with data stream of the message to read.

Declaration

```
public NetDataReader reader
```

Field Value

TYPE	DESCRIPTION
NetDataReader	

tinyNetConn

The connection from where this message came from.

Declaration

```
public TinyNetConnection tinyNetConn
```

Field Value

TYPE	DESCRIPTION
TinyNetConnection	

Methods

Dump(Byte[], Int32)

Dumps the specified payload.

Declaration

```
public static string Dump(byte[] payload, int sz)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Byte[]	payload	The payload.
System.Int32	sz	The size of the payload.

Returns

TYPE	DESCRIPTION
System.String	

ReadMessage<TMsg>()

Reads the message.

Declaration

```
public TMsg ReadMessage<TMsg>()where TMsg : ITinyNetMessage, new ()
```

Returns

TYPE	DESCRIPTION
TMsg	

Type Parameters

NAME	DESCRIPTION
TMsg	The type id of the message.

ReadMessage<TMsg>(TMsg)

Reads the message.

Declaration

```
public void ReadMessage<TMsg>(TMsg msg)where TMsg : ITinyNetMessage
```

Parameters

TYPE	NAME	DESCRIPTION
TMsg	msg	A message where the data will be deserialized to.

Type Parameters

NAME	DESCRIPTION
TMsg	The type id of the message.

Class TinyNetMsgType

built-in system network messages

Inheritance

System.Object
TinyNetMsgType

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetMsgType
```

Fields

AddPlayer

Declaration

```
public const ushort AddPlayer = 37
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Connect

Declaration

```
public const ushort Connect = 32
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

CRC

Declaration

```
public const ushort CRC = 14
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Disconnect

Declaration

```
public const ushort Disconnect = 33
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Error

Declaration

```
public const ushort Error = 34
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Highest

The highest system message id used.

Declaration

```
public const ushort Highest = 41
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Input

Declaration

```
public const ushort Input = 6
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

InternalHighest

Declaration

```
public const ushort InternalHighest = 31
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

LocalChildTransform

Declaration

```
public const ushort LocalChildTransform = 16
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

LocalClientAuthority

Declaration

```
public const ushort LocalClientAuthority = 15
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

NetworkInfo

Declaration

```
public const ushort NetworkInfo = 11
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

NotReady

Declaration

```
public const ushort NotReady = 36
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

ObjectDestroy

Declaration

```
public const ushort ObjectDestroy = 1
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

ObjectHide

Declaration

```
public const ushort ObjectHide = 13
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

ObjectSpawnMessage

Declaration

```
public const ushort ObjectSpawnMessage = 3
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

ObjectSpawnScene

Declaration

```
public const ushort ObjectSpawnScene = 10
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Owner

Declaration

```
public const ushort Owner = 4
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

PeerClientAuthority

Declaration

```
public const ushort PeerClientAuthority = 17
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Ready

Declaration

```
public const ushort Ready = 35
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

RemovePlayer

Declaration

```
public const ushort RemovePlayer = 38
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

RequestAddPlayer

Declaration

```
public const ushort RequestAddPlayer = 39
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

RequestRemovePlayer

Declaration

```
public const ushort RequestRemovePlayer = 40
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Rpc

Declaration

```
public const ushort Rpc = 2
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Scene

Declaration

```
public const ushort Scene = 41
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

SpawnFinished

Declaration

```
public const ushort SpawnFinished = 12
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

SpawnPlayer

Declaration

```
public const ushort SpawnPlayer = 5
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

StateUpdate

Declaration

```
public const ushort StateUpdate = 8
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

SyncEvent

Declaration

```
public const ushort SyncEvent = 7
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

SyncList

Declaration

```
public const ushort SyncList = 9
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Methods

MsgTypeToString(UInt16)

Converts the type id to a readable string.

Declaration

```
public static string MsgTypeToString(ushort value)
```

Parameters

TYPE	NAME	DESCRIPTION
System.UInt16	value	The message type id.

Returns

TYPE	DESCRIPTION
System.String	

Class TinyNetNotReadyMessage

Inheritance

System.Object
TinyNetNotReadyMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetNotReadyMessage : ITinyNetMessage
```

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetObjectDestroyMessage

Inheritance

System.Object
TinyNetObjectDestroyMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetObjectDestroyMessage : ITinyNetMessage
```

Fields

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public virtual void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public virtual void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetObjectHideMessage

Inheritance

System.Object
TinyNetObjectHideMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetObjectHideMessage : ITinyNetMessage
```

Fields

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public virtual void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public virtual void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetObjectSpawnFinishedMessage

Inheritance

System.Object
TinyNetObjectSpawnFinishedMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetObjectSpawnFinishedMessage : ITinyNetMessage
```

Fields

state

Declaration

```
public byte state
```

Field Value

TYPE	DESCRIPTION
System.Byte	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetObjectSpawnMessage

Inheritance

System.Object
TinyNetObjectSpawnMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetObjectSpawnMessage : ITinyNetMessage
```

Fields

assetIndex

Declaration

```
public int assetIndex
```

Field Value

TYPE	DESCRIPTION
System.Int32	

initialState

Declaration

```
public byte[] initialState
```

Field Value

TYPE	DESCRIPTION
System.Byte[]	

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

position

Declaration

```
public Vector3 position
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public virtual void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public virtual void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetObjectSpawnSceneMessage

Inheritance

System.Object
TinyNetObjectSpawnSceneMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetObjectSpawnSceneMessage : ITinyNetMessage
```

Fields

initialState

Declaration

```
public byte[] initialState
```

Field Value

TYPE	DESCRIPTION
System.Byte[]	

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

position

Declaration

```
public Vector3 position
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Vector3	

sceneId

Declaration

```
public int sceneId
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetObjectStateUpdate

Inheritance

System.Object
TinyNetObjectStateUpdate

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetObjectStateUpdate : ITinyNetMessage
```

Fields

dirtyFlag

Declaration

```
public int dirtyFlag
```

Field Value

TYPE	DESCRIPTION
System.Int32	

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

state

Declaration

```
public byte[] state
```

Field Value

TYPE	DESCRIPTION
System.Byte[]	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public virtual void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public virtual void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetOwnerMessage

Something about player controllers objects, but since they are not gameobjects in TinyBirdNet this message is useless.

Inheritance

System.Object
TinyNetOwnerMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetOwnerMessage : ITinyNetMessage
```

Fields

connectId

Declaration

```
public short connectId
```

Field Value

TYPE	DESCRIPTION
System.Int16	

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetReadyMessage

Inheritance

System.Object
TinyNetReadyMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetReadyMessage : ITinyNetMessage
```

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetRemovePlayerMessage

Inheritance

System.Object
TinyNetRemovePlayerMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetRemovePlayerMessage : ITinyNetMessage
```

Fields

playerControllerId

Declaration

```
public short playerControllerId
```

Field Value

TYPE	DESCRIPTION
System.Int16	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetRequestAddPlayerMessage

Inheritance

System.Object
TinyNetRequestAddPlayerMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetRequestAddPlayerMessage : ITinyNetMessage
```

Fields

amountOfPlayers

Declaration

```
public ushort amountOfPlayers
```

Field Value

TYPE	DESCRIPTION
System.UInt16	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetRequestRemovePlayerMessage

Inheritance

System.Object
TinyNetRequestRemovePlayerMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetRequestRemovePlayerMessage : ITinyNetMessage
```

Fields

playerControllerId

Declaration

```
public short playerControllerId
```

Field Value

TYPE	DESCRIPTION
System.Int16	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetRPCMessage

Inheritance

System.Object
TinyNetRPCMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetRPCMessage : ITinyNetMessage
```

Fields

networkID

Declaration

```
public int networkID
```

Field Value

TYPE	DESCRIPTION
System.Int32	

parameters

Declaration

```
public byte[] parameters
```

Field Value

TYPE	DESCRIPTION
System.Byte[]	

rpcMethodIndex

Declaration

```
public int rpcMethodIndex
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

msgType

Declaration

```
public ushort msgType { get; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetShortMessage

Inheritance

System.Object
TinyNetShortMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetShortMessage : ITinyNetMessage
```

Constructors

TinyNetShortMessage()

Declaration

```
public TinyNetShortMessage()
```

TinyNetShortMessage(Int16)

Declaration

```
public TinyNetShortMessage(short v)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Int16	v	

Fields

value

Declaration

```
public short value
```

Field Value

TYPE	DESCRIPTION
System.Int16	

Properties

msgType

Declaration

```
public ushort msgType { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Class TinyNetStringMessage

Inheritance

System.Object
TinyNetStringMessage

Implements

ITinyNetMessage

Inherited Members

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

Namespace: TinyBirdNet.Messaging

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyNetStringMessage : ITinyNetMessage
```

Constructors

TinyNetStringMessage()

Declaration

```
public TinyNetStringMessage()
```

TinyNetStringMessage(String)

Declaration

```
public TinyNetStringMessage(string v)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	v	

Fields

value

Declaration

```
public string value
```

Field Value

TYPE	DESCRIPTION
System.String	

Properties

msgType

Declaration

```
public ushort msgType { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.UInt16	

Methods

Deserialize(NetDataReader)

Declaration

```
public void Deserialize(NetDataReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataReader	reader	

Serialize(NetDataWriter)

Declaration

```
public void Serialize(NetDataWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
NetDataWriter	writer	

Implements

ITinyNetMessage

Namespace TinyBirdUtils

Classes

TinyLogger

Helper that removes debug calls from release builds.

Currently not working due to a bug in Unity [Conditional].

Class TinyLogger

Helper that removes debug calls from release builds.

Currently not working due to a bug in Unity [Conditional].

Inheritance

System.Object
TinyLogger

Inherited Members

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

Namespace: TinyBirdUtils

Assembly: Assembly-CSharp.dll

Syntax

```
public class TinyLogger
```

Methods

Log(Object, Object)

Declaration

```
public static void Log(object message, Object context = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	message	
UnityEngine.Object	context	

LogError(Object, Object)

Declaration

```
public static void LogError(object message, Object context = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	message	
UnityEngine.Object	context	

LogWarning(Object, Object)

Declaration

```
public static void LogWarning(object message, Object context = null)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Object	message	
UnityEngine.Object	context	