Not native English speaker, sorry for my poor English before everything.

**What is this project:**

**A stream like network API wrapper of UNet LLAPI**

**Core Classes Description:**

**All runtime core classes are under namespace Network;**

**All Packets and PacketID are under namespace Network.Packets;**

**Runtime Core Classes:**

**NetworkConst:**

Settings for Terminals are defined here.

**Terminal:**

Network event processor, base class of ClientTerminal and ServerTerminal.­

Provide basic send packet method, please mind that send method which used in server is slightly different from which used in client. You can modify or extend send methods as you like, it won’t cause any problem if the ByteArray is setup and sent correctly.

Function BufferProcess() is the network event process function, should be driven by unity update event. Every new connection will be registered to m\_ConnectedTerminalInfoMap here, with some of its connection information. Packets will be assembly here and dispatch via PacketDispatcher. Description of PacketDispatcher is down below.

**ServerTerminal:**

Singleton of STerminal, STerminal is inherit from Terminal, act as server instance. Provide some basic functions of sending packets to client. **PLEASE DO NOT USE STerminal IMMEDIATELY, USE ITS SINGLETON VERSION INSTEAD.**

**ClientTerminal:**

Singleton of CTerminal, CTerminal is inherit from Terminal, act as client instance. Provide some basic functions of sending packets to server. **PLEASE DO NOT USE CTerminal IMMEDIATELY, USE ITS SINGLETON VERSION INSTEAD.**

**Other Terminals:**

If the subclasses of Terminal which is already provided are not suitable for you, you can modify existing subclasses of Terminal or create your own class to extend functions you need.

**PacketID:**

PacketID is the enumeration of packet identifier, packet identifier is stored as short value in bytearray, you can define packet id by yourself or just let it generate by tools.

**PacketFactory:**

Packet Factory is responsible for assembly packet from received buffer data and disassembly packet into data buffer. This class is divided into 3 files, which is Assemblers.cs, Disassemblers.cs and PacketFactory.cs. Assemblers.cs and Disassemblers.cs is auto generated by tools, provide assembly functions and disassembly functions and mapper of these functions.

**ByteArray:**

Wrapper of byte[], provide stream like interfaces for read and write.

**PacketDispatcher:**

Dispatch incoming packet to corresponding handler for further process. One type of packet is only corresponding to one handler. You can make it correspond to multi-handlers by slightly modification on PacketDispatcher, It won’t cause any problem, but I don’t recommend it;

**Editor Toolkit Classes:**

**NetworkEditorConst:**

Pathes and code templates are defined here.

**PacketGenerator:**

Generate runtime packet classes by class definitions under namespace Network.Packets.Structs, PacketID can be defined manually in class Network.Packets.Structs.PacketID, or just leave it to be generated by tools. Check TEST\_PACKET.cs to know what kind of data type do this framework support.

**PacketID:**

Define packet id manually here.

**How to Get It Work:**

1. **Setup:**

Copy file “82-C# PacketStruct-NewPacketStructDefinition.cs.txt” under “Assets/NetworkFrameworkBaseOnLLAPI/Editor/Templates/” to “%UNITY\_EDITOR%/Data/Resources/ScriptTemplates/”. This will add an item as figure.1 show. By the way “%UNITY\_EDITOR%” means the Editor folder under your unity program install path.

1. **Create Packet Structures:**

Create C# script via menu item added at step 1 under path “Assets/NetworkFrameworkBaseOnLLAPI/Editor/Network/PacketDefinition”. Data types shown below can define in custom packet structures.

1. byte
2. short
3. int
4. Int64
5. float
6. bool
7. string
8. packet defined by step 2
9. List of type a-h.
10. **Generate Game Runtime Codes:**

Click menu item as figure.2 show, generate codes which are used in real game. Use those generated codes as Examples show.

Runtime codes will be generated at “Assets/NetworkFrameworkBaseOnLLAPI/Scripts/Network/Packets/” and “Assets/NetworkFrameworkBaseOnLLAPI/Scripts/Utilities/”. Paths mentioned above can be modify in file “NetworkEditorConst.cs”. Packets will be generated with a prefix which is defined in NetworkEditorConst.GENERATED\_PACKET\_PREFIX. You can modify it as you want, and use empty string as prefix won’t cause any problem.

**Every time you try to generate packet codes, a backup will be made under “/Assets/../PacketCodeBackup/%CURRENT\_TIMESTAMP%/”. If unfortunately, you failed on generate packet code and editor pop up lots of compile error, you can find your backup codes here.**

1. **Read Example Codes.**
2. **Enjoy coding.**

**Figures**

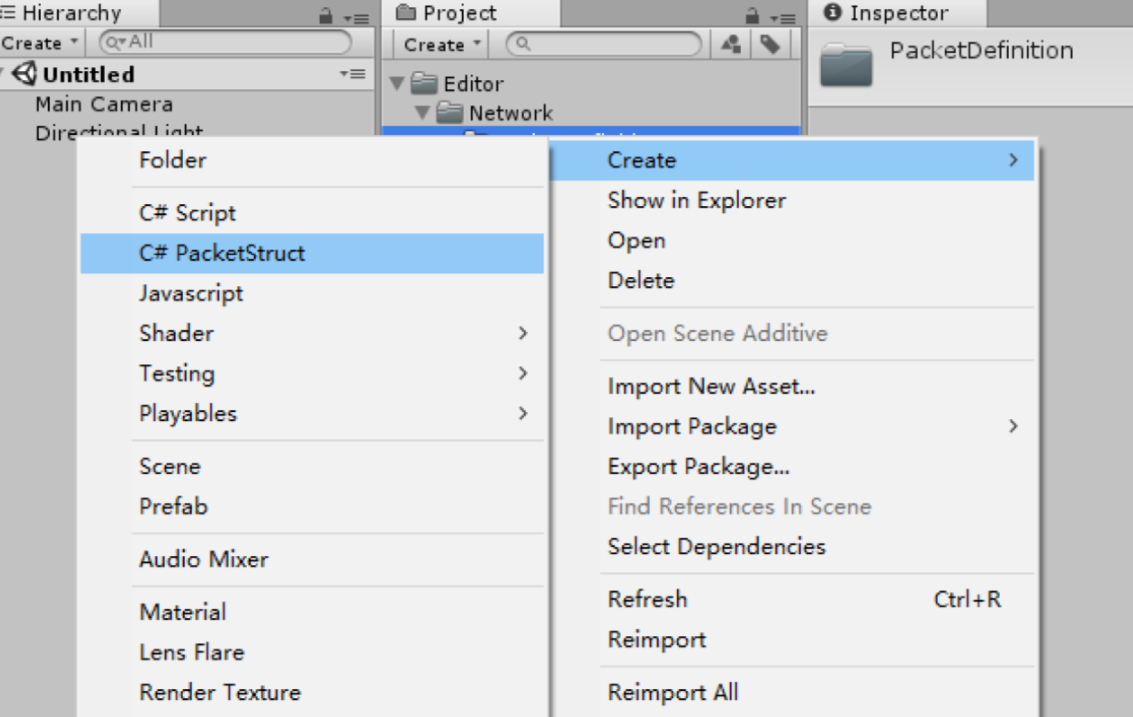
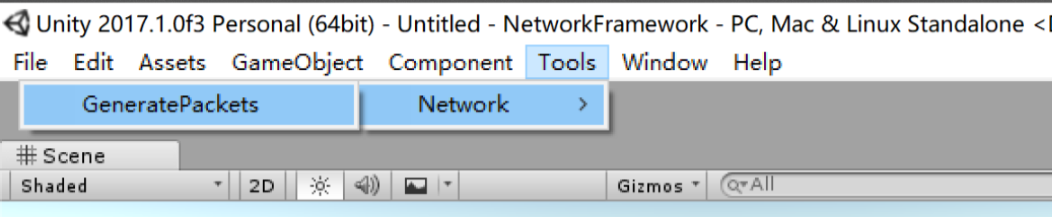
Figure.1 

Figure.2 

**Examples:**

**Example Codes:**

Server Example: Server.cs

Client Example: Client.cs

**Example Scene:**

Server Example: Server

Client Example: Client