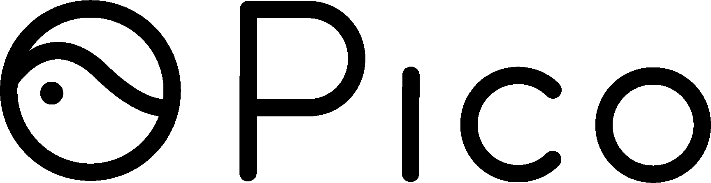
PicoVR Broadcast Platform Development Instructions



Pico Technology Co., Ltd.

2020, May

Revision History

Category: Software User Manual

Classification: General Level  
Contributor：Xiao.Shi,Aaron.Zhang,Taylor.Ynag,

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Contributor** | **Decsription** |
| V1.0 | 2020-04-03 | Xiao.Shi,Aaron.Zhang,Taylor.Ynag | 1. Initial Commits; |
| V1.1 | 2020-5-18 | Laker.Liu |  |

目录

[1. Background 4](#_Toc38298483)

[2. Introduction 4](#_Toc38298484)

[3. Manual 4](#_Toc38298485)

[3.1. Server Manual 4](#_Toc38298486)

[3.1.1. Objects Descriptions 5](#_Toc38298487)

[3.1.2. Assets Descriptions 5](#_Toc38298488)

[3.1.3. Script Descriptions 6](#_Toc38298489)

[3.1.4. Features and Interfaces 6](#_Toc38298490)

[3.1.5. Instructions for Editor debugging in on Server side 11](#_Toc38298491)

[3.2. Client Instructions 12](#_Toc38298492)

[3.2.1. Main Scene instructions 12](#_Toc38298493)

[3.2.2. NetworkClient.cs： 16](#_Toc38298494)

[3.2.3. Core scripts： 19](#_Toc38298495)

# Background

PicoVR Broadcast Platform is developed by Pico Technology, which aims to provide synchronized multi-user experiences. In this solution, multiple VR standalone headsets will work as clients and PC/Pads as server. Server controls all clients all together using the same local network.

To provide better support for partners, now we are releasing Pico Broadcast Platform to open-source platform for our partners.

# Introduction

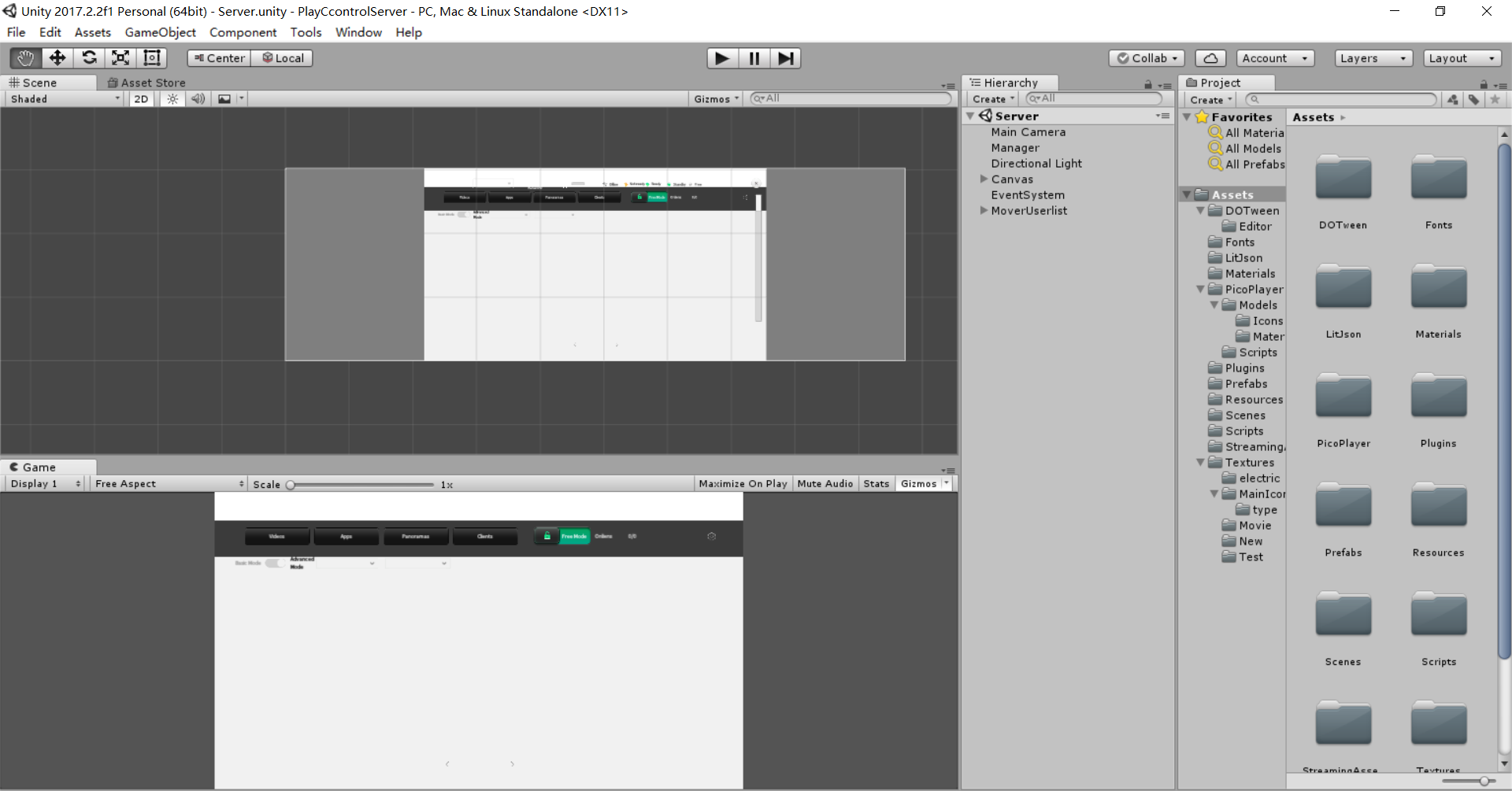
Pico Broadcast Platform consists of two parts: Server and Client. Open-sourced codes includes both parts. This Instructions are to describe these two parts briefly.

* The Unity Version of the project is 2017.2.2f1
* You need copy the "pre resource" folder under the broadcast server to the root directory of the device system, if the folder already exists in the directory, merge the folder;
* The APK packaged on the client needs to be signed, and the signing account can be provided by contacting Pico technical support.

# Manual

## Server Manual

Notes: The following parts is brief introductions to server part. For more detailed implementations, please refer to Unity source code.



The main Unity scene of Server is as above.

### Objects Descriptions

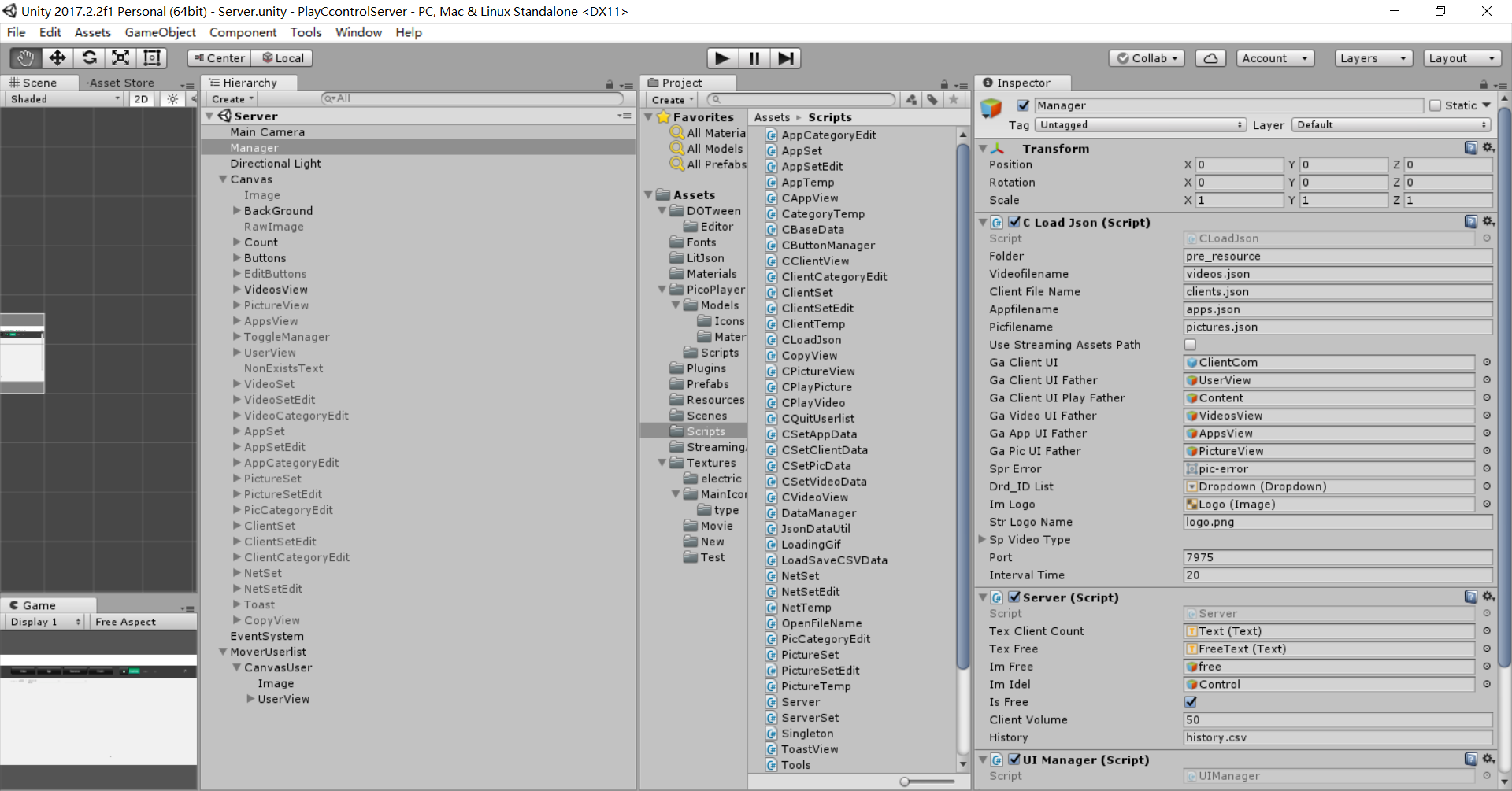
* **Manager**：Core manager of server which handles: Read broadcast configure files, assets initialization, scripts initialization, server initialization etc.
* **Canvas**，MoverUserlist: UI System, Edits, Display etc.

### Assets Descriptions

Assets category which includes plugins and etc.

* **Scripts**：Source code of scripts
* **Textures**：Texture resources used in Software

Screenshot of main scene’s contents



### Script Descriptions

#### CLoadJson.cs

CLoadJson is used to initialize each module. Singleton initialize each module by loading configure files.

### Features and Interfaces

* Start() ： Load configure files , initialize UI, initialize sockets etc.
* void InitSocket() : Initialize Server interfaces
* void SocketSend() : Send sockets
* void SocketReceive() : Receive Sockets
* void SocketQuit() : Close connections
* void fnLoadClientDataOver() : Process client data
* void fnLoadVideoDataOver() : Process video list
* void fnLoadAppDataOver() : Process application list
* void fnDropUserIdSelect : Select User ID to sync
* void fnSetClientStatus : After connection established, use SN code to set client status

Notes: Please refer to source code for details

#### Server.cs

Server is using Unity RPC network transmission, interfaces details:

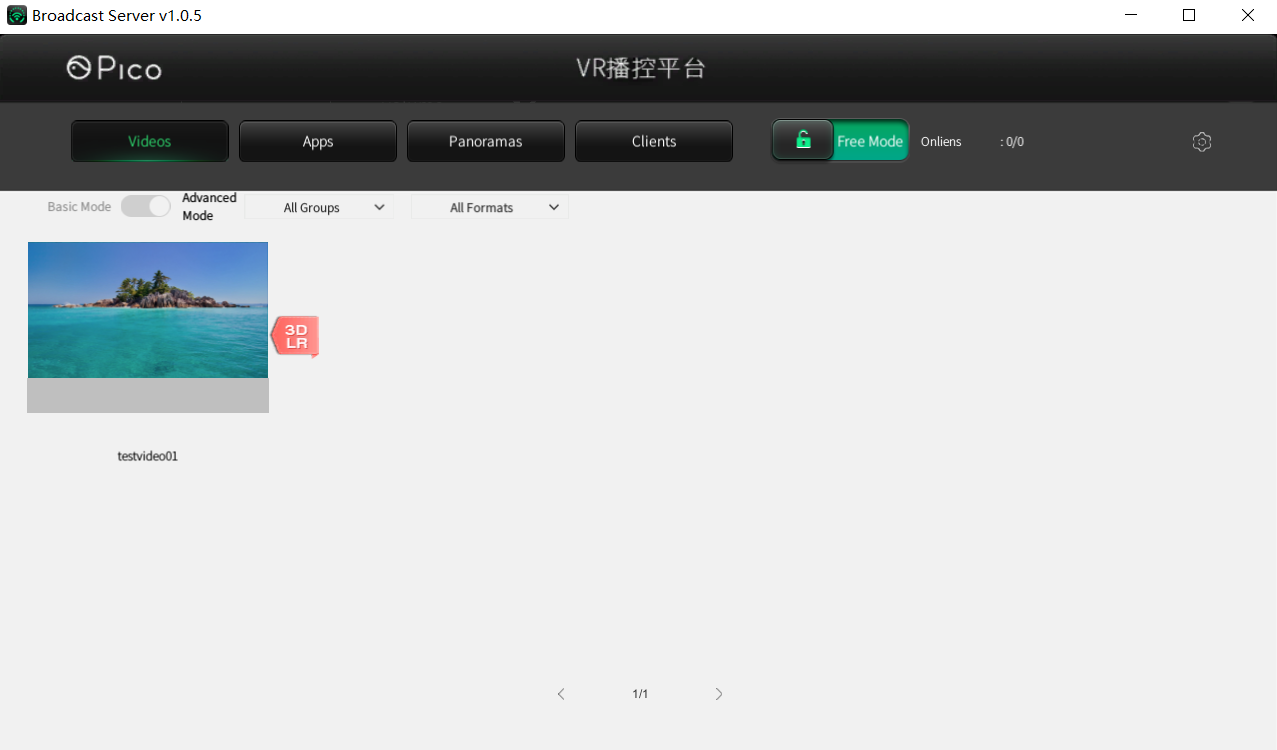
* public void fnStartServer() Start Server in Simulcast mode
* public void fnStopServer() Stop Server in Simulcast mode
* public void fnStartVideo()Start Video in Simulcast mode
* public void fnPlayVideo(long playTime = 0)
* public void fnPlayPlayingVideo(long playTime = 0) ： Play video in Video Player in Simulcast mode
* public void fnPauseVideo() ： Pause Video in Simulcast mode
* public void fnStopVideo() ： Stop Video or Restart in Simulcast mode
* public void fnPlayPicture(): Play playing picture in Simulcast mode
* public void fnStopPicture() ： Stop playing piucture in Simulcast mode
* public void fnSetClientFreeOrIdle() ： Set Free mode
* public void fnStopOrStartSynchroClient(string strSn, bool IsSynchro)： Synchronize or stop client
* public void fnServerSeekToClient() ： Seek servers to clients

Notes: Find interfaces or usages details in Server.cs

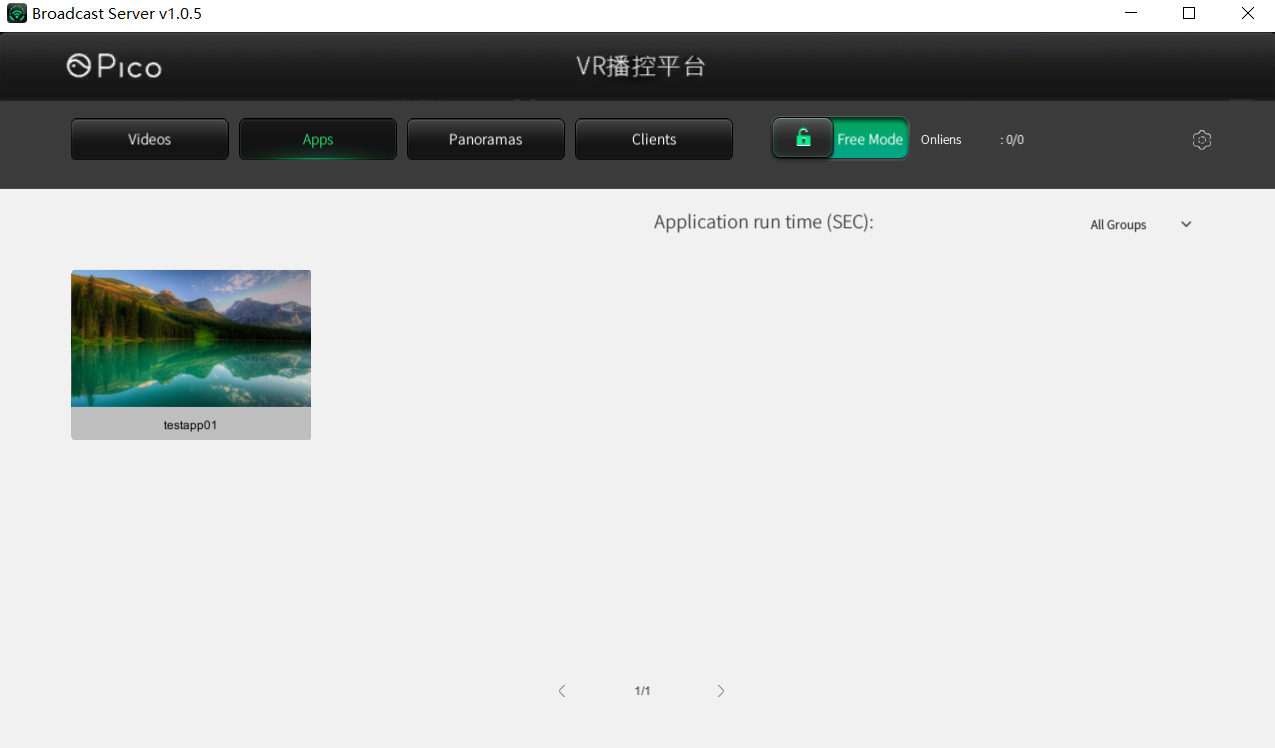
#### UIManager：

UIManager is used for managing UI for each panel

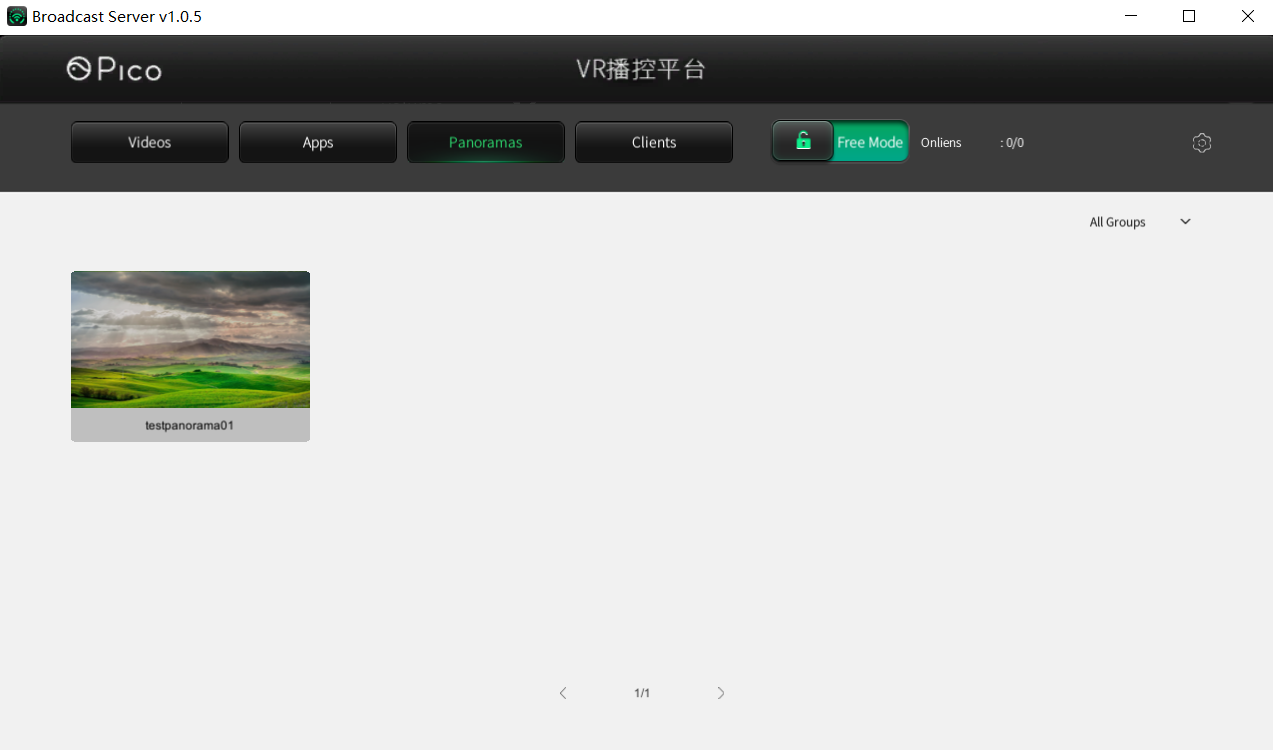
* Switch UI panels and push/pop logic of UI
* GetUI()
* RunUI()
* PushUI()
* PopUI()
* ReplaceUI()



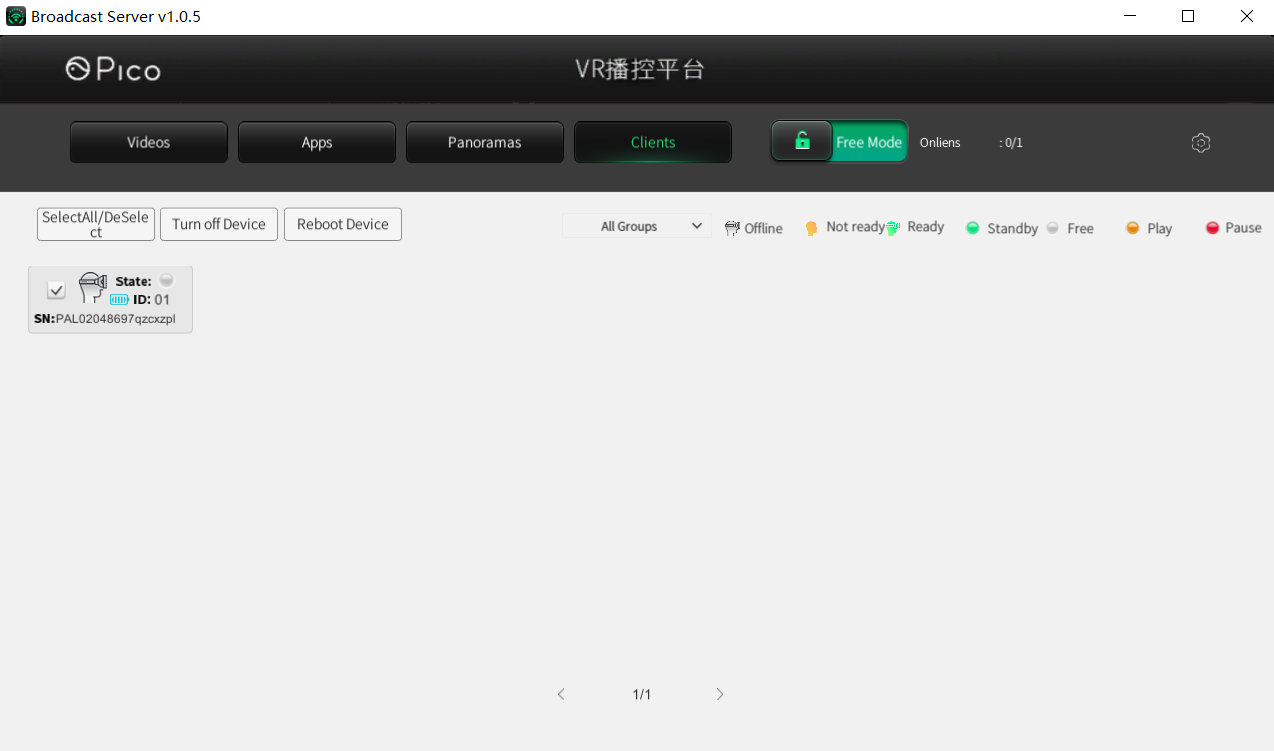
* Video Module：CVideoView.cs



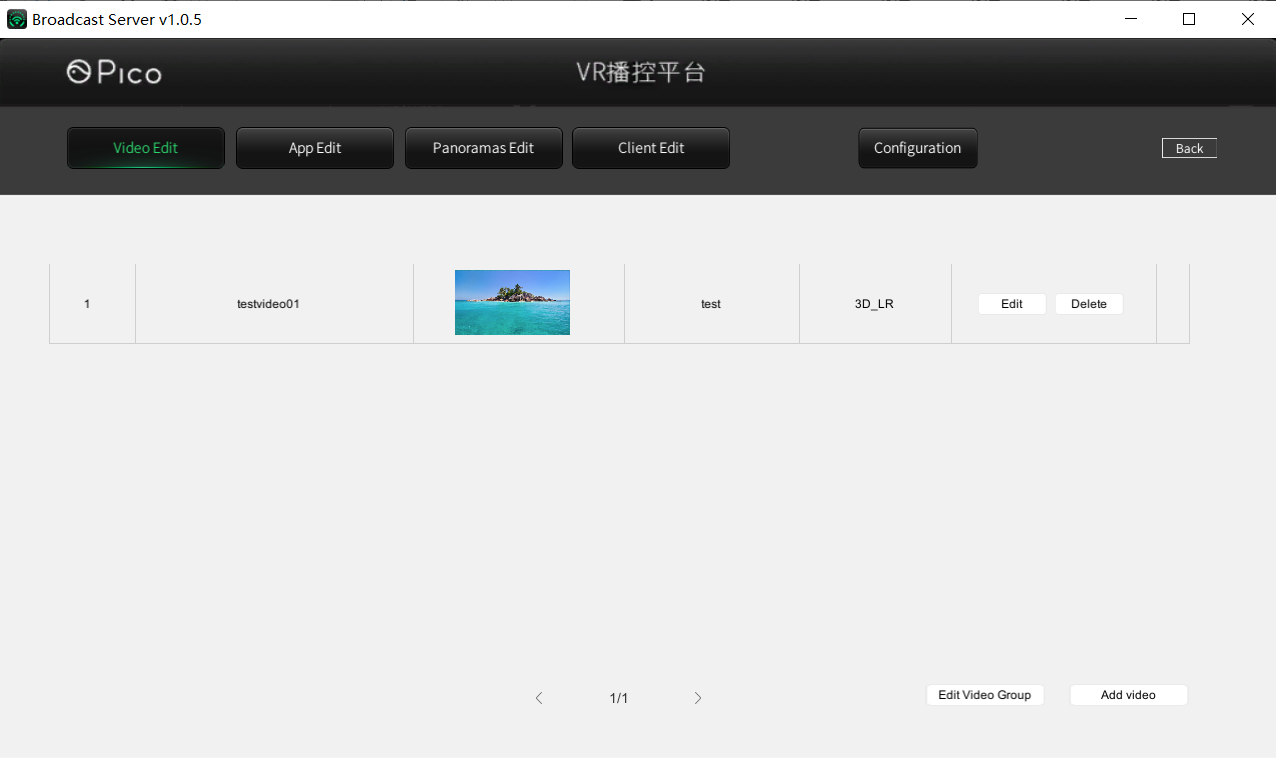
* App Module：CAppView.cs



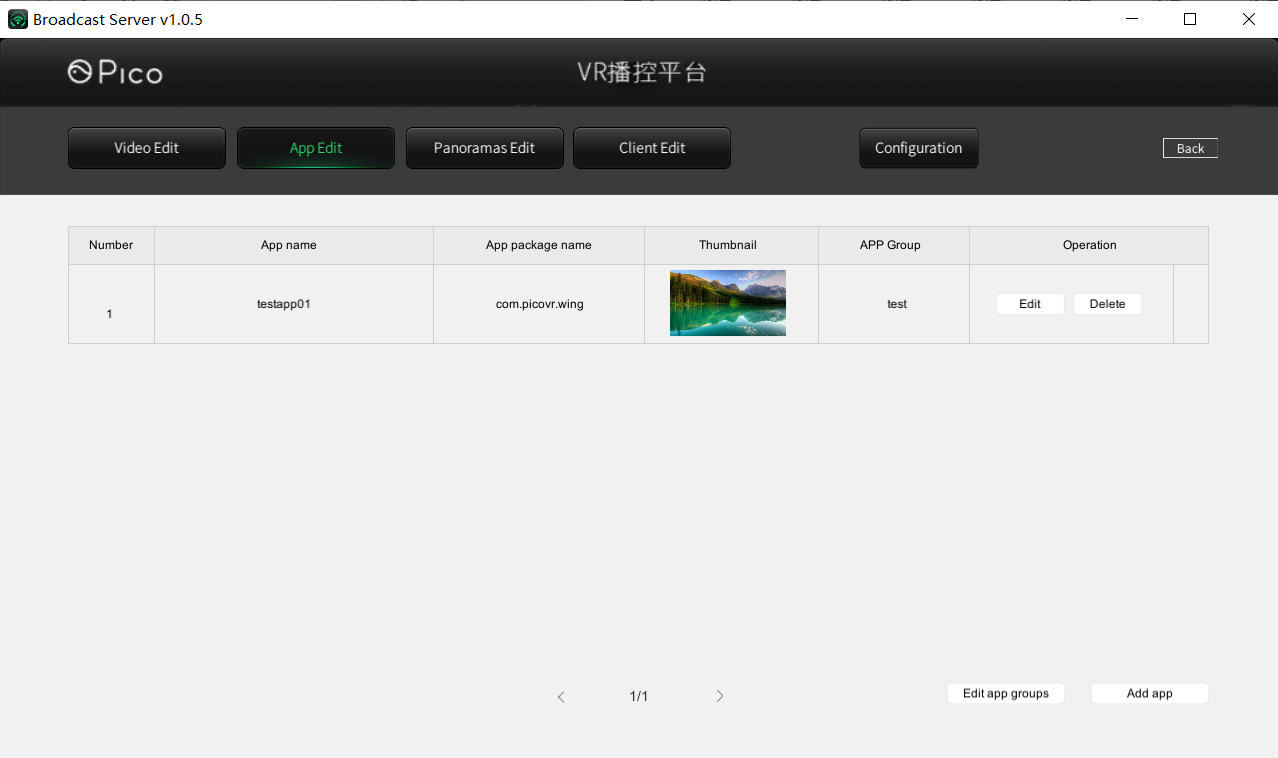
* Panoramic Picture Module：CPictureView.cs



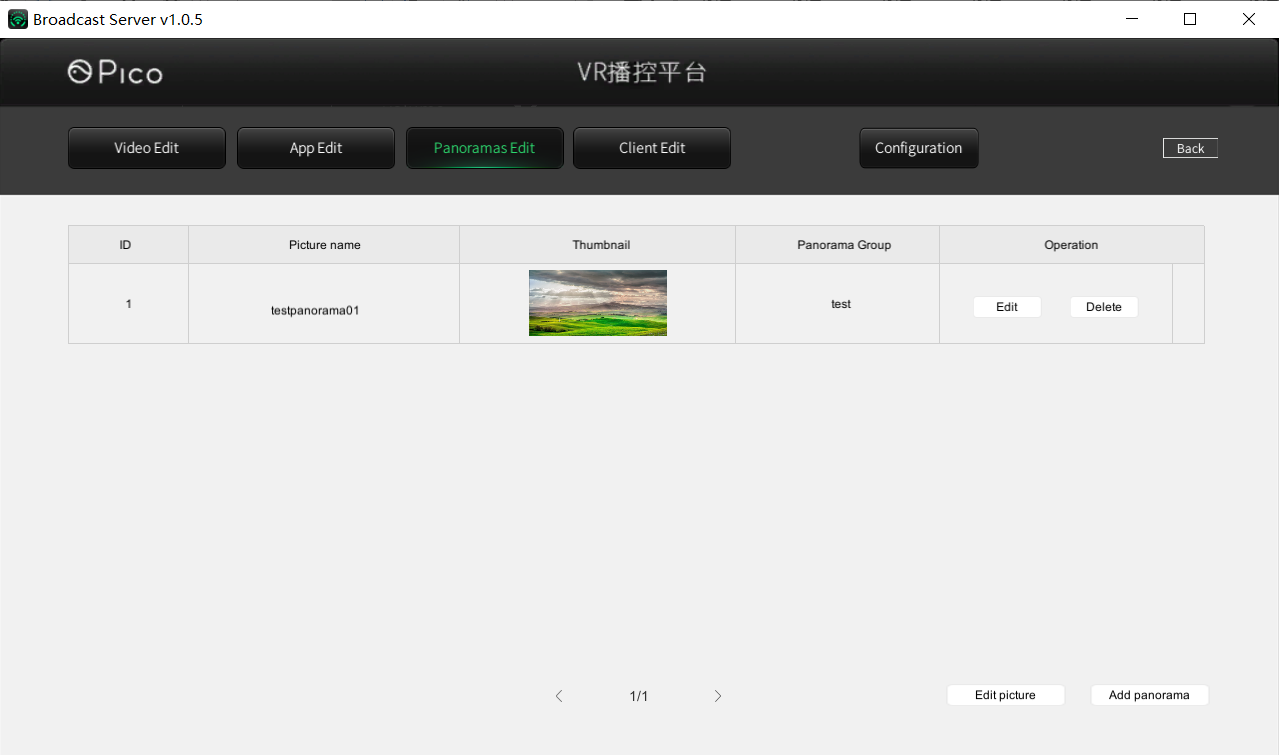
* Client Module：CClientView.cs



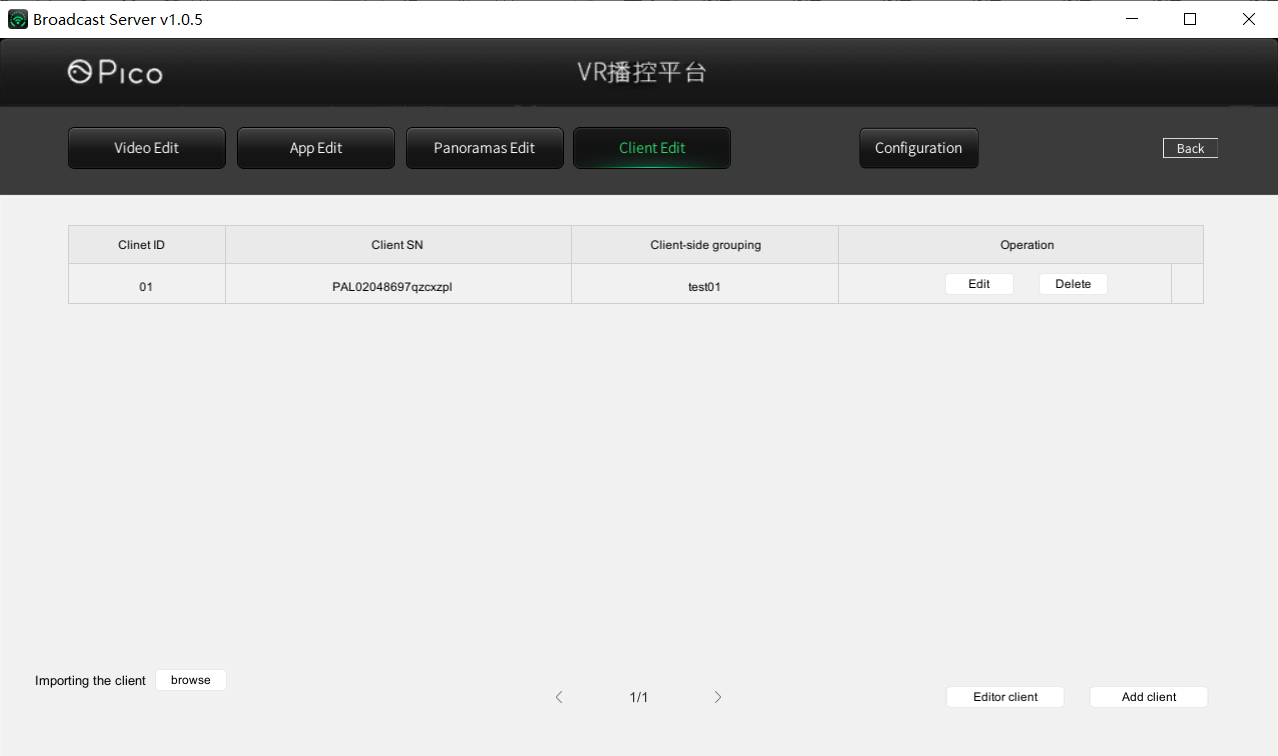
* Video management：VideoSet.cs VideoSetEdit.cs



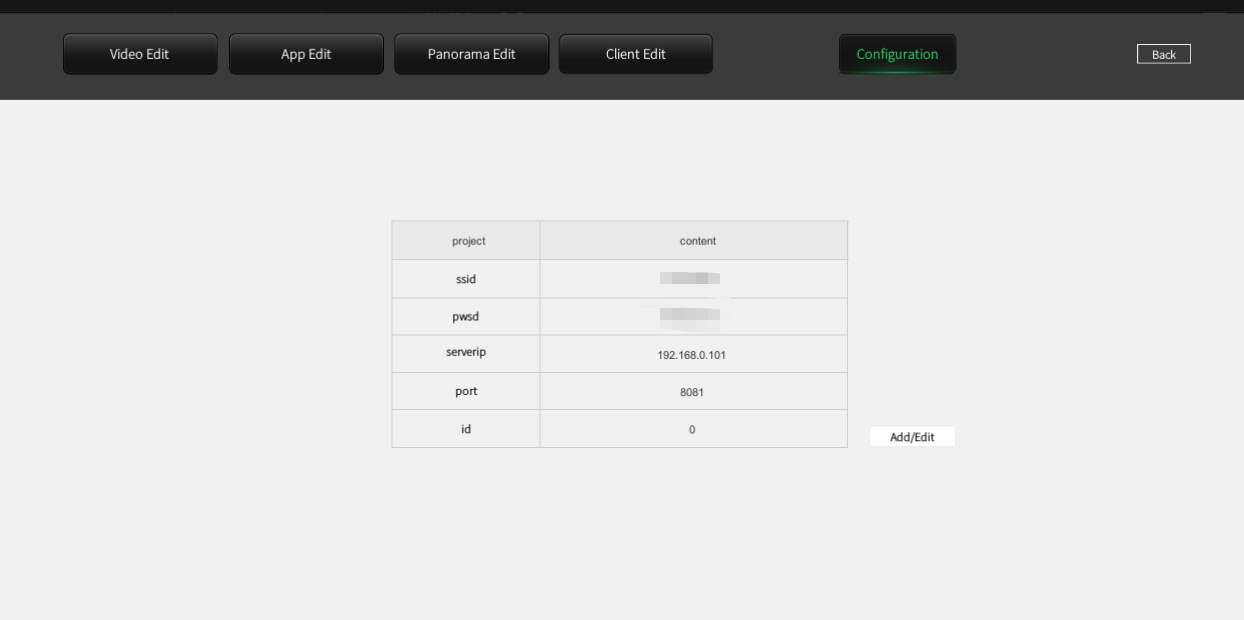
* Application management：AppSet.cs AppSetEdit.cs



* Panoramic picture management：PictureSet.cs PictureSetEdit.cs



* Client Editting: ClientSet.cs ClientSetEdit.cs



* Network configuration management ： NetSet.cs NetSetEdit.cs
* Note: Add network configuration for connection between client and Server. The specific adding method is as follows:

I. Open the broadcast server folder and start the "broadcast server. exe" program;

II. Click the "Settings" - > "network configuration" icon on the right side of the title bar to enter the "network configuration" interface;

III. click "add / modify"

Enter the relevant information in the input box:

SSID: input the configured Wifi account;

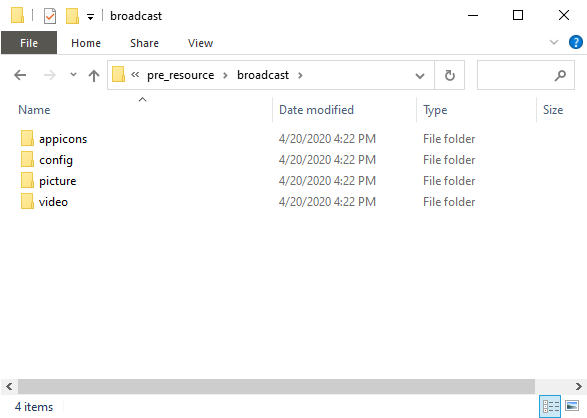
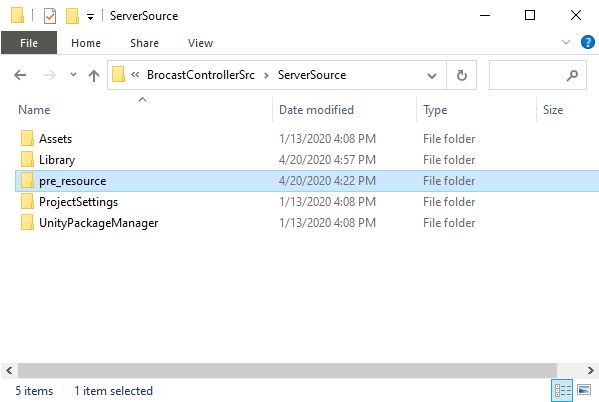
Pswd: enter the configured Wifi password;

ServerIP: connect the PC to the router, check the IP address of the PC, and enter it;

Port and ID are not entered and remain in the default state.

### Instructions for Editor debugging in on Server side

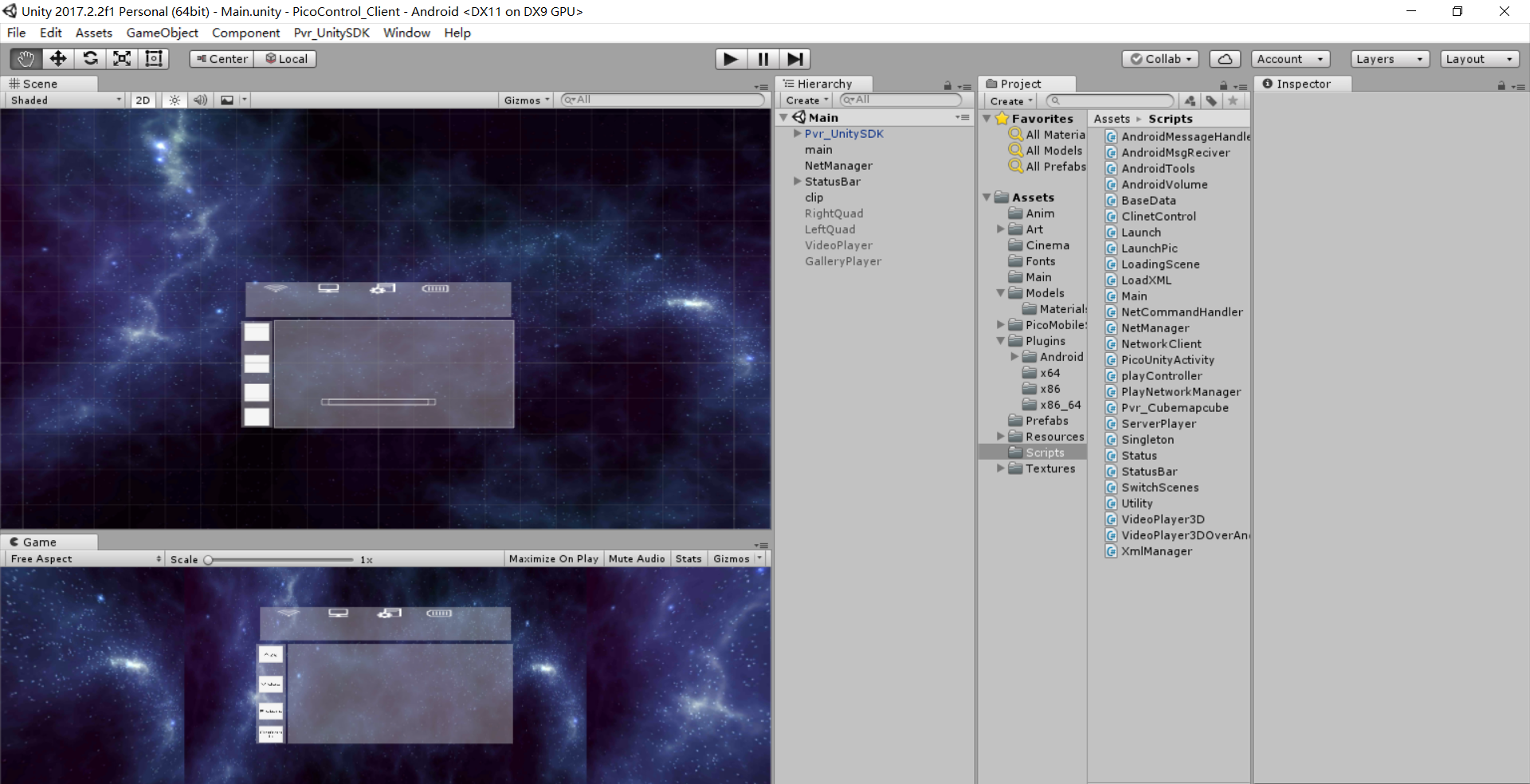
* Place configuration files under path [Server\pre\_resource\broadcast] of Server project



## Client Instructions

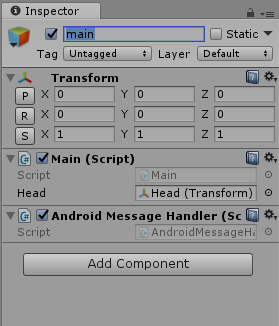
* Notes: The following parts is brief introductions to client part. For more detailed implementations, please refer to Unity source code

### Main Scene instructions



The above screenshot is Main scene of Client

#### Main



**Main.cs**

Notes： Define singleton, status, initializing UI/assets/scripts

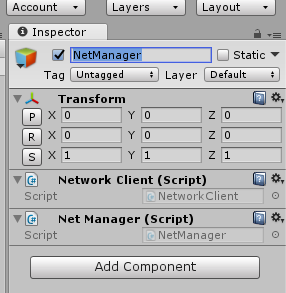
* void Awake() : Gameobject initialization
* Start() : Read configure files, upgrading config from ver 2.0 to 3.0 (leave unprocessed if upgraded), network status monitoring, accept Android data initialization
* private IEnumerator OnClientResume() ： Operate when client returns values and use SendToServerMassage to send messages to server
* private IEnumerator OpenFreeMode() ： Enable Free mode
* private void OnNetStateChanged(bool connected): Operate when network status change
* public void RefreshFiles(string value): Read configure file folder and load each asset

**AndroidMessageHandler.cs:**

Receive android messages and operate，for communications between Unity and Android, please refer to Unity manuals.

* public void Android\_SendTimeToUnity(string time) : Time to receive messages from Android side
* public void Android\_SendBatteryInfo(string info) ： Receive battery status from Android
* public void Android\_SendWifiConnectedToUnity(string connected): Receive Wi-Fi connection infomation
* public void Android\_SendWifiStatusToUnity(string state): Receive Wi-Fi status infomation
* public void Android\_SendWifiLevelToUnity(string level): Receive Wi-Fi signal level infomation
* public void Android\_SendSensorStatusChange(string value): Receive Sensor status infomation

#### NetManager



### NetworkClient.cs：

Notes：Client Network operating：

* public void SendToServerMessage(string command, string parameter): Send messages to server
* private void CheckConnection() ： Check connection status
* private void NetStateChanged(bool connected) ： Operation when network status changes
* private void RequestMessage(string command, string parameter, NetworkMessageInfo info): Request data processing
* NetManager.cs: Network managers
* private void OnNetworkCommand(string command, string parameter): Wait to process command messages
* NetCommandHandler.cs: Receive messages from server and process
* public void OnCommand(string command, string parameter) ：command is the command information， parameter is the transmitted String to be parsed（video）
* private bool ParsePlayParameter(string parameter, out int seekPos, out string fullPath, out VideoType mode, out string type): Parse transmitted String（Gallery）
* private bool ParseGalleryParameter(string parameter, out string name, out string category)

#### Launch.cs

Launch.cs: Used to manage the logical relationship between the picture playing module and the video playing module.

* public void StartLaunch(object goal,UIPointerEventArgs currentUI): Select from app mode, video mode and picture mode
* public void Video(string videoname, string videotype):Play video,use videoname and videotype to judge videoname and videotype

#### 3.2.2.2 playController.cs

playController.cs:Used to control the video player.

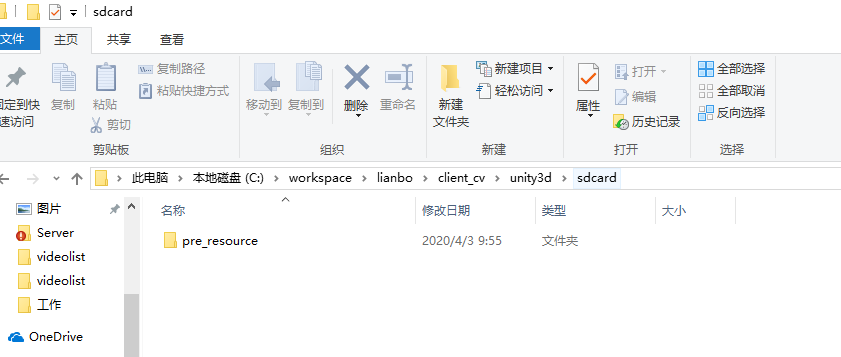
* public void PlayOrPause():Play video or pause video

### Core scripts：

* BitVideoPlayer.cs : Video player control, please refer comments in scripts
* VideoUIControlBit.cs VideoPlayerUIManagerBit.cs: Display and control UI

#### Editor debugging of Client

* Place configure files in clientCV project under path: (client\_cv\unity3d\sdcard\pre\_resource\broadcast)；



* Editor support only debugging part of the features, for overall debugging please apply on actual Android device

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*