### 1. Establish connection：

TCP connects to the server and specifies socket to establish long connections.

### 2. Message data structure：

public static byte[] createMessageByJson(String jsonBody,short msgType){

short size = (short) (jsonBody.length() + 6);

if(size <= 6) return null;

ByteBuffer bf = ByteBuffer.allocate(size);

bf.putShort(size);

bf.putShort(msgType);

bf.put((byte)0);

bf.put((byte)1);

bf.put(jsonBody.getBytes());

bf.flip();

byte[] result = new byte[size];

bf.get(result);

return result;

}

**Message data structure:**

2byte size | 2byte type | 2byte Stationary | json main body

Info：

size，value type： short（16bit integer）；

type，value type： short（16bit integer）；

Stationary，0 and 1 two bite in sequence.

The 3 part is package head, where the size includes package head.

Json main body，json object mathod main body。

Suppose a message 1002 is used to play, then its JSON is as follows：

{    **"PlayId"**:**1**,  **"VideoName"**:**"MovieTest"**,    **"VideoType"**:**"\*"**,    **"VideoPath"**:**"/sdcard/1.mp4"**,    **"StereoType"**:**1**,    **"UnixTime"**:**"11111111"**}

Size is message size;

Msgtype is the type of message:

2 byte Stationary as (0，1);

Follows by bodyjson。

### 3. Message Type：

Among them, the message with parameters is JSON body, and the message without parameters has no JSON body.

//Heartbeat server to client

public static final short TYPE\_REQ\_SEQ\_SERVER\_TIME = 1001;

//Heartbeat client to server

public static final short TYPE\_PUSH\_SERVER\_TIME = 1001;

public class Server\_TIME{ }

//Play Video

public static final short TYPE\_PLAY\_VIDEO = 1002;

json body：

public class Push\_Play\_Video {

public int PlayId; //ID number

public String VideoName; //Video Title

public String VideoType; //Video Type（2d，3d ect.）

public String VideoPath; //Video Path

public int StereoType; //Video type parameter 2

public String UnixTime; //time stamp

}

The specific play types corresponding to each video type of the video type can be redefined.

Among them, VideoType fixed selection "\*", StereoType according to video type is as follows：

0 ： normal 2d video

1 ： 3D Left and Right

2 ： 3D Up and Down

3： 180° 2d

4 ：180° left and right 3d

5： 180° Up and down 3d

6： Panoramic 2D

7： Panoramic left and right 3D

8： Panoramic up and down 3D

//Pause

public static final short TYPE\_PAUSE\_VIDEO = 1003;

//Stop play and quit

public static final short TYPE\_STOP\_VIDEO = 1004;

//Resume

public static final short TYPE\_RESUME\_VIDEO = 1005;

//Seek

public static final short TYPE\_SEEK\_VIDEO = 1006;

public class Push\_Seek {

public int time;

}

Unit：second

//Set video sound：

public static final short TYPE\_SET\_VOLUME = 1007;

public class Push\_Set\_Vol {

public int volume\_value;

}

Range： 0 - 15;

//Shielding button is temporarily unsupported.

public static final short TYPE\_BLOCK\_KEY = 1008;

//Power off

public static final short TYPE\_POWER\_OFF = 1009;

//Restart

public static final short TYPE\_RESTART = 1010;

public static final short TYPE\_RECENT = 1011;

//Start App

public static final int TYPE\_START\_ACTIVITY = 2001;

Play message flow：

Server send 1002.

The all-in-one machine sets the server to listen for IP and port：

adb shell am start com.deepoon.eggchair/.MainActivity

Simple Server Demo

At present, this demo only implements the example message startup application, and the project is server in the compressed package zip.

For other messages, please refer to <message.CS>, if you only need to refer to message encapsulation and unpacking, refer to <Message.CS>.

Dependency Library： Newtonsoft.json

Download address：http://json.codeplex.com/