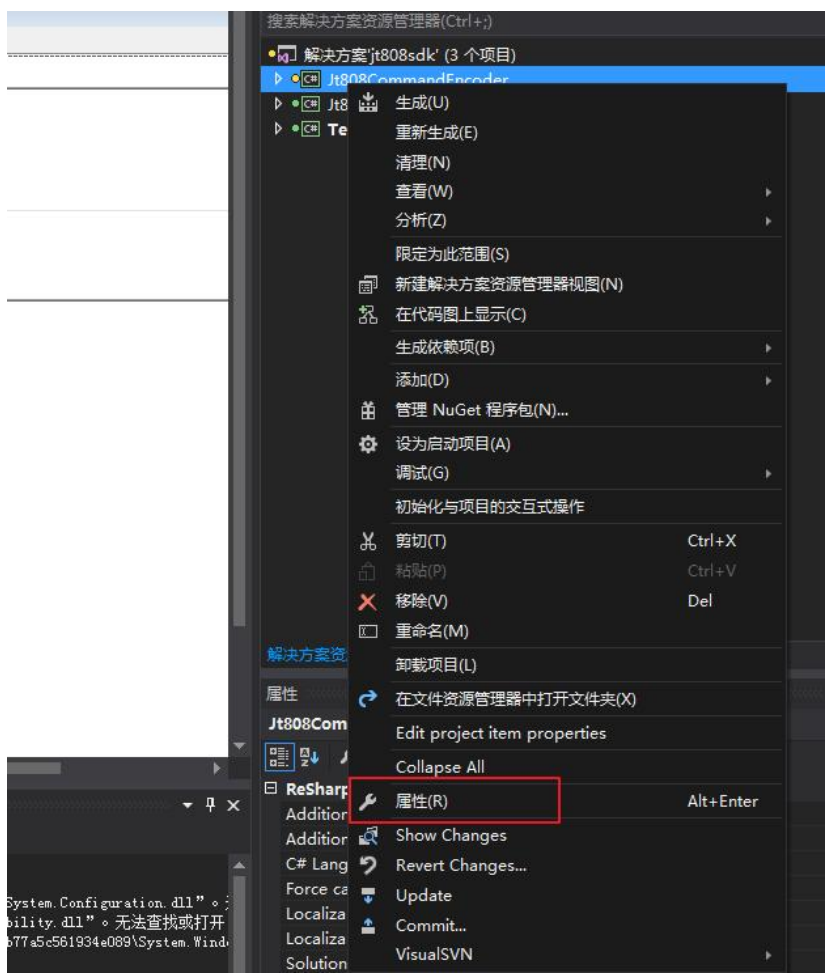


1.Introduction to Development Package

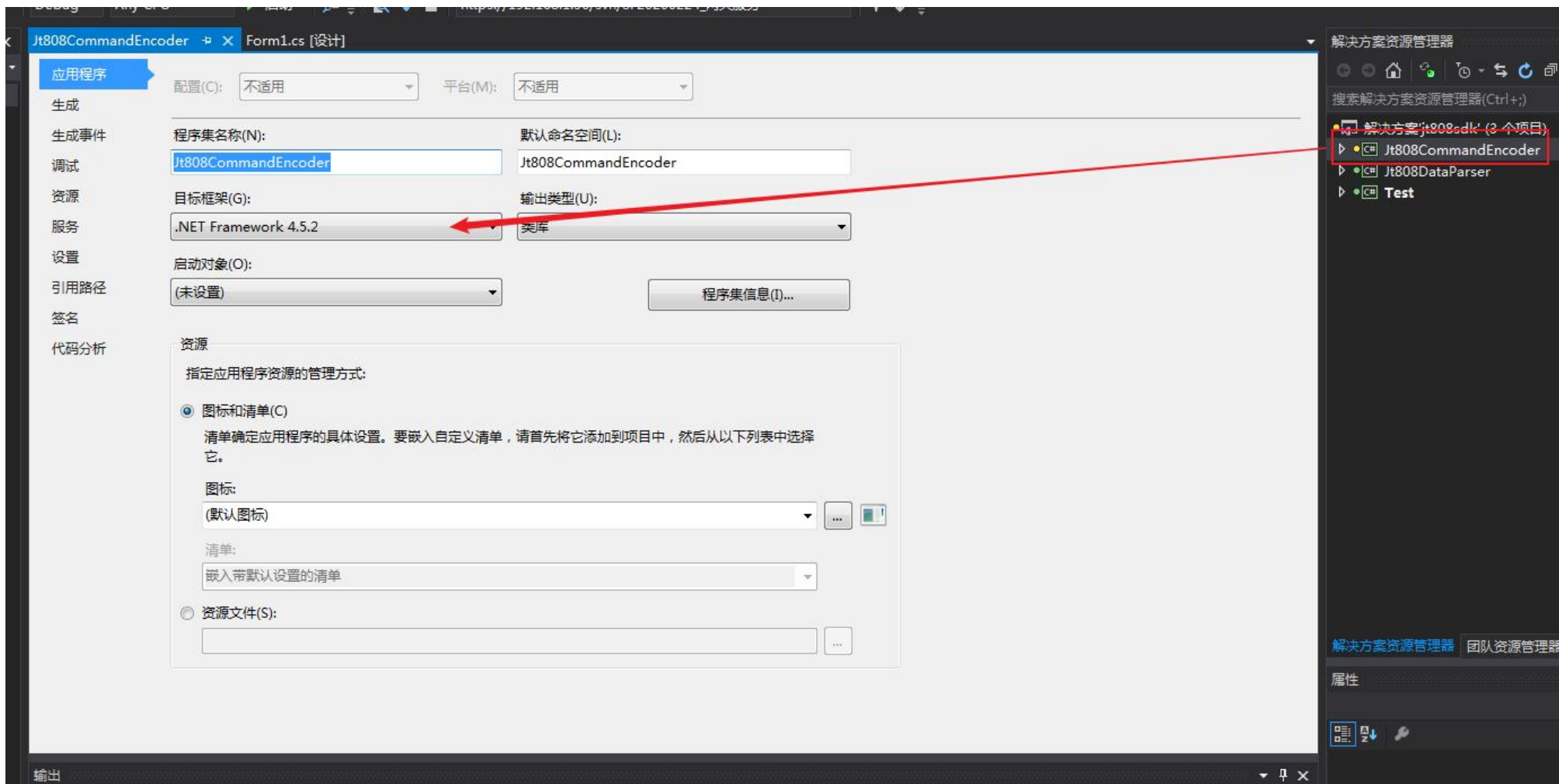
This development is mainly aimed at AOVX products, a toolkit for decoding device uplink data and packaging platform downlink control commands. The target framework used is .NET Framework 4.5.2. If your development program is not using this framework, you can obtain the framework source code of the development package and modify the target framework of the toolkit. The modification method is as follows:

1.1.Modifying the Objective Framework

Select the project in the tool source code and click on "Properties"



Then find the same framework for your project in the target framework drop-down box.



2. Source Code Description

2.1. Introduction to Source Code

The toolkit includes a total of 3 projects:

Jt808CommandEncoder: When sending control commands to platform devices, package the platform's downstream commands, and then the system that manages device access can distribute the packaged data to the devices.

Jt808DataParser: When the system that manages device access receives data reported by the device, it can call the parsing data method in this project to decode the data reported by the device to the platform and parse it into JSON format strings.

Test: testing tool used to verify data decoding and downstream command packaging.

2.2.Jt808CommandEncoder

CommandEncoder: It is the entry method class for platform downstream command packages.

Method encapsulation is based on the command message ID. Each message ID is encapsulated into two different return types of messages, while others return a hex string and byte [].

```

/// <summary>
/// Set Device Parameters (0x8103)
/// </summary>
/// <param name="deviceId"></param>
/// <param name="msgFlowId"></param>
/// <param name="dicParameters"></param>
/// <returns>hex string</returns>
1 个引用
public static string setDeviceParameterForHex(string deviceId, int msgFlowId, Dictionary<Int32, Object> dicParameters) {
    return PackageUtil.packageHexCommand(deviceId, msgFlowId, 0x8103, dicParameters);
}

/// <summary>
/// Set Device Parameters (0x8103)
/// </summary>
/// <param name="deviceId"></param>
/// <param name="msgFlowId"></param>
/// <param name="dicParameters"></param>
/// <returns>byte[]</returns>
0 个引用
public static byte[] setDeviceParameterForBytes(string deviceId, int msgFlowId, Dictionary<Int32, Object> dicParameters)
{
    return PackageUtil.packageBytesCommand(deviceId, msgFlowId, 0x8103, dicParameters);
}

```

2.2.1 0x8103（Set deive parameters）

public static string setDeviceParameterForHex(string deviceId, int msgFlowId, Dictionary<Int32, Object> dicParameters)

public static byte[] setDeviceParameterForBytes(string deviceId, int msgFlowId, Dictionary<Int32, Object> dicParameters)

方法：setDeviceParameterForHex：对命令进行组包并返回 16 进制字符串

方法：setDeviceParameterForBytes：对命令进行组包并返回字节数组

参数 1：deviceId：设备 S/N 码，即设备 12 位的编号

参数 2：msgFlowId：下发指令的流水号（2 个字节（1~65535）），这个需要进行存储，用来匹配设备对此命令的应答

参数 3：dicParameters：下发参数字典；其中：key 对应的是下图中的“参数 ID”

参数项	参数ID	参数类型	参数长度	参数单位	适配系列	备注
终端心跳间隔	0x0001	DWORD	4	秒	GAV	
服务器APN	0x0010	STRING	/	/	GAV	
APN 名称	0x0011	STRING	/	/	GAV	
APN 密码	0x0012	STRING	/	/	GAV	
主服务器地址	0x0013	STRING	/	/	GAV	支持域名和IP
备份服务器地址	0x0017	STRING	/	/	GAV	支持域名和IP
主服务器端口	0x0018	DWORD	4	/	GAV	TCP或者UDP端口
休眠上报间隔	0x0027	DWORD	4	秒	V	
行驶上报间隔	0x0029	DWORD	4	秒	V	
拐点角度	0x0030	DWORD	4	度	V	小于180度
最高速度	0x0055	DWORD	4	km/h	V	
超速持续时间	0x0056	DWORD	4	秒	V	
车辆里程表读数	0x0080	DWORD	4	1/10km	V	例如:103 = 10.3km
终端ID	0xF000	STRING	/	/	GAV	版本最大12位BCD码
行驶电压	0xF001	WORD	2	毫伏	V	
停车电压	0xF002	WORD	2	毫伏	V	
休眠电压	0xF003	WORD	2	毫伏	V	
NTP服务器地址	0xF004	STRING	/	/	GAV	支持域名和IP
NTP服务器端口	0xF005	DWORD	4	/	GAV	
时区	0xF006	BYTE	1	/	GAV	[-12,12]
协议类型	0xF007	BYTE	1	/	GAV	[0:JT808 1:TAIP]
协议加密方式	0xF009	BYTE	1	/	GAV	[0:NULL 1:RSA 2:AES 3:XTEA]
定位星系	0xF00A	BYTE	1	/	GAV	[0:gps+bd 1:gps+glo 2:gps+ga]
WIFI使能	0xF00B	BYTE	1	/	GAV	[0:关 1:开]
WIFI工作模式*	0xF00C	BYTE	1	/	GAV	[0:ap 1:sta]
WIFI最大AP数	0xF00D	BYTE	1	/	GAV	
WIFI单次扫描时间	0xF00E	WORD	2	秒	GA	
BT使能	0xF00F	BYTE	1	/	GAV	[0:关 1:开]
BT工作模式	0xF010	BYTE	1	/	GAV	[0:host 1:slave]
BT最大节点数	0xF011	BYTE	1	/	GAV	
BT节点超时时间	0xF012	BYTE	1	分	GAV	BT自动扫描,用于判断节点离线
BT单次扫描时间	0xF013	WORD	2	秒	GA	
BT上报掩码	0xF014	BYTE	1	/	GAV	
输入GPIO模式	0xF015	WORD	2	/	V	高字节[通道号] 低字节[0:数字 1:模拟]
GPIO方向*	0xF016	WORD	2	/	V	
传输协议	0xF017	BYTE	1	/	GAV	[0:TCP 1:UDP 2:MQTT]
上报掩码	0xF018	DWORD	4	/	GAV	
Gsensor使能	0xF019	BYTE	1	/	GA	[0:关 1:开]
Gsensor灵敏度	0xF01A	BYTE	1	/	GAV	[0-255]
Gsensor量程	0xF01B	BYTE	1	/	GAV	[0:2g 1:4g 2:8g 3:16g]
Gsensor次数	0xF01C	WORD	2	/	GA	
Gsensor时间	0xF01D	DWORD	4	秒	GAV	
Gsensor触发间隔	0xF01E	DWORD	4	秒	GA	
Gsensor上报掩码	0xF01F	BYTE	1	/	GAV	
Light使能	0xF020	BYTE	1	/	GA	[0:关 1:开]
Light阈值	0xF021	WORD	2	/	GA	
Light触发间隔	0xF022	DWORD	4	秒	GA	
温湿度使能	0xF023	BYTE	1	/	G	[0:关 1:开]
温度上限	0xF024	WORD	2	/	G	
温度下限	0xF025	WORD	2	/	G	
湿度上限	0xF026	WORD	2	/	G	
湿度下限	0xF027	WORD	2	/	G	
温湿度触发间隔	0xF028	DWORD	4	秒	G	
GPS使能	0xF029	BYTE	1	/	G	[0:关 1:开]
工作模式	0xF02A	BYTE	1	/	GA	
备份服务器端口	0xF02B	DWORD	4	/	GAV	TCP或者UDP端口
缓存开关*	0xF02C	BYTE	1	/	GAV	
ACK开关*	0xF02D	BYTE	1	/	GAV	
上报周期	0xF02E	DWORD	4	秒	GA	
采样周期	0xF02F	DWORD	4	秒	GA	
透传AT指令	0xF030	STRING	/	/	GAV	参考AT指令手册

Value corresponds to the parameter ID that needs to be set.

2.2.2 0x8104 (Query device parameters)

```
public static string queryDeviceParameterForHex(string deviceId, int msgFlowId)

public static byte[] queryDeviceParameterForBytes(string deviceId, int msgFlowId)
```

Method: queryDeviceParameterForHex: Package the command and return a hexadecimal string.

Method: queryDeviceParameterForBytes: Package the command and return a byte array.

Parameter 1: deviceId: Device S/N code, which is the 12 digit number of the device.

Parameter 2: msgFlowId: Serial number of the issued instruction (2 bytes (1~65535)), which needs to be stored to match the device's response to this command.

2.2.3 0x8105 (Device control commands)

```
public static string deviceControlCommandForHex(string deviceId, int msgFlowId, Dictionary<int, Object> dicParameter)

public static byte[] deviceControlCommandForBytes(string deviceId, int msgFlowId, Dictionary<int, Object> dicParameter)
```

Method: queryDeviceParameterForHex: Package the command and return a hexadecimal string.

Method: queryDeviceParameterForBytes: Package the command and return a byte array.

Parameter 1: deviceId: Device S/N code, which is the 12 digit number of the device.

Parameter 2: msgFlowId: Serial number of the issued instruction (2 bytes (1~65535)), which needs to be stored to match the device's response to this command.

Parameter 3: dicParameters: Command parameter dictionary; Where: key corresponds to the "command word" in the following figure.

指令	命令字	命令参数	适配系列	备注
重启	4	/	GAV	
恢复出厂	5	/	GAV	
OTA升级	32	TYPE;MODE;VERSION;PROTOCOL;URL;MD5	GAV	TYPE:0表示app升级,1表示core升级 MODE:0表示完整版,1表示差分包 VERSION:目标版本号 PROTOCOL:0表示FTP协议,1表示HTTP协议 URL:实际升级使用的完整URL连接 MD5:固件MD5值
油路	33	MODE	V	MODE:0:开 1:关
电源	34	MODE	V	MODE:0:关 1:开
GPIO输出	35	CHANNEL;MODE	V	CHANNEL:Bit 0-15 可同时配置多路 MODE:0:关 1:开
透传AT	36	COMMAND	GAV	COMMAND:参考AT指令手册

Value corresponds to a specific parameter of the corresponding command word.

If the command word is 4 or 5, then value corresponds to null or "".

If the command word is 32, then value corresponds to a dictionary<String, String>, where key is TYPE, MODE, VERSION, PROTOCOL, URL, MD5;Value is their corresponding content.

If the command word is 33, then value is 0 (on) or 1 (off).

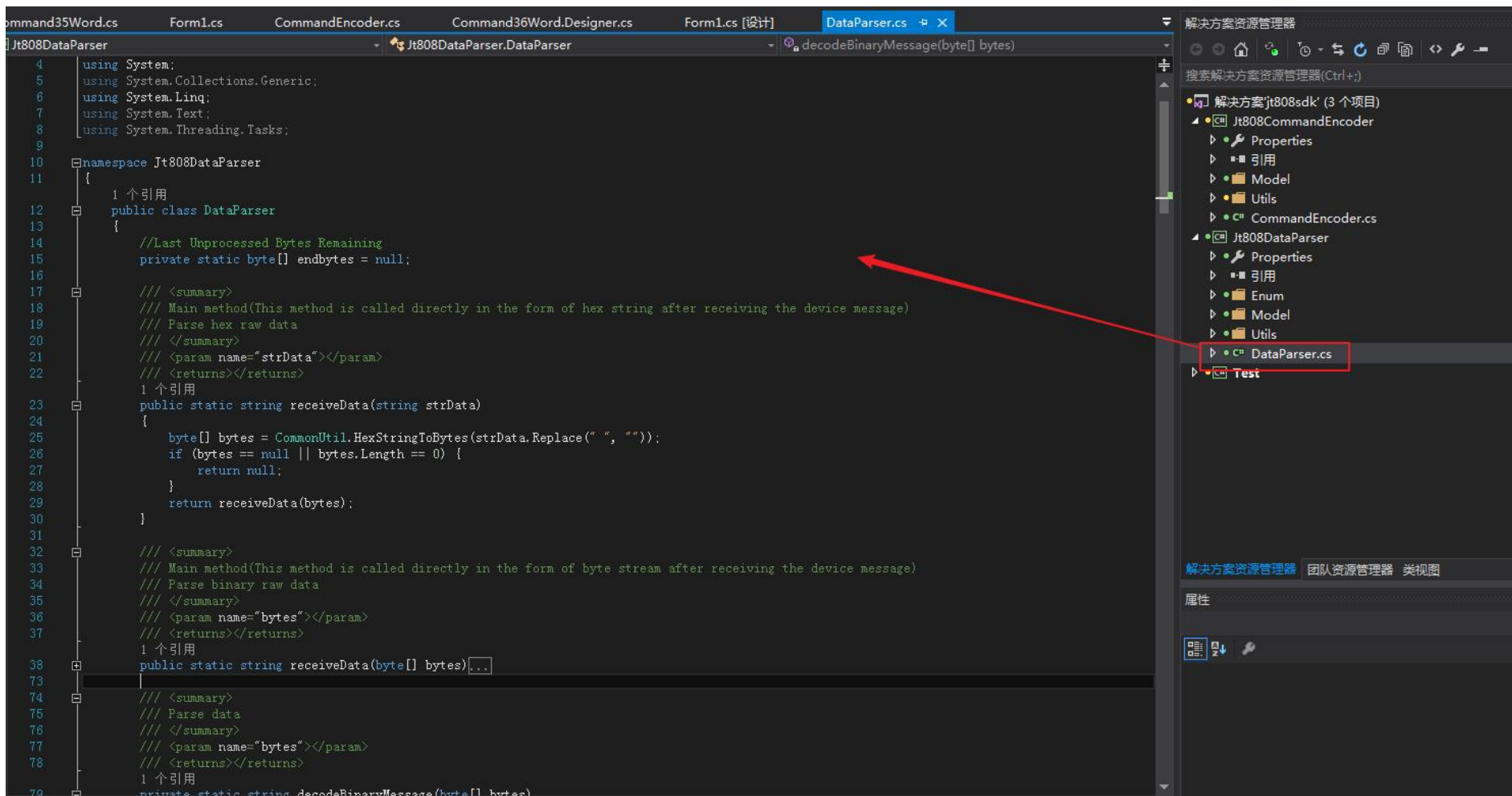
If the command word is 34, then value is 0 (off) or 1 (on).

If the command word is 35, then value corresponds to a dictionary Dictionary<String, String>, CHANNEL, MODE;Value is their corresponding content.

If the command word is 36, then value corresponds to the AT command string.

2.3.Jt808DataParser

DataParser: It is the device uplink data decoding entry method class.



```

4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Jt808DataParser
11 {
12     1 个引用
13     public class DataParser
14     {
15         ///Last Unprocessed Bytes Remaining
16         private static byte[] endbytes = null;
17
18         /// <summary>
19         /// Main method(This method is called directly in the form of hex string after receiving the device message)
20         /// Parse hex raw data
21         /// </summary>
22         /// <param name="strData"></param>
23         /// <returns></returns>
24         1 个引用
25         public static string receiveData(string strData)
26         {
27             byte[] bytes = CommonUtil.HexStringToBytes(strData.Replace(" ", ""));
28             if (bytes == null || bytes.Length == 0) {
29                 return null;
30             }
31             return receiveData(bytes);
32         }
33
34         /// <summary>
35         /// Main method(This method is called directly in the form of byte stream after receiving the device message)
36         /// Parse binary raw data
37         /// </summary>
38         /// <param name="bytes"></param>
39         /// <returns></returns>
40         1 个引用
41         public static string receiveData(byte[] bytes) {...}
42
43         /// <summary>
44         /// Parse data
45         /// </summary>
46         /// <param name="bytes"></param>
47         /// <returns></returns>
48         1 个引用
49         private static string decodeBinaryMessage(byte[] bytes)

```

If a hexadecimal string is passed in, use

```
public static string receiveData(string strData)
```

Parse hexadecimal strings into JSON strings and output.

If byte [] is passed in, use

```
public static string receiveData(byte[] bytes)
```

Parse byte[] into JOSN strings and output

The unified format returned by the decoding method is:

```
public class Result
{
    /// <summary>
    /// Device ID
    /// </summary>
    public string DeviceID { get; set; }

    /// <summary>
    /// Message type
    /// </summary>
    public string MsgType { get; set; }

    /// <summary>
    /// Message serial number
    /// </summary>
    public int MsgFlowId { get; set; }
```

```

    /// <summary>

    /// Message body(Json)

    /// </summary>

    public object DataBody { get; set; }

    /// <summary>

    /// Need reply content

    /// </summary>

    public string ReplyMsg { get; set; }

}

```

Depending on the MsgType, the entity class corresponding to the DataBody will be different.

ReplyMsg is whether the platform needs to respond to this message. If data needs to be responded to, ReplyMsg is the corresponding response content (hexadecimal string). If no response is needed, ReplyMsg is a null value.

2.3.1.MsgType=0x0200

Then the DataBody is LocationData

```

public class LocationData

{

    /// <summary>

    /// Device packaging time,(UTC+0)

    /// </summary>

    public string GpsTime { get; set; }
}

```



```
/// <summary>

/// Latitude

/// </summary>

public double Latitude { get; set; }

/// <summary>

/// Longitude

/// </summary>

public double Longitude { get; set; }

/// <summary>

/// Location type(1:GNSS Located;0:Not Located)

/// </summary>

public int LocationType { get; set; }

/// <summary>

/// Speed(km/h)

/// </summary>

public int Speed { get; set; }

/// <summary>

/// Direction(0~360)

/// </summary>

public int Direction { get; set; }
```

```

/// <summary>

/// Mileage(km)

/// </summary>

public long Mileage { get; set; }

/// <summary>

/// Altitude(m)

/// </summary>

public int Altitude { get; set; }

/// <summary>

/// ACC

/// </summary>

public int ACC { get; set; }

/// <summary>

/// Gnss signal

/// </summary>

public int GpsSignal { get; set; }

/// <summary>

/// GSM signal

/// </summary>

public int GSMSignal { get; set; }

```

```

    /// <summary>
    /// Alarm value list
    /// </summary>

    public List<int> AlarmTypeList { get; set; }

    /// <summary>
    /// Expansion Information Map
    /// </summary>

    public Dictionary<string, Object> ExtraInfoMap { get; set; }

}

```

2.3.2.MsgType=0x0001

Then the DataBody is GeneralResponse

```

public class GeneralResponse
{
    /// <summary>
    /// Response serial number
    /// </summary>

    public int ResponseSequence { get; set; }

    /// <summary>
    /// Response message ID
    /// </summary>
}

```

```
public string ResponseID { get; set; }

/// <summary>
/// Response result
/// </summary>

public int Result { get; set; }

}
```

2.3.3.MsgType=0x0002

Then DataBody is null, 0x0002 is the heartbeat packet

2.3.4.MsgType=0x0100

Then the DataBody is DeviceRegistration

```
public class DeviceRegistration
{
    public int ProvinceId { get; set; }

    public int CityId { get; set; }

    public string ProducerId { get; set; }

    public string DeviceType { get; set; }

    public string DeviceID { get; set; }

    public int LicensePlateColor { get; set; }

    public string VehicleNo { get; set; }
}
```

```
}
```

2.3.5.MsgType=0x0102

Then the DataBody is String, and the content is the device authentication code

2.3.6.MsgType=0x0104

Then the DataBody is DeviceParameter

```
public class DeviceParameter
{
    /// <summary>
    /// Response serial number
    /// </summary>

    public int ReplyMsgFlowId { get; set; }

    /// <summary>
    /// Number of response parameters
    /// </summary>

    public int ItemCount { get; set; }

    /// <summary>
    /// Response parameter list
    /// </summary>

    public Dictionary<string,object> ItemMap { get; set; }
```

```
}
```

2.3.7.MsgType=0x0900

Then the DataBody is TransparentMessage

```
public class TransparentMessage
{
    /// <summary>
    /// Transparent message type
    /// </summary>
    public int MsgType;
    /// <summary>
    /// Transparent message content
    /// </summary>
    public string MsgContent;
}
```

2.4.Test

Testing tools

“Uplink decoding” is the decoding of data reported by devices.

"Downlink encoding" is the grouping of instruction data.

Data decoder tools (V2023.3.7)

Uplink decoding

Downlink encoding

device id

730057932364

message serial number

0001

Encoding

Set Device Parameters (0x8103)

Query Device Parameter (0x8104)

Device Control (0x8105)

Downlink Transparent Transmission (0x8900)

Platform RSA Public Key (0x8A00)

0x0001|DWORD|Device heartbeat interval

0x0010|STRING|Server APN

0x0011|STRING|APN username

0x0012|STRING|APN password

0x0013|STRING|Main server address

0x0017|STRING|Backup server address

0x0018|DWORD|Server port

0x0027|DWORD|Report interval in sleep mode

0x0029|DWORD|Report interval in run mode

0x0030|DWORD|Inflection point angle

0x0055|DWORD|The highest speed

0x0056|DWORD|The time of overspeed duration

0x0080|DWORD|The data of odometer

0xF000|STRING|Device ID

0xF001|WORD|The voltage in run mode

0xF002|WORD|Stop voltage

0xF003|WORD|The voltage in sleep mode

0xF004|STRING|NTP server address

0xF005|DWORD|NTP server port

0xF006|BYTE|Timezone

0xF007|BYTE|The type of protocol

0xF009|BYTE|The encryption of protocol

0xF00A|BYTE|Positioning Galaxy

0xF00B|BYTE|WIFI enable

0xF00C|BYTE|WIFI work mode*

0xF00D|BYTE|The max AP of WIFI

0xF00E|WORD|WiFi single scan time

0xF00F|BYTE|BT enable

0xF010|BYTE|BT work mode

0xF011|BYTE|Maximum number nodes of BT

0xF012|BYTE|The timeout of BT nodes

0xF013|WORD|BT single scan time

0xF014|BYTE|BT report mask

0xF015|WORD|Input GPIO mode

0xF016|BYTE|GPIO direction*

0xF017|BYTE|Transmit protocol

0xF018|DWORD|DWORD|Report mask

0xF019|BYTE|Gsensor enable

0xF01A|BYTE|Gsensor sensitivity

0xF01B|BYTE|Gsensor range

0xF01C|WORD|Gsensor times

0xF01D|DWORD|Gsensor time

0xF01E|DWORD|Gsensor trigger interval

0xF01F|BYTE|Gsensor report mask

0xF020|BYTE|Light enable

0xF021|WORD|Light hreshold

0xF022|DWORD|Light trigger interval

0xF023|BYTE|Temp&hum enable

0xF024|WORD|Upper temperature limit

0xF025|WORD|Lower temperature limit

0xF026|WORD|Upper humidity limit

0xF027|WORD|Lower humidity limit

Parameter ID

Parameter Type

Parameter Length

Unit

Device Type

Parameter Value

0x0010

STRING

/

/

GAV

CMIIOT

0x0011

STRING

/

/

GAV

TEST

0x0012

STRING

/

/

GAV

123456

0x0013

STRING

/

/

GAV

124.223.60.234

0x0018

DWORD

4

/

GAV

8608

Encoded message:

7E8103003C7300579323640001050000001005434D494F540000001104544553540000001206313233343536000000130F3132342E3232332E3036302E3233340000001804000019D0F37E

The above is the command package I used to set the primary IP for the device:

Server APN:CMIOTAPN username:TESTAPN password:123456Main server address:124.223.60.234Server port:6608

Encoded message below: will show the command content for packaging.

7E8103003C7300579323640001050000001005434D494F540000001104544553540000001206313233343536000000130F3132342E3232332E3036302E3233340000
001804000019D0F37E