Sino CASTEL
OBD Gateway API
HGAPI

Document Modification Records

Version NO.	Modification Date	Name	Contents
V0.1			Draft
V0.2			Modify part of the interface and function
V0.3	2012-01-13		1.Modify the name of alarm settings 2Modify the name of support the event structure of the data stream 3modify the structure of alarm to increase terminal threshold of the alarm 4modify the fault type of event structure 5modify the snapshot type of data structure , freeze frame snapshot data types 6modify the field structure of HTOBDInfo resolve 7Increase the obd Information to obtain HGAPI local cache / number of interface functions
V0.4	2012-02-01		1. Modify the position state description of HTGPSData 2. Increase the data flow code instructions
V0.5	2012-02-03		1. Increase examples of each used function 2. Delete HTSendCommand return value SMALL_SEQ value

Content

1 Introduction

1.1 The purpose of the document

Define the function of the CASTEL OBD Gateway APIs (HGAPI), connect detailed interface

1.2 Scope of application

Third - party platform develop, product and design.

1.3 Terms and Abbreviations

Name	Abbreviation	Explanation
OBD device	OBD	
OBD gateway	GW	Responsible for access to OBD -end gateway system
Castel gateway API	HGAPI	A set of interface functions for OBD terminal access
User		Use HGAPI system

1.4 Reference documentation

《HGAPI.h》
《HGDataDef.h》

2 HGAPI Outline

2.1 OBD Monitor schematic

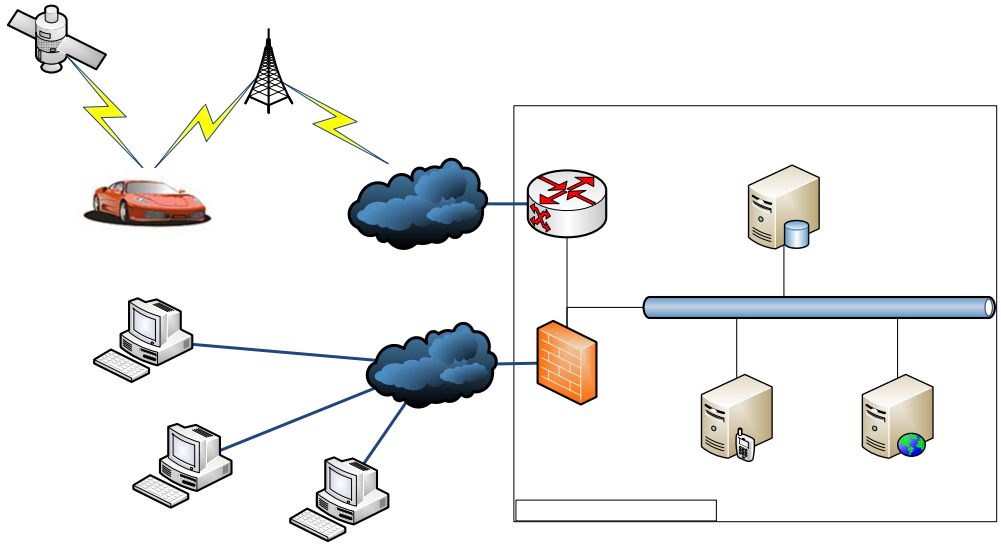


Chart 1 Monitor schematic

OBD monitor contains three parts: the terminal equipment, central platform and monitoring customer terminal

- terminal equipment is responsible for receiving satellite¬ positioning information and sent it to the central platform ;
- center platform is responsible for the information receiving ¬terminal device and resolution, and save the parsed data ,then push data to the monitor
- monitor client terminal is responsible for monitoring the OBD¬ equipment

2.2 HGAPI Position

HGAPI is located in the front end position of the platform, as shown below

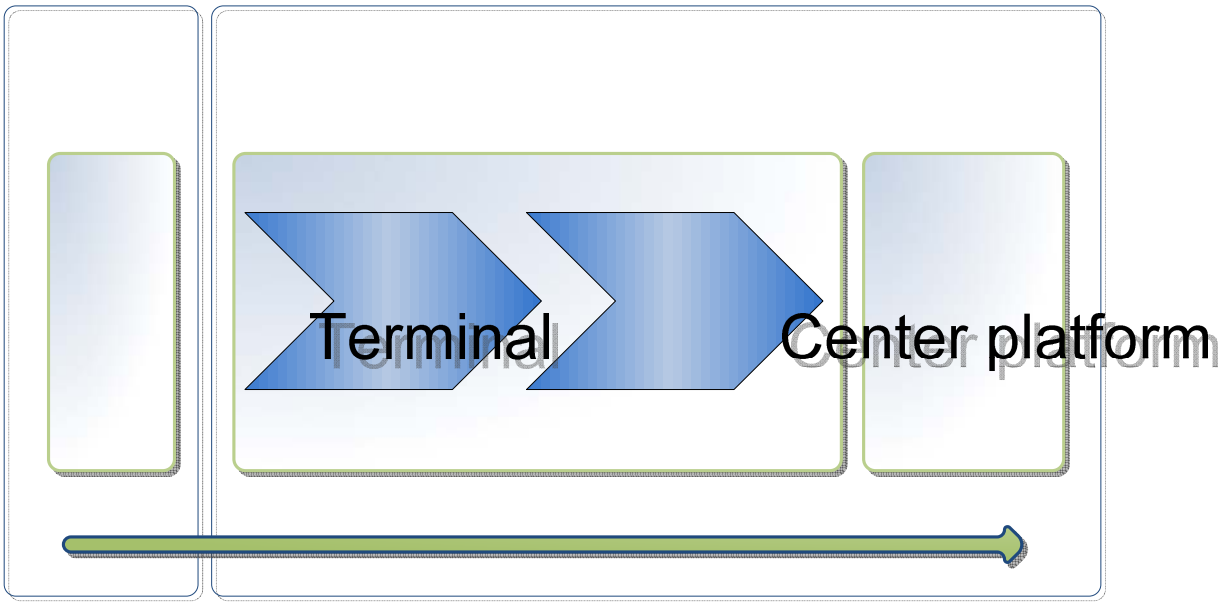


Chart 2 HGAPI position

2.3 **HGAPI Function**

- OBD access
- Receive OBD data，parse and the pass back to the user
- Send OBD instruction

3 HGAPI Instruction

HGAPI uses standard C language interface, use the rules unless there are special instructions, otherwise are __cdecl

Event callback function use __stdcall function
Support a third party platform developers in the form of dll file
The platform supports windows 32bit system currently

3.1 **Sequence Diagram**

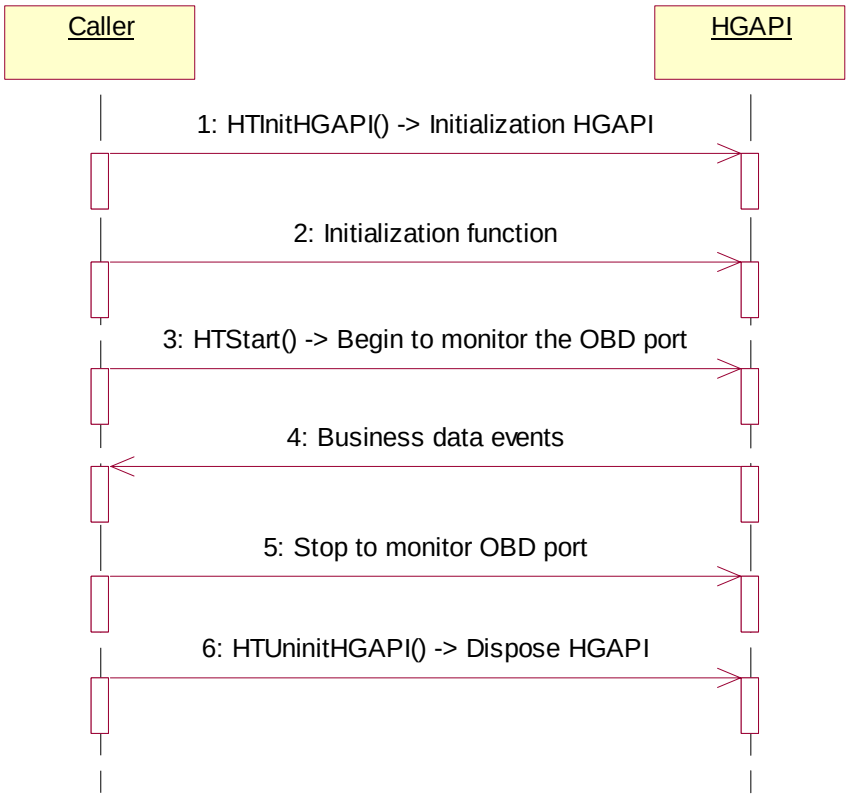


Chart 3 HGAPI sequence diagram

HTInitHGAPI function must be called before the all other HGAPI functions, if succeed, it will continue call other functions

- HTUninitHGAPI function must be called before procedures closed , otherwise the result is undefined ;
- Initialization function should be called according to the caller's need, such as increased delete the OBD data.
- After **HTStart** function is called successfully, the OBD will be possible to land on.
- After **HTStop** function is called successfully,the OBD online will be forced to close connection.
- OBD business data event is a callback function，and designed by parameters of the **HTInitHGAPI** function，This callback function starts some thread callback by HGAPI internal，and the number of threads is specified by the parameters in the HTInitHGAPI function , from1 to 8.
- OBD business data event will be triggered between HTStart and HTStop function，

  HTStop function returns and it is no longer triggering this event.

4 Interface Function

This chapter describes all HGAPI interface functions

HGAPI interface functions can be divided into the following categories

- Performance function
- Configuration management function
- OBD instruction function
- Callback function

API functions can be used in the basic functions of the API

, such as initiation, anti-initiation, start function, and end function.

Configuration management functions for use in the configuration HGAPI internal parameters, such as the basic information of the OBD

OBD instruction function sends instructions to OBD

The callback function uploads the data in the OBD, and calls it back to the caller

4.1 Performance Function

4.1.1 HTInitHGAPI

Function prototype	HTRet **HTInitHGAPI**(void);			
Function Description	Initiation HGAPI			
	Parameters type	Name	I/O	Instruction
Parameter	void			
Return value	HTRet		OUT	Return value： HT_SUCCESS HT_INITED HT_UNKNOW_ERROR
Note	Dll is the first called function , if unsuccessful, subsequent operations will fail This function is couple with HTUninitHGAPI before Dll release call HTUninitHGAPI function			
Code Example	HTRet ret = HTInitHGAPI(); if (ret == HT_SUCCESS)			

	{
	//success init api
	}
	else
	{
	//failure init api
	}

4.1.2 HTUninitHGAPI

Function prototype	HTRet **HTUninitHGAPI**(void);			
Function Description	Anti-initiation HGAPI，release API related information			
	Parameters type	Name	I/O	Instruction
Parameters	void			
Return value	HTRet		OUT	Return value： HT_SUCCESS HT_NOT_INITED HT_UNKNOW_ERROR
Note	Must call **HTUninitHGAPI** function before release DLL，Otherwise it will cause undefined error ➢ After use this function, all the API function call will fail unless call HTInitHGAPI again			
Code Example	HTRet ret = HTUninitHGAPI();			

4.1.3 HTStart

Function prototype	HTRet **HTStart**(HTUshort usListenPort, HTCallbackOBDEventFunc pFunc, HTUlong ulCallbackThreadCount)			
Function Description	Begin to accept the OBD connection			
	Parameters type	Name	I/O	Instruction

<table>
<tr><td rowspan="3">Parameters</td><td>HTUshort</td><td>usListenPort</td><td>IN</td><td>Monitor OBD TCP port</td></tr>
<tr><td>HTCallbackOBDEventFunc</td><td>pFunc</td><td>IN</td><td>OBD event callback function，not allowed NULL</td></tr>
<tr><td>HTUlong</td><td>ulCallbackThreadCount</td><td>IN</td><td>Callback number of threads , from 1 to 8</td></tr>
<tr><td>Return value</td><td>HTRet</td><td></td><td>OUT</td><td>Return value：
HT_SUCCESS
HT_NOT_INITED
HT_NULL_CALLBACK_FUNC
HT_TOO_MANY_THREADCOUNT
HT_OPENED
HT_UNKNOW_ERROR</td></tr>
<tr><td>Note</td><td colspan="4">➢ If succeed，OBD can land on from a pointed usListenPort
➢ If succeed, the event callback function will operate, HGAPI create ulCallbackThreadCount to callback the events.
➢ If the function returns successfully，however, there is another program open this port, API will try to open the port automatically. Shut down the port and it can be used normally. In this case, API will open automatically, and OBD can connect successfully.
➢ If it return HT_OPENED，that means HTStart function has been called.
➢ After open successfully, you need to close the HTStop function to stop monitor OBD port.</td></tr>
<tr><td>Code example</td><td colspan="4">// When call HTStart function; a callback function pointer should be brought.
HTRet ret = HTStart(12345, &HTCallbackOBDEvent, 2)；

// The implementation of the callback function, according to the judgment of the callback function pEventData type to cast complementally.

void __stdcall HTCallbackOBDEvent(const HTEventData* pEventData)
{
if (pEventData->ctType == HGCT_OBD_LOGIN) //log on
{
HTEventLogin* pLogin = (HTEventLogin*)pEventData;
//do something for login event
}
else if (pEventData->ctType == HGCT_OBD_LOGOUT) //quit
{
HTEventLogin* pLogout = (HTEventLogout*)pEventData;
//do something for logout event
}
else //
{
//.....
}
}</td></tr>
</table>

4.1.4 HTStop

<table>
<tr><td>Function prototype</td><td colspan="4">HTRet HTStop(HTUchar ucDiscardCallbackEvent);</td></tr>
<tr><td>Function Description</td><td colspan="4">Close the OBD monitor port</td></tr>
<tr><td></td><td>Parameters type</td><td>Name</td><td>I/O</td><td>Instruction</td></tr>
<tr><td>Parameters</td><td>HTUchar</td><td>ucDiscardAllCallbackEvent</td><td>IN</td><td>Whether to throw away all of the callback function event
If=1，All event callback function will no longer trigger , HGAPI throw away the event directly.
If=0，Queue function of event will called continually. Before the callback function event has not been completed ，HTStop function will not stop. If there is a lot of the queue event, it may take a long time, and The processing time upon to the caller.</td></tr>
<tr><td>Return value</td><td>HTRet</td><td></td><td>OUT</td><td>Return value：
HT_SUCCESS
HT_NOT_INITED
HT_UNKNOW_ERROR</td></tr>
<tr><td>Note</td><td colspan="4">➢ If succeed, the OBD cannot connect the landing.
➢ ucDiscard AllCallbackEvent=0, HTStopfunction will wait for the end of the all callback function and then return. Event callback function stop operation after the function returns.
➢ If return HT_LISTEN_ERROR，instruction system has another program to open the port. Shutdown the opened point and it will operate normally.
➢ If not call HTStop function, it will call HTUninitHGAPI directly，HTStop function will be called automatically in HTUninitHGAPI, default, HTStop function the parameters ucDiscardAllCallbackEvent = 1. HTStop function will normally be called in the main thread. If you need to wait for all the callback function call，you should pay great attention to the competition of resources such as the callback function of the main thread，Otherwise，it is easy to cause deadlock. Deadlock is usually due to the main thread calls HTStop function. The function needs to wait until the end of the event callback function call completely before return，Event callback function is very likely to visit the main thread of some resources, leads waiting for the main thread returns</td></tr>
<tr><td>Code example</td><td colspan="4">HTRet ret = HTStop(1);</td></tr>
</table>

4.2 Configuration Management Function

4.2.1 HTAddOBD

Function prototype	HTRet **HTAddOBD**(const HTObdInfo* pInfo, HTUlong ulOBDCount);			
Function Description	Add OBD information			
	Parameters type	Name	I/O	Instruction
Parameters	const HTObdInfo*	pInfo	IN	OBD information structure pointer , the number of structure is determined by the ulOBDCount If ulOBDCount=0 or=1，that pInfo number is 1 OBD information structure contains OBD number and OBD Fixed packet number The initial value of the fixed package number is 0
	HTUlong	ulOBDCount	IN	pInfo point to the number of content
Return value	HTRet		OUT	Return value： HT_SUCCESS HT_NOT_INITED HT_INVALID_PARAM HT_UNKNOW_ERROR
Note	If succeed， OBD information will be saved in HGAPI，HGAPI has independent of the configuration file，configuration file will save this OBD info，and the configuration file will load automatically when start it next time. If OBD number is more than 20 characters, the OBD number cannot be inserted into the configuration file If the OBD number is empty, the OBD number cannot be inserted into the configuration file If the OBD number information already exists in the configuration file can be regard as update the OBD information or return HT_SUCCESS If OBD NO. =~0，means=0xffffffff，there are two cases:1. OBD info file already exists，there is no need to modify the number of OBD, if OBD is not exists, the fix package number should be set as 0. Fixed package number is an important symbol of the OBD data¬ retransmission，If you are unsure whether the OBD exist in the configuration file，Recommended input parameters fixed package number is ~ 0.This will ensure that the OBD number set to 0，the original the OBD number of remain unchanged The terminal with the OBD number logs on the HGAPI is legitimate, otherwise HGAPI direct refusal, and does not inform the caller.			

	If parameter pInfo is NULL， return HT_INVALID_PARAM. If you need to remove the OBD information from the configuration file, **HTRemoveOBD** need to be called.

Code example	//add 10 obd
	HTObdInfo obds[10] = {0};
	for (int i = 0; i < 10; ++i)
	{
	char szBuf[20];
	sprintf(szBuf, "OBD%.7d", i);
	strcpy(obds[i].obdID, szBuf);
	obds[i].ulFixNumber = ~0;
	}
	
	HTRet ret = HTAddOBD(obds, 10000);
	//CHECK_RET
	
	//add 10 obd
	HTObdInfo obd = {0};
	strcpy(obd.obdID, "OBD1000000");
	obd.ulFixNumber = ~0;
	HTRet ret = HTAddOBD(obd, 1);
	//CHECK_RET

4.2.2 HTRemoveOBD

Function prototype	HTRet **HTRemoveOBD**(const HTChar* szOBDID);			
Function Description	Delete OBD information			
	Parameters type	Name	I/O	Instruction
parameters	const HTChar*	szOBDID	IN	OBD number
Return Value	HTRet		OUT	Return value： HT_SUCCESS HT_NOT_INITED HT_INVALID_PARAM HT_OBD_NOT_EXIST HT_UNKNOW_ERROR
Note	➢ If succeed, the OBD is removed from the configuration file and the OBD numbered terminal			

	connection is illegal ➢ If you send a command queue contains the instruction of this OBD will be removed from the queue and will not trigger HTEventCommandReply event ➢ If parameter szOBDID is NULL， return HT_INVALID_PARAM ➢ If OBD is not exists，return HT_OBD_NOT_EXIST ➢ If you need to increase the OBD information again, you need to call HTAddOBD
Code example	HTRet ret = HTRemoveOBD("OBD1000000"); //CHECK_RET

4.2.3 HTGetOBDCount

Function prototype	HTRet **HTGetOBDCount**(HTUlong* pulCount);			
Function Description	Get the number of OBD in the HGAPI			
	Parameters type	Name	I/O	Instruction
parameters	HTUlong*	pulCount	OUT	If successful, return to the OBD number
Return Value	HTRet		OUT	Return value： HT_SUCCESS HT_NOT_INITED HT_UNKNOW_ERROR
Note	➢ If successful , * pulCount save the number of OBD in the HGAPI cache ➢ Only return HT_SUCCESS , * pulCount is effective , otherwise * pulCount does not make any changes			

Code example	unsigned long ulOBDCount = 0;
	HTRet ret = HTGetOBDCount(&ulOBDCount);
	//CHECK_RET			

4.2.4 HTGetOBD

Function prototype	HTRet **HTGetOBD**(HTObdInfo* pInfo, HTUlong ulBufCount, HTUlong* pulOBDCount);
Function Descriptio	Get information of OBD from HGAPI

n				
	Parameters type	Note	I/O	Instruction
parameters	HTObdInfo*	pInfo	IN	OBD information on an array of pointers
	HTUlong	ulBufCount	IN	Specify pInfo pointer point to the size of the array
	HTUlong*	pulCount	OUT	If succeed, return to the OBD number
Return Value	HTRet		OUT	Return value： HT_SUCCESS HT_NOT_INITED HT_UNKNOW_ERROR
Note	➢ If succeed, *pulOBDCoun will save the return OBD number ➢ If the pInfo is not enough to put down the actual contained OBD data, *pulOBDCount==ulBufCount ➢ Only return HT_SUCCESS , * pulCount is effective , otherwise * pulCount will not make any changes			

Code example	unsigned long ulOBDCount = 0;
	HTRet ret = HTGetOBDCount(&ulOBDCount);
	//CHECK_RET
	HTObdInfo* pInfo = new HTObdInfo[ulOBDCount] ;
	unsigned long ulActuallyCount = 0;
	ret = HTGetOBD(pInfo, ulOBDCount, &ulActuallyCount);
	//CHECK_RET
	for (unsigned long ul = 0; ul < ulActuallyCount; ++ul)
	{
	//...loop
	}
	delete []pInfo;			

4.2.5 HTGetOnlineOBDCount

Function prototype	HTRet **HTGetOnlineOBDCount**(HTUlong* pulCount);			
Function Description	Get the number of the OBD online			
	Parameters type	Note	I/O	Instruction
parameters	HTUlong*	pulCount	OUT	If succeed，return the number of OBD online.
Return	HTRet		OUT	Return value：

Value				HT_SUCCESS HT_NOT_INITED HT_UNKNOW_ERROR
Note	➢ If succeed，*pulCount saves the number of OBD online now. ➢ If have not called **HTStart**，the return number is 0 ➢ Only return to HT_SUCCESS，should *pulCount be valid，otherwise *pulCount cannot do any changes.			

Code example	unsigned long ulOnlineOBDCount = 0;
	HTRet ret = HTGetOnlineOBDCount(&ulOnlineOBDCount);
	//CHECK_RET			

4.3 OBD Instruction Function

4.3.1 HTSendCommand

Function prototype	HTRet **HTSendCommand**(const HTProtocolData* pData, HTUlong* pulSeq);			
Function Description	Send OBD instruction			
	Parameters type	Name	I/O	Instruction
parameters	const HTProtocolData*	pData	IN	
	HTUlong*	pulSeq	OUT	If succeed, it will return a series number which is only identity. When HTEventCommandReply event calls back, it uses series number as an instruction to return.
Return Value	HTRet		OUT	Return value： HT_SUCCESS HT_NOT_INITED HT_INVALID_PARAM HT_OBD_NOT_EXIST HT_NOT_SUPPORT_TYPE HT_UNKNOW_ERROR
Note	➢ Call this function has nothing to do with OBD online. ➢ Return value HT_SUCCESS，merely indicates that the instruction has been send to the queue , but whether to send successful is unknown ➢ If the data contained in the data is wrong, it will return HT_INVALID_PARAM			

	➢ If the OBD online, send queue will send this OBD instructions initiative, and wait for return. HTEventCommandReply event has been triggered after return, and rtType=HTRT_SUCCESS or HTRT_FAILURE ➢ If OBD does not respond within 15 seconds, then HGAPI will repeatedly send the command until it return ➢ If the OBD upload information indicates a particular type of instruction has been modified, HTEventCommandReply event has been triggered first, in this event, rtType= HTRT_TERMINAL_MODIFY, and then trigger HTEventParamInfo event. ➢ If the parameter pData contains unrecognized data type, return HT_NOT_SUPPORT_TYPE. ➢ Once instruction is issued, it cannot be canceled, even if it is also in the send queue.

Code example	// Sent the instruction of set fuel parameters
	HTConfFuelParam ht = {0};
	strcpy(ht.obdID, "OBD1000000");
	ht.ptType = HGP_CONF_FUEL_PARAM;
	ht.ftType = HTFT_LPG;
	ht.ucEngineDisplacement = 16;
	HTUlong ulSeq = 0;
	HTRet ret = HTSendCommand((HTProtocolData*)&ht, &ulSeq);
	//CHECK_RET

4.3.2 HTRequireCurrentPos

Function prototype	HTRet **HTRequireCurrentPos**(const HTChar* szOBDID);			
Function Description	Request OBD current location			
	Parameters type	Name	I/O	Instruction
parameters	const HTChar*	szOBDID	IN	OBD number
Return Value	HTRet		OUT	Return value： HT_SUCCESS HT_NOT_INITED HT_INVALID_PARAM HT_OBD_NOT_EXIST HT_OBD_OFFLINE HT_UNKNOW_ERROR
Note	➢ If send successfully, return HT_SUCCESS ➢ Return HT_SUCCESS only show that it has been sent successfully, but cannot indicate the OBD will			

	return. If it returns current position, it returns HTEventCurrentPos through the event callback function. ➢ If parameter szOBDID is NULL， return HT_INVALID_PARAM ➢ If the OBD is not online, it cannot send this command, return HT_OBD_OFFLINE
Code example	HTRet ret = HTRequireCurrentPos("OBD1000000"); //CHECK_RET

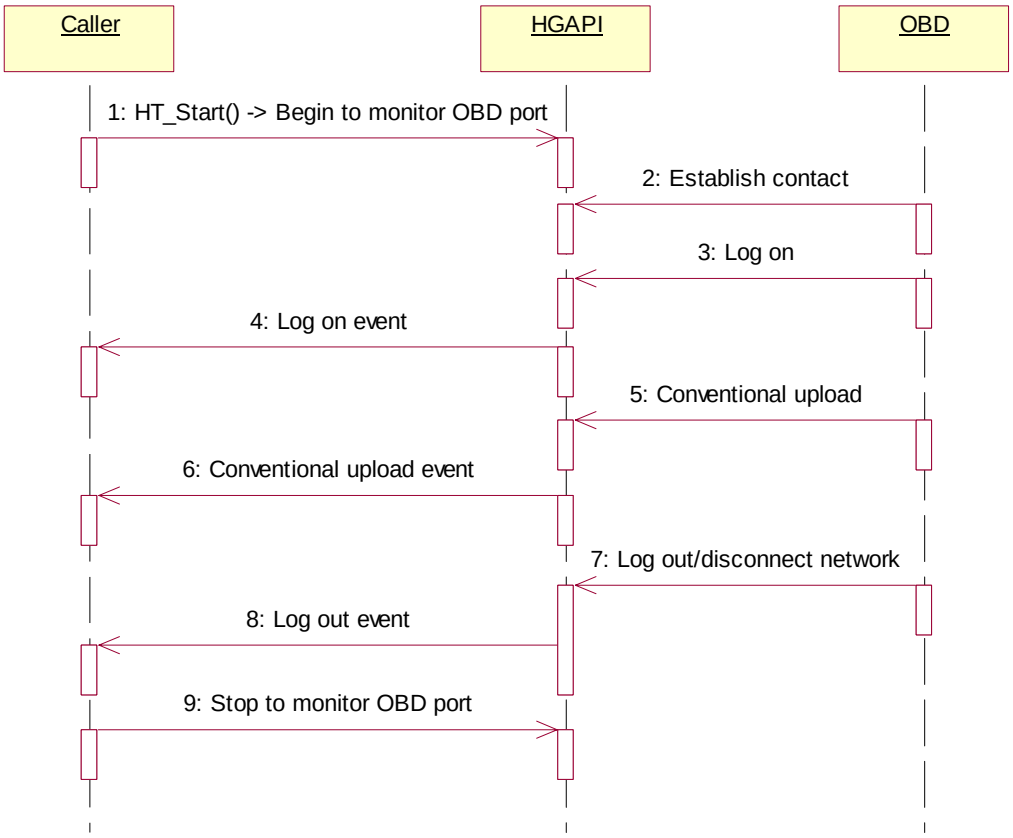
4.4 Callback Function

4.4.1 HTCallbackOBDEventFunc

Function prototype	typedef void (__stdcall *HTCallbackOBDEventFunc)(const HTEventData* pEventData);			
Function Description	event callback function			
	Parameters type	Name	I/O	Instruction
parameters	const HTEventData*	pEventData	IN	event callback function information
Return Value	void			
Note	➢ This function is provided by the caller and it specified in HTStart parameters The following diagram is a simple callback function, When HTStart has been called successfully, and the OBD can connect HGAPI normally and sends business data. Meanwhile, it will trigger different HTEventDat event according to different data uploaded by OBD. According to the HTEventData of HTCallbackType type, it cast into the callback data types and then deal with business. ➢ This callback function will create multiple specified threads callback at the same time the according to the HTStart , so it is necessary to pay attention to the multi-threaded processing ➢ In order to ensure the callback processing order , the same OBD number of events will in turn to call in the same thread			

Code example	void __stdcall HTCallbackOBDEvent(const HTEventData* pEventData)
	{
	if (pEventData->ctType == HGCT_OBD_LOGIN) //log on
	{
	HTEventLogin* pLogin = (HTEventLogin*)pEventData;
	//do something for login event
	}
	else if (pEventData->ctType == HGCT_OBD_LOGOUT) //quit			

```
    {
        HTEventLogin* pLogout = (HTEventLogout*)pEventData;
        //do something for logout event
    }
    else    //
    {
        //.....
    }
}
```



5 HGAPI Data type

Detail data types, structure definition, please refer to the header file HGDataDef.h

5.1 Basic data types

Type	Length(Byte)	Note
HTChar	1	
HTUchar	1	
HTShort	2	
HTUshort	2	
HTLong	4	
HTUlong	4	
HTInt	4	
HTUint	4	
HTFloat	4	
HTDouble	4	
HTTime	4	Since the early morning of January 1, 1970 , the second, UTC time, is the type of time_t in c language in fact, length 32bit

5.2 Interface data type

5.2.1 HTRet

Type Description
HGAPI performs return value function
typedef enum _HTRet { HT_SUCCESS = 0, //success

```
    HT_INITED,                      //initialization dll
    HT_NOT_INITED,                  //have not initialization dll
    HT_NULL_CALLBACK_FUNC,          //null callback function
    HT_TOO_MANY_THREADCOUNT,        //too many threadcount
    HT_LISTEN_ERROR,                //listen error
    HT_OPENED,                      //have been called
    HT_INVALID_PARAM,               //wrong parameter
    HT_OBD_NOT_EXIST,               //OBD is not exist
    HT_OBD_OFFLINE,                 //OBD is offline
    HT_NOT_SUPPORT_TYPE,            //not support data type
    HT_UNKNOW_ERROR = ~0,           //unknown error
} HTRet;
```

Enumeration values	Instruction
HT_SUCCESS	success
HT_INITED	initialization dll
HT_NOT_INITED	have not initialization dll
HT_NULL_CALLBACK_FUNC	null callback function
HT_TOO_MANY_THREADCOUNT	too many threadcount
HT_LISTEN_ERROR	listen error
HT_OPENED	have been called
HT_INVALID_PARAM	wrong parameter
HT_OBD_NOT_EXIST	OBD is not exist
HT_OBD_OFFLINE	OBD is offline
HT_NOT_SUPPORT_TYPE	not support data type
HT_UNKNOW_ERROR	unknown error

- The return value of the all interface functions are included in the list above
- If succeed，return HT_SUCCESS，otherwise, it returns other data.

5.3 **Business basis of enumerated type**

5.3.1 **HTProtocolType**

Type Description
When send commands to type, Call HGAPI function **HTSendCommand** , the type of call HGAPI the function HTSendCommand use to send commands

```
typedef enum _HTProtocolType
{
    HGP_CONF_ALARM = 1,             //alarm configuration
    HGP_CONF_NORMAL_UPLOAD,         //normal upload configuration
```

HGP_CONF_FUEL_PARAM, //fuel parameter configuration } HTProtocolType;

Enumeration values	Instruction
HGP_CONF_ALARM	alarm configuration
HGP_CONF_NORMAL_UPLOAD	normal upload configuration
HGP_CONF_FUEL_PARAM	fuel parameter configuration

➢ Above list contains all current HGAPI support instruction types

5.3.2 HTCallbackType

Type Description
HGAPI callback function HTCallbackOBDEventFunc parameter HTEventData callback data type.

typedef enum _HTCallbackType
{
HGCT_OBD_LOGIN = 1, //log in
HGCT_OBD_LOGOUT = 2, //log out
HGCT_OBD_TRIP_START = 3, //trip start
HGCT_OBD_TRIP_FINISH = 4, //trip finish
HGCT_OBD_SUPPORT_DATA = 5,//OBD support data type list HGCT_OBD_ALARM = 6, //beginning or end the alarm
HGCT_OBD_FAULT = 7, //fault
HGCT_OBD_SNAPSHOT = 8, //snapshot data
HGCT_OBD_NORMAL_UPLOAD = 9,//normal upload data
HGCT_OBD_CURRENT_POS = 10, //call command
HGCT_OBD_COMMAND_REPLY = 11,//reply command
HGCT_OBD_PARAM_UPOLOAD = 12,//upload parameters
} HTCallbackType;

Enumeration values	Instruction
HGCT_OBD_LOGIN	log in
HGCT_OBD_LOGOUT	log out
HGCT_OBD_TRIP_START	trip start
HGCT_OBD_TRIP_FINISH	trip finish
HGCT_OBD_SUPPORT_DATA	OBD support data type list
HGCT_OBD_ALARM	beginning or end the alarm
HGCT_OBD_FAULT	fault
HGCT_OBD_SNAPSHOT	snapshot data
HGCT_OBD_NORMAL_UPLOAD	normal upload data
HGCT_OBD_CURRENT_POS	call command

HGCT_OBD_COMMAND_REPLY	reply command
HGCT_OBD_PARAM_UPOLOAD	upload parameters
➢ support for the type of callback event	

5.3.3 HTAlarmType

Type Description
Alarm type is used in setting and eliminating the alarm type and upload

```
typedef enum _HTAlarmType
{
    HTAT_OVERSPEED = 1,          //0x01 overspeed
    HTAT_LOWVATT = 2,            //0x02 lowvater
    HTAT_WATER = 3,              //0x03 water temp alarm
    HTAT_HARD_ACC = 4,           //0x04 acceleration
    HTAT_HARD_BROKE = 5,         //0x05  abrupt deceleration
    HTAT_PARK_ACCON = 6,         //0x06  park without turn off the engine
    HTAT_DRAG = 7,               //0x07 drag
    HTAT_RPM_HIGH = 8,           //0x08 high speed
    HTAT_POWNON = 9,             //0x09 power on speed
    HTAT_TAILGAS = 10,           //0x0A excessive exhaust
    HTAT_HARD_LANE = 11,         //0x0B hard lane
    HTAT_HARD_TURN = 12,         //0x0C hard rurn
    HTAT_FATIGUE_DRIVE = 13,     //0x0D fatigue drive
} HTAlarmType;
```

Enumeration values	Instruction
HTAT_OVERSPEED	overspeed
HTAT_LOWVATT	lowvater
HTAT_WATER	water temp alarm
HTAT_HARD_ACC	acceleration
HTAT_HARD_BROKE	abrupt deceleration
HTAT_PARK_ACCON	park without turn off the engine
HTAT_DRAG	drag
HTAT_RPM_HIGH	high speed
HTAT_POWNON	power on speed
HTAT_TAILGAS	excessive exhaust
HTAT_HARD_LANE	hard lane
HTAT_HARD_TURN	hard rurn
HTAT_FATIGUE_DRIVE	fatigue drive
➢	

5.3.4 HTFuelType

Type Description	
Fuel type used in a set of fuel parameters function	
typedef enum _HTFuelType { HTFT_GAS = 0x10, HTFT_LPG = 0x20, HTFT_HYBIRD = 0x30, HTFT_DIESEL_A = 0x40, HTFT_DIESEL_B = 0x50, } HTFuelType;	
Enumeration values	Instruction
HTFT_GAS	
HTFT_LPG	
HTFT_HYBIRD	
HTFT_DIESEL_A	
HTFT_DIESEL_B	
➢	

5.3.5 HTCommandReplyType

Type Description	
Reply command type	
typedef enum _HTCommandReplyType { HTRT_SUCCESS, //success HTRT_FAILURE, //failure HTRT_TERMINAL_MODIFY, //terminal has been set,cancel the command } HTCommandReplyType;	
Enumeration values	Instruction
HTRT_SUCCESS	success
HTRT_FAILURE	failure
HTRT_TERMINAL_MODIFY	terminal has been set,cancel the command
➢	

5.4 Business basic data structures

5.4.1 HTObdID

Type Description
The OBD number is unique identified number of OBD terminal , the maximum length of a string of 20 bytes
typedef HTChar HTObdID[21];
➢

5.4.2 HTObdInfo

Type Description
OBD information, add obd information to the dll

//OBD info data
typedef struct _HTObdInfo
{
HTObdID obdID; // fixed data code,if not or does not to modify the configutation,fill in~0,namely xffffffff
} HTObdInfo;

➢

5.4.3 HTProtocolData

Type Description		
Used in send the OBD instruction function		
typedef struct _HTProtocolData { HTObdID obdID; HTProtocolType ptType; } HTProtocolData;		
Type	Name	Instruction
HTObdID	obdID	OBD number
HTProtocolType	ptType	Command type
➢ Used to the parameter HTProtocolData* of function **HTSendCommand.** ptType have different values according to different types of instruction. HGAPI converts to the type of configuration data according to different ptType.		

➢ More detailed information of configuration data type refer to the configuration data type chapter

5.4.4 HTEventData

Type Description		
Callback function parameter of HGAPI functions mainly for indentifying the data type of callback function.		
typedef struct _HTEventData { HTObdID obdID; HTCallbackType ctType; } HTEventData;		
Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ctType	Command type
➢ Used in parameter HTEventData* of the callback function HTCallbackOBDEventFunc. ptType have different values according to different time of callback. The user needs to convert the callback data type according to different ctType. ➢ More detailed information of callback data type refer to the data type chapter		

5.4.5 HTGPSData

Type Description		
GPS data structure contains a GPS data		

typedef struct _HTGPSData
{
HTTime tmTime; //GPS time
HTInt nX; //x * 360000
HTInt nY; //y * 360000
HTUshort usSpeed; //km/hour
HTUshort usDirect; //1/10degree
HTUchar ucState: 2; //Locate state=0，unlocated, =1，dloacte=3，dlocate
HTUchar ucStalCount: 4; //the number of satellites
} HTGPSData;		

Type	Name	Instruction
HTTime	tmTime	GPS time
HTInt	nX	Longitude, Unit 1/ 100 seconds East longitude is positive , west

		longitude is negative
HTInt	nY	Latitude, Unit 1/ 100 seconds Latitude is positive , south latitude is negative
HTUshort	usSpeed	Speed，units km/h
HTUshort	usDirect	Direction, 1 / 10 degrees , range from 0 to 3599 The north direction is 0 degrees , clockwise rotation increases
HTUchar	ucState	Locate the state. = 0 is not positioned > 0 , positioning
HTUchar	ucStalCount	The number of satellites， OBD terminal receives GPS satellite number
➢		

5.4.6 HTGSensor

Type Description		
The GSensor data structure contains a Gsensor data		

typedef struct _HTGSensor
{
HTTime tmTime; //time
HTChar x; //the x-direction acceleration
 	
HTChar y;
HTChar z;
} HTGSensor;		

Type	Name	Instruction
HTTime	tmTime	Time
HTChar	x	X - direction acceleration Units g A positive number indicates the positive direction of acceleration Negative number indicates the opposite direction of acceleration
HTChar	y	Y - direction acceleration Units g A positive number indicates the positive direction of acceleration

		Negative number indicates the opposite direction of acceleration
HTChar	z	Z - direction acceleration Units g A positive number indicates the positive direction of acceleration Negative number indicates the opposite direction of acceleration
➢		

5.4.7 HTConData

Type Description		
Condition data includes a set of condition data		

typedef struct _HTConData
{
HTTime tmTime; //time
HTFloat* pfVal; //working condition value list，number=ucConTypeCount
} HTConData;		

Type	Name	Instruction
HTTime	tmTime	Time
HTFloat*	pfVal	X - direction acceleration Units g A positive number indicates the positive direction of acceleration Negative number indicates the opposite direction of acceleration
➢		

5.4.8 HTOBDState

Type Description
OBD state data indicates the status of the current OBD

typedef struct _HTOBDState
{
HTUchar voltlow: 1; //low voltage
HTUchar dragalarm: 1; //drag alarm
HTUchar overspeed: 1; //overspeed alarm
HTUchar overwatertemp: 1; //over water temp alarm

```
    HTUchar hardacc: 1;              //acceleration alarm
    HTUchar hardbrake: 1;           // abrupt deceleration


    HTUchar parkenginerun: 1;      // park without turn off the engine
    HTUchar gasoil: 1;              // excessive exhaust alarm
    HTUchar rpmhigh: 1;              //speed
    HTUchar poweron: 1;              //power on
} HTOBDState;
```

Type	Name	Instruction
HTUchar	voltlow	low voltage
HTUchar	dragalarm	Drag alarm
HTUchar	overspeed	overspeed alarm
HTUchar	overwatertemp	over water temp alarm
HTUchar	hardacc	acceleration alarm
HTUchar	hardbrake	abrupt deceleration
HTUchar	parkenginerun	park without turn off the engine
HTUchar	gasoil	excessive exhaust alarm
HTUchar *	rpmhigh	speed
HTUchar	poweron	power on

➢ all values are = 1 means exist this state , = 0 indicates that no such state

5.4.9 HTAlarmItem

Type Description
OBD state data indicates the status of the current OBD

```
typedef struct _HTAlarmItem
{
    HTAlarmType atType;              //alarm type
    HTUchar ucEnabled;              //Whether or not open the alarm type，=0 close，=1 open
    HTUchar ucBeep;                  //Whether or not open the the sound of alarm，=0 close，=1 open
    HTFloat  fThreshold;            //alarm threshold,different type of alarm indicates threshold value range
} HTAlarmItem;
```

Type	Name	Instruction
HTAlarmType	atType	Alarm type
HTUchar	ucEnabled	Whether or not open the alarm type
HTUchar	ucBeep	Whether or not open the the sound of alarm
HTFloat	fThreshold	Alarm threshold,different type of

		alarm indicates threshold value range
➢ atType = HTAT_OVERSPEED, speed > fThreshold km/h(km/h)alarm ➢ atType = HTAT_LOWVATT,voltage < fThreshold v(V)alarm ➢ atType = HTAT_WATER,water temp > fThreshold 0C(°C) alarm ➢ atType = HTAT_HARD_ACC, acceleration > fThreshold g(m/s2) alarm ➢ atType = HTAT_HARD_BROKE, abrupt deceleration < fThreshold g(m/s2) alarm ➢ atType = HTAT_PARK_ACCON, park without turn off the engine > fThreshold min(min) alarm ➢ atType = HTAT_DRAG, ignore ➢ atType = HTAT_RPM_HIGH, speed > fThreshold rpm(rev/min) alarm ➢ atType = HTAT_POWNON, ignore		

5.4.10 HTDataInfo

Type Description		
Data flow information		

typedef struct _HTDataInfo
{
HTUshort usDataType; //data type
HTFloat fVal; //data value
} HTDataInfo;		

Type	Name	Instruction
HTUshort	usDataType	Data flow type refer to Appendix 6.1 , data flow code description table
HTFloat	fVal	Data type of the corresponding data values
➢ different types of data flow , fVal indicates different values		

5.5 Configuration data types

Configuration data type is used in **HTSendCommand** and HTEventParamInfo. Parameter type is HTProtocolData and need to be converted into the following type of configuration data in actual processing.

In **HTSendCommand**， as the content of set command, send to the terminal of OBD.

HTEventParamInfo， as the OBD upload OBD equipment parameters returns this data to know the local settings of the OBD equipment

5.5.1 HTConfAlarm

Type Description		
Alarm type Configuration		

typedef struct _HTConfAlarm
{
HTObdID obdID; //obd id
HTProtocolType ptType; //type instruction=HGP_CONF_ALARM
 	
HTUlong ulConfCount; //Alarm number of configuration，decide the number of pItem
HTAlarmItem arrayAlarmItem[0]; //decidef by ulConfCount
} HTConfAlarm;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTProtocolType	ptType	Instruction type ， fixed= HGP_CONF_ALARM
HTUlong	ulConfCount	Set the number of alarm The number of alarm>0
HTAlarmItem[]	arrayAlarmItem	Set alarm detail info refer to HTAlarmItem
➢ HTAlarmItem is an array，the length of array decided by ulConfCount， ➢ This data structure is used to set the OBD upload and cancel police alarm in any condition. ➢ This structure is used to **HTSendCommand** function parameter，convert HTConfAlarm*to HTProtocolData* directly.		

5.5.2 HTConfNormalUpload

Type Description
Fixed upload configuration parameters

typedef struct _HTConfNormalUpload
{
HTObdID obdID; //obd id
HTProtocolType ptType; // type instruction =HGP_CONF_NORMAL_UPLOAD
 	
HTUchar ucEnableGPS; //upload gps data switch, 0off, 1on
HTUchar ucEnableConnData; //upload work condition data switch, 0off , 1on
HTUchar ucEnableGSensor; //upload GSensor data switch, 0off, 1on
HTUchar ucUploadInterval; //upload time interval, multiply of 10~60
HTUchar ucConnInterval; //ucEnableConnData=0ignor,Working conditions of the acquisition time interval (seconds) , only 2,10

HTUchar ucConnCount; //ucEnableConnData=0ignor，ucConnInterval=0ignor, need to collect the number of the working condition type ， from 1 to 10.

HTUshort arrayConnType [0]; //ucEnableConnData=0 ignor，ucConnInterval=0 ignor, need to collect the number of the working condition type，the number is decided by ucConnCount

} HTConfNormalUpload;

Type	Name	Instruction
HTObdID	obdID	OBDnumber
HTProtocolType	ptType	Instruction type ， fixed= HGP_CONF_NORMAL_UPLOAD
HTUchar	ucEnableGPS	Upload GPS logo =0 not upload =1upload
HTUchar	ucEnableConnData	Whether or not to upload the condition data =0 not upload =1upload
HTUchar	ucEnableGSensor	Whether or not to upload Gsenso data =0 not upload =1upload
HTUchar	ucUploadInterval	Upload time interval, multiply of 10~60，must be in multiples of 10 If ucEnableConnData=0, this field is ignored
HTUchar	ucConnInterval	Acquisition conditions data time interval (seconds) are only to be 0，2，10 If ucEnableConnData=0, this field is ignored =0 That do not do any modifications to the original acquisition condition data ，using the original set =2 Condition data collected every two seconds =10 Condition data collected every ten seconds
HTUchar	ucConnCount	Collect the number of working situations If ucEnableConnData=0, this field is ignored
HTUshort[]	arrayConnType	Collect the array of the working condition type ， the number is decided by ucConnCount

		Condition data types, refer to Appendix 6.1 , data flow code sheets If ucEnableConnData=0, this field is ignored
➢ This structure is used to set the OBD fixed upload the data ➢ This structure is used to **HTSendCommand** function parameter , convert HTConfNormalUpload* into TProtocolData* directly.		

5.5.3 **HTConfFuelParam**

Type Description		
Fuel consumption parameters configuration		

typedef struct _HTConfFuelParam
{
HTObdID obdID; //obd id
HTProtocolType ptType; //construction type=HGP_CONF_FUEL_PARAM
HTUchar ucEngineDisplacement; //engine displacement, 1/10, eg,the displacement is.6L
HTFuelType ftType; //type of fuel
} HTConfFuelParam;		

Type	Name	Instruction
HTObdID	obdID	OBDnumber
HTProtocolType	ptType	Instruction type , fixed= HGP_CONF_FUEL_PARAM
HTUchar	ucEngineDisplacement	Engine displacement units 1/10L
HTFuelType	ftType	Type of fuel，detail info refer to HTFuelType
➢ This structure is used to set the parameters of fuel consumption, only the correct fuel consumption of the set parameters is possible to calculate the OBD driving fuel consumption correctly. ➢ This structure is used to **HTSendCommand**function parameter，convert HTConfFuelParam* into HTProtocolData* directly.		

5.6 **Callback data type**

5.6.1 **HTEventSystem**

Type Description

System error message indicates that the system has a serious error		

typedef struct _HTEventSystem
{
HTObdID obdID; //invalid
HTCallbackType ctType; //callback event type =HGCT_SYSTEM_ERROR
 	
HTRet ret; //error code
HTChar szErrorInfo[128]; //error info
} HTEventSystem;		

Type	Name	Instruction
HTObdID	obdID	The OBD number here is invalid
HTCallbackType	ctType	Event type , fixed= HGCT_SYSTEM_ERROR
HTRet	ret	Error code
HTChar[128]	szErrorInfo	Error info
➢ Fatal error occur in system, through this callback returns an error code and error messages ➢ fatal error may be a temporary impact system operation , it is possible to the normal functioning of the follow-up, e.g.: Port is occupied, not enough memory		

5.6.2 HTEventLogin

Type Description		
OBD landing indicates that the OBD online		

typedef struct _HTEventLogin
{
HTObdID obdID;
HTCallbackType ctType; //callback event type=HGCT_OBD_LOGIN
 	
HTUlong ulTripID; //trip ID
HTTime tmTime; //equipment time(GPS time)
HTUchar ucHaveGPS; //whether conclude GPS data
HTGPSData gpsData; //GPS data，ucHaveGPS=0ignor this field
HTOBDState state; //OBD state information
} HTEventLogin;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ctType	Event type , fixed= HGCT_OBD_LOGIN
HTUlong	ulTripID	Trip ID Every time the vehicle starts will generate a new trip ID , so each

		Trip ID indicates that a car has gone through ignition and Flameout.
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time, this time may not be accurate
HTUchar	ucHaveGPS	Whether or not contain GPS data =1，contain =0，not contain
HTGPSData	gpsData	Detail info about GPS data refer to HTGPSData
HTOBDState	state	Detail info about State OBD refer to HTOBDState
➢ After the OBD lands you can upload other business information. If OBD not logged in , all instructions cannot sent successfully ➢ After the OBD lands , if there is the original call HTSendCommand but not sent successfully, HGAPI will continue to send ➢ Corresponding OBD underline is determined by the HTEventLogout.		

5.6.3 HTEventLogout

Type Description		
OBD cancel indicates the OBD is offline		

typedef struct _HTEventLogout
{
HTObdID obdID;
HTCallbackType ctType; //callback event type=HGCT_OBD_LOGOUT
 	
HTUchar ucOBDLogout; //OBD equipment quit automatically，=1quit automatically，=0 net disconnected
HTTime tmTime; //equipment time (GPStime)
HTUchar ucHaveGPS; //Whether contain GPS data
HTGPSData gpsData; //GPS data，ucHaveGPS=0 ignore this field
HTOBDState state; //OBD state information
} HTEventLogout;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ctType	Event type ， fixed= HGCT_OBD_LOGOUT
HTUchar	ucOBDLogout	OBD equipment quit automatically

		=0 OBD is disconnected =1 OBD send quit package automatically.
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time, this time may not be accurate
HTUchar	ucHaveGPS	Whether or not contain GPS data =1，contain =0，not contain
HTGPSData	gpsData	Detail info about GPS data refer to HTGPSData
HTOBDState	state	Detail info about State OBD refer to HTOBDState
➢ Trigger this event indicates that the OBD had been canceled off the assembly line		

5.6.4 HTEventTripStart

Type Description		
Trip start event		

typedef struct _HTEventTripStart
{
HTObdID obdID;
HTCallbackType ctType; //callback event type=HGCT_OBD_TRIP_START
HTUlong ulTripID; //trip ID
HTUlong ulFixNumber; //fixed data package code
HTTime tmTime; // equipment time (GPStime)
HTUchar ucHaveGPS; // Whether contain GPS data
HTGPSData gpsData; //GPS data，ucHaveGPS=0 ignore this field
HTOBDState state; //OBD state information
} HTEventTripStart;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type ， fixed= HGCT_OBD_TRIP_START
HTUlong	ulTripID	Trip ID
HTUlong	ulFixNumber	Fixed data package code
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time, this time may not be accurate
HTUchar	ucHaveGPS	Whether or not contain GPS data

		=1，contain =0，not contain
HTGPSData	gpsData	Detail info about GPS data refer to HTGPSData
HTOBDState	state	Detail info about State OBD refer to HTOBDState
➢ When the vehicle ignition , OBD will increment to generate a new trip ID ➢ Unless there is a new trip to start the event trigger , otherwise this ID will not change in subsequent upload data		

5.6.5 HTEventTripFinish

Type Description		
The end of the trip event		

typedef struct _HTEventTripFinish
{
HTObdID obdID;
HTCallbackType ctType; // callback event type =HGCT_OBD_TRIP_FINISH
HTUlong ulTripID; // trip ID
HTUlong ulFixNumber; // fixed data package code
HTTime tmTime; // equipment time (GPStime)
HTUlong ulTripMileage; //trip milage
HTFloat fFuelWear; //Trip total fuel consumption L
HTUchar ucHaveGPS; // Whether or not contain GPS data
HTGPSData gpsData; // GPS data，ucHaveGPS=0 ignore this field
HTOBDState state; //OBD state information
} HTEventTripFinish;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type，fixed= HGCT_OBD_TRIP_FINISH
HTUlong	ulTripID	trip ID
HTUlong	ulFixNumber	Fixed data package code This number as an increment id , if the data is lost in the middle ,the OBD can complement transfer data according to local cached data
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time, this time may not be accurate

HTUlong	ulTripMileage	Trip mileage，units M
HTFloat	fFuelWear	Trip total fuel consumption units L
HTUchar	ucHaveGPS	Whether or not contain GPS data =1，contain =0，not contain
HTGPSData	gpsData	Detail info about GPS data refer to HTGPSData
HTOBDState	state	Detail info about State OBD refer to HTOBDState
➢ This structure is used to set the parameters of fuel consumption , only the correct fuel consumption of the set parameters be possible to correctly calculate the OBD driving fuel consumption		

5.6.6 HTEventSupportData

Type Description		
OBD support work condition data		

typedef struct _HTEventSupportData
{
HTObdID obdID;
HTCallbackType ctType; // callback event type =HGCT_OBD_SUPPORT_DATA
HTUlong ulTripID; // trip ID
HTUlong ulFixNumber; // fixed data package code
HTTime tmTime; // equipment time (GPStime)
HTUlong ulDataCount; //support the number of data flow
HTUshort arrayDataType[0]; //data flow type
} HTEventSupportData;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type , fixed= HGCT_OBD_SUPPORT_DATA
HTUlong	ulTripID	trip ID
HTUlong	ulFixNumber	Fixed data package code This number as an increment id , if the data is lost in the middle ,the OBD can complement transfer data according to local cached data
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time,

		this time may not be accurate
HTUlong	ulDataCount	Support the number of working type, indicates the number of usDataType
HTUshort[]	arrayDataType	Working type array, the number is indicated by ulDataCount. Working data type refers to appendix 6.1 data flow description list.
➢ upload OBD where the vehicles to support the type of working conditions		

5.6.7 HTEventSnapshot

Type Description		
Snapshot data		

typedef struct _HTEventSnapshot
{
HTObdID obdID;
HTCallbackType ctType; // callback event type =HGCT_OBD_SNAPSHOT
HTUlong ulTripID; // trip ID
HTUlong ulFixNumber; // fixed data package code
HTTime tmTime; // equipment time (GPStime)
HTUchar ucIsSnapshotData; //whether snapshot data or not，=1 yes=0 is Freeze frame data
HTUlong ulDataCount; //number of data，=0 indicates the vehile not support freeze frame data
HTDataInfo arrayData[0]; //data flow information，number=ulDataCount
} HTEventSnapshot;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type ， fixed= HGCT_OBD_SNAPSHOT
HTUlong	ulTripID	trip ID
HTUlong	ulFixNumber	Fixed data package code This number as an increment id , if the data is lost in the middle ,the OBD can complement transfer data according to local cached data
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time,

		this time may not be accurate
HTUchar	ucIsSnapshotData	Whether snapshot data or not， =1 yes =0 is Freeze frame data
HTUlong	ulDataCount	Information number Indicates the number of array Data
HTDataInfo[]	array Data	Information array

- return for a moment snapshot of data
- Snapshot data indicates the working data that return to that time supported by OBD.
- Freeze frame data indicates the working data that return the vehicle of the moment obd data, generally it upload when vehicle breakdown.

5.6.8 HTEventAlarm

Type Description
OBD alarm event，beginning or end

```
typedef struct _HTEventAlarm
{
    HTObdID obdID;
    HTCallbackType ctType;          // callback event type =HGCT_OBD_ALARM

    HTUlong ulTripID;               // trip ID
    HTUlong ulFixNumber;            // fixed data package code
    HTTime tmTime;                  // equipment time (GPStime)
    HTUlong ulTripMileage;          //trip mileage
    HTUchar ucHaveGPS;              //whether contain GPS data
    HTGPSData gpsData;              //GPS data，ucHaveGPS=0 ignore this field
    HTOBDState state;               //OBD state information
    HTUchar ucAlarmStart;           //=0end alarm，=1，begin alarm
    HTAlarmType atType;             //alarm type
    HTFloat fThreshold;             //threshold of terminal value is set during the alarm,indicates different value according to atType
    HTFloat fVal0;                  //alarm value, according to atType
    HTFloat fVal1;                  // alarm value, according to atType
} HTEventAlarm;
```

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type ， fixed= HGCT_OBD_ALARM
HTUlong	ulTripID	trip ID
HTUlong	ulFixNumber	Fixed data package code

		This number as an increment id , if the data is lost in the middle ,the OBD can complement transfer data according to local cached data
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time, this time may not be accurate
HTUlong	ulTripMileage	Trip mileage，units M
HTUchar	ucHaveGPS	Whether or not contain GPS data =1，contain =0，not contain
HTGPSData	gpsData	Detail info about GPS data refer to HTGPSData
HTOBDState	state	Detail info about State OBD refer to HTOBDState
HTUchar	ucAlarmStart	Alarm type =0 end alarm =1 begin alarm
HTAlarmType	atType	Alarm type
HTFloat	fThreshold	Threshold of terminal value is set during the alarm,indicates different value according to atType
HTFloat	fVal0	Alarm value 0
HTFloat	fVal1	Alarm value 1

- different alarm type , fThreshold , have different meanings , the specific value reference to the value set in the HTConfAlarm structure
- different alarm types , fVal0, fVal1 have different meanings
- atType= HTAT_OVERSPEED, fVal0:speed km/h(km/h) fVal1: ignore
- atType= HTAT_LOWVATT, fVal0: voltage (v) fVal1: ignore
- atType= HTAT_WATER, fVal0: water temp(℃) fVal1: ignore
- atType= HTAT_HARD_ACC, fVal0: when speed(km/h) fVal1: last second speed
- atType= HTAT_HARD_BROKE, fVal0: when speed(km/h)fVal1: last second speed
- atType= HTAT_PARK_ACCON, fVal0: parking (min) fVal1: ignore
- atType= HTAT_DRAG, fVal0: ignore fVal1: ignore
- atType= HTAT_RPM_HIGH, fVal0: rotate speed (rotate/min) fVal1: ignore
- atType= HTAT_POWNON, fVal0: ignore:fVal1: ignore

5.6.9 HTEventFault

Type Description
Fault event

typedef struct _HTEventFault
{
HTObdID obdID;
HTCallbackType ctType; // callback event type =HGCT_OBD_FAULT
HTUlong ulTripID; // trip ID
HTUlong ulFixNumber; // fixed data package code
HTTime tmTime; // equipment time (GPStime)
HTUchar ucPendingFault; //not sure fault mark=0，sure，=1not sure
HTUlong ulFaultCount; //number of fault，=0 no fault
HTUshort arrayFaultDataType[0];//fault type list，decided by ulFaultCount
} HTEventFault;

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type , fixed= HGCT_OBD_FAULT
HTUlong	ulTripID	Trip ID
HTUlong	ulFixNumber	Fixed data package code This number as an increment id , if the data is lost in the middle ,the OBD can complement transfer data according to local cached data
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time, this time may not be accurate
HTUchar	ucPendingFault	Not sure fault mark =0，sure， =1not sure
HTUlong	ulFaultCount	number of fault
HTUshort	arrayFaultDataType	Fault type array, the number is decided by ulFaultCount The type ,refer to appendix 6.1 data flow code list
➢ It triggered when OBD monitor vehicle breaks down		

5.6.10 **HTEventNormalData**

Type Description
Upload the data event conventionally

```
typedef struct _HTEventNormalData
{
    HTObdID obdID;
    HTCallbackType ctType;      // callback event type =HGCT_OBD_NORMAL_UPLOAD

    HTUlong ulTripID;                   // trip ID
    HTUlong ulFixNumber;                // fixed data package code
    HTTime tmTime;                      // equipment time (GPStime)
    HTUlong ulTripMileage;              // trip mileage
    HTOBDState state;                   //OBD state info
    HTUchar ucConTypeCount;             // working condition type number
    HTUchar ucGPSCount;                 //GPS data number
    HTUchar ucConCount;                 //working number number
    HTUchar ucGSensorCount;             //GSensor data number

    HTUshort*  pConType;        //working condition type,number=ucConTypeCount, =0， point invaild
    HTGPSData* pGPSData;                //GPS data,number=ucGPSCount, =0，point invaild
    HTGSensor*  pGSensorData;           //GSensor data,number=ucGSensorCount, =0， point invaild
    HTConData* pConData;                //working condition data array,number=ucConCount, =0，point invaild
} HTEventNormalData;
```

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type , fixed= HGCT_OBD_NORMAL_UPLOAD
HTUlong	ulTripID	Trip ID
HTUlong	ulFixNumber	fixed data package code This number as an increment id , if the data is lost in the middle ,the OBD can complement transfer data according to local cached data
HTTime	tmTime	Equipment time Before not positioned or not positioned with the server time, this time may not be accurate
HTUlong	ulTripMileage	Trip mileage,units M
HTOBDState	state	OBD state，detail information refer to HTOBDState
HTUchar	ucConTypeCount	Working condition type number, point the number of pConType array. if=0，no working condition data，

		ignore ucConCount and pConData
HTUchar	ucGPSCount	Number of GPS data
HTUchar	ucConCount	number of working condition data
HTUchar	ucGSensorCount	Number of Gsensor data
HTUshort*	pConType	working array type
HTGPSData*	pGPSData	GPS data array ucGPSCount=0，ignore
HTGSensor*	pGSensorData	Gsensor array ucGSensorCount =0，ignore
HTConData*	pConData	Working data array ucConCount=0，ignore
➢ Upload events is triggered by the rules of the parameters set by the HTConfNormalUpload conventionally.		

The relationship among of ucConTypeCount, pConType, ucConCount and pConData, There are some examples to instruction below

If ucConTypeCount=3, pConType point the array that contains 3working condition type，
E.g.: pConType [0] =0x2105,
Phonotype [1] =0x2106,
PConType [1] =0x210C,

If ucConCount=5, pConData point the array HTConData contains 5groupworking condition
PConData [0].tmTime = xxxx, →this array working condition time
pConData[0].pfVal[0] = 100, →working condition type is pConType[0],value is100，
pConData[0].pfVal[1] = 200, → working condition type is pConType[1],value is100，
pConData[0].pfVal[2] = 300, → working condition type is pConType[2],value is100，，

UcConTypeCount point the columns of below table
UcConCount point the rows of below table

	time	1st working condition type pConType[0]	2nd working condition type pConType[1]	3rd working condition type pConType[2]
1st working condition	ConData[0].tmTime	pConData[0].pfVal[0]	pConData[0].pfVal[1]	pConData[0].pfVal[2]
1st working condition	ConData[1].tmTime	pConData[1].pfVal[0]	pConData[1].pfVal[1]	pConData[1].pfVal[2]
pConData[2]	ConData[2].tmTime	pConData[2].pfVal[0]	pConData[2].pfVal[1]	pConData[2].pfVal[2]
pConData[3]	ConData[3].tmTime	pConData[3].pfVal[0]	pConData[3].pfVal[1]	pConData[3].pfVal[2]
pConData[4]	ConData[4].tmTime	pConData[4].pfVal[0]	pConData[4].pfVal[1]	pConData[4].pfVal[2]

5.6.11 HTEventCurrentPos

Type Description		
return current position event		

typedef struct _HTEventCurrentPos
{
HTObdID obdID;
HTCallbackType ctType; //callback event typ=HGCT_OBD_CURRENT_POS
 	
HTUchar ucHaveGPS; // whether contain GPS data
HTGPSData gpsData; //GPS data，ucHaveGPS=0 ignore this field
HTOBDState state; //OBD state info
} HTEventCurrentPos;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ptType	Event type , fixed= HGCT_OBD_CURRENT_POS
HTUchar	ucHaveGPS	Whether contain GPS data =1，contain =0，not contain
HTGPSData	gpsData	GPS data, detail info refer to HTGPSData
HTOBDState	state	OBD state，detail info refer to HTOBDState
➢ **HTRequireCurrentPos** requests for the current position, OBD will trigger the event when it receives the request and returns back		

5.6.12 HTEventCommandReply

Type Description		
Reply command		

typedef struct _HTEventCommandReply
{
HTObdID obdID;
HTCallbackType ctType; //callback event type=HGCT_OBD_COMMAND_REPLY
 	
HTUlong ulSeq; //command ID
HTProtocolType ptType; //command type
HTCommandReplyType rtType; //command send result
} HTEventCommandReply;		

Type	Name	Instruction

HTObdID	obdID	OBD number
HTCallbackType	ctType	Event type , fixed= HGCT_OBD_COMMAND_REPLY
HTProtocolType	ptType	Command type
HTUlong	ulSeq	Command ID
HTCommandReplyType	rtType	Command send result
➢ After use **HTSendCommand** successfully, the OBD is triggered because of timeout		

5.6.13 HTEventParamInfo

Type Description		
OBD parameter upload event		

typedef struct _HTEventParamInfo
{
HTObdID obdID;
HTCallbackType ctType; //callback event type=HGCT_OBD_PARAM_UPOLOAD
 	
HTProtocolData* pParam; //specific type is determined by the upload data
} HTEventParamInfo;		

Type	Name	Instruction
HTObdID	obdID	OBD number
HTCallbackType	ctType	Event type , fixed= HGCT_OBD_PARAM_UPOLOAD
HTProtocolData*	pParam	Parameter data
➢ OBD uploads local set parameter data automatically		

6 Appendix

6.1 Data flow code list

Data flow code list is mainly used in event function such as snapshot data, freeze frame, and support for data flow. Set OBD fixed upload data also use this data flow code.

Data flow code (hex)	Data flow code (decimals)	Data flow name	Abbreviation（Chinese）	Complete name（Chinese）
0x2101	8449	DTC_CNT	故障码存储量	对应所存储的冻结桢的故障码
0x2102	8450	DTCFRZF	一个故障码	一个故障码
0x2103	8451	FUELSYS1	燃油系统状态 1	燃油系统状态 1
0x2104	8452	LOAD_PCT (%)	计算负荷	计算负荷
0x2105	8453	ETC(℃)	水温	发动机冷却液温度
0x2106	8454	SHRTFT1 (%)	短时燃油修正 B1	短时燃油修正(气缸列 1 和 3)
0x2107	8455	LONGFT1 (%)	长期燃油修正 B1	长期燃油修正(气缸列 1 和 3)
0x2108	8456	SHRTFT2 (%)	短时燃油修正 B2	短时燃油修正(气缸列 2 和 4)
0x2109	8457	LONGFT2 (%)	长期燃油修正 B2	长期燃油修正(气缸列 2 和 4)
0x210a	8458	FRP(kPa)	燃油压力	燃油压力计量
0x210b	8459	MAP(kPa)	进气歧管压力	进气歧管绝对压力
0x210c	8460	RPM(/min)	转速	发动机转速
0x210d	8461	VSS(km/h)	车速	车速
0x210e	8462	SPARKADV(°)	点火正时	第一缸点火正时提前角(不包括机械提前)
0x210f	8463	IAT(℃)	进气温度	进气温度
0x2110	8464	MAF(g/s)	空气流量	空气流量传感器的空气流量
0x2111	8465	TP (%)	绝对节气门位置	绝对节气门位置
0x2112	8466	AIR_STAT	二次空气状态指令	二次空气状态指令
0x2113	8467	O2B1S1(V)	氧传感器电压 B1S1	氧传感器电压
0x2114	8468	02B1S1 (V) SHRTFTB1S1	输出电压（B1S1）短	传统 0 到 1V 氧传感器输出电压(Bx-Sy)及与此传感器关联

		(%)	时燃油修正（B1S1）	的短时燃油修正
0x2115	8469	O2B1S2 (V) SHRTFTB1S2 (%)	输出电压（B1S2）短时燃油修正（B1S2）	传统 0 到 1V 氧传感器输出电压(Bx-Sy)及与此传感器关联的短时燃油修正
0x2116	8470	O2B2S1(V)	氧传感器电压 B2S1	氧传感器电压 B2S1
0x2117	8471	O2B1S2(V)	氧传感器电压 B1S2	氧传感器电压 B1S2
0x2118	8472	O2B3S1(V)	氧传感器电压 B3S1	氧传感器电压 B3S1
0x2119	8473	O2B3S2(V)	氧传感器电压 B3S2	氧传感器电压 B3S2
0x211a	8474	O2B4S1(V)	氧传感器电压 B4S1	氧传感器电压 B4S1
0x211b	8475	O2B4S2(V)	氧传感器电压 B4S2	氧传感器电压 B4S2
0x211c	8476	OBD	OBD 的车辆设计要求	OBD 系统的车辆设计要求
0x211d	8477	O2S	氧传感器的位置	氧传感器的位置
0x211f	8479	RUNTM(sec)	自发动机起动的时间	自发动机起动的时间
0x2121	8481	MIL_DIST(KM)	MIL 激活后的行驶里程	在 MIL 激活状态下行驶的里程
0x2122	8482	FRP(kPa)	油轨压力（歧管真空度）	相对于歧管真空度的油轨压力
0x2123	8483	FRP(kPa)	油轨压力（大气压）	相对于大气压力的油轨压力
0x2124	8484	EQ_RATB1S1	线性氧传感器等效比和电压(B1S1)	线性或宽带式氧传感器的等效比(lambda)和电压
0x2125	8485	EQ_RATB1S2	线性氧传感器等效比和电压(B1S2)	线性或宽带式氧传感器的等效比(lambda)和电压
0x2126	8486	EQ_RATB2S1	线性氧传感器等效比和电压(B2S1)	线性或宽带式氧传感器的等效比(lambda)和电压
0x2127	8487	EQ_RATB2S2	线性氧传感器等效比和电压(B2S2)	线性或宽带式氧传感器的等效比(lambda)和电压
0x2128	8488	EQ_RATB3S1	线性氧传感	线性或宽带式氧传感器的等效

			器等效比和电压(B3S1)	比(lambda)和电压
0x2129	8489	EQ_RATB3S2	线性氧传感器等效比和电压(B3S2)	线性或宽带式氧传感器的等效比(lambda)和电压
0x212A	8490	EQ_RATB4S1	线性氧传感器等效比和电压(B4S1)	线性或宽带式氧传感器的等效比(lambda)和电压
0x212B	8491	EQ_RATB4S2	线性氧传感器等效比和电压(B4S2)	线性或宽带式氧传感器的等效比(lambda)和电压
0x212C	8492	EGR_PTC (%)	EGR 指令开度	EGR 指令开度
0x212d	8493	EGR_ERR (%)	EGR 开度误差率	EGR 开度误差 (实际开度 — 指令开度)/指令开度*100%
0x212e	8494	EVAP_PCT (%)	蒸发冲洗控制指令	蒸发冲洗控制指令
0x212f	8495	FLI (%)	燃油液位输入	燃油液位输入
0x2130	8496	EVAP_VP(inH2O)	故障码清除后的暖机循环数	自故障码被清除之后经历的暖机循环个数
0x2131	8497	CLR_DIST(KM)	故障码清除后的行驶里程	自故障码被清除之后的行驶里程
0x2132	8498	EVAP-VP(PA)	蒸发系统的蒸汽压力	蒸发系统的蒸汽压力
0x2133	8499	BARO{KPA}	大气压	大气压
0x2134	8500	EQ_RAT11	线性氧传感器等效比和电流(B1S1)	线性或宽带式氧传感器的等效比(lambda)和电流
0x2135	8501	EQ_RAT12	线性氧传感器等效比和电流（B1S2）	线性或宽带式氧传感器的等效比(lambda)和电流
0x2136	8502	EQ_RAT21	线性氧传感器等效比和电流（B2S1）	线性或宽带式氧传感器的等效比(lambda)和电流
0x2137	8503	EQ_RAT22	线性氧传感器等效比和电流（B2S2）	线性或宽带式氧传感器的等效比(lambda)和电流
0x2138	8504	EQ_RAT31	线性氧传感器等效比和电流（B3S1）	线性或宽带式氧传感器的等效比(lambda)和电流
0x2139	8505	EQ_RAT32	线性氧传感	线性或宽带式氧传感器的等效

			器等效比和电流（B3S2）	比(lambda)和电流
0x213a	8506	EQ_RAT41	线性氧传感器等效比和电流（B4S1）	线性或宽带式氧传感器的等效比(lambda)和电流
0x213b	8507	EQ_RAT42	线性氧传感器等效比和电流（B4S2）	线性或宽带式氧传感器的等效比(lambda)和电流
0x213c	8508	CATEMP11(℃)	催化器温度 B1S1	催化器温度 B1S1
0x213d	8509	CATEMP21(℃)	催化器温度 B2S1	催化器温度 B2S1
0x213e	8510	CATEMP12(℃)	催化器温度 B1S2	催化器温度 B1S2
0x213f	8511	CATEMP22(℃)	催化器温度 B2S2	催化器温度 B2S2
0x2142	8514	VPWR(V)	控制模块电压	控制模块电压
0x2143	8515	LOAD_ABS (%)	绝对负荷	绝对负荷值
0x2144	8516	EQ_RAT	等效比指令	等效比指令
0x2145	8517	TP_R (%)	相对节气门位置	相对节气门位置
0x2146	8518	AAT(℃)	环境空气温度	环境空气温度
0x2147	8519	TP_B (%)	绝对节气门位置 B	绝对节气门位置 B
0x2148	8520	TP_C (%)	绝对节气门位置 C	绝对节气门位置 C
0x2149	8521	APP_D (%)	加速踏板位置 D	加速踏板位置 D
0x214a	8522	APP_E (%)	加速踏板位置 E	加速踏板位置 E
0x214b	8523	APP_F (%)	加速踏板位置 F	加速踏板位置 F
0x214c	8524	TAC_PCT (%)	节气门执行器控制指令	节气门执行器控制指令
0x214d	8525	MIL_TIME	MIL 激活后运转时间	MIL 处于激活状态下的发动机运转时间
0x214e	8526	CLR_TIME	故障码清除之后的时间	自故障码清除之后的时间
0x214F	8527	FUELSYS2	燃油系统状态 2	燃油系统状态 2
0x2150	8528	SHRTFTB1S1 (%)	短期燃油修正（B1S1）	短期燃油修正（B1S1）

0x2151	8529	SHRTFTB2S1 (%)	短期燃油修正（B2S1）	短期燃油修正（B2S1）
0x2152	8530	O2B1S1(V)	氧传感器电压 B1S1	氧传感器输出电压 B1S1
0x2153	8531	O2B1S2(V)	氧传感器电压 B1S2	氧传感器输出电压 B1S2
0x2154	8532	O2B2S1(V)	氧传感器电压 B2S1	氧传感器输出电压 B2S1
0x2155	8533	O2B2S2(V)	氧传感器电压 B2S2	氧传感器输出电压 B2S2
0x2156	8534	O2S11(mA)	氧传感器电流 B1S1	线性或宽带式氧传感器电流 B1S1
0x2157	8535	O2S12(mA)	氧传感器电流 B1S2	线性或宽带式氧传感器电流 B1S2
0x2158	8536	O2S21(mA)	氧传感器电流 B2S1	线性或宽带式氧传感器电流 B2S1
0x2159	8537	O2S22(mA)	氧传感器电流 B2S2	线性或宽带式氧传感器电流 B2S2