UHFPrimeREADER.DLL
Dynamic Link Library User Guide
V1.0

Contents

1.Introduction

The SDK support C, C++ and other languages that can call C library interfaces, es delphi,C#,VB6.0,VB.NET etc.
At present,the SDK only can support Windwos Operation system (32 bits & 64 bits)
Upper computer application operate Prime Reader serials ISO18000-6C format reader through UHFPrimeReader.DLL, Support upper computer application open multiples reader under multiple communication port

2.Application Integration

The SDK include the file as below：

File name	Application
UHFPrimeReader.dll	DLL， Include app API port

Language	Description
C#	UHF_RFID_API.cs copy to project contents, and join project,copy UHFPrimeReader.DLL to exe file output contents

3.API

3.1 Connect/Close Reader

3.1.1 OpenDevice

Definition	`Int OpenDevice(HANDLE *hComm,char *ComPort, byte Baudrate);`		
Description	Open serial port connected to reader ,default baud rate is 115200,8 data bits,4 stop bit,no check		
Parameter	**Name**	**Type**	**Remarks**
	`hComm`	`HANDLE *`	Return the handle connected to reader, all API operation need use this handle
	`strComPort`	`char *`	Input reader address, if don't know real address, can input broadcast address 0xFF, After call library success, this parameter will return real address
	`Baudrate`	`Byte`	Set communication baud rate 0x00：9600； 0x01：19200；0x02：38400；0x03：57600；0x04：115200（Default）；
Return (int)	Success：0；Failure：NOT 0；（Check return value error code sheet）		

Parameter Code	`string[] ports = SerialPort.GetPortNames();`
	`HANDLE m_handler = HANDLE.Zero;`
	int state = `OpenDevice(out m_handler, port);`		

3.1.1 OpenNetConnection

Definition	Int OpenNetConnection(HANDLE *hComm, char *ip, WORD port,DWORD timeoutMs)		
Description	Open network port connected to reader, the default IP address is 192.168.1.200, defaultport is 2022. use tcpclient way connect reader		
Parameter	**Nmae**	**Type**	**Remarks**
	hComm	HANDLE *	Return the handle connected to reader, all API operation need use this handle
	ip	char *	Input reader IP address, the reader default IP address is 192.168.1.200
	port	WORD	Input reader default port，0~65535；
	timeoutMs	DWORD	The connection timeout period set，unit :ms。
Return (int)	Success：0；Failure：NOT 0；（Check return value error code sheet）		
Parameter	NO		

3.1.2 CloseDevice

Definition	int CloseDevice(HANDLE hComm);		
Description	Close serial port or network port connected to reader		
Parameter	**Nmae**	**Type**	**Remarks**
	hComm	HANDLE	A handle connected with reader, when open serial port returned handle
Return (int)	Success：0；Failure：NO 0；（Check return value error code sheet）		
Parameter Code	`int state = UHF_RFID_API. CloseDevice (m_handler);`		

3.1.3 OpenHidConnection

Definition	Int OpenHidConnection (HANDLE *hComm, WORD index)		
Illustration	Open the device corresponding to the HID device serial number and return the handle.		
Parameter	**Name**	**Type**	**Remark**
	hComm	HANDLE *	Returns the handle to the connection to the reader/writer, which is required for all API operations thereafter
	index	WORD	Equipment serial number
Return (int)	Success：0；Fail：Not 0；（View the return value error code table）		
Reference Code	None		

3.1.4 CFHid_GetUsbCount

Definition	`Int CFHid_GetUsbCount(void);`		
Illustration	Obtain the HID device with a vid of 0x0483 and a pid of 0x5750, and return the number of HID devices.		
Parameter	**Name**	**Type**	**Remark**
	-	-	-
Return (int)	Success： Not 0， This value is the number of HID devices； Fail： 0；		
Reference Code	None		

3.1.5 `CFHid_GetUsbInfo`

Definition	`Int CFHid_GetUsbInfo(WORD index, char *  pucDeviceInfo);`		
Illustration	Obtain the address of the specified device. The device address suffix is kbd, which is the keyboard. Unable to connect.		
Parameter	**Name**	**Type**	**Remark**
	`index`	`WORD`	Specify the equipment serial number, which is determined by CFHide_ GetUsbCount Get;
	`pucDevice Info`	`char *`	Device address.
Return (int)	Success: 0 ; Fail: Not 0 ; (View the return value error code table)		
Reference Code	None		

3.1.6 CloseDevice【Close card reader connection】

definition	int `CloseDevice(HANDLE hComm);`		
illustrate	Close the serial port or network port connected to the reader.		
parameter	**name**	**type**	**Remark**
	`hComm`	`HANDLE`	The handle connected to the reader and writer, the handle returned when opening the serial port
	Success: 0; Failure: non-0; (See return value error code table)		
return	`int state = UHF_RFID_API. CloseDevice (m_handler);`		

3.2. 18000-6C (EPC G2)

3.2.1. InventoryContinue

Definition	int InventoryContinue(HANDLE hComm, BYTE btInvCount, DWORD dwInvParam);			
Description	Check whether have tag comppliant with the protocol exists in the valid range			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	A handle connected with reader
	invCount	BYTE	[in]	Inventory type：0x00： Inventory according time:0x03： Custom inventory，select and query parameter use SelectOrSortSet and QueryCfgSet command set
	invParam	DWORD	[in]	Inventory parameter:4 byte ,unit: S, value as 0 will continue inventory When inventory type as custom inventory, max inventory 256S
Return (int)	Default no Return （Check return value error code sheet）			

Parameter Code	Int state =0;
	Int count =0;
	Int invParam =0;
	state =InventoryContinue(m_handler, count, invParam);// Set cycle inventory			

3.2.2. GetTagInfo

Definition	int GetTagInfo(`HANDLE hComm, TagInfo* tag_info, WORD timeout`);			
Description	Check whether have tag comppliant with the protocol exists in the valid range			
Parameter	**Nmae**	**Type**	**Direction**	**Remarks**
	`hComm`	`HANDLE`	[in]	The handle connected to reader,all API operation need use this handle

	tag_info	TagInfo	[out]	`typedef struct`
				`{`
				`    WORD reserve;`
				`    SHORT rssi;`
				`    BYTE antenna;`
				`    BYTE channel;`
				`    BYTE reserve;`
				`    BYTE reserve;`
				`    BYTE codeLen;`
				`    BYTE code[255];`
				`}TagInfo;`

	timeout	`WORD`	[in]	Waiting data time, Unit:ms
Return (int)	STAT_OK: Command operation success, at the same time return inventoried other tag data STAT_CMD_INVENTORY_STOP: finish inventory or no tag around			

Parameter Code	`TagInfo info;`
	`int state = UHF_RFID_API.GetTagUii(m_handler, out info, 1000);`			

3.2.3. InventoryStop

Definition	int InventoryStop(`HANDLE hComm, WORD timeout);`			
Description	Initialize the callback library After initialize the API, inventory command and mix inventory command received tag can upload by callback way			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	`hComm`	`HANDLE`	[in]	Connected reader handle, all API operation need use this handle
	timeout	WORD	[in]	Waiting data time, Unit:ms
Return	NO			
Parameter Code	`Int state = UHF_RFID_API.InventoryStop(m_handler, 10000);`			

3.2.4. ReadTag

Definition	`int ReadTag(HANDLE hComm, byte option, BYTE* accPwd, byte memBank, WORD wordPtr, WORD wordCount);`			
Description	This command read tag whole or part resersed area,EPC storager, TID storager or data in USER storeager,read start from indicated address, use word as unit			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	`hComm`	`HANDLE`	[in]	Connected reader handle, all API operation need use this handle
	`option`	BYTE	[in]	0x00，Retain Byte 。
	`accPwd`	BYTE*	[in]	4 bytes, access password, to allow tag enter safety state,default as 0x00000000。
	`memBank`	BYTE	[in]	1 byte，select storage area need read 0x00 – Reserved area； 0x01 – EPC Storage area； 0x02 – TID Storage area； 0x03 –USER Storage area, other value reserve If appear other value among command, will return parameter error message
	WordPtr	`WORD`	[in]	1 byte, the start word address to read (word)
	`wordCount`	BYTE	[in]	1 byte, the data length to read (word)
Return (int)	0: Success。NOT 0: Failure，(check the return value error code sheet)			

Parameter Code	`BYTE* m_arrPwd = new BYTE* { 0, 0, 0, 0 };`
	`byte  memBank = 0;`
	`byte wordPtr = 0;`
	`byte wordcount = 2;`
	`int state = ReadTag(m_handler, 0, accPwd,`
	`•                                    (byte)memBank, wordPtr,`
	`          wordCount);//`
	`Read the kill password`			

3.2.5. GetReadTagResp

Definition	int GetReadTagResp(HANDLE hComm, TagResp* resp,byte wordCount, BYTE* readData, WORD timeout);			
Description	This command can write a few words to reserved memory,EPC storager, TID storager or USER storage by one time			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	resp	TagResp	[out]	Return read data。
				typedef struct
				{
				    BYTE tagStatus;
				    BYTE antenna;
				    BYTE crc[2];
				    BYTE pc[2];
				    BYTE codeLen;
				    BYTE code[255];
				}TagResp;

	wordCount	BYTE	[out]	1 byte, return the number of tag data words that read
	readData	BYTE*	[out]	The tag data that read,length is WordCount×2 bytes
	timeout	WORD	[in]	Waiting tag return time, unit:ms
Return (int)	0:Success。Not 0: Failure，(check the return value error code sheet)			
Parameter Code	No			

3.2.6. WriteTag

Definition	int WriteTag(HANDLE hComm, byte option, BYTE* accPwd, byte memBank, WORD wordPtr, byte wordCount, BYTE* writeData);			
Description	This command can write a few words to reserved memory,EPC storager, TID storager or USER storage by one time			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	option	BYTE	[in]	0x00，Retain byte。
	accPwd	BYTE*	[in]	4 bytes, access password, to allow tag enter safety state,default as 0x00000000。
	memBank	BYTE	[in]	1 byte，select storage area need read 0x00 – Reserved area； 0x01 – EPC Storage area； 0x02 – TID Storage area； 0x03 –USER Storage area, other value reserve If appear other value among command, will return parameter error message
	WordPtr	WORD	[in]	2 bytes, the start word adress to read
	wordCount	BYTE	[in]	1 byte, the word length to read
	Writedata	BYTE*	[in]	wordcount*2 bytes，data need to be written
Return (int)	0:success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.7. GetTagResp

Definition	int GetTagResp(HANDLE hComm, WORD cmd, out TagResp resp, WORD timeout);			
Description	This command respond to command for obtaining write,kill and lock			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	cmd	WORD	[in]	Command word, Write： 0x0004; lock： 0x0005; Kill： 0x0006;

	resp	TagResp	[out]	`typedef struct`
				`{`
				`    BYTE tagStatus;`
				`    BYTE antenna;`
				`    BYTE crc[2];`
				`    BYTE pc[2];`
				`    BYTE codeLen;`
				`    BYTE code[255];`
				`}TagResp;`

	timeout	WORD	[in]	Waiting command respond time, unit:ms
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.8. LockTag

Definition	int LockTag(HANDLE hComm, BYTE* accPwd, byte erea, byte action);			
Description	This command can set reserved area as readable/writable,always readable/writable, with password readable/writable; can separate set EPC storage, TID storage and USER storage as read/write.always writable, with password writable, always not-writable; EPC storeage, TID storage or USER storeage are always readable. And then, TID storage just reader only, can not write.			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	accPwd	byte[4]	[in]	4 bytes, access password, to allow tag enter safety state,default as 0x00000000
	erea	BYTE	[in]	0x00：Kill password area；0x01：Access password area； 0x02：EPC； 0x03：TID；0x04：User；
	action	BYTE	[in]	1 Byte , When select as 0x00 or 0x01，SetProtect value represent the definition as below: 0x00 – Set as readable/writable 0x01 – Set as always readable/writable 0x02 – Set as with password readable/writable 0x03 – Set as non-readable/non-writable When Select as 0x02、0x03、0x04，SetProtectvalue represent the definition as below: 0x00 – Set as writable 0x01 – Set as always writable 0x02 –Set as with passwrod writable 0x03 –Set as always non-writable
Return (int)	0:Success。NOT 0:Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.9. KillTag

Definition	KillTag(HANDLE hComm, BYTE* accPwd);			
Description	This command use to destruct tag, after destructed, won't process reader command never (need select tag before operate that command)			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	accPwd	byte[4]	[in]	4 bytes, access password, to allow tag enter safety state,default as 0x00000000
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.10. SetSelectMask

Definition	int SetSelectMask(HANDLE hComm, WORD maskPtr, byte maskBits, BYTE* mask);			
Description	This command can select tag according rule			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	maskPtr	WORD	[in]	2 bytes, default as 0x0000。
	maskBits	byte	[in]	1 byte, need match EPC number bit length,default as 0x00；if maskBits as 0，means didn't indicated tag,multiple tags operation（will operation for all tags at current area)
	mask	BYTE*	[in]	• Need matching data, effective data length as lengthbit, if length is odd, need add 0 to the lower part of the mask code
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO CF FF 00 07 0F 00 00 60 E2 80 68 94 00 00 40 0B 19 B6 16 01 CB D0 Select tag with EPC number as E2 80 68 94 00 00 40 0B 19 B6 16 01 。			

3.2.11. SetCoilPRM

Definition	int SetCoilPRM(HANDLE hComm, byte qVal, byte reserved);			
Description	This command can set Q value size			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	qVal	byte	[in]	1 byte, Q value size have relation to tags quantity around, the number of tags is 2 to the Q
	reserved	byte	[in]	1 byte, Rserved, default as 0
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.12. GetCoilPRM

Definition	GetCoilPRM(HANDLE hComm, out byte pqVal, out byte reserved);			
Description	This command use to obtain Q value			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	qVal	byte	[out]	1 byte, Q value size have relation to tags quantity around, the number of tags is 2 to the Q
	reserved	byte	[in]	1 byte, Rserved, default as 0
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.13. SelectOrSortSet

Definition	int SelectOrSortSet(HANDLE hComm, byte prot, SelectSortParam param);			
Description	Check whether exist tag that comply with protocol in the valid range			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	prot	byte	[in]	Protocol No： Set as 0x00;

	param	SelectSortParam	[in]	`typedef struct`
				`{`
				`    BYTE target;`
				`    BYTE trucate;`
				`    BYTE action;`
				`    BYTE membank;`
				`    WORD m_ptr;`
				`    BYTE len;`
				`    BYTE mask[31];`
				`}SelectSortParam;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.14. SelectOrSortGet

Definition	int SelectOrSortGet(HANDLE hComm, byte proto, SelectSortParam param);			
Description	Check whether exist tag that comply with protocol in the valid range			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	prot	byte	[in]	Protocol No：Set as 0x00;

	param	SelectSortParam	[out]	`typedef struct`
				`{`
				`    BYTE target;`
				`    BYTE trucate;`
				`    BYTE action;`
				`    BYTE membank;`
				`    WORD m_ptr;`
				`    BYTE len;`
				`    BYTE mask[31];`
				`}SelectSortParam;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.15. QueryCfgSet

Definition	int QueryCfgSet(HANDLE hComm, byte proto, QueryParam param);			
Description	Check whether exist tag that comply with protocol in the valid range			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	prot	byte	[in]	Protocl No： Set as 0x00;

	param	QueryParam	[in]	`typedef struct`
				`{`
				`    BYTE condition;`
				`    BYTE session;`
				`    BYTE target;`
				`}QueryParam;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.2.16. QueryCfgGet

Definition	int QueryCfgGet(HANDLE hComm, byte proto, QueryParam param);			
Description	Check whether exist tag that comply with protocol in the valid range			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	prot	byte	[in]	Protocl No： Set as 0x00;
	param	QueryParam	[out]	`typedef struct { BYTE condition; BYTE session; BYTE target; }QueryParam;`
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3. Reader Custom Comamand

3.3.1. GetInfo

Definition	int GetInfo (HANDLE hComm, DeviceInfo* devInfo);			
Description	Obtain reader information, reader software version and other informations			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	devInfo	DeviceInfo	[out]	`typedef struct`
				`{`
				`    BYTE firmVersion[32];`
				`    BYTE hardVersion[32];`
				`    BYTE SN[12];`
				`    BYTE PARA[12];`
				`}DeviceInfo;`
				
				`struct PARA`
				`{`
				`    BYTE RFIDPRO;`
				`    WORD STRATFREI;`
				`    WORD STRATFRED;`
				`    WORD STEPFRE;`
				`    BYTE CN;`
				`    BYTE POWER;`
				`    BYTE ANTENNA;`
				`    BYTE REGION;`
				`    BYTE RESERVED;`
				`};`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.2. GetDeviceInfo

Definition	int `GetDeviceInfo(HANDLE hComm, DeviceFullInfo* devInfo);`			
Description	Obtain reader information, reader software version and other informations			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	devInfo	`DeviceFullInfo`	[out]	`typedef struct`
				`{`
				`    BYTE DevicehardVersion[32];`
				`    BYTE DevicefirmVersion[32];`
				`    BYTE DeviceSN[12];`
				`    BYTE hardVersion[32];`
				`    BYTE firmVersion[32];`
				`    BYTE SN[12];`
				`}DeviceFullInfo;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	No			

3.3.3. GetDevicePara

Definition	int `GetDevicePara(HANDLE hComm, DevicePara* devInfo);`			
Description	Obtain reader information, reader software version and other informations			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	devInfo	`DevicePara`	[in]	`typedef struct`
				`{`
				`BYTE DEVICEARRD;`
				`BYTE RFIDPRO;`
				`BYTE WORKMODE;`
				`BYTE INTERFACE;`
				`BYTE BAUDRATE;`
				`BYTE WGSET;`
				`BYTE ANT;`
				`BYTE REGION;`
				`BYTE STRATFREI[2];`
				`BYTE STRATFRED[2];`
				`BYTE STEPFRE[2];`
				`BYTE CN;`
				`BYTE RFIDPOWER;`
				`BYTE INVENTORYAREA;`
				`BYTE QVALUE;`
				`BYTE SESSION;`
				`BYTE ACSADDR;`
				`BYTE ACSDATALEN;`
				`BYTE FILTERTIME;`
				`BYTE TRIGGLETIME;`
				`BYTE BUZZERTIME;`
				`BYTE INTENERLTIME;`
				`}DevicePara;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			
参考代码	无			

3.3.4. SetDevicePara

Definition	int SetDevicePara(HANDLE hComm, DevicePara devInfo);			
Description	Obtain reader information, reader software version and other informations			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	devInfo	DevicePara	[in]	`typedef struct`
				`{`
				`BYTE DEVICEARRD;`
				`BYTE RFIDPRO;`
				`BYTE WORKMODE;`
				`BYTE INTERFACE;`
				`BYTE BAUDRATE;`
				`BYTE WGSET;`
				`BYTE ANT;`
				`BYTE REGION;`
				`BYTE STRATFREI[2];`
				`BYTE STRATFRED[2];`
				`BYTE STEPFRE[2];`
				`BYTE CN;`
				`BYTE RFIDPOWER;`
				`BYTE INVENTORYAREA;`
				`BYTE QVALUE;`
				`BYTE SESSION;`
				`BYTE ACSADDR;`
				`BYTE ACSDATALEN;`
				`BYTE FILTERTIME;`
				`BYTE TRIGGLETIME;`
				`BYTE BUZZERTIME;`
				`BYTE INTENERLTIME;`
				`}DevicePara;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.5 SetDevicePara_J

illustrate	Obtain the information of the reader, the software version of the reader and many other information			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	HANDLE	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.
	DEVICEARRD	BYTE	[in]	1 Byte Communication address of the device The default is 0x00 and cannot be 0xFF
	RFIDPRO	BYTE	[in]	1 Byte Device RF 0x00:ISO 18000-6c 0X01: GB/T29768 0x02： GJB7377.1 Currently only supports ISO18000-6C
	WORKMODE	BYTE	[in]	1 Byte The working mode of the device 0: answer mode 1: active mode 2: trigger mode
	INTERFACE	BYTE	[in]	1 Byte Device communication interface Default 0x80 0x80: RS232 0x40: RS485 0x20: RJ45 0x10: WiFi
	BAUDRATE	BYTE	[in]	1 Byte Serial baud rate default 4 0: 9600bps 1: 19200bps 2: 38400bps 3: 57600bps 4: 115200bps
	WGSET	BYTE	[in]	1 Byte Configuration parameters of the Wiegand data output interface Default 0x00
	ANT	BYTE	[in]	8 Bytes The default is 0x01, which means antenna 1 All the antenna numbers of the device, which represent the selected antenna by bit, and the corresponding bit value is 1, which means that the antenna is used. A value of 0 means that the antenna is not used; starting from the lower bit, the 0th bit means antenna 1, the 1st bit means antenna 2, and so on, up to 8 antennas can be represented; different modules support different antennas, depending on the specific depending on the situation;

	REGION	BYTE	[in]	1 Byte National frequency range: 0x00: User-defined according to requirements; 0x01: US[902.75~927.25] 0x02: Korea[917.1~923.5] 0x03: EU[865.1~868.1] 0x04: JAPAN[952.2~953.6] 0x05: MALAYSIA[919.5~922.5] 0x06: EU3[865.7~867,5] 0x07: CHINA_BAND1[840.125~844.875] 0x08: CHINA_BAND2[920.125~924.875]
	STRATFREI1	BYTE	[in]	1 Byte High byte of the integer part of the megahertz start frequency Such as 920.125MHz STRATFREI=920=0x0398, High byte=0x03, low byte=0x98;
	STRATFREI2	BYTE	[in]	1 Byte Low byte of the integer part of the megahertz start frequency
	STRATFRED1	BYTE	[in]	1 Byte The high byte of the fractional part of the megahertz start frequency Such as 920.125MHz STRATFREI= 125= 0x007D, High byte = 0x00, low byte = 0x7D;
	STRATFRED2	BYTE	[in]	1 Byte Low byte of the fractional part of the megahertz start frequency
	STEPFRE1	BYTE	[in]	1 Byte High byte of frequency step (KHz) Such as 125KHz, STEPFRE=125=0x007D, high byte = 0x00, Low byte = 0x7D;
	STEPFRE2	BYTE	[in]	1 Byte Low byte of frequency step (KHz)
	CN	BYTE	[in]	1 Byte number of channels
	RFIDPOWER	BYTE	[in]	1 Byte The RFID output power of the device, the unit is: dBm, the value range is: [0,30]dBm, others are invalid.
	INVENTORYAREA	BYTE	[in]	1 Byte The memory area of the device to access the tag 0x01 Default EPC 0x02 TID 0x03 USER 0x04 EPC+TID 0x05 EPC+USER 0x06 EPC+TID+USER Other values are reserved

	QVALUE	BYTE	[in]	1 Byte The initial Q value used when querying EPC tags The default value is 4 The value range is 0~15
	SESSION	BYTE	[in]	1 Byte Session value used when querying EPC tags Default is 0 Value range [0, 3]
	ACSADDR	BYTE	[in]	1 Byte The starting address of the device to access the label storage area Default value 0x00
	ACSDATALEN	BYTE	[in]	1 Byte The data length of the device to access the tag storage area Default value 0x00
	FILTERTIME	BYTE	[in]	1 Byte filter time default value 0 Value range [0, 255] Unit second (s)
	TRIGGLETIME	BYTE	[in]	1 Byte The query duration after the device receives the trigger signal default value 0 Value range [0, 255] Unit second (s)
	BUZZERTIME	BYTE	[in]	1 Byte Buzzer beeping time default value 1 When the value is 0, it means no tweet Value range [0, 255] Unit 10 milliseconds (10ms)
	INTENERLTIME	BYTE	[in]	1 Byte query interval default value 1 Value range [0, 255] Unit 10 milliseconds (10ms)
return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.3.6 RebootDevice

Definition	int RebootDevice(HANDLE hComm);			
Description	After operate command, reader restore factory default parameter, includ frequency,power,antenna enable etc			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.7 SetRFPower

Definition	int SetRFPower(HANDLE hComm, byte power, byte reserved);			
Description	Set reader output power			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	power	BYTE	[in]	1 byte, reader power , range 0~33dBm。
	reserved	byte	[in]	Reserved
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.8. GetRFPower

Definition	int GetRFPower(HANDLE hComm, out byte power, out byte reserved);			
Description	Obtain reader power			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle。
	power	BYTE	[out]	1 byte， reader power ，range 0~33dBm。
	reserved	byte	[out]	Reserved
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.9. SetFreq

Definition	int SetFreq(HANDLE hComm, ref FreqInfo frqInfo);			
Description	This command sets the upper limit and lower limit of the reader working frequency. The upper frequency must be greater than or equal to the lower frequency.			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	frqInfo	FreqInfo	[in]	`typedef struct {`
				 `BYTE region;`
				 `WORD StartFreq;`
				 `WORD StopFreq;`
				 `WORD StepFreq;`
				 `BYTE cnt;`
				 `}FreqInfo;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

Frequency set as below：

0x00: User define according demand ；
0x01：US [902.75~927.25]
0x02：Korea [917.1~923.5]
0x03：EU [865.1~868.1]
0x04：JAPAN [952.2~953.6]
0x05：MALAYSIA [919.5~922.5]
0x06：EU3 [865.7~867,5]
0x07：CHINA_BAND1 [840.125~844.875]
0x08：CHINA_BAND2 [920.125~924.875]

Each frequency band calculation formula：

Chinese band2：	*Fs = 920.125 + N * 0.25 (MHz) N∈[0, 19]*。
US band：	*Fs = 902.75 + N * 0.5 (MHz) N∈[0,49]*。
Korean band：	*Fs = 917.1 + N * 0.2 (MHz) N∈[0, 31]*。
EU band:	*Fs = 865.1 + N*0.2(MHz) N∈[0, 14]*。
Ukraine band:	*Fs = 868.0 + N*0.1(MHz) N∈[0, 6]*。
Chinese band1：	*Fs = 840.125 + N * 0.25 (MHz) N∈[0, 19]*。
US band3：	*Fs = 902 + N * 0.5 (MHz) N∈[0,52]*。

3.3.10. GetFreq

Definition	int SetBaudRate(BYTE * ComAddr, BYTE baud, int FrmHandle);			
Description	Set reader communication baud rate			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	frqInfo	FreqInfo	[in]	`typedef struct`
				`{`
				`    BYTE region;`
				`    WORD StartFreq;`
				`    WORD StopFreq;`
				`    WORD StepFreq;`
				`    BYTE cnt;`
				`}FreqInfo;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.11 SetAntenna

definition	`int SetAntenna(HANDLE hComm, BYTE* antenna);`			
illustrate	This command is used to set the antenna.			
parameter.	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.
	antenna	byte	[in]	1 byte, the corresponding Bit value is 1, which means that the antenna is used, and the value is 0, which means that the antenna is not used; starting from the low bit, the 0th bit indicates the 1st antenna, the 1st bit indicates the 2nd antenna, and so on , can represent up to 8 antennas, and different modules support different antennas, depending on the specific situation;
return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.3.12 GetAntenna

definition	`int GetAntenna(HANDLE hComm, BYTE* antenna);`			
illustrate	This command is used for 4-port reader antenna configuration.			
parameter	**name**	**type**	**directio n**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.
	antenna	byte	[out]	1 byte, the corresponding Bit value is 1, which means that the antenna is used, and the value is 0, which means that the antenna is not used; starting from the low bit, the 0th bit indicates the 1st antenna, the 1st bit indicates the 2nd antenna, and so on , can represent up to 8 antennas, and different modules support different antennas, depending on the specific situation;
return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.3.13. SetRFIDType

Definition	int SetRFIDType(HANDLE hComm, byte type);			
Description	This command use to set module protocol type			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	type	byte	[in]	1 byte，0x00：ISO 18000-6C；0x01：GB/T 29768；0x02：GJB 7377.1； At present only support ISO 18000-6C。
Return (int)	0:Success。NOT 0: Failure，（check the return value error code sheet)			
Parameter Code	NO			

3.3.14. GetRFIDType

Definition	int GetRFIDType(HANDLE hComm, out byte type);			
Description	This command use to obtain module protocol type			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	type	byte	[out]	1 byte，0x00：ISO 18000-6C；0x01：GB/T 29768；0x02：GJB 7377.1； At present only support ISO 18000-6C。
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.15. GetTemperature

Definition	int GetTemperature(HANDLE handler, out byte tempCur, out byte tempLimit);			
Description	This command use to obtain the current temperature and threshold			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	tempCur	byte	[out]	1 byte，current temperature unit as degrees Celsius 。
	tempLimit	byte	[out]	1 byte，current temperature threshold， over this temperatuere ,module will stop work to wait temperature reduce
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.16. SetTemperature

Definition	int SetTemperature(HANDLE handler, byte tempLimit, byte resv);			
Description	This command use to set temperature and threshold			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	tempLimit	byte	[in]	1 byte, current temperature threshold, over this temperatuere ,module will stop work to wait temperature reduce, generally set 50~90°C。
	resv	byte	[in]	保留
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.17 GetNetInfo

Definition	int `GetNetInfo(HANDLE hComm, NetInfo *type);`			
Description	This command use to obtain device network parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	`type`	`NetInfo`	[out]	`typedef struct`
				`{`
				`    BYTE IP[4];`
				`    BYTE MAC[6];`
				`    BYTE PORT[2];`
				`    BYTE NetMask[4];`
				`    BYTE Gateway[4];`
				`}NetInfo;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	No			

3.3.18 SetNetInfo

Definition	int `SetNetInfo(HANDLE hComm, NetInfo type);`			
Description	This command use to set device network parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	`type`	`NetInfo`	[in]	`typedef struct`
				`{`
				`BYTE IP[4];`
				`BYTE MAC[6];`
				`BYTE PORT[2];`
				`BYTE NetMask[4];`
				`BYTE Gateway[4];`
				`}NetInfo;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.19. GetRemoteNetInfo

Definition	int `GetRemoteNetInfo(HANDLE hComm, RemoteNetInfo *type);`			
Description	This command use to obtain remote network parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	`type`	`NetInfo`	[out]	`typedef struct`
				`{`
				`BYTE Enable;`
				`BYTE IP[4];`
				`BYTE PORT[2];`
				`BYTE HeartTime;`
				`}RemoteNetInfo;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.7.16. SetRemoteNetInfo

Definition	Int `SetRemoteNetInfo(HANDLE hComm, RemoteNetInfo type);`			
Description	This command use to set remote network parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	`type`	`NetInfo`	[in]	`typedef struct`
				`{`
				`    BYTE Enable;`
				`    BYTE IP[4];`
				`    BYTE PORT[2];`
				`    BYTE HeartTime;`
				`}RemoteNetInfo;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.21. GetPermissonPara

Definition	Int `GetPermissonPara(HANDLE hComm, PermissonPara* PermissonPara);`			
Description	This command use to obtain reading permission parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	`PermissonPara`	`PermissonPara`	[out]	`typedef struct`
				`{`
				`    BYTE CodeEn;`
				`    BYTE Code[4];`
				`    BYTE MaskEn;`
				`    BYTE StartAdd;`
				`    BYTE MaskLen;`
				`    BYTE MaskData[12];`
				`    BYTE MaskCondition;`
				`}PermissonPara;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.22. SetPermissonPara

Definition	Int `SetPermissonPara(HANDLE hComm, PermissonPara PermissonPara);`			
Description	This command use to set reading permission parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	`PermissonPara`	`PermissonPara`	[in]	`typedef struct`
				`{`
				`    BYTE CodeEn;`
				`    BYTE Code[4];`
				`    BYTE MaskEn;`
				`    BYTE StartAdd;`
				`    BYTE MaskLen;`
				`    BYTE MaskData[12];`
				`    BYTE MaskCondition;`
				`}PermissonPara;;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.23. GetGpioPara

Definition	int `GetGpioPara(HANDLE hComm, GpioPara* GpioPara);`			
Description	This command use to obtain GPIO parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle

	`GpioPara`	`NetInfo`	[out]	`typedef struct`
				`{`
				`    BYTE KCEn;`
				`    BYTE RelayTime;`
				`    BYTE KCPowerEn;`
				`    BYTE TriggleMode;`
				`    BYTE BufferEn;`
				`    BYTE ProtocolEn;`
				`    BYTE ProtocolType;`
				`    BYTE`
				`ProtocolFormat[10];`
				`}GpioPara;`

Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.24. SetGpioPara

Definition	Int `SetGpioPara(HANDLE hComm, GpioPara GpioPara);`			
Description	This command use to set GPIO parameter			
Parameter	**Name**	**Type**	**Direction**	**Remarks**
	hComm	HANDLE	[in]	Connected reader handle, all API operation need use this handle
	`GpioPara`	`NetInfo`	[in]	`typedef struct { BYTE KCEn; BYTE RelayTime; BYTE KCPowerEn; BYTE TriggleMode; BYTE BufferEn; BYTE ProtocolEn; BYTE ProtocolType; BYTE ProtocolFormat[10]; }GpioPara;`
Return (int)	0:Success。NOT 0: Failure，(check the return value error code sheet)			
Parameter Code	NO			

3.3.25 GetLongPermissonPara

definition	Int `GetPermissonPara(HANDLE hComm, PermissonPara* PermissonPara);`			
illustrate	This command is used to obtain read permission.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.
	`PermissonPara`	`LongPermissonPara *`	[out]	`typedef struct { BYTE CodeEn; BYTE Code[4]; BYTE MaskEn; BYTE StartAdd; BYTE MaskLen; BYTE MaskData[31]; BYTE MaskCondition; }LongPermissonPara;`
return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.3.26 SetLongPermissonPara

definition	Int `SetPermissonPara(HANDLE hComm, LongPermissonPara PermissonPara);`			
illustrate	This command is used to set the card reading authority.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.

	`PermissonPara`	`LongPermissonPara`	[in]	`typedef struct`
				`{`
				`    BYTE CodeEn;`
				`    BYTE Code[4];`
				`    BYTE MaskEn;`
				`    BYTE StartAdd;`
				`    BYTE MaskLen;`
				`    BYTE MaskData[31];`
				`    BYTE MaskCondition;`
				`}LongPermissonPara;`

return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.3.27 Close_Relay

definition	`int Close_Relay(HANDLE hComm, BYTE time);`			
illustrate	该命令用于闭合继电器。			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.
	`time`	`Byte`	[out]	`Time represents the closing time, 0 means it is always closed, and the unit is S.`
return (int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.3.28 Release_Relay

definition	`int Release_Relay(HANDLE hComm, BYTE time);`			
illustrate	This command is used to release the relay.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.
	`time`	`Byte`	[out]	`invalid`
return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.4 Commands for Gate Reader

3.4.1 SetGPIOWorkParam

definition	`int SetGPIOWorkParam(HANDLE hComm, GPIOWorkParam gpio);`			
illustrate	This command is used to set GPIO working parameters.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.

	gpio	`GPIOWorkParam`	[in]	`typedef struct {`
				`    BYTE Mode;`
				`    BYTE GPIEnable;`
				`    BYTE InLevel;`
				`    BYTE GPOEnable;`
				`    BYTE PutLevel;`
				`    BYTE PutTime[8];`
				`}GPIOWorkParam;`

return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.4.2 GetGPIOWorkParam

definition	`int GetGPIOWorkParam(HANDLE hComm, GPIOWorkParam* gpio);`			
illustrate	This command is used to obtain GPIO working parameters.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.

	gpio	`GPIOWorkParam`	[out]	`typedef struct {`
				`    BYTE Mode;`
				`    BYTE GPIEnable;`
				`    BYTE InLevel;`
				`    BYTE GPOEnable;`
				`    BYTE PutLevel;`
				`    BYTE PutTime[8];`
				`}GPIOWorkParam;`

return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.4.3 SetGateWorkParam

definition	`int SetGateWorkParam(HANDLE hComm, GateWorkParam gate);`			
illustrate	This command is used to set the working parameters of the gate reader.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.

	gate	`GateWorkParam`	[in]	`typedef struct {`
				`    BYTE GateMode;`
				`    BYTE GateGPI1;`
				`    BYTE GateGPI2;`
				`    BYTE GatePower;`
				`    BYTE GateRead;`
				`    BYTE EASMode;`
				`    BYTE EASGPO;`
				`}GateWorkParam;`

return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.4.4 GetGateWorkParam

definition	`int GetGateWorkParam(HANDLE hComm, GateWorkParam gate);`			
illustrate	This command is used to obtain the working parameters of the gate reader			
parameter	**name**	**type**	**directio n**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter. .

	gate	`GateWorkParam`	[out]	`typedef struct {`
				`    BYTE GateMode;`
				`    BYTE GateGPI1;`
				`    BYTE GateGPI2;`
				`    BYTE GatePower;`
				`    BYTE GateRead;`
				`    BYTE EASMode;`
				`    BYTE EASGPO;`
				`}GateWorkParam;`

return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.4.5 GetGateStatus

definition	`int GetGateStatus(HANDLE hComm, GateParam* status, WORD timeout);`			
illustrate	This command is used to get access gate reader.			
parameter	**name**	**type**	**directio n**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.

	status	`GateParam`	[in]	`typedef struct {`
				`    BYTE DIR;`
				`    BYTE GPI;`
				`    BYTE SYSTIME[4];`
				`}GateParam;`

	timeout	`WORD`	[in]	Waiting time for command response, in ms.
return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.4.6 SetEASMask

definition	`int SetEASMask(HANDLE hComm, EASMask* eas);`			
illustrate	This command is used to set EAS data matching.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.

	eas	`EASMask`	[in]	`typedef struct {`
				`    BYTE Addr;`
				`    BYTE Len;`
				`    BYTE Data[32];`
				`}EASMask;`

return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.4.7 GetEASMask

definition	`int SetEASMask(HANDLE hComm, EASMask* eas);`			
illustrate	This command is used to obtain EAS data matching.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle to the reader/writer connection, which is required for all API operations thereafter.

	eas	`EASMask`	[out]	`typedef struct {`
				`    BYTE Addr;`
				`    BYTE Len;`
				`    BYTE Data[32];`
				`}EASMask;`

return(int)	0: indicates success. Not equal to 0: indicates failure, please check the return value error code table.			
Reference Code	none			

3.5 Access System Reader related commands

3.5.1 GetAccessInfo

definition	`int GetAccessInfo(HANDLE hComm, AccessInfo* access);`			
illustrate	This command is used to obtain access control information.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle of the reader/writer connection, which is required for all API operations thereafter.

	access	`AccessInfo`	[out]	`typedef struct {`
				`    BYTE STATE;`
				`    WORD CUSTOMERCOUNT;`
				`}AccessInfo;`

return (int)	0: indicates success. Not equal to 0: Indicates failure, please check the return value error code table.			
Reference Code	none			

3.5.2 BeginWhiteList

definition	`int BeginWhiteList(HANDLE hComm, BYTE Option, DWORD infoCount);`			
illustrate	This command is used to notify the device to enter the whitelist data update or acquisition process.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle of the reader/writer connection, which is required for all API operations thereafter.
	Option	`BYTE`	[in]	0x01: Enter the status of the host computer transmitting whitelist data to the device; 0x02: Enter the state where the device transmits whitelist data to the host computer; Other values: invalid.
	infoCount	`DWORD`	[in]	When Option=1, it indicates the total number of customer information CUSTOMERINFO that needs to be updated; when Option=2, INFOCOUNT = 0
return (int)	0: indicates success. Not equal to 0: Indicates failure, please check the return value error code table.			
Reference Code	none			

3.5.3 GetWhiteList

definition	`int GetWhiteList(HANDLE hComm, WhiteList* whiteList, WORD timeout)`			
illustrate	This command is used to obtain the whitelist.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle of the reader/writer connection, which is required for all API operations thereafter.

	whiteList	`WhiteList`	[out]	`typedef struct {`
				`    BYTE STATUS;`
				`    WORD FRAMENUM;`
				`    BYTE INFOCOUNT;`
				`    BYTE WHITELIST[4096];`
				`}WhiteList;`

	timeout	`WORD`	[in]	Connection timeout setting, unit ms.
return (int)	0: indicates success. Not equal to 0: Indicates failure, please check the return value error code table.			
Reference Code	none			

WhiteList Parameter Description：

- Status： command response code
  0x8C: Upload data;
  Other values: invalid;
- FRAMENUM： Transmission packet number, counting from 0, length 2 bytes.
- INFOCOUNT： The number of customer information CUSTOMERINFO in the uploaded data packet, length 1 byte, maximum 2048/32 = 64.
- WHITELIST： Whitelist, N Bytes, N maximum 251, N = 32*INFOCOUNT. It consists of multiple customer information CUSTOMERINFO.

3.5.4 SetWhiteList

definition	`int SetWhiteList(HANDLE hComm, WORD len, const BYTE* pParam)`			
illustrate	This command is used to set the whitelist.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle of the reader/writer connection, which is required for all API operations thereafter.
	len	`WORD`	[in]	The length of the long pass whitelist
	pParam	`BYTE`	[out]	The number of whitelists returned by the device for the current update
return (int)	0: indicates success. Not equal to 0: Indicates failure, please check the return value error code table.			
Reference Code	none			

3.5.5 EndWhiteList

definition	`int EndWhiteList(HANDLE hComm, DWORD* infoCount);`			
illustrate	This command is used to end setting the whitelist.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle of the reader/writer connection, which is required for all API operations thereafter.
	infoCount	`DWORD`	[out]	The device returns the total number of whitelists for this update
return (int)	0: indicates success. Not equal to 0: Indicates failure, please check the return value error code table.			
Reference Code	none			

3.5.6 GetAccessOperateParam

definition	`int GetAccessOperateParam(HANDLE hComm, AccessOperateParam* aParam);`			
illustrate	This command is used to obtain access control operation parameters.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	`HANDLE`	[in]	The handle of the reader/writer connection, which is required for all API operations thereafter.

	aParam	`AccessOperateParam`	[out]	`typedef struct {`
				`    BYTE LISTENABLE;`
				`    BYTE GPI1EXEFUNC;`
				`    BYTE GPI2EXEFUNC;`
				`    BYTE GPI3EXEFUNC;`
				`    BYTE GPI4EXEFUNC;`
				`    BYTE RECVGPIEXEFUNC[4];`
				`    BYTE NUMAUTHORIZEDACTIONEXEGPO;`
				`    BYTE NUMUNAUTHORIZEDACTIONEXEPART;`
				`    BYTE RECVACTIONEXEFUNC[6];`
				`}AccessOperateParam;`

return (int)	0: indicates success. Not equal to 0: Indicates failure, please check the return value error code table.			
Reference Code	none			

AccessOperateParam Parameter Description:

▪LISTENABLE: Whitelist enable parameters, 1Byte。

bit	Definition

7	0
6	0
5	0
4	0
3	0
2	0
1	0
0	0: Turn off the whitelist; 1: Turn on the whitelist.

- GPI1EXEFUNC: GPI1 input execution function parameter, length 1Byte, default 0x01 triggers card reading, can be configured for other action triggers.

bit	Definition
7	0
6	0
5	0
4	0: Disable the trigger GPO1 output; 1: Enable the trigger GPO1 output.
3	0x00: No action; 0x01: read card; 0x02: Channel is open; 0x03: Channel closed; Other values are invalid.
2	
1	
0	

- GPI2EXEFUNC: GPI2 input execution function parameter, length 1Byte, default 0x01 triggers card reading, can be configured for other action triggers.

bit	Definition
7	0
6	0
5	0
4	0: Disable the trigger GPO1 output; 1: Enable the trigger GPO1 output.
3	0x00: No action; 0x01: read card; 0x02: Channel is open; 0x03: Channel closed; Other values are invalid.
2	
1	
0	

- GPI3EXEFUNC: GPI3 input execution function parameter, length 1Byte, default 0x03 trigger channel is closed, can be configured for other action triggers.

bit	Definition
7	0
6	0
5	0

4	0: Disable the trigger GPO1 output; 1: Enable the trigger GPO1 output.

3	0x00: No action;
	0x01: read card;
	0x02: Channel is open;
	0x03: Channel closed;
	Other values are invalid.

2	
1	
0	

- GPI4EXEFUNC: GPI4 input execution function parameters, length 1Byte, default 0x02 trigger channel is enabled, and can be configured for other action triggers.

bit位	Definition
7	0
6	0
5	0
4	0: Disable the trigger GPO1 output; 1: Enable the trigger GPO1

3	0x00: No action;
	0x01: read card;
	0x02: Channel is open;
	0x03: Channel closed;
	Other values are invalid.

2	
1	
0	

- RECVGPIEXEFUNC: Reserved input execution function parameters, length 4Bytes, default all are 0x00.
- NUMAUTHORIZEDACTIONEXEGPO: When the authorization tag is detected, the action executes the GPO parameter, that is, the gate switch controls the GPO parameter, the length is 1Byte, and the default is 0x80.

bit	Definition
7	0: Do not select RELAY; 1: Select RELAY.
6	0
5	0
4	0
3	0
2	0
1	0
0	0: Do not select GPO1; 1: Select GPO1.

- NUMUNAUTHORIZEDACTIONEXEPART: Action execution GPO parameters when an unauthorized tag is detected, that is, warning switch control GPO parameters, length 1Byte, default 0x01.

bit	Definition
7	0: Do not select RELAY; 1: Select RELAY.
6	0
5	0
4	0
3	0

2	0
1	0
0	0: Do not select GP01; 1: Select GP01.

- RECVACTIONEXEFUNC: The reserved output execution function parameters are 6 Bytes in length and all are 0x00 by default.

3.5.6 SetAccessOperateParam

definition	`int SetAccessOperateParam(HANDLE hComm, AccessOperateParam aParam);`			
illustrate	This command is used to set access control operation parameters.			
parameter	**name**	**type**	**direction**	**Remark**
	hComm	HANDLE	[in]	The handle of the reader/writer connection, which is required for all API operations thereafter.

	aParam	AccessOperateParam	[in]	`typedef struct {`
				`    BYTE LISTENABLE;`
				`    BYTE GPI1EXEFUNC;`
				`    BYTE GPI2EXEFUNC;`
				`    BYTE GPI3EXEFUNC;`
				`    BYTE GPI4EXEFUNC;`
				`    BYTE RECVGPIEXEFUNC[4];`
				`    BYTE NUMAUTHORIZEDACTIONEXEGPO;`
				`    BYTE NUMUNAUTHORIZEDACTIONEXEPART;`
				`    BYTE RECVACTIONEXEFUNC[6];`
				`}AccessOperateParam;`

return (int)	0: indicates success. Not equal to 0: Indicates failure, please check the return value error code table.			
Reference Code	none			

Appendix 1, Return value error code sheet

Definition		Annotation
#define STAT_OK	0x00000000	
#define STAT_PORT_HANDLE_ERR	0xFFFFFF01	Handle error, or input serial port parameter
#define STAT_PORT_OPEN_FAILED	0xFFFFFF02	Open serial port failure
#define STAT_DLL_INNER_FAILED	0xFFFFFF03	Internal dynamic library
#define STAT_CMD_PARAM_ERR	0xFFFFFF04	Parameter value incorrect or out of bounds,or module do not support that parameter
#define STAT_CMD_SERIAL_NUM_EXIT	0xFFFFFF05	Serial number existed
#define STAT_CMD_INNER_ERR	0xFFFFFF06	• The command execution failed due to an internal error in the module pocedure •
#define STAT_CMD_INVENTORY_STOP	0xFFFFFF07	Didn't inventoried tag or inventory finished
#define STAT_CMD_TAG_NO_RESP	0xFFFFFF08	Tag response timeout
#define STAT_CMD_DECODE_TAG_DATA_FAI	0xFFFFFF09	Failed to call tag data
#define STAT_CMD_CODE_OVERFLOW	0xFFFFFF0A	Tag data exceed the max transmission length of serial port
#define STAT_CMD_AUTH_FAIL	0xFFFFFF0B	authentication failure
#define STAT_CMD_PWD_ERR	0xFFFFFF0C	Command error
#define STAT_CMD_SAM_NO_RESP	0xFFFFFF0D	SAM card no response
#define STAT_CMD_SAM_CMD_FAIL	0xFFFFFF0E	PSAM card command execute failure
#define STAT_CMD_RESP_FORMAT_ERR	0xFFFFFF0F	reader response format incorrect
#define STAT_CMD_HAS_MORE_DATA	0xFFFFFF10	Command executed successfully, but subrequent data didn't

#define STAT_CMD_BUF_OVERFLOW	0xFFFFFF11	Incoming cache too small, data overflows
#define STAT_CMD_COMM_TIMEOUT	0xFFFFFF12	Wait reader response overtime
#define STAT_CMD_COMM_WR_FAILED	0xFFFFFF13	Write data to serial port failure
#define	0xFFFFFF14	Read serial port data
#define STAT_CMD_NOMORE_DATA	0xFFFFFF15	No more data
#define STAT_DLL_UNCONNECT	0xFFFFFF16	Network connect have not been established
#define STAT_DLL_DISCONNECT	0xFFFFFF17	Network already
#define STAT_CMD_RESP_CRC_ERR	0xFFFFFF18	Reader respond to crc check error

Appendix 2，ErrorCode sheet

Erroe code		Description
#defineSTAT_GB_TAG_LOW_POWE R	0xFFFFF 0	Tag power supply insufficient, and tag do not have enough power to completed the
#define STAT_GB_TAG_OPR_LIMIT	0xFFFFF F4	Insufficient tag operation permissions, unauthorized
#define STAT_GB_TAG_MEM_OVF	0xFFFFF F42	Tag operation store overflows, or the target
#define STAT_GB_TAG_MEM_LCK	0xFFFFF F4 3	The tag storage area locked, write operation or erase operation to locked unwritable storage area, to do read operation for locked
#define STAT_GB_TAG_PWD_ERR	0xFFFFF F4	Tag operation command error, access command error
#define STAT_GB_TAG_AUTH_FAIL	0xFFFFF F4	Tag failed to be authenticated
#define STAT_GB_TAG_UNKNW_ERR	0xFFFFF F4	Tag operation occurred unkonwn error