RFID Reader Development Guide For C/C++ (windows)

V0.1.4.0

Contents

1. Preface....4
   - 1.1. Overview....4
   - 1.2.Applicable Devices....4
   - 1.3.Copyright Statements....4
   - 1.2. Basic Flow of Reader and Write....5
2. Quick Start....5
3. Connection Description....8
   - 3.1. RS232 Connection....8
   - 3.2. RS485 Connection....8
   - 3.3. TCP Client Connection....9
   - 3.4. TCP Server Listening....9
   - 3.5. USB-HID Connection....9
   - 3.6. Close Connection....9
   - 3.7. Close TCP Server Listening....10
4. Events Description....10
   - 4.1. ISO18000-6C Tag Reports Event....10
   - 4.2. ISO18000-6C Tag Reports Over Event....11
   - 4.3. ISO18000-6B Tag Reports Event....11
   - 4.4. ISO18000-6B Tag Reports Over Event....12
   - 4.5. GPI Triggers Start Event....13
   - 4.6. GPI Trigger Over Event....13
   - 4.7. TCP Listening Connection Event....14
   - 4.8. TCP Connection Disconnect Event....14
5. Messages Configuration and Query Description....15
   - 5.1. Send Synchronized Messages....15
6. Messages Description....16
   - 6.1. Reader Configuration and Management....16
     - 6.1.1. Reboot Reader....16
     - 6.1.2. Configure and Query COM Parameter....16
     - 6.1.3. Configure GPO State Parameter....17
     - 6.1.4. Query GPI State Parameter....17
     - 6.1.5. Configure and Query GPI Trigger Parameter....17
     - 6.1.6. Query Version for Software Baseband....18
     - 6.1.7. Query the information of the Reader Capabilities....19
     - 6.1.8. Query the Information of the Reader....19
     - 6.1.9. Buzzer control....20
   - 6.2. RFID Configuration and Operation....20
     - 6.2.1. Stop Command....20
     - 6.2.2. Configure and Query Power of the Reader....21
     - 6.2.3. Configure and Query the working frequency band of the reader....22
     - 6.2.4. Configure and Query EPC Baseband Parameter....22
     - 6.2.5. Configure and Query Tag for Uploading Parameter....23

6.2.6. Inventory ISO18000-6C Tag 23
6.2.7. Write ISO18000-6C Tag 24
6.2.8. Lock ISO18000-6C Tag 25
6.2.9. Kill ISO18000-6C Tag 25
6.2.10. Inventory ISO18000-6B Tag 26
6.2.11. Write ISO18000-6B Tag 26
6.2.12. Lock ISO18000-6B Tag 27
6.2.13. Locking Query ISO18000-6B Tag 27
7. Parameter Description 28
7.1.1. ISO18000-6C Tag Select Tag Parameter 28
7.1.2. ISO18000-6C Tag Read TID Parameter 28
7.1.3. ISO18000-6C Tag Read User Data Area Parameter 28
7.1.4. ISO18000-6B Tag Read User Data Area Parameter 29
8. Appendix 1 29

1. Preface

1.1. Overview

We provide the function that can run on Windows for the convenience of add-on development. This library is written by C++ language and zipped up to be a standard DII library.

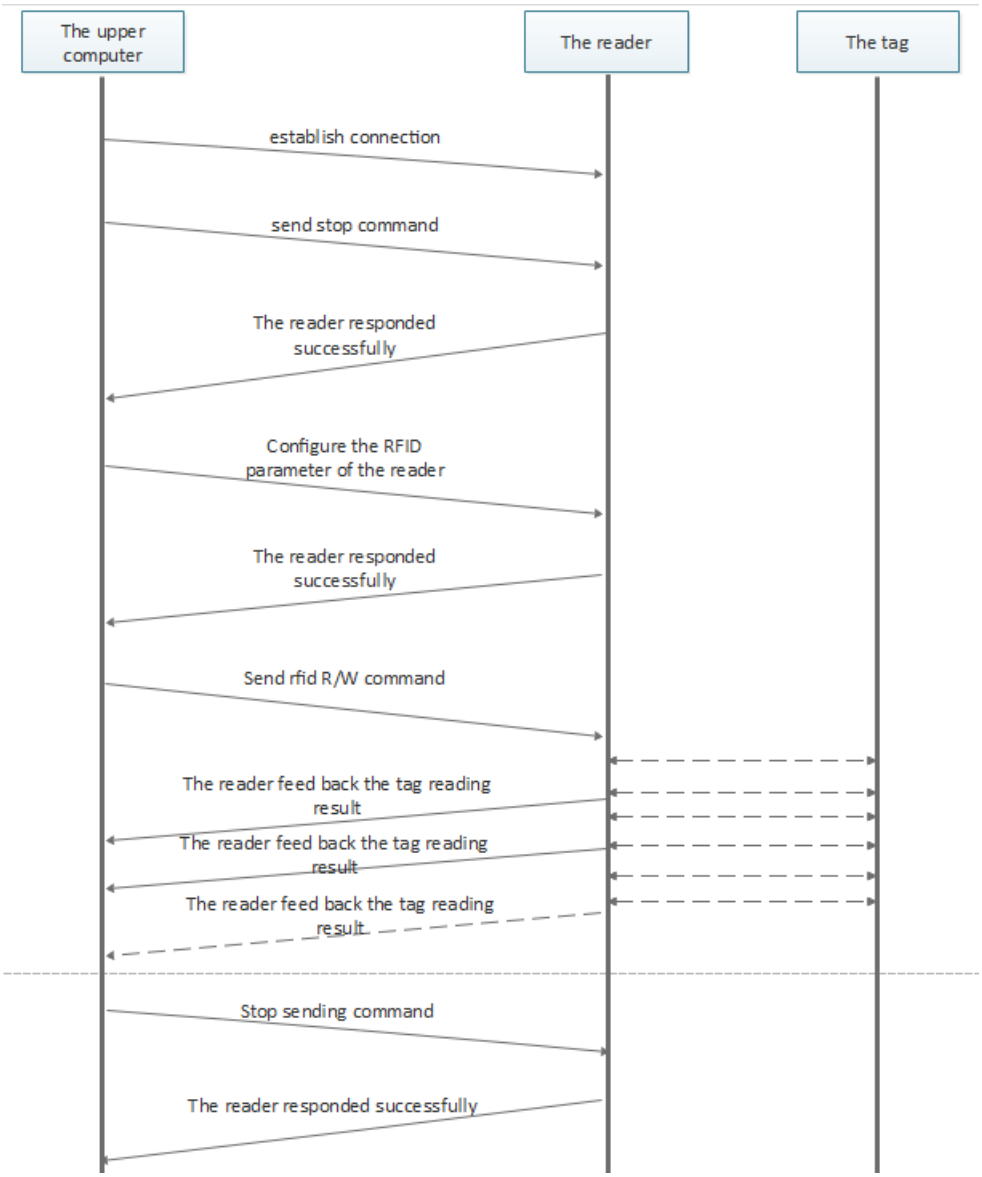
1.2. Applicable Devices

Function Module	Applicable Devices Type
Reader Configuration and Management	Identium 103
RFID Configuration and Operation	All UHF Device

1.3. Copyright Statements

© 2026 Identium Tech Solutions Pvt. Ltd. All rights reserved. This guide is provided to Identium customers for building applications with Identium readers. Support: info@identium.in

1.2. Basic Flow of Reader and Write



2. Quick Start

```
#include <stdio.h>
#include <stdlib.h>
#include "delegate.h"
#include "message.h"
#include "GClient.h"

#pragma comment(lib, "GReader")
```

```
GClient* gclient;
const char* initParam = "E5DC18EDBEEE6EAD954D7332E6D90361";

void __stdcall TagEpcLog(LogBaseEpcInfo msg)
{
    // Any blocking inside the callback will affect the normal use of the API!
    // Any blocking or low efficiency in the callback will affect the normal use of
the API!

    if (msg.Result == 0) {
        printf("EPC: %s\n", msg.Epc);
        printf("TID: %s\n", msg.Tid);
    }
}

void __stdcall TagEpcOver(void)
{
    printf("Epc log over. \n");
}

int main()
{
    gclient = CreateRS232(initParam, "COM12:115200", 3, InitInventory);

    if (gclient == NULL)
    {
        printf("failed to open connection. \n");
        return 0;
    }

    //Debug log
    gclient->isPrint = TRUE;

    RegCallBack(gclient, ETagEpcLog, (void*)TagEpcLog);
    RegCallBack(gclient, ETagEpcOver, (void*)TagEpcOver);

    //STOP COMMAND
    MsgBaseStop stop;
    memset(&stop, 0, sizeof(stop));
    SendSynMsg(gclient, EMESS_BaseStop, &stop);
    if (stop.rst.RtCode != 0) {
        printf("Failed to MsgBaseStop: %s. \n", stop.rst.RtMsg);
    }
    else
```

```
{
    printf("Successed to MsgBaseStop. \n");
}

//Inventory EPC COMMAND
MsgBaseInventoryEpc msg;
memset(&msg, 0, sizeof(msg));
msg.AntennaEnable = AntennaNo_1;
//msg.AntennaEnable = AntennaNo_1 | AntennaNo_2 | AntennaNo_3 | AntennaNo_4;
msg.InventoryMode = 1;

//msg.Filter.Area = 2; // TID
//msg.Filter.Start = 0;
//memcpy(msg.Filter.HexData, "E2801130200020190FFD019B", 25);
//msg.Filter.BitLen = strlen(msg.Filter.HexData) * 4;

msg.ReadTid.Mode = 0;
msg.ReadTid.Len = 6;

//msg.ReadUserdata.Start = 0;
//msg.ReadUserdata.len = 4;

SendSynMsg(gclient, EMESS_BaseInventoryEpc, &msg);
if (msg.rst.RtCode != 0)
{
    printf("Failed to MsgBaseInventoryEpc: %s. \n", msg.rst.RtMsg);
}
else {
    printf("Successed to MsgBaseInventoryEpc. \n");
}

Sleep(5000);

//STOP COMMAND
SendSynMsg(gclient, EMESS_BaseStop, &stop);
if (stop.rst.RtCode != 0) {
    printf("Failed to MsgBaseStop: %s. \n", stop.rst.RtMsg);
}
else
{
    printf("Successed to MsgBaseStop. \n");
}

Sleep(500);
```

```
    //CLOSE
    if (gclient!=NULL)
    {
        Close(gclient);
    }

    return 0;
}
```

3. Connection Description

3.1. RS232 Connection

Method	GClient* __stdcall CreateRS232(const char* initParam, const char* readerName, int timeout, ConnetionAttemptEventStatusType type)
Description	initParam：Initialization parameters readerName: connection string, such as ”COM1:115200” timeout: Connection confirmation timeout (seconds), such as “3” type：Connection status enum

3.2. RS485 Connection

Method	GClient* __stdcall CreateRS485(const char* initParam, const char* readerName, int timeout, ConnetionAttemptEventStatusType type)
Description	initParam：Initialization parameters readerName: connection string, such as ”COM1:115200:1” timeout: Connection confirmation timeout (seconds), such as “3” type：Connection status enum

3.3. TCP Client Connection

Method	GClient* __stdcall CreateTcpClient(const char* initParam, const char* readerName, int timeout, ConnetionAttemptEventStatusType type)
Description	initParam： Initialization parameters readerName: connection string, like ”192.168.1.168:8160”, default port 8160 timeout: connection confirmation timeout (seconds), such as “3” type： connection status enum

3.4. TCP Server Listening

Method	BOOL __stdcall CreateTcpServer(const char* initParam, short port, GServer* gserver, ConnetionAttemptEventStatusType type)
Description	initParam： Initialization parameters port: Listen on local port, such as 8160 gserver: Server-side structure, obtain the connected GClient in the GClientConnected callback type： connection status enum

3.5. USB-HID Connection

Method	GClient* __stdcall CreateUSBHID(const char* initParam, int timeout, ConnetionAttemptEventStatusType type)
Description	initParam： Initialization parameters timeout: connection confirmation timeout (seconds), such as “3” type： connection status enum

3.6. Close Connection

Method	BOOL __stdcall Close(GClient* client)

Description	Close and release the linked resources. Noted: For expired link objects, it need to actively call this method to release resources.

3.7. Close TCP Server Listening

Method	`BOOL __stdcall CloseGServer(GServer* gserver)`
Description	Close TCP Server listening

4. Events Description

4.1. ISO18000-6C Tag Reports Event

Event	`delegateTagEpcLog call_TagEpcLog`

Description	`void __stdcall TagEpcLog(char* readerName, LogBaseEpcInfo msg)`
	`{`
	`  if (msg.Result == 0) {}`
	`}`
	`RegCallBack(client, ETagEpcLog, (void*)TagEpcLog);`
	6C tag report events forwardly：When the reader is reading, the tag will report via this event。
	Examples are detailed in Quick Start
	LogBaseEpcInfo： detailed in “Reporting Object”

Reporting Object

Object	LogBaseEpcInfo

Attribute	Epc: Hexadecimal EPC character string Pc：PC Value AntId：Antenna No. Rssi：Signal strength Result：Tag reading result, 0 means success and non-zero value means failure Tid：Hexadecimal TID character string Userdata：Hexadecimal Userdata character string Reserved：Hexadecimal reserved area character string
Description	6C tag report parameter forwardly

4.2. ISO18000-6C Tag Reports Over Event

Event	delegateTagEpcOver call_TagEpcOver

Description	void __stdcall TagEpcOver(char* readerName, LogBaseEpcOver msg)
	{
	printf("TagEpcOver \n");
	}
	RegCallBack(s, ETagEpcOver, (void*)TagEpcOver);
	6C tag reports the over parameter forwardly to ensure the asynchronous messages are synchronized.

4.3. ISO18000-6B Tag Reports Event

Event	delegateTag6bLog call_Tag6bLog

Description	void __stdcall Tag6bLog(char* readerName, LogBase6bInfo msg)
	{
	if (msg.Result == 0) {}
	}
	RegCallBack(s, ETag6bLog, (void*)Tag6bLog);
	6B tag report events forwardly：When the reader is reading, the tag will report via this event.
	Examples are detailed in **"Quick Start"**.
	LogBase6bInfo：detailed in **"Reporting Object"**

Reporting Object

Object	LogBase6bInfo
Attribute	Tid：Hexadecimal TID Character string AntId：Antenna No. Rssi：Singal strength Result：Tag reading result, 0 means success and non-zero value means failure Userdata: Hexadecimal Userdata character string
Description	6b tag reports the end parameter forwardly.

4.4. ISO18000-6B Tag Reports Over Event

Event	delegateTag6bOver call_Tag6bOver

Description	void __stdcall Tag6bOver(char* readerName, LogBase6bOver msg)
	{
	printf("Tag6BOver \n");
	}
	RegCallBack(s, ETag6bOver, (void*)Tag6bOver);
	6B tag reports the over parameter forwardly to ensure the asynchronous messages are synchronized.

4.5. GPI Triggers Start Event

Event	delegateGpiStart call_GpiStart

Description	void __stdcall GpiStart(char* readerName, LogAppGpiStart msg)
	{
	printf("Gpi Start \n");
	}
	RegCallBack(s, EGpiStart, (void*)GpiStart);
	When the trigger start condition is met, the reader will actively upload a notification message to notify the upper computer that the trigger operation has started.

Reporting Object

Object	LogAppGpiStart
Attribute	GpiPort：GPI port No.(The index starts from) Level：GPI Port Level (electrical level state,0 means low electrical level,1 means high electrical level) TriggerTime：Current System Trigger time
Description	GPI triggers start and report events.

4.6. GPI Trigger Over Event

Event	delegateGpiOver call_GpiOver

Description	void __stdcall GpiOver(char* readerName, LogAppGpiOver msg)
	{
	printf("Gpi Over \n");
	}
	RegCallBack(s, EGpiOver, (void*)GpiOver);
	When the trigger stop condition is met, the reader will actively upload a notification message to notify the upper computer that the trigger operation has started.

Reporting Object

Object	LogAppGpiOver
Attribute	GpiPort：GPI port No.(The index starts from) Level：GPI Port Level (electrical level state,0 means low electrical level,1 means high electrical level) TriggerTime：Current Trigger time
Description	GPI trigger over and report the parameter forwardly

4.7. TCP Listening Connection Event

Event	`typedef void(*delegateGClientConnected)(char* readerName, GClient* client);`

Description	`void GClientConnected(char* readerName, GClient* client)`
	`{`
	
	`}`
	`RegGServerCallBack(gserver, GClientConnected);`
	Description：
	➢ TCP listens successfully, this event is reported when the remote connection is successful.
	➢ After the event is reported, the upper computer (caller) needs to receive and store the GClient to communicate with the module.

4.8. TCP Connection Disconnect Event

Event	`typedef void(__stdcall* delegateTcpDisconnected)(char* readerName);`

Description	`void __stdcall TcpDisconnected(char* readerName)`
	`{`
	`  printf("Disconnected \n");`
	`}`
	`RegCallBack(s, ETcpDisconnected, (void*)TcpDisconnected);`
	Description:
	➢ The connection is under TCP. When the remote connection is actively disconnected or the physical layer is abnormal, the event will be reported.
	➢ After the reporting of the events, the upper computer(the caller) need to release connection object, or the event reporting will loops until the connection object is released.
	➢ It is decided by the upper computer(the caller) itself that whether the remote device should be re-connected or not in order to meet different requests.

5. Messages Configuration and Query Description

5.1. Send Synchronized Messages

Method	`void __stdcall SendSynMsg(GClient * s, MESSAGE type, void* msg)`
Parameter	s: Connected Object type: Message Type Msg: Message
Instruction	msg.rst.RtCode: message return code, 0 means success, and non-zero value means failure. Msg.rst.RtMsg: The reason of the failed operation Send synchronized messages, detailed in **Code Example** Tips: "Reader configuration and management", "RFID configuration and operation" and other message are sent with this method.

Code Example

```
 // stop command,idle state
MsgBaseStop stop;
memset(&stop, 0, sizeof(stop));//memset Clear a structure type variable or database
SendSynMsg(s, EMESS_BaseStop, &stop);
if (stop.rst.RtCode != 0) {
    printf("failed to MsgBaseStop: %s \n", stop.rst.RtMsg);
}
else
{
    printf("Success to MsgBaseStop. \n");
```

```
}
```

6. Messages Description

6.1. Reader Configuration and Management

6.1.1. Reboot Reader

Category	EMESS_AppReboot
Message	MsgAppReboot
Description	Normally the restart message of the device will be executed after the modification of the configuration that need to come in to effect after reboot.

6.1.2. Configure and Query COM Parameter

Message category	EMESS_AppSetSerialParam
Configuration messages	MsgAppSetSerialParam
Message category	EMESS_AppGetSerialParam
Configuration messages	MsgAppGetSerialParam
Attribute	BaudrateIndex: Baud rate index（0，9600 bps； 1，19200 bps； 2，115200 bps； 3，230400 bps; 4， 460800bps）
Description	(Persistent configuration, which means the information will be saved when it is powered off) configure the COM parameter of the device.

6.1.3. Configure GPO State Parameter

Namespace	EMESS_AppSetGpo
Configuration Class	MsgAppSetGpo
Attribute	Gpo1: 0 （low，relay disconnected） 1 （high，relay closed） Gpo2: 0 （low，relay disconnected） 1 （high，relay closed） Gpo3: 0 （low，relay disconnected） 1 （high，relay closed） Gpo4: 0 （low，relay disconnected） 1 （high，relay closed） ……
Description	(Persistent configuration, which means the information will be saved when it's powered off) Configure the device GPO parameter。 Note: For GPO that do not require state control, no assignment is needed.

6.1.4. Query GPI State Parameter

Namespace	EMESS_AppGetGpiState
Query Class	MsgAppGetGpiState
Attribute	Gpi1: 0 Low， 1 High Gpi2: 0 Low， 1 High Gpi3: 0 Low， 1 High Gpi4: 0 Low， 1 High
Description	Query the GPI status of the device. Notes: The index number starts from 1.

6.1.5. Configure and Query GPI Trigger Parameter

Namespace	EMESS_AppSetGpiTrigger

Configuration Class	MsgAppSetGpiTrigger
Message	EMESS_AppGetGpiTrigger
Query Class	MsgAppGetGpiTrigger
Attribute	GpiPort: GPI port No, the index starts from 0 TriggerStart: Trigger start（0 trigger close， 1 low electrical trigger， 2 high electrical trigger， 3 rising edge trigger， 4 falling edge trigger， 5 random edge trigger） TriggerCommand: trigger binding command(Hex, can be null) TriggerOver: trigger stop（0 non-stop，1 low electrical trigger，2 high electrical trigger，3 rising edge trigger， 4 falling edge trigger， 5random edge trigger， 6 delayed stop） OverDelayTime: delayed stop time(take effect only if the stop condition is “delayed stop”) LevelUploadSwitch: uploading switch of the IO electrical level changes when triggering non-stop(0 do not upload,1 upload)
Description	(Persistent configuration, which means the information will be saved when it is powered off configure the device GPI trigger parameter) Notes：This configuration needs to be modified when the device is idle(that is, the configuration can be changed under loop reading)

6.1.6. Query Version for Software Baseband

Namespace	EMESS_AppGetBaseVersion
Query Class	MsgAppGetBaseVersion
Attribute	BaseVersion: Baseband software version
Description	Used to obtain the version number of the baseband software.

6.1.7. Query the information of the Reader Capabilities

Namespace	EMESS_BaseGetCapabilities
Query Class	MsgBaseGetCapabilities
Attribute	MaxPower: Maximum Supported Power MinPower： Minimum Supported Power AntennaCount： Antenna Quantity FrequencyArray： Support Frequency 0， National Standard 920~925MHz 1， National Standard 840~845MHz 2， National Standard 840~845MHz and 920~925MHz 3， FCC， 902~928MHz 4， ETSI， 866~868MHz 5， JP, 916.8~920.4 MHz 6， TW, 922.25~927.75 MHz 7， ID, 923.125~925.125 MHz 8， RUS, 866.6~867.4 MHz 9， TEST 802.75~998.75MHz, ProtocolArray： List of supported protocols 0， ISO18000-6C/EPC C1G2 1， ISO18000-6B 2， National Standard GB/T 29768-2013
Description	N/A

6.1.8. Query the Information of the Reader

Namespace	EMESS_AppGetReaderInfo
Query Class	MsgAppGetReaderInfo

Attribute	SerialNum: Reader Serial No. PowerOnTime： Power-on time BaseBuildDate：baseband compiling time AppVersion：application software version（such as：“0.1.0.0”） AppBuildDate：application compiling time SystemVersion：version of the operation system
Description	N/A

6.1.9. Buzzer control

Namespace	EMESS_AppSetBuzzerCtrl
Query Class	MsgAppSetBuzzerCtrl
Attribute	Status: 0，stop；1，ring Mode: 0，ring once；1，always ring
Description	N/A

6.2. RFID Configuration and Operation

6.2.1. Stop Command

Namespace	EMESS_BaseStop
Category	MsgBaseStop
Attribute	N/A

Description	Stop all RFID operations of the reader and make the reader idle Tips: When the reader is reading, all configuration messages will not be able to send, a stop command must be sent.

6.2.2. Configure and Query Power of the Reader

Namespace	EMESS_BaseSetPower
Configuration Class	MsgBaseSetPower
Category	EMESS_BaseGetPower
Query Class	MsgBaseGetPower

Attribute	DicPower: the corresponding antenna power of the reader
	`typedef struct {`
	`  unsigned char AntennaNo;//Antenna No.`
	`  unsigned char Power;// Power`
	`}Dictionary;`
	DicCount：Number of valid values for DicPower
	setReadOrWriteFalg：Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid)
	ReadOrWrite：（Optional）
	Configuration class: 0, configure both read and write power (default); 1, configure only read power; 2, configure only write power;
	Query Class: 0, query the read power; 1, query the write power; the default is to query the read and write power;
	setPowerDownSaveFlag：Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid)
	PowerDownSave：Parameter persistence (0, not saved when power off; 1, saved when power off). If this parameter is not included, it will be saved when power off by default.（Optional）

Description	Used to configure the current power of the reader.

6.2.3. Configure and Query the working frequency band of the reader

Namespace	EMESS_BaseSetFreqRange
Configuration Class	MsgBaseSetFreqRange
Category	EMESS_BaseGetFreqRange
Query Class	MsgBaseGetFreqRange
Attribute	FreqRangeIndex: Frequency band index, specific correspondence, see **Error! Reference source not found.**
Description	(Permanent configuration, saved when power off) Used to configure the current working frequency band of the reader.

6.2.4. Configure and Query EPC Baseband Parameter

Namespace	EMESS_BaseSetBaseband
Configuration Class	MsgBaseSetBaseband
Category	EMESS_BaseGetBaseband
Query Class	MsgBaseGetBaseband
Attribute	setBaseSpeedFlag： Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid) BaseSpeed: EPC baseband speed（Optional） setQValueFlag： Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid) QValue: Defaule Q Value（Optional）(0~15) setSessionFlag： Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid) Session:（Optional）(0,Session0; 1,Session1; 2,Session2; 3,Session3) setInventoryFlag： Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid) InventoryFlag: Inventory flag parameter (optional) (0, only use Flag A for inventory; 1, only use Flag B for inventory; 2, use Flag A and Flag B in turn).

Description	(Persistent configuration, saved when power off) Used to configure the baseband parameters used by the reader/writer.

6.2.5. Configure and Query Tag for Uploading Parameter

Namespace	EMESS_BaseSetTagLog
Configuration Class	MsgBaseSetTagLog
Category	EMESS_BaseGetTagLog
Query Class	MsgBaseGetTagLog
Attribute	setRepeatedTimeFlag：Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid) RepeatedTime: Repeat tag filtering time (optional) (indicates that within a card reading instruction execution cycle, the same tag content will only be uploaded once within the specified repeated filtering time, 0~65535, time unit: 10ms)。 setRssiTVFlag：Optional parameter identifier (true: optional parameters are valid; false: optional parameters are invalid) RssiTV: RSSI threshold（Optional）（When the tag RSSI value is lower than the threshold, the tag data will not be uploaded and discarded.）
Description	(Persistent configuration, saved when power off) Used to configure reader upload parameters.

6.2.6. Inventory ISO18000-6C Tag

Category	EMESS_BaseInventoryEpc
Information	MsgBaseInventoryEpc

Attribute	AntennaEnable: Antenna Port(using antenna enumeration is detailed in Quick Start) InventoryMode: Continuous/Single Read (0: single read，the reader read one time on each enacling antenna then stop reading and automatically enter idle state; 1: continuous read，the reader keeps reading and stop reading after receiving a stop command..) Filter: select reading parameter（optional）（detailed in parameter description） ReadTid: TID read parameter（optional）（detailed in parameter description） ReadUserdata: user data area read parameter（optional）（detailed in parameter description） ReadReserved: reserved area read parameter（optional）（detailed in parameter description） StrHexPassword: access password（optional）
Description	Using for configuring the reader parameter for tag reading and start reading.Any reading operation for the tag data require the EPC code of the tag, so a EPC code can be acquired through any reading operation.

6.2.7. Write ISO18000-6C Tag

Namespace	EMESS_BaseWriteEpc
Category	MsgBaseWriteEpc
Attribute	AntennaEnable: Antenna Port Area：tag data area to be written(0，reserved area；1，EPC area；2，TID area；3，user data area) Start：word initial address of the tag data area to be writted StrHexWriteData：data content to be written（optional）(hex) Filter: select reading parameter（optional）（detailed in parameter description） StrHexPassword: Access Password（optional）
Description	➢ Reader writes EPC Tags, the writing that this command refined should be single operation. ➢ ISO18000-6C The protocol specifies that the minimum data unit for R/W operations is a word. ➢ EPC are include CRC-16(the zeroth word) + PC(the first word) + EPC： CRC16: the zeroth word, not writable.

Write results	0， Write Success 1， Antenna port parameter error 2， Choice parameter error 3， Write parameter error 4， CRC checksum error 5， Insufficient power 6， Data area overflow 7， Data area is locked 8， Wrong access password 9， Other Tag errors 10， Loss the Tag 11， Reader sends command error

6.2.8. Lock ISO18000-6C Tag

Namespace	EMESS_BaseLockEpc
Category	MsgBaseLockEpc
Attribute	AntennaEnable: Antenna Port Area：tag data area to be written(0，reserved area；1，EPC area；2，TID area；3，user data area) Mode：Lock operation type (0, unlock; 1, lock; 2, permanently unlock; 3, permanently locked) Filter: select reading parameter（optional）（detailed in parameter description） StrHexPassword: Access Password（optional）
Description	Lock or unlock the tag. The operation defined by this command is a single operation.

6.2.9. Kill ISO18000-6C Tag

Namespace	EMESS_BaseDestroyEpc
Category	MsgBaseDestoryEpc

Attribute	AntennaEnable: antenna Port StrHexPassword: destroy passwords Filter: Select read parameters（optional）（detailed in parameter description）
Description	Kill the ISO18000-6C tag. The tag inactivated will be in permanent failure and this operation is irreversible. The operation this command defined is single.

6.2.10. Inventory ISO18000-6B Tag

Namespace	EMESS_BaseInventory6b
Category	MsgBaseInventory6b
Attribute	AntennaEnable: antenna Port InventoryMode: Continuous/single reading (0: Single reading mode, the reader only performs one card reading operation on each enabled antenna, then ends the card reading operation and automatically enters the idle state; 1: Continuous reading mode, The reader continues to read the card until the reader receives the stop command and ends reading the card) Area: Read content (0, read only 6B TID; 1, read 6B TID + user data; 2, read only user data) ReadUserdata: User data area reading parameters (optional) (see parameter description for details) StrHexMatchTid：TID code of the 6B tag to be matched (optional) (hex)
Description	Used for the data reading for ISO18000-6B tags

6.2.11. Write ISO18000-6B Tag

Namespace	EMESS_BaseWrite6b
Category	MsgBaseWrite6b
Attribute	AntennaEnable: Antenna Port StrHexMatchTid：TID code of the 6B tag to be matched (hex) Start：word initial address of the tag data area to be written StrHexWriteData：Data content to be written (hex)

Description	Write 6B tag, The writing operation this command defined is single.

6.2.12. Lock ISO18000-6B Tag

Namespace	EMESS_BaseLock6b
Category	MsgBaseLock6b
Attribute	AntennaEnable: Antenna Port StrHexMatchTid： The TID code of the 6B tag to be matched (hex) Address： The byte address of the data to be locked
Description	Lock 6B tag. The operation is irrevocable and reversible. The locking operation this command defined is single.

6.2.13. Locking Query ISO18000-6B Tag

Namespace	EMESS_BaseLockGet6b
Category	MsgBaseLockGet6B
Attribute	AntennaEnable: Antenna Port StrHexMatchTid： The TID code of the 6B tag to be matched (hex) Address： The byte address of the data to be locked
Description	Query the locking state for the 6B tag. The query operation this command defined is single.

7. Parameter Description

7.1.1. ISO18000-6C Tag Select Tag Parameter

Category	ParamFilter
Attribute	Area: data area to be matched (1，EPC area；2，TID area；3，user data area) BitStart: starting bit address of the matching data BitLen: the data bit length to be matched HexData: the data content to be matched(hex)
Description	Optional parameter (The start bit address when matching EPCs is usually 32)

7.1.2. ISO18000-6C Tag Read TID Parameter

Category	ParamEpcReadTid
Attribute	Mode：TID reading mode configuration，(0，TID reading length is self-adapted，but the max length should not be longer than the length defined by byte 1；1，read TID according to the length defined by byte 1) Len：The reader need to read the length of the word(word，16bits，similarly hereinafter)of TID data.
Description	Optional parameter

7.1.3. ISO18000-6C Tag Read User Data Area Parameter

Category	ParamEpcReadUserdata
Attribute	Start：starting word address Len：The word length of user data that need to be read.

Description	Optional parameter

7.1.4. ISO18000-6B Tag Read User Data Area Parameter

Category	Param6bReadUserdata
Attribute	Start：Starting byte address for user data Len： The byte length of user data
Description	optional parameter

8. Appendix 1

List of the frequency band the Readers Supported

Index	Description
0	National standard 920~925MHz
1	National standard 840~845MHz
2	National standard 840~845MHz and 920~925MHz
3	FCC，902~928MHz
4	ETSI，866~868MHz
5	JP, 916.8~920.4 MHz
6	TW, 922.25~927.75 MHz
7	ID, 923.125~925.125 MHz
8	RUS, 866.6~867.4 MHz
9	TEST 802.75~998.75MHz