# RFID Reader Development Guide For C#

**Identium Tech Solutions Pvt. Ltd.**

**V0.4.0.0**

# Contents


1. Preface ..... 4
   1.1. Overview ..... 4
   1.2. Applicable Devices ..... 4
   1.3. Copyright Statements ..... 4
   1.4. Basic Flow of Read 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 Monitoring ..... 9
4. Events Description ..... 10
   4.1. ISO18000-6C Tag Reports Events ..... 10
   4.2. ISO18000-6CTag Reports End Events ..... 11
   4.3. ISO18000-6BTag Reports End Events ..... 12
   4.4. ISO18000-6B Tag Reports End Events. ..... 12
   4.5. National Standard Tag Reports Events ..... 13
   4.6. National Standard Tag Report End Events ..... 14
   4.7. GPI Triggers Start Events ..... 15
   4.8. GPI Trigger End Events ..... 15
   4.9. TCP Connection Disconnect Events ..... 16
   4.10. TCP Connection Events ..... 16
5. Messages Configuration and Query Description ..... 17
   5.1. Send Synchronized Messages ..... 17
6. Messages Description ..... 19
   6.1. Reader Configuration and Management ..... 19
      6.1.1. Restart the Reader ..... 19
      6.1.2. Configuration and Query COM Parameter ..... 20
      6.1.3. Configure GPO State Parameter ..... 20
      6.1.4. Query GPI State Parameter ..... 21

6.1.5. Config and Query GPI Trigger Parameter ..... 21
6.1.6. Query Version for Software Baseband ..... 22
6.1.7. Query the Information of the Reader ..... 22
6.1.8. Query RFID ability of the Reader ..... 23
6.2. RFID Configuration and Operation ..... 23
6.2.1. Stop Command ..... 23
6.2.2. Configure and Query Power of the Reader ..... 24
6.2.3. Configure and Query Working frequency band of the Reader ..... 24
6.2.4. Config and Query EPC Baseband Parameter ..... 25
6.2.5. Configure and Query Tag for Uploading Parameter ..... 25
6.2.6. Read EPC Tag ..... 26
6.2.7. Write EPC Tag ..... 27
6.2.8. Lock EPC Tag ..... 27
6.2.9. Inactivate EPC Tag ..... 28
6.2.10. Read 6B Tag ..... 28
6.2.11. Write 6B Tag ..... 28
6.2.12. Lock 6B Tag ..... 29
6.2.13. 6B Tag Locking Query ..... 29
6.2.14. Read GB Tag ..... 30
6.2.15. Write GB Tag ..... 30
6.2.16. Lock GB Tag ..... 31
6.2.17. Inactivate GB Tag ..... 31
7. Parameter Description ..... 32
7.1. 6C/GB Tag Select Tag Parameter ..... 32
7.2. 6C Tag Read TID Parameter ..... 32
7.3. 6C Tag Read User Data Area Parameter ..... 33
7.4. Parameter of User Data Area for 6B Tag Reading ..... 33
7.5. Parameter for User Data Area for GB Tag Reading ..... 33
8. Appendix 1 ..... 34
9. Appendix 2 ..... 34

# 1. Preface

## 1.1. Overview

We provide the function library that work on .net for the convenience of add-on development. This library is written in C# language and packaged as standard DLL library “GReaderApi.dll”, and it support .net framework 2.0 and above versions.

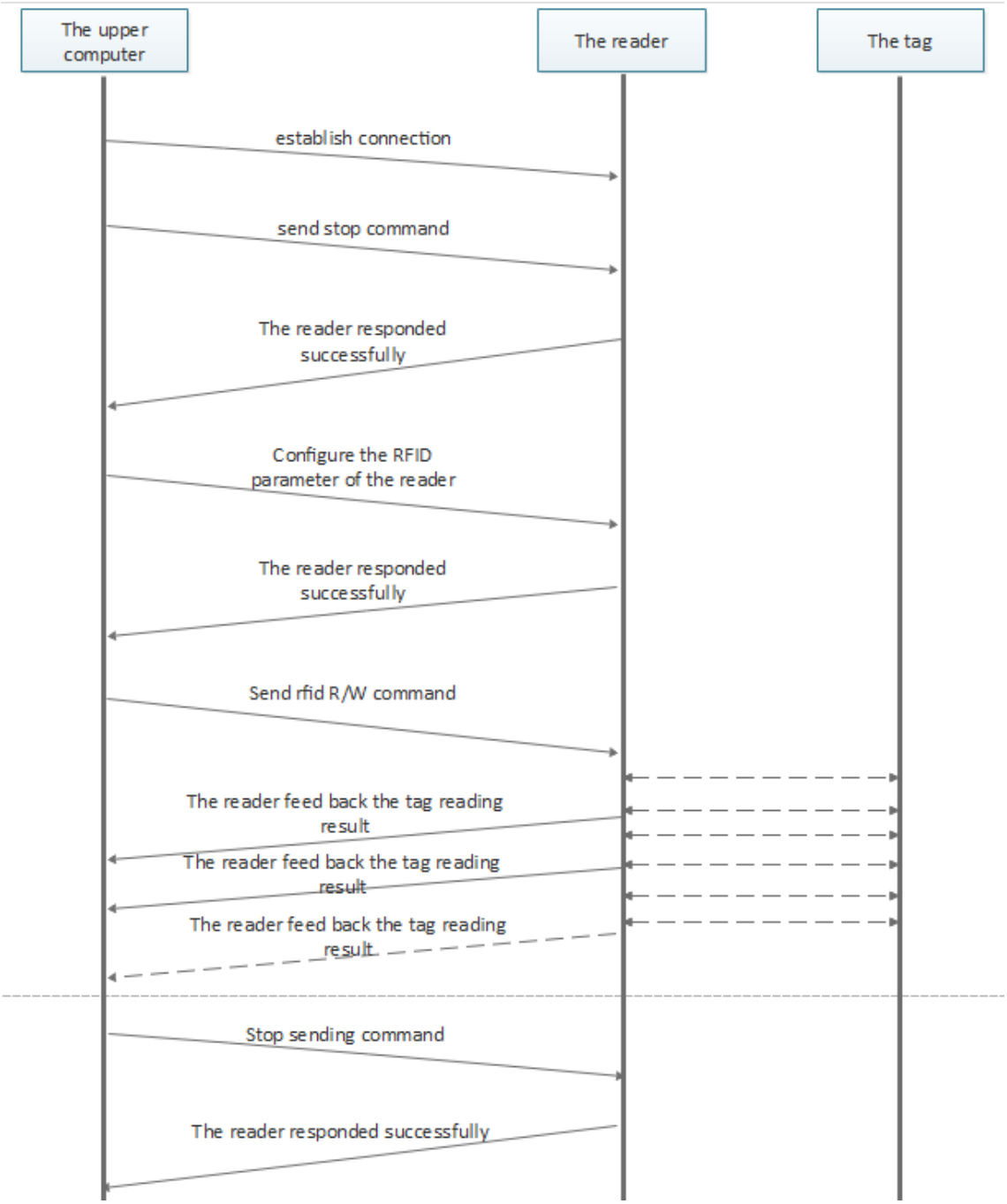
## 1.2. Applicable Devices

| Function Mold | Applicable Devices Type |
|---|---|
| Reader Configure and Management | **Identium 103** |
| RFID configure and operation | **All UHF device model** |

## 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

## 1.4. Basic Flow of Read and Write




# 2. Quick Start

```
using GDotnet.Reader.Api.DAL;
using GDotnet.Reader.Api.Protocol.Gx;
using System;
using System.Collections.Generic;
```

```
using System.Text;

// ============================================================
//     Copyright (C) 2026 Identium Tech Solutions Pvt. Ltd.
//
//     Identium 103 SDK sample code.
//
//     Support: info@identium.in
// ============================================================
namespace GDotnet.Reader.Api
{
    static class Example
    {
        static void Main()
        {
            GClient clientConn = new GClient();
            eConnectionAttemptEvent stateType    state;
            // clientConn.OpenTcp("192.168.1.168:6180", 3000, out    state)
            if (clientConn.OpenSerial("COM16:115200", 3000, out    state))
            {
                // subscribe tag reporting events
                clientConn.OnEncapedTagEpcLog += new delegateEncapedTagEpcLog(OnEncapedTagEpcLog);
                clientConn.OnEncapedTagEpcOver += new delegateEncapedTagEpcOver(OnEncapedTagEpcOver);

                // stop command,    idle state
                MsgBaseStop msgBaseStop = new MsgBaseStop();
                clientConn.SendSynMsg(msgBaseStop);
                if (0 == msgBaseStop.RtCode)
                {
                    Console.WriteLine("Stop successful.");
                }
                else { Console.WriteLine("Stop error."); }

                // Power configuration, set the power of the 4 antennas as 30dBm
                MsgBaseSetPower msgBaseSetPower = new MsgBaseSetPower();
                msgBaseSetPower.DicPower = new Dictionary<byte, byte>()
                {
                    {1, 30},
                    {2, 30},
                    {3, 30},
                    {4, 30}
                };
                clientConn.SendSynMsg(msgBaseSetPower);
                if (0 == msgBaseSetPower.RtCode)
```

```
                {
                    Console.WriteLine("Power configurationuration successful.");
                }
                else { Console.WriteLine("Power configurationuration error."); }
                Console.WriteLine("Enter any character to start reading the tag.");
                Console.ReadKey();

                // 4 antennas read, read the EPC data are and TID data area
MsgBaseInventoryEpc msgBaseInventoryEpc = new MsgBaseInventoryEpc();
                msgBaseInventoryEpc.AntennaEnable = (ushort)(eAntennaNo._1 | eAntennaNo._2 |
eAntennaNo._3 | eAntennaNo._4);
                msgBaseInventoryEpc.InventoryMode = (byte)eInventoryMode.Inventory;
                msgBaseInventoryEpc.ReadTid = new ParamEpcReadTid();                    // tid parameter
                msgBaseInventoryEpc.ReadTid.Mode = (byte)eParamTidMode.Auto;
                msgBaseInventoryEpc.ReadTid.Len = 6;
                clientConn.SendSynMsg(msgBaseInventoryEpc);
                if (0 == msgBaseInventoryEpc.RtCode)
                {
                    Console.WriteLine("Inventory epc successful.");
                }
                else { Console.WriteLine("Inventory epc error."); }
                Console.ReadKey();

                // stop reading,    idle state
                clientConn.SendSynMsg(msgBaseStop);
                if (0 == msgBaseStop.RtCode)
                {
                    Console.WriteLine("Stop successful.");
                }
                else { Console.WriteLine("Stop error."); }
            }
            else
            {
                Console.WriteLine("Connect failure.");
            }
            Console.ReadKey();
        }

        #region API events Events

        public static void OnEncapedTagEpcLog(EncapedLogBaseEpcInfo msg)
        {
            // The use of API will be influenced    if there is blockage in callback
            // There are many tag callbacks. Please cache the tag data first and then do business processing.
```

```
                if (null != msg)
                {
                    Console.WriteLine(msg.logBaseEpcInfo.ToString());
                }
            }

            public static void OnEncapedTagEpcOver(EncapedLogBaseEpcOver msg)
            {
                if (null != msg)
                {
                    Console.WriteLine("Epc log over.");
                }
            }

            #endregion
        }
    }
```

# 3. Connection Description

## 3.1. RS232 Connection

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GClient |
| **Method** | **public** bool OpenSerial**(**String readerName**,** int timeout**, out** eConnectionAttemptEvent stateType state**)** |
| **Description** | readerName: connection string, such as"COM1:115200"("COM No.:baud rate")<br>timeout: connection time is confirmed exceeded limits(ms), like "1000"<br>state: Connection state enumeration |

## 3.2. RS485 Connection

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |

| Category | GClient |
| --- | --- |
| Method | **public** bool OpenSerial485**(**String readerName**,** int timeout**, out** eConnectionAttemptEvent stateType state**)** |
| Description | readerName: connection string, such as ”COM1:115200:1”(“COM No: baud rate:485 address ”)<br>timeout: connection time is confirmed exceeded limits like “1000”(millisecond)<br>state: Connection state enumeration<br>Occasional failure is normal in half-duplex, please try again. |

## 3.3. TCP Client Connection

| Namespace | GDotnet.Reader.Api.DAL |
| --- | --- |
| Category | GClient |
| Method | **public** bool OpenTcp**(**String readerName**,** int timeout**, out** eConnectionAttemptEvent stateType state**)** |
| Description | readerName: connection string (such as “192.168.1.168:6180”). Reader default IP ”192.168.1.168”, default port “6180”<br>timeout: connection time is confirmed exceeded limits like “1000”(millisecond)<br>state: Connection state enumeration |

## 3.4. TCP Server Monitoring

| Namespace | GDotnet.Reader.Api.DAL |
| --- | --- |
| Category | GServer |
| Method | **public** bool Open**(**int port**)** |
| Description | port: Local interface monitored by upper computer<br>UHF reader need to be configured to be "client mode" when monitored with this method .<br>The configuration method of "client mode" is detailed in the "RFID Demonstration Software Operating Manual". |

# 4. Events Description

## 4.1. ISO18000-6C Tag Reports Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GClient |
| **Events** | **public** delegateTagEpcLog OnTagEpcLog**;** |
| **Description** | **public delegate** void delegateTagEpcLog**(**LogBaseEpcInfo msg**);**<br>6C tag reports events forwardly: when the reader is reading, the tag will report via this event. Examples are detailed in Quick Start.<br>LogBaseEpcInfo: detailed in Reporting Object |

Reporting Object

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| Object | LogBaseEpcInfo |

| | |
|---|---|
| **Attribute** | Epc: Hexadecimal EPC character string<br>BEpc: EPC byte array<br>Pc: PC value<br>AntId: Antenna No.<br>Rssi: Signal strength<br>Result: Tag reading result, 0 means success and non-zero value means failure<br>1,Tag not responding<br>2,CRC error<br>3,Data area locked<br>4,Data area overflow<br>5,Access password error<br>6,Other tags errors<br>7,Other reader errors<br>Tid: Hexadecimal TID character string<br>BTid: TID byte array<br>Userdata: Hexadecimal Userdata character string<br>BUser: Userdata byte array<br>Reserved: Hexadecimal reserved area character string<br>BRes: reserved area byte array |
| **Description** | 6C report the parameter forwardly |

## 4.2. ISO18000-6CTag Reports End Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | GClient |
| **Attribute** | **public** delegateTagEpcOver OnTagEpcOver**;** |
| **Description** | **public delegate** void delegateTagEpcOver**(**LogBaseEpcOver msg**);**<br>6C tag reports the end parameter forwardly to ensure the asynchronous messages are synchronized. |

## 4.3. ISO18000-6BTag Reports End Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GClient |
| **Events** | **public** delegateTag6bLog OnTag6bLog**;** |
| **Description** | **public delegate** void delegateTag6bLog**(**LogBase6bInfo msg**);**<br>6B tag reports the events forwardly. When the reader is reading, the tag will report via this event.<br>The example is detailed Quick Start.<br>LogBase6bInfo:detailed in Reporting Object |

**Reporting Object**

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Object** | LogBase6bInfo |
| **Attribute** | AntId: Antenna No.<br>Rssi: Signal strength<br>Result: Tag reading result, 0 means success and non-zero value means failure<br>1,Tag not responding<br>2,CRC error<br>3,Other reader errors<br>Tid: Hexadecimal TID character string<br>BTid: TID byte array<br>Userdata: Hexadecimal Userdata character string<br>BUser: Userdata byte array |
| **Description** | 6C tag reports forwardly. |

## 4.4. ISO18000-6B Tag Reports End Events.

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |

| Category | GClient |
| --- | --- |
| Attribute | **public** delegateTag6bOver OnTag6bOver**;** |
| Description | **public delegate** void delegateTag6bOver**(**LogBase6bOver msg**);**<br>6B tag reports end parameter forwardly to ensure the asynchronous messages are synchronized. |

## 4.5. National Standard Tag Reports Events

| Namespace | GDotnet.Reader.Api.DAL |
| --- | --- |
| Category | GClient |
| Events | **public** delegateTagGbLog OnTagGbLog**;** |
| Description | **public delegate** void delegateTagGbLog**(**LogBaseGbInfo msg**);**<br>GB tag reports the events forwardly. When the reader is reading, the tag will report via this event.<br>The example is detailed Quick Start.<br>LogBaseGbInfo: Detailed in Reporting Object |

Reporting Object

| Namespace | GDotnet.Reader.Api.Protocol.Gx |
| --- | --- |
| Object | LogBaseGxInfo |

| | |
|---|---|
| **Attribute** | Epc: Hexadecimal EPC character String (tag coding area)<br>BEpc: EPC byte array(tag coding are)<br>Pc: PC value<br>AntId: Antenna No.<br>Rssi: signal strength<br>Result: Tag reading result, 0 means success and non-zero value means failure<br>1,Tag not responding<br>2,CRC error<br>3,Data area locked<br>4,Data area overflow<br>5,Password reading error<br>6,Permission denied<br>7,Identification failed<br>8,Other tag errors<br>9,Other reader errors<br>Tid: Hexadecimal TID character string (tag information area)<br>16 BTid: TID byte array (tag information area)<br>Userdata: Hexadecimal Userdata character string<br>BUser: Userdata byte array |
| **Description** | 6C Tag reports the parameter forwardly. |

## 4.6. National Standard Tag Report End Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | GClient |
| **Attribute** | **public** delegateTagGbOver OnTagGbOver**;** |
| **Description** | **public delegate** void delegateTagGbOver**(**LogBaseGbOver msg**);**<br>GB tag reports the end parameters forwardly to ensure the asynchronous messages are synchronized. |

## 4.7. GPI Triggers Start Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GClient |
| **Events** | **public** delegateGpiStart OnGpiStart**;** |
| **Description** | **public delegate** void delegateGpiStart**(**LogBaseGpiStart msg**);**<br>GPI triggers start and reports events forwardly: The event will be reported when the GPI meet the start configuration.<br>LogBaseGpiStart: detailed in Reporting Object |

**Reporting Object**

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Object** | LogBaseGpiStart |
| **Attribute** | GpiPort: GPI port No. (The index starts from )<br>Level: electrical level state,0 means low electrical level , 1 means high electrical level<br>TriggerTime: Trigger time |
| **Description** | GPI triggers start and report events. |

## 4.8. GPI Trigger End Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GClient |
| **Events** | **public** delegateGpiOver OnGpiOver**;** |
| **Description** | **public delegate** void delegateGpiOver**(**LogBaseGpiOver msg**);**<br>GPI triggers start and report events forwardly :The event will be reported when the GPI meet the start configuration.<br>LogBaseGpiOver: Detailed in **Reporting Object** |

**Reporting Object**

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Object** | LogBaseGpiOver |
| **Attribute** | GpiPort: GPI port No. (The index starts from )<br>Level: electrical level state,0 means low electrical level , 1 means high electrical level<br>TriggerTime: Trigger time |
| **Description** | GPI trigger end and report the parameter forwardly |

## 4.9. TCP Connection Disconnect Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GClient |
| **Events** | **public** delegateTcpDisconnected OnTcpDisconnected**;** |
| **Description** | **public delegate** void delegateTcpDisconnected**(**String readerName**);**<br>**Description**:<br>➢ The connection is under TCP. When the remote connection is actively disconnected or the physical layer is abnormal, the event will be reported.<br>➢ After the reporting of the event, the upper computer( the caller) need to release connection object, or the event reporting will loops.<br>➢ 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.<br>readerName: the name of the connection object. |

## 4.10. TCP Connection Events

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GServer |

| | |
|---|---|
| **Events** | **public** delegateGClientConnected OnClientConnected**;** |
| **Description** | **public delegate** void delegateGpiOver**(**GClient client**);**<br>TCP is under monitoring .It will be triggered to report the event when the remote device connect the upper computer forwardly.<br>GClient: detailed in " **Reporting Object** "。 |

**Reporting Object**

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Object** | GClient |
| Attribute | N/A |
| **Description** | **Description**: this connection object is the identical to the other actively connection objects and share the same usage. |

# 5. Messages Configuration and Query Description

## 5.1. Send Synchronized Messages

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.DAL |
| **Category** | GClient |
| **Method** | **public** void SendSynMsg**(**Message msg**)** |
| **Method 1** | **public** void SendSynMsg**(**Message msg**,** int timeout**)** |
| **Method 2** | **public** void SendSynMsgRetry**(**Message msg**,** int timeout**,** int retry**)** |
| **Return value** | msg.RtCode: message return code, o means success, and non-zero value means failure.<br>msg.RtMsg: the reason of the failed operation. |

| Description | Send synchronized messages, detailed in **Code Example**<br>Tips: "Reader configuration and management", "RFID configuration and operation" and other messages are sent with this method |
|---|---|

**Code Example 1**

```
// Stop command, idle state
MsgBaseStop msgBaseStop = new MsgBaseStop();
clientConn.SendSynMsg(msgBaseStop);
if (0 == msgBaseStop.RtCode)
{
    Console.WriteLine("Stop successful.");
}
else { Console.WriteLine("Stop error."); }
```

**Code Example 1**

```
// Power configuration, set the power of the 4 antennas to be 30dBm.
MsgBaseSetPower msgBaseSetPower = new MsgBaseSetPower();
msgBaseSetPower.DicPower = new Dictionary<byte, byte>()
{
    {1, 30},
    {2, 30},
    {3, 30},
    {4, 30}
};
clientConn.SendSynMsg(msgBaseSetPower);
if (0 == msgBaseSetPower.RtCode)
{
    Console.WriteLine("Power configuration successful.");
}
else { Console.WriteLine("Power configuration error."); }
```

**Code Expamle 3**

```
if (null != this.clientConn)
{
    // Query the power of the antennas
    MsgBaseGetPower msg = new MsgBaseGetPower();
    this.clientConn.SendSynMsg(msg);
    if (0 == msg.RtCode && null != msg.DicPower)
    {
        foreach (var item in msg.DicPower)
        {
```

```
                    switch (item.Key)
                    {
                        case 1:
                            {
                                cmbAnt1.SelectedIndex = item.Value;
                            }
                            break;
                        case 2:
                            {
                                cmbAnt2.SelectedIndex = item.Value;
                            }
                            break;
                        case 3:
                            {
                                cmbAnt3.SelectedIndex = item.Value;
                            }
                            break;
                        case 4:
                            {
                                cmbAnt4.SelectedIndex = item.Value;
                            }
                            break;
                        default:
                            break;
                    }
                }
            }
```

# 6. Messages Description

## 6.1. Reader Configuration and Management

### 6.1.1. Restart the Reader

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgAppReset |

| | |
|---|---|
| **Attribute** | N/A |
| **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 restart. |

### 6.1.2. Configuration and Query COM Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Configuration Class** | MsgAppSetSerialParam |
| **Query Class** | 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.<br>Notes: This configuration needs to be modified when the device is idle (that is, the configuration can be changed under loop reading). |

### 6.1.3. Configure GPO State Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Configuration Class** | MsgAppSetGpo |
| **Attribute** | Gpo1: 0 (low, relay disconnected) 1 (high, relay closed)<br>Gpo2: 0 (low, relay disconnected) 1 (high, relay closed)<br>Gpo3: 0 (low, relay disconnected) 1 (high, relay closed)<br>Gpo4: 0 (low, relay disconnected) 1 (high, relay closed)<br>…… |

| | |
|---|---|
| **Description** | (Persistent configuration, which means the information will be saved when it is powered off) configure the device GPO parameter.<br>Notes: For GPO that do not require state control, no assignment is needed. |

### 6.1.4. Query GPI State Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Query Class** | MsgAppGetGpiState |
| **Attribute** | DicGpiState: it is corresponding to the GPI electrical level state (Dictionary<byte, byte> , key: GPI index No, value: electrical level state(0 low,1 high)). |
| **Description** | Query the GPI status of the device.<br>Notes: The index number starts from 1. |

### 6.1.5. Config and Query GPI Trigger Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Configuration Class** | MsgAppSetGpiTrigger |
| **Query Class** | MsgAppGetGpiTrigger |
| **Attribute** | GpiPort: GPI port No, the index starts from 1<br>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 )<br>TriggerCommand: trigger binding command(Hex, can be null)<br>BtriggerCommand: trigger binding command(Byte[],can be null)<br>TriggerOver: trigger stop(0 non-stop ,1 low electrical trigger ,2 high electrical trigger ,3 rising edge trigger ,4 falling edge trigger ,5 random edge trigger ,6 delayed stop)<br>OverDelayTime: delayed stop time(take effect only if the stop condition is "delayed stop")<br>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.<br>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** | GDotnet.Reader.Api.Protocol.Gx |
| **Query Class** | MsgAppGetBaseVersion |
| **Attribute** | Version: version No. Of the baseband software(such as“0.1.0.0”) |
| **Description** | N/A |

### 6.1.7. Query the Information of the Reader

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Query Class** | MsgAppGetReaderInfo |
| **Attribute** | Imei: sequential number of the reader<br>PowerOnTime: Power on time<br>BaseBuildDate: baseband compiling time<br>AppVersion: application software version( such as:“0.1.0.0”)<br>AppBuildDate: application compiling time<br>SystemVersion: version of the operation system |
| **Description** | N/A |

### 6.1.8. Query RFID ability of the Reader

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Query Class** | MsgBaseGetCapabilities |
| **Attribute** | MaxPower: the maximum power supported<br>MinPower: the minimum power supported<br>AntennaCount: the quantity of the antennas<br>FrequencyArray: the list of supported frequency band<br>0， National standard 920~925MHz<br>1， National standard 840~845MHz<br>2， National standard 840~845MHz and 920~925MHz<br>3， FCC,902~928MHz<br>4， ETSI,866~868MHz<br>ProtocolArray: List of protocols supported,<br>0， ISO18000-6C/EPC C1G2<br>1， ISO18000-6B<br>2， National standard GB/T 29768-2013<br>3， National military standard GJB 7383.1-2011 |
| **Description** | N/A |

## 6.2. RFID Configuration and Operation

### 6.2.1. Stop Command

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseStop |
| **Attribute** | N/A |

| | |
|---|---|
| **Description** | Stop all RFID operations of the reader and make the reader idle.<br>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** | GDotnet.Reader.Api.Protocol.Gx |
| **configuration Class** | MsgBaseSetPower |
| **Query Class** | MsgBaseGetPower |
| **Attribute** | DicPower: the corresponding antenna power of the reader( Dictionary<byte, byte> , key: antenna index No. value: antenna power value)<br>IsPersistence： 0, Will not save when power is off; 1, Power off save (default). |
| **Description** | (Persistent configuration, which means the information will be saved when it is powered off) configure reader power of each antenna port |

## 6.2.3. Configure and Query Working frequency band of the Reader

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **configuration Class** | MsgBaseSetFreqRange |
| **Query Class** | MsgBaseGetFreqRange |
| **Attribute** | FreqRangeIndex: frequency band index, the specific corresponding relationship is detailed in appendix1. |
| **Description** | (Persistent configuration, which means the information will be saved when it is powered off) for configuration of the current working frequency band of the reader. |

## 6.2.4. Config and Query EPC Baseband Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **configuration Class** | MsgBaseSetBaseband |
| **Query Class** | MsgBaseGetBaseband |
| **Attribute** | BaseSpeed: EPC baseband speed (Optional).<br>QValue: Default Q value(Optional) (0~15).<br>Session: (Optional) (0,Session0; 1,Session1; 2,Session2; 3,Session3).<br>InventoryFlag: parameter of inventory taking mark(Optional) (0, use only Flag A to take inventory ;1 , use only Flag B to take inventory;2, use Flag A and Flag B alternately).<br>IsPersistence： 0, Will not save when power is off; 1, Power off save (default). |
| **Description** | ((Persistent configuration, which means the information will be saved when it is powered off) for configuration of the baseband parameter of the reader. |

## 6.2.5. Configure and Query Tag for Uploading Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Configuration Class** | MsgBaseSetFreqRange |
| **Query Class** | MsgBaseGetFreqRange |
| **Attribute** | RepeatedTime: repeated tag filtering time (optional) (means during a reading instruction execution period, the same tag content can only be uploaded once in repeated tag filtering time, 0~65535,the time unit is 10ms).<br>RssiTV: RSSI threshold value (optional) (give up uploading and discard when the RSSI value of the tag is lower than the threshold value). |
| **Description** | (Persistent configuration, which means the information will be saved when it is powered off) configure the parameter that the tag to upload forwardly. |

## 6.2.6. Configure and Query Resident parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Configuration Class** | MsgBaseSetResidenceTime |
| **Query Class** | MsgBaseGetResidenceTime |
| **Attribute** | AntTime: The maximum dwell time of a single antenna during multi antenna card reading, unit: 10ms。<br>FrequencyTime: In frequency hopping mode, the maximum dwell time of a single frequency, unit: 10ms。 |
| **Description** | (Persistent configuration, which means the information will be saved when it is powered off)<br>Configure dwell time of the antenna and dwell time of the frequency. |

## 6.2.7. Read EPC Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseInventoryEpc |
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start)<br>InventoryMode: Continuous/Single Read (0: single read, the reader read one time on each enabling antenna then stop reading and automatically enter idle state; 1: continuous read, the reader keeps reading and stop reading after receiving a stop command..<br>Filter: select reading parameter (optional) ( detailed in parameter description)<br>ReadTid: TID read parameter (optional) ( detailed in parameter description)<br>ReadUserdata: user data area read parameter (optional) (detailed in parameter description )<br>ReadReserved: reserved area read parameter (optional)(detailed in parameter description )<br>HexPassword: access password (optional) |
| **Description** | Used 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. |

## 6.2.8. Write EPC Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseWriteEpc |
| **Attribute** | AntennaEnable: antenna port(Using antenna enumeration is detailed in Quick Start )<br>Area: tag data area to be written (0,reserved area;1,EPC area;2,TID area;3, user data area )<br>Start: word initial address of the tag data area to be written<br>HexWriteData: data content to be written (optional) (hex)<br>BwriteData: data content to be written<br>Filter: select reading parameter (optional) (detailed in parameter description)<br>HexPassword: Access Password (optional) |
| **Description** | ➢ Reader writes EPC Tags, the writing that this command refined should be single operation.<br>➢ ISO18000-6C The protocol specifies that the minimum data unit for R/W operations is a word.<br>➢ EPC are include CRC-16(the zeroth word) + PC( the first word) + EPC:<br>CRC16: the zeroth word, not writable.<br>PC: the first word , the first 5 bit is the word length of EPC , which means the computing method of PC to move the word length of EPC 11 bit to the left. |

## 6.2.9. Lock EPC Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseLockEpc |
| **Attribute** | AntennaEnable: antenna port(Using antenna enumeration is detailed in Quick Start )<br>Area: tag data area to be locked(0, inactivated password area; 1, access password; 2, EPC area; 3, TID area; 4, user data area )<br>Mode: locking operation type(0,unlock;1,lock;2,unlock permanently;3,lock permanently)<br>Filter: Select read parameters (optional) ( detailed in parameter description )<br>HexPassword: access password (optional) |
| **Description** | To lock or unlock the tag. The operation this command defined is single. |

### 6.2.10. Inactivate EPC Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseDestoryEpc |
| **Attribute** | AntennaEnable: antenna port(Using antenna enumeration is detailed in Quick Start )<br>HexPassword: Destroy passwords<br>Filter: Select read parameters (optional) (detailed in parameter description ) |
| **Description** | Inactivate the tag. The tag inactivated will be in permanent failure and this operation is irreversible. The operation this command defined is single. |

### 6.2.11. Read 6B Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseInventory6b |
| **Attribute** | AntennaEnable: antenna port(Using antenna enumeration is detailed in Quick Start )<br>InventoryMode: Continuous/Single Read (0: Single Read ,the reader read one time on each enabling antenna then stop reading and automatically enter idle state; 1: continuous read, the reader keeps reading and stop reading after receiving a stop command.)<br>Area: Reading content (0, read only 6B TID;1,read 6B TID+ user data;2, read only user data )<br>ReadUserdata: user data area read parameter (optional) (detailed in parameter description )<br>HexMatchTid: TID code of the 6B tag to be matched (optional) (hex)<br>BMatchTid: TID code of the 6B tag to be matched (optional) |
| **Description** | Used for the data reading for ISO18000-6B tag. |

### 6.2.12. Write 6B Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseWrite6b |

| | |
|---|---|
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start )<br>HexMatchTid: TID code of the 6B tag to be matched (optional) (hex)<br>BMatchTid: TID code of the 6B tag to be matched<br>Start: Byte initial address for the data area of the tag to be written<br>HexWriteData: data content to be written (optional) (hex)<br>BwriteData: data content to be written |
| **Description** | Write 6B tag. The writing operation this command defined is single. |

### 6.2.13. Lock 6B Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseLock6b |
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start )<br>HexMatchTid: TID code of the 6B tag to be matched (optional) (hex)<br>BMatchTid: TID code of the 6B tag to be matched<br>LockIndex: 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. |

### 6.2.14. 6B Tag Locking Query

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseLock6bGet |
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start )<br>HexMatchTid: TID code of the 6B tag to be matched (optional) (hex)<br>BMatchTid: TID code of the 6B tag to be matched<br>LockIndex: byte address of the data to be locked and queried. |

| | |
|---|---|
| **Description** | Query the locking state for the 6B tag. The query operation this command defined is single. |

## 6.2.15. Read GB Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseInventoryGb |
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start )<br>InventoryMode: Continuous/Single Read (0: Single Read ,the reader read one time on each enabling antenna then stop reading and automatically enter idle state; 1: continuous read, the reader keeps reading and stop reading after receiving a stop command.)<br>Filter: Select read parameters (optional) ( detailed in parameter description )<br>ReadTid: TID read parameter (optional) ( detailed in parameter description )<br>ReadUserdata: user data area read parameter (optional) (detailed in parameter description )<br>HexPassword: access password (optional) |
| **Description** | Used for configuring the reader parameter for tag reading and start reading. Any reading operation for the tag data require the tag code, so a EPC code can be acquired with any reading operation. |

## 6.2.16. Write GB Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseWriteGb |
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start )<br>Area: tag data area to be written (0x10,tag coding area 0x20, tag secure area 0x30~0x3F,User subarea 0~15 area)<br>Start: word address of the data area of the tag to be written<br>HexWriteData: data content to be written (optional) (hex)<br>BwriteData: data content to be written<br>Filter: Select read parameters (optional) ( detailed in parameter description )<br>HexPassword: access password (optional) |

| | |
|---|---|
| **Description** | ➢ The reader write GB tag. The writing operation this command defined is single.<br>➢ The GB protocol specifies that the minimum data unit for R/W operations is a word. |

## 6.2.17. Lock GB Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseLockGb |
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start )<br>Area: the tag data area to be locked (0x00 tag information area, 0x10 tag coding area, 0x20 tag secure area, 0x30~0x3F user subarea 0~15)<br>Mode: locking operation type<br>0x00,Readable and writable<br>0x01,Readable but not Writable<br>0x02,Unreadable but Writable<br>0x03,Unreadable and unwritable<br>0x11,Set no identification is required under secure mode; the operation area must be tag secure area.<br>0x12,Set identification is required but no secure communications is required under secure mode. The operation area must be tag secure area.<br>0x13,Set identification and secure communications are required under secure mode. The operation area must be tag secure area.<br>Filter: Select read parameters (optional) ( detailed in parameter description )<br>HexPassword: access password (optional) |
| **Description** | Lock or unlock the tag. The operation this command defined is single. |

## 6.2.18. Inactivate GB Tag

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | MsgBaseDestoryGb |

| | |
|---|---|
| **Attribute** | AntennaEnable: antenna port(using antenna enumeration is detailed in Quick Start )<br>HexPassword: Destroy passwords<br>Filter: Select read parameters (optional) ( detailed in parameter description ) |
| **Description** | Inactivate the tag. And the inactivated tag will be in permanent failure This operation is irreversible. The operation this command defined is single. |

# 7. Parameter Description

## 7.1. 6C/GB Tag Select Tag Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | ParamEpcFilter |
| **Attribute** | Area: data area to be matched (1,EPC area ;2,TID area;3, user data area )<br>(GB tag,0x00 tag information area ,0x10 tag coding area ,0x20 tag secure area ,<br>0x30~0x3F user subarea 0~15)<br>BitStart: starting bit address of the matching data<br>BitLength: the data bit length to be matched<br>HexData: the data content to be matched (optional) (hex)<br>BData: the data content to be matched |
| **Description** | optional parameter (The first 32 bit of the EPC area is PC value. So usually the starting bit address is 32 when distinguishing EPC.) |

## 7.2. 6C Tag Read TID Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **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)<br>Len: The reader need to read the length of the word(word, 16bits, similarly hereinafter) of TID data. |
| **Description** | optional parameter |

## 7.3. 6C Tag Read User Data Area Parameter

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | ParamEpcReadUserdata |
| **Attribute** | Start: starting word address<br>Len: The word length of user data that need to be read. |
| **Description** | optional parameter |

## 7.4. Parameter of User Data Area for 6B Tag Reading

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | Param6bReadUserdata |
| **Attribute** | Start: starting byte address for user data<br>Len: byte length for user data |
| **Description** | optional parameter |

## 7.5. Parameter for User Data Area for GB Tag Reading

| | |
|---|---|
| **Namespace** | GDotnet.Reader.Api.Protocol.Gx |
| **Category** | ParamGbReadUserdata |
| **Attribute** | ChildArea: user subarea<br>Start: starting byte address for user data<br>Len: byte length for 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 |

# 9. Appendix 2