Technical reference manual

No.	Date	Note
1	20170512	Initial Version
2	20200908	Add directory setting for log files and configure files 3.4.37 3.4.38

SAFETY INSTRUCTIONS

 Caution

Improper operation may cause harm to your physical health.

Improper operation may cause damage to the device.

 Attention

If ignored, it may prevent your operation from proceeding smoothly.

If ignored, it may lead to unwanted results.

Content

1 API operation class composition

The API specified in this document is a reference platform for the communication between readers and application software, and is suitable for the secondary development of the company's fixed reader application. It is also a required file for system integrators or end users when programming (including PortalAPI.dll, PortalCore.dll, PortalConfigFile.xml, language\ErrCode(**).xml),API is delivered to the user along with the hardware device.

To ensure the secondary development program proceed smoothly, please include the above mentioned files in the secondary development software.

1.1 PortalAPI.dll

API dynamic linked library.

1.2 Config file(PortalConfigFile.xml)

Config file is used to configure connection information and log levels. As shown in the following picture:

```
<?xml version="1.0" encoding="utf-8"?>
<Terminals IsSingleReader="true" CurrentReaderName="Portal1" IsBeep="False">
  <LogLevel>INFO</LogLevel>
  <Client Name="Portal1" Group="Group1">
    <Port Type="TcpClient" Protocol="GRP">192.168.1.201:7088</Port>
  </Client>
</Terminals>
```

1.2.1 LogLevel enum

In ascending order, there are 5 levels of LogLevel: DEBUG, INFO, WARN, ERROR, FATAL. The default level is INFO.

Note: if it wasn't for debugging, do not change it to DEBUG. If the log level is set to debug, the API will write a large amount of debugging information to logs, which seriously affects the running efficiency.

1.2.2 Client enum

Name Indicates the name of the access control connection , it is a user-defined string

Type attribute of Port indicates the connection type. The optional values is :TcpClient and RS232.

Port Indicates the connection character string. For example, 192.168.1.201:7088 indicates that the IP address is 192.168.1.201 and the port is 7088. The two are separated by colons; the serial port connection character string COM1,115200 indicates serial port 1. The baud rate is 115200 BPS. The serial port number and baud rate are separated by commas.

1.3 Language package(language folder)

The language package is located in the Language folder of the API. It contain multiple language files in the format of the file name '****.xml'. '* * * *' is the value of System.Globalization.CultureInfo.LCID , such as Chinese value is 2052. At present, just English and Chinese is available.

1.4 Error Code Class

The error code class is used in conjunction with 'ErrCode(****).xml' file in the language folder. It provides error code messages to software.

2 Gate operation Class (CommonGate)

The Gate operation class includes functions such as establishing/disconnecting connection, access control board Settings, and access control service

namespace: PortalAPI

Code set: API (in PortalAPI.dll)

2.1 properties

2.1.1 Gate Name (GateName property)

purpose: get or set the name of gate device(access control system),

Usage:

```
public String GateName{ get; set;}
```

Note:

Gate Name is a user-defined name. It specifies the name of the gate. The value cannot be repeated in the same configuration file

2.1.2 Status of connection (IsConnected property)

purpose: get or set the connection status of gate device

Usage:

```
public Boolean IsConnected{ get; set;}
```

Note:

This property consists of 1 byte. All subclasses of CommonGate class inherit this property.

2.1.3 Protocol of RFID module (ReaderProtocol property)

purpose: get or set the protocol of rfid module in Gate device. Optional values is:CRP, LRP.

Usage:

```
public ProtocolVersion ReaderProtocol{ get; set;}
```

Note:

At present,CRP and LRP is optional.Normally it does not need to be changed by users because It has been set before delivery

2.1.4 Reader module 1 Enable (Reader1Enable property)

purpose: get or set and enable vable of Reader module 1

Usage:

```
public Boolean Reader1Enable{ get; }
```

2.1.5 Reader module 2 Enable (Reader2Enable property)

purpose: get or set and enable vable of Reader module 2

Usage:

```
public Boolean Reader2Enable{ get; }
```

Note:

Gate device can contain two reader modules. Normally this property is set before delivery.

2.2 Event

2.2.1 OnTagDataReceived Event

```
purpose: Receive real-time data from gate device
Usage:
    public event TagDataReceivedHandle OnTagDataReceived;
delegate prototype:
    public delegate void TagDataReceivedHandle(
        String gateName,
        RxdTagData tagData);
Tagdata class (RxdTagData) :
    Tag EPC: public Byte[] EPC{ get; set; }
    Tag TID: public Byte[] TID{ get; set; }
    antenna No.: public Byte Antenna{ get; set; }
    Port No.: public Unshort PortNum{ get; set; }
    Tag RSSI: public Double RSSI{ get; set; }
```

2.2.2 OnGateBuffTagDataReceived Event

```
purpose: Receive offline data from gate device
Usage:
    public event GateRxdBuffDataReceivedHandle OnGateBuffTagDataReceived;
delegate prototype:
    public delegate void GateRxdBuffDataReceivedHandle(
        String gateName,
        Gate_RxdBuffData tagData);
```

Gate_RxdBuffData class:

property	type	description
TagNUM	Uint16	Serial number of the tag data, used to mark and reply confirm. Note: API replies automatically, the customer does not need to process.
Time	DateTime	The time when tag is read. UTC time
Chanel	Byte	Channel No. 01H: Channel 1 02H: Channel 2
InOutState	Byte	entry and exit direction of tag movement 00H/NULL: entry and exit direction cannot be determined 01H: entry 02H: exit

EPC	Byte[]	EPC data
PC	Byte[]	The length of EPC data
Antenna	Byte	Antenna number when the label is read. (There are two modules, the antenna number of module 2 is 5-8)
RSSI	Double	RSSI value when tag is read
TID	Byte[]	Tag TID

Note: If the offline buffer function is enabled, offline data will be automatically uploaded when gate device is idle

2.2.3 OnTriggerRxdDataState Event

purpose: When the gate triggers reading or stops, this message need to be uploaded。
Usage:

```
    public event TriggerRxdDataStateHandle OnTriggerRxdDataState;
```

delegate prototype:

```
    public delegate void TriggerRxdDataStateHandle(
        String gateName,
        Gate_TriggerRxdDataStateParameter state);
```

Gate_TriggerRxdDataStateParameter class：

property	type	description
IsStart	Boolean	True：Start False：Stop

2.2.4 OnRxdInOutState Event

Purpose: Used to count the number of visitors.
Usage:

```
    public event RxdInOutStateHandle OnRxdInOutState;
```

delegate prototype：

```
    public delegate void RxdInOutStateHandle(
        String gateName,
        Gate_InOutStateParameter state);
```

Gate_InOutStateParameter class:

property	type	description
Inter	Int32	Number of people present
In	Int32	Number of people get in
Out	Int32	Number of people get out

State	Byte	In-out state 0x00 No people in or out 0x01 It can not be judged because of the first infrared trigger data is wrong, 0x02 It can not be judged because of the second infrared trigger data is wrong 0x03 some one get in 0x04 some one get out

2.2.5 OnRxdIrTriggerState Event

Purpose: When the gate is triggered, this message need to be uploaded。
Usage:

```
public event RxdIrTriggerStateHandle OnRxdIrTriggerState;
```

Delegate prototype:

```
public delegate void RxdIrTriggerStateHandle(
    String gateName,
    Dictionary<byte, bool> irState);
```

Infrared trigger state(irState parameters):

Key: trigger port, port1~port8

Value: Trigger or not

2.2.6 Gate_OnCallBackEvent

Purpose: In top-mounted gate,when all phase states are polled, this message will be uploaded to the application layer.
Usage:

```
public event CallBackEventHandle OnCallBackEvent;
```

Delegate prototype:

```
public delegate void CallBackEventHandle(String readerName, object arg);
```

2.3 Constructor

2.3.1 Constructor

Purpose: used to create a gate class instance.
Usage:

```
public CommonGate(string gateName)
```

Parameter:

gateName: The name of the gate connected.It should be consistent with the configuration file.

Note:

API supports multiple Gates. Only one instance can be created in one client.

2.4 Method

2.4.1 Connect

Purpose: used to create a connection between host and gate.
Usage:

```
public Boolean Connect()
```

Parameter:
    Null.
Return:
    Whether the connection is successful.
    True connect successfully;
    False connect fail
Note:
    The connection parameters are set in the configuration file.

2.4.2 Disconnect

Purpose: disconnect host and gate
Usage:

```
public void Disconnect()
```

Parameter:
    Null.
Return:
    Null.

2.4.3 Gate_Ctrb_Restart

Purpose: This method is used to soft restart the gate control board
Usage:

```
public bool Gate_Ctrb_Restart(out String errStr)
```

Parameter:
    errStr: error message.
Return:
    Operation success or fail,
    True operation success
    False operation fail

2.4.4 Gate_Ctrb_SetUtcTime

Purpose: used to configurate the UTC time of gate
Usage:

```
public bool Gate_Ctrb_SetUtcTime(
    DateTime utc,
    out String errStr)
```

Parameter:
    utc: UTC time.
    errStr: Error message。
Return:
    Operation success or fail,
    True operation success
    False operation fail.
Note:
    UTC is 8 hours different from Beijing time.

2.4.5 Gate_Ctrb_GetUtcTime

Purpose: used to get the UTC time of gate
Usage:
```
public DateTime Gate_Ctrb_GetUtcTime(out String errStr)
```
Parameter:
errStr: Error message.
Return:
UTC time。
Note:
UTC is 8 hours different from Beijing time.

2.4.6 Gate_Ctrb_LinkConfirm

Purpose: This method is used to confirm the connection status between the gate and the host computer.
Usage:
```
public bool Gate_Ctrb_LinkConfirm(out String errStr)
```
Parameter:
errStr: Error message。
Return:
Operation success or fail,
True operation success
False operation fail.
Note:
Both gate and host can send a connection status confirmation message. The peer party must reply to the confirmation message immediately after receiving this message. If the initiator cannot receive the confirmation message from the peer party, the connection is considered invalid. API automatically replies the connection confirmation message of gate.

2.4.7 Gate_Ctrb_GetMac

Purpose: used to get the MAC address of the gate.
Usage:
```
public byte[] Gate_Ctrb_GetMac(out String errStr)
```
Parameter:
errStr: Error message。
Return:
Gate’s MAC address,Unsigned byte, as 44-87-FC-94-BD-BF.

2.4.8 Gate_Ctrb_SetIP

Purpose: used to set gate’s IP address.
Usage:
```
public bool Gate_Ctrb_SetIP(
    Gate_IpParameter ipParam,
    out String errStr)
```
Parameter:
ipParam: IPParameter.

Gate_IpParameter class

property	type	description

Ip	String	IP address, unsigned byte, as 192.168.1.2 / C0-A8-01-02
Subnet	String	Subnet mask, as above
Gateway	String	Gateway, as above
Port	UInt16	2 bytes，Port number of the gate in server mode

errStr：Error message。

Return：

Operation success or fail,

True operation success

False operation fail.

2.4.9 Gate_Ctrb_GetIP

Purpose：used to get the IP address of gate.

Usage：

```
public Gate_IpParameter Gate_Ctrb_GetIP(
    out String errStr)
```

Parameter：

errStr：Error message。

Return：

Gate’s IP address. Same data format as above.

2.4.10 Gate_Ctrb_DefaultParam

Purpose：Restore factory Settings.

Usage：

```
public bool Gate_Ctrb_DefaultParam(out String errStr)
```

Parameter：

errStr：Error message。

Return：

Operation success or fail,

True operation success.

False operation fail.

Note：

This instruction is used for the host to restore the default configuration of gate.

2.4.11 Gate_Ctrb_SetModuleType

Purpose：used to set the type of RFID module.

Usage：

```
public bool Gate_Ctrb_SetModuleType(
      ProtocolVersion protocol,
      out String errStr)
```

Parameter：

protocol：RFID module, at present CRP,LRP is available。

errStr：Error message。

Return：

Operation success or fail,

True operation success

False operation fail.

Note：

This method is used to set the module model connected to the main control board. The module model is defined as the manufacturer type. At present, just LRP,CRP is available.

2.4.12 Gate_Ctrb_GetModuleType

Purpose: used to get the type of RFID module.
Usage:

```
public ProtocolVersion Gate_Ctrb_GetModuleType(
    out String errStr)
```

Parameter:
errStr: Error message。
Return:
Protocol type of RFID module, the definition is the same as above.

2.4.13 Gate_Ctrb_UpdataApp

Purpose: used to update the firmware in gate’s control board。
Usage:

```
public bool Gate_Ctrb_UpdataApp(
    Gate_CtrbUpdataAppPackage package,
    out String errStr)
```

Parameter:
package: Upgrade package information.

Gate_IpParameter classs

property	type	description
SerialNumber	UInt32	Upgrade packet sequence number，The value starts with 0x00000000 and ends with 0xFFFFFFFF.
UpdataPackage	Byte[]	Upgrade data

errStr: Error message。
Return:
Operation success or fail,
True operation success,
False operation fail.
Note:
This method is used for online upgrade of gate’s application software and baseband software. During the upgrade process, the host splits the upgrade file into a certain number of bytes, and sends an upgrade packet numbered 0x00000000 to start the upgrade. During the upgrade process, the upgrade packet numbered 0xFFFFFFFF is sent to mark the end of the upgrade.Gate must confirm the upgrade packet of each upper computer, and the host computer can send the next upgrade packet only after the current packet is successfully confirmed

2.4.14 Gate_Ctrb_VersionQuery

Purpose: inquire the software version of gate.
Usage:

```
public String Gate_Ctrb_VersionQuery(out String errStr)
```

Parameter:
errStr: Error message。
Return:

Strings of gate’s version, such as: 20160903V105.

2.4.15 Gate_SetWorkReader

Purpose: Set the working status of the RFID module.
Usage:

```
public bool Gate_SetWorkReader(
    Byte[] param,
    out String errStr)
```

Parameter:
param:
Set Parameter, 11 bytes. Definition as below:

Byte 0	polling Parameter	0x01	Enable polling
		0x00	Disable polling
Byte 1	Reader 1	0x01	valid
		0x00	invalid
Byte 2	Reader 2	0x01	valid
		0x00	invalid
Byte 3-10		reserved	

errStr: Error message.
Return:
Operation success or fail,
True operation success,
False operation fail.

2.4.16 Gate_GetWorkReader

Purpose: Get the working status of the RFID module。
Usage:

```
public byte[] Gate_GetWorkReader(out String errStr)
```

Parameter:
errStr: Error message。
Return:
Parameter of rfid module, 11 bytes, same as 2.4.15

2.4.17 Gate_SetIrDelay

Purpose: Set the delay of infrared trigger
Usage:

```
public bool Gate_SetIrDelay(byte sec, out String errStr)
```

Parameter:
sec: delay,the unit is second
errStr: Error message。
Return:
True operation success;
False operation fail.
Note:
When the item passes through the gate , the infrared photoelectric sensor triggers the gate to read the card, and the time of reading the card is the delay time.

2.4.18 Gate_GetIrDelay

Purpose: Set the delay of infrared trigger.
Usage:

```
public byte Gate_GetIrDelay(out String errStr)
```

Parameter:
Null.
Return:
Delay time, the unit is second.

2.4.19 Gate_SetPeopleCount

Purpose: Set a initial value for head count number.
Usage:

```
public bool Gate_SetPeopleCount(
    Gate_PeopleCountParameter param,
    out String errStr)
```

Parameter:
param: head count parameter。
errStr: Error message
Gate_PeopleCountParameter class

property	type	description
Inter	Int32	Number of people present 0 don't display other display and show the value
In	Int32	Number of people get in 0 don't display other display and show the value
Out	Int32	Number of people get out 0 don't display other display and show the value
DisplayMode	Byte	Display mode 00H don't display 01H display number of people present 02H display number of people get in 03H display number of people get out 04H display number of people present+number of people get in 05H display number of people get in+number of people get out

Return:
True operation success;
False operation fail.

Note:
head count change, API received the message through Event OnRxdInOutState.

2.4.20 Gate_GetPeopleCount

Purpose: Get head count number in the gate
Usage:

```
public Gate_PeopleCountParameter Gate_GetPeopleCount(
    out String errStr)
```

Parameter:
errStr: Error message。
Return:
head count parameter.
The definition of Gate_PeopleCountParameter, pls refer to 2.4.19

2.4.21 Gate_SetBufMode

Purpose: used to set offlie mode.
Usage:

```
public bool Gate_SetBufMode(
    Gate_BufModeInfo bufMode,
    out String errStr)
```

Parameter:
bufMode: offline mode parameter。

Gate_BufModeInfo class

property	type	description
IsEnable	Boolean	True enable False Disable
OffLineDataCount	UInt16	Clear offline data or not 0 Clear offline data other Keep offline data

errStr: Error message。
Return:
True operation success;
False operation fail.
Note:
Host receive the offline data through the event OnTagDataReceived.

2.4.22 Gate_GetBufMode

Purpose: get offline mode information.
Usage:

```
public Gate_BufModeInfo Gate_GetBufMode(
    out String errStr)
```

Parameter:
Null.
Return:
offline mode parameter

Gate_BufModeInfo class

property	type	description
IsEnable	Boolean	True enable False Disable
OffLineDataCount	UInt16	Number of offline data records

2.4.23 Gate_SetReaderClient

Purpose: used to set Client mode of gate
Usage:

```
public bool Gate_SetReaderClient(
    Gate_ReaderClientParameter param,
    out String errStr)
```

Parameter:
param: client mode Parameter。
Gate_ReaderClientParameter class

property	type	description
IsEnable	Boolean	True enable client mode False close client mode
IP	String	IP address of the server
Port	UInt16	port number of the server

errStr: Error message。
Return:
True operation success;
False operation fail.

2.4.24 Gate_GetReaderClient

Purpose: used to get client mode information of gate
。
Usage:

```
public Gate_ReaderClientParameter
    Gate_GetReaderClient(out String errStr)
```

Parameter:
errStr: Error message。
Return:
client mode Parameter in Gate_ReaderClientParameter class,pls refer to 2.4.23.

2.4.25 Gate_SetGpo

Purpose: Set the output of GPIO.
Usage:

```
public bool Gate_SetGpo(
    Byte port,
    Byte level,
    out String errStr)
```

Parameter:

port: GPO number.

01H	GPO1
02H	GPO2
03H	GPO3
04H	GPO4
05H	GPO5
06H	GPO6
07H	GPO7
08H	GPO8

Level: output level。

00H	low level
01H	high level

errStr: Error message。

Return:

True operation success;

False operation fail.

2.4.26 Gate_GetGpi

Purpose: get the level of GPIN.

Usage:

```
public Dictionary<byte, byte> Gate_GetGpi(
    Byte port,
    out String errStr)
```

Parameter:

port: GPIN number.

01H	GPI1
02H	GPI2
03H	GPI3
04H	GPI4
05H	GPI5
06H	GPI6
07H	GPI7
08H	GPI8

Return:

GPIN number and level info.

Key: GPIN number,As defined above.

Value: input level, defined as 2.4.25.

2.4.27 Gate_SetTriggerFrame

Purpose: used to set trigger reading command.

Usage:

```
public bool Gate_SetTriggerFrame(
```

```
        TriggerFrameParameter param,
        out String errStr)
```

Parameter:

param: trigger reading parameters.

errStr: Error message。

TriggerFrameParameter class

property	type	description
Protocol	ProtocolVersion	Protocol protocol CRP, LRP is efficient
ScanParam	ScanTagParameter	Scan parameter, defined as below
Address	Byte	Module 1: 01H Module 2: 02H

ScanTagParameter class

property	type	description
Antenna	Byte	Antenna number
IsLoop	Boolean	True Loop read False single read
TidParameter	Byte[]	TID reading parameter Null unread TID data Non-null 2 bytes defined as below: Byte 0：TID read mode configuration， 0，adaptive to TID length, but the maximum length does not exceed the length defined by byte 1； 1，Read TID at the length defined by byte 1 Byte 1： The length of the TID data (word, Multiples of 16)

UserDataParameter	Byte[]	User memory parameter Null　unread User memory User memory reading is not supported, so set this parameter to blank.
SelectTagParamete r	Byte[]	Tag selected parameter Null　none set。 Non-null　set the parameter defined as below. Byte 0：Data zone to match. 1，EPC； 2，TID； 3，User memory。 Byte 1+Byte 2：start bit address need to match, Byte1 High 8-bit of start bit address，　Byte2 Low 8-bit of start bit address。 Byte 3：data lenght need to match Byte 4~Byte N：Data that needs to be matched.

Return:
    True operation success;
    False operation fail.

2.4.28 Gate_GetTriggerFrame

Purpose: used to get paramter of trigger reading command 。
Usage:

```
    public TriggerFrameParameter Gate_GetTriggerFrame(
        Byte protocolNum,
        out String errStr)
```

Parameter:
    protocolNum: protocol type。
        01H　module 1
        02H　module 1
    errStr: Error message。

Return:
    TriggerFrameParameter class, refer to 2.4.28

2.4.29 Gate_ Inventory

Purpose: Manually control the gate to read.
Usage:

```
    public bool Gate_Inventory(
```

```
        ScanTagParameter param,
        Byte readerAddress,
        out String errStr)
Parameter:
    param: parameter when reading used

    readerAddress:  modulue address
            01H   module 1
            02H   module 2
Return:
    True  operation success;
    False operation fail.
Note:
    Tag data is received through OnTagDataReceived Event or OnGateBuffTagDataReceived Event.
```

2.4.30 Gate_StopInventory

```
Purpose: Manually stop reading.
Usage:
    public bool Gate_StopInventory(out String errStr)
Parameter:
    errStr: Error message。

Return:
    True  operation success;
    False operation fail.
```

2.4.31 Gate_SetPowerList

```
Purpose: used to set RF output level.
Usage:
    public bool Gate_SetPowerList(
        Dictionary<byte, Dictionary<byte, byte>> powerList,
        out String errStr)
Parameter:
    powerList: power level list

        Key: antenna port (01H~08H)

        Value: power level (0~36) dBm

    errStr: Error message。

Return:
    True  operation success;
    False operation fail.
```

2.4.32 Gate_GetPowerList

```
Purpose: used to get RF output level
Usage:
    public Dictionary<byte, Dictionary<byte, byte>>
        Gate_GetPowerList(out String errStr)
Parameter:
```

errStr: Error message。

Return:
Power level list

2.4.33 Gate_SetInventoryMode

Purpose: used to set RFID inventory mode.
Usage:

```
public bool Gate_SetInventoryMode(
    GateInventoryModeParameter param,
    out String errStr)
```

Parameter:
param: inventory mode parmater class.

GateInventoryModeParameter class

property	type	description
Address	Byte	Module address 01H module 1 02H module 2
Mode	Byte	Inventory mode 0x00 -Fast switching antenna mode 0x01-Custom mode
Session	Byte	Parameter definded in ISO18000-6C Value range 0,1,2,3
Target	Byte	Inventoried Flag， 0x00 Flag A; 0x01 Flag B; 0x11-Flag A and Flag B in turn
Repeat	Byte	The number of times the inventory process is repeated

errStr: Error message。

Return:
True operation success;
False operation fail.

2.4.34 Gate_GetInventoryMode

Purpose: used to get RFID inventory mode parameter.
Usage:

```
public GateInventoryModeParameter Gate_GetInventoryMode(
    Byte address,
    out String errStr)
```

Parameter:
address: module address(0x01 or 0x02)
errStr: Error message.

Return:
GateInventoryModeParameter class, refer to 2.4.34.

2.4.35 Set_IR_Enable

Purpose: enable/disable the Infrared trigger function 。
Usage:

```
public bool Set_IR_Enable(bool enable, out String errStr)
```

Parameter:

enable:

False: disable; True: enable.

errStr: Error message。

Return:

True operation success;

False operation fail.

2.4.36 Get_IR_Enable

Purpose: get the status of Infrared trigger function
Usage:

```
public byte Get_IR_Enable(out String errStr)
```

Parameter:

errStr: Error message。

Return:

Enable statue, 0: disable; 1: enable.

2.4.37 SetPath

Purpose: This section describes how to save log files to a specified directory

Usage:

```
public static void Log.SetPath(string path)
```

Call method:

```
Log.SetPath(@"D:\dbug");
```

Return:

Null.

2.4.38 ConfigFile.SetPath

Purpose: This section describes how to save configuration files to a specified directory.

Usage:

```
public static void SetPath(string path)
```

Call method:

```
ConfigFile.SetPath(@"D:\dbug");
```

Return:

Null

2.4.39 Gate_SetTriggerInOut

Purpose：Set channel direction of gate.

Usage：

```
public bool Gate_SetTriggerInOut(
IODirectionParameter[] param,
out string errStr)
```

Parameter：

param：Parameter array。

property	type	description
IsConfig	Bool	Configure it or not. True -configure it； False -don't configure it
Direction	Bool	True: forward direction; False: reverse direction.

errStr：Error message。

Return：

True operation success;
False operation fail.

2.4.40 Gate_GetTriggerInOut

Purpose：Get the channel direction of gate。

Usage：

```
public IODirectionParameter[] Gate_GetTriggerInOut(out string errStr)
```

Parameter：

errStr：Error message。

Return：

IODirectionParameter[]：

property	type	description
IsConfig	Bool	Configure it or not. True -configure it； False -don't configure it
Direction	Bool	True: forward direction; False: reverse direction.

2.4.41 Gate_SetEas

Purpose：Set the EAS alarm enable and alarm duration.

Usage：

```
public bool Gate_SetEas(EasParameter param, out string errStr)
```

Parameter：

param：eas parameter class.

EasParameter class

property	type	description

Mode	Byte	Alarm mode 0x01: Bit matching alarm 0x02: No alarm 0x12: Bit mismatching alarm 0x13: all tag alarm
Time	Byte	alarm duration, unit: second

errStr：Error message。

Return：

True operation success;

False operation fail.

2.4.42 Gate_GetEas

Purpose：Set the EAS alarm enable and alarm duration.

Usage：

```
public EasParameter Gate_GetEas(out string errStr)
```

Parameter：

errStr：Error message。

Return：

property	type	description
Mode	Byte	Alarm mode 0x01: Bit matching alarm 0x02: No alarm 0x12: Bit mismatching alarm 0x13: all tag alarm
Time	Byte	alarm duration, unit: second

2.4.43 Gate_SetEasMaskMatch

Purpose：configurate the EAS alarm matching group.

Usage：

```
public bool Gate_SetEasMaskMatch(EasMatchParameter param, out string errStr)
```

Parameter：

param：EAS matching group.

property	type	description
NO	Byte	Serial number，0~9，10 groups at most
Enabled	Bool	True：enable current group False：disable current group
EPC	Byte[]	One or more anti-theft bits can be configured

Mask	Byte[]	Mask code

errStr：Error message。

Return：

True operation success;

False operation fail.

Note

For example, if the first character of the epc hexadecimal string is 8, the EPC hexadecimal string is 8000000000000000 and the mask is F000000000000000.

2.4.44 Gate_GetEasMaskMatch

Purpose：get the EAS alarm matching group。

Usage：

```
public EasMatchParameter Gate_GetEasMaskMatch(byte index, out string errStr)
```

Parameter：

Index：匹配组序号

errStr：Error message。

Return：

property	type	description
NO	Byte	Serial number，0~9，10 groups at most
Enabled	Bool	True：enable current group False：disable current group
EPC	Byte[]	One or more anti-theft bits can be configured
Mask	Byte[]	Mask code

2.4.45 SetRFLinkProfile

Purpose：Set the baseband rate of the reader module.

Usage：

```
public bool SetRFLinkProfile(byte address, byte profileid, out string errStr)
```

Parameter：

param：baseband rate Parameter。

property	type	description
Address	Byte	Module address 01H or 02H
Profileid	Byte	0xD0：Tari 25uS，FM0 40KHz 0xD 1：Tari 25uS，Miller 4 250KHz 0xD 2：Tari 25uS，Miller 4 300KHz 0xD 3：Tari 6.25uS，FM0 400KHz

errStr: Error message。

Return:
True operation success;
False operation fail.

2.4.46 GetRFLinkProfile

Purpose: inquire the baseband rate of the reader module。

Usage:

```
public byte GetRFLinkProfile(byte address, out string errStr)
```

Parameter:

Address: module address

errStr: Error message。

Return:

property	type	description
Profileid	Byte	0xD0: Tari 25uS, FM0 40KHz 0xD 1: Tari 25uS, Miller 4 250KHz 0xD 2: Tari 25uS, Miller 4 300KHz 0xD 3: Tari 6.25uS, FM0 400KHz

2.4.47 SetFrequency

Purpose: used to set the frequency band and frequency.

Usage:

```
public bool SetFrequency(byte address, FrequencyParameter freqParam, out String errStr)
```

Parameter:

address: module address, 01H or 02H。

freqParam: frequency parameter class.

property	type	description
FrequencyBand	Byte	0x01:FCC 0x02:ETSI 0x03:CHN 0x04:custom
FrequencyList	Byte[]	0~59 correspond to 865.00MHz~928.00MHz

errStr: Error message。

Return:
True operation success;
False operation fail.

2.4.48 GetFrequency

Purpose: inquire frequency and frequency band from gate

Usage:

```
public FrequencyParameter GetFrequency(byte address, out String errStr)
```

Parameter:

Address: module address

errStr: Error message。

Return:

property	type	description
FrequencyBand	Byte	0x01:FCC 0x02:ETSI 0x03:CHN 0x04: Custom
FrequencyList	Byte[]	0~59 correspond to 865.00MHz~928.00MHz

2.4.49 SetBeepLevel

Purpose: Set the volume of alarm.

Usage:

```
public bool SetBeepLevel(byte level, out string errStr)
```

Parameter:

level: 0~255。

errStr: Error message。

Return:

True operation success;
False operation fail.

2.4.50 GetBeepLevel

Purpose: Inquire the volume of alarm.

Usage:

```
public byte GetBeepLevel(out string errStr)
```

Parameter:

errStr: Error message。

Return:

0~255

2.4.51 SetUrlAddress

Purpose: Set the URL link of HTTP server

Usage:

```
public bool SetUrlAddress(RemoteServerParameter serverParamter, out string errStr)
```

Parameter:

RemoteServerParameter

property	type	description
serverMode	ServerMode	assign to：Mode_HTTP
Url	string	HTTP server link

errStr: Error message。

Return:

True operation success;

False operation fail.

2.4.52 GetUrlAddress

Purpose: inquiry the URL link of HTTP server

Usage:

```
public RemoteServerParameter GetUrlAddress(out string errStr)
```

Parameter:

errStr: Error message。

Return:

property	type	description
serverMode	ServerMode	assign to：Mode_HTTP
Url	string	HTTP server link

2.4.53 SetFilterParameter

Purpose: Set tag filtering parameters,Configure tag’s start string and the minimum RSSI value

Usage:

```
public bool SetFilterParameter(FilterParameter param, out string errStr)
```

Parameter:

param: filtering Parameter。

property	type	description
filterMode	Int	Filter type， 0: unfilter， 1: Match start string，

		2:RSSI filter, 3: Match start string+RSSI filter。
filterStr	string	tag’s start string
rssi	Int	-31~ -98

errStr：Error message。

Return：
True operation success;
False operation fail.

2.4.54 GetFilterParameter

Purpose：get tag filtering parameters

Usage：

```
public FilterParameter GetFilterParameter(out string errStr)
```

Parameter：

errStr：Error message。

Return：

FilterParameter class, refer to 2.4.53

2.4.55 SetLightMode

Purpose：Set the lighting mode for inventory and alarm

Usage：

```
public bool SetLightMode(LightMode lightmode, out String errStr)
```

Parameter：

lightmode：

StartMode：0:off；1:on；

FlashMode：0:No flicker；1:flicker

errStr：Error message。

Return：
True operation success;
False operation fail.

2.4.56 ReaderInfoQuery

Purpose：inquire firmware version information.

Usage：

```
public ReaderInfo ReaderInfoQuery(byte address, out String errStr)
```

Parameter：

errStr：Error message。

Return：

property	type	description
Version	string	Version number string
Name	string	App name
PoweredOnTime	Int	Power-on time of the device

2.4.57 GetSerialNumber

Purpose: get serial number of the device。

Usage:

```
public string GetSerialNumber(out String errStr)
```

Parameter:

errStr: Error message。

Return:

Serial number string

3 Appendix A Checksum calculation method(C language)

```
unsigned char CheckSum(unsigned char *uBuff, unsigned char uBuffLen)
{
    unsigned char i,uSum=0;
    for(i=0;i<uBuffLen;i++)
    {
        uSum = uSum + uBuff[i];
    }
    uSum = (~uSum) + 1;
    return uSum;
}
```

4 Appendix B Error code table

code	Definition
00H	Executed successfully
20H	instruction or data received is incomplete
21H	wrong CRC check
22H	wrong instruction parameter
23H	Instruction length error
26H	unsupported instruction types
30H	Execution failure
31H	unknown error