Hopeland

Focus on RFID core technology

Hopeland RFID READER - PC

Development Guide

.net

Editor: Tang yue

Shenzhen Hopeland Technologies Co., Ltd

V 2.17.0

Menu

1. Overview

1.1 Content Overview

For facilitating the user software development, we provide library running in the .Net platform. The library is written in C # language and encapsulated into a standard dynamic link library, the development environment for the .Net Framework 2.0.

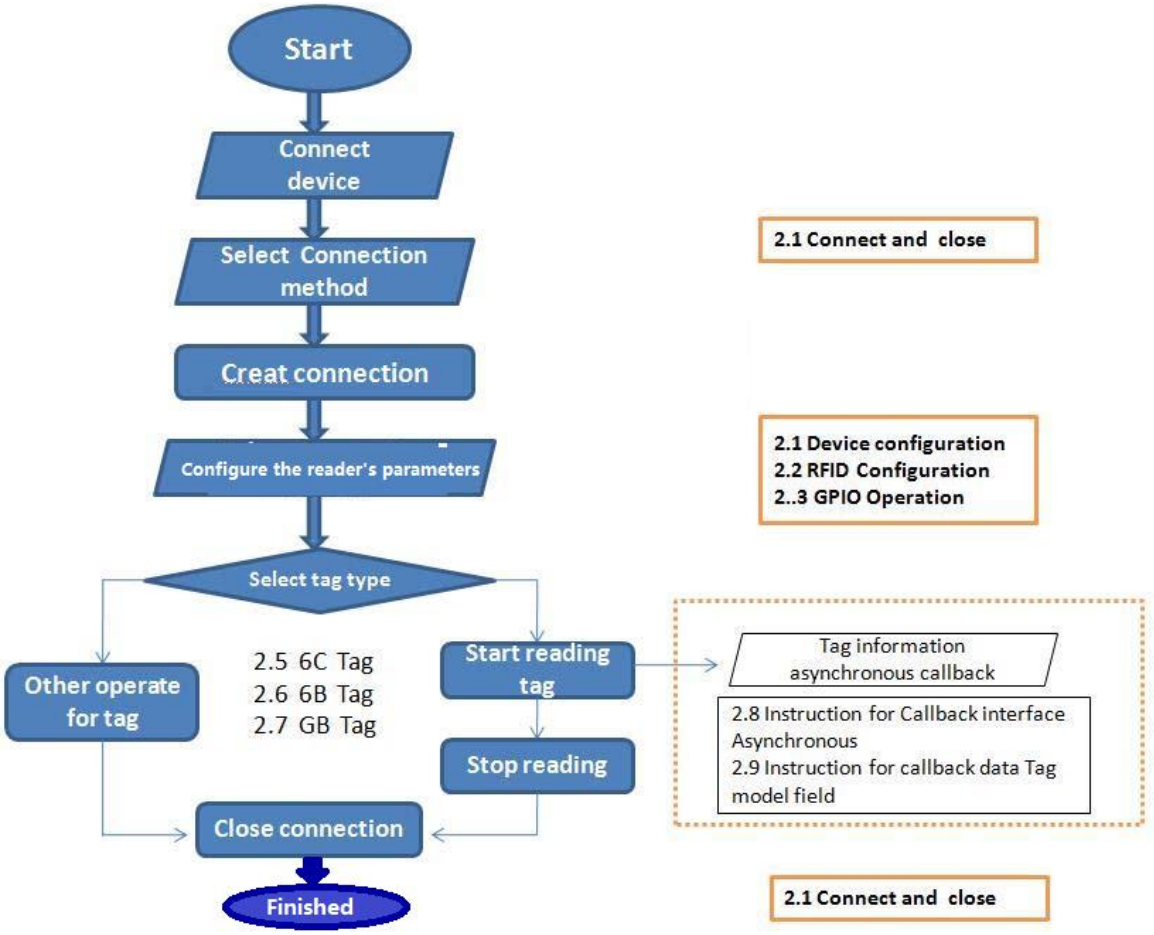
This guide introduces the corresponding technical indicators, application development notes and precautions, application interface function description and so on.

Statement:

1. For the operation of RFID management software (DEMO), please refer to the user manual of related products.

2. If this document still couldn't meet your development requirements, please kindly contact with manufacturer for support

1.2 Development process



1.3 Applicable Models

This document lists all RFID devices' API, the following table lists supportable function of different models (please refer to specific notes of function module details)

Function module	Applicable models
Connection operation	CL7206A/CL7206B/CL720C series
Device configuration	All products
RFID configuration	All products
GPO operation	CL7206B/CL7206C series
6C tag operation	All products
6B tag operation	All products

1.4 Copyright Statement

2. API modules function description

2.1 connect & disconnect

2.1.1 Create serial port connect

Namespace	ClouReaderAPI.CLReader
Function	static bool CreateSerialConn(string seriaParam, IAsynchronousMessage log)
Parameter	seriaParam: serial port connection Parameter, eg:"COM1:115200" log: Data callback interface, all tags data will be called back from this interface.
Return	True: successful; false: failed
Remark	1. Connection created by this method, "seriaParam" will be this connection channel's ID, which is different with other connection channel. 2. log: Data callback interface, please refer to 2.7callback interface introductions for more information.

2.1.2 Create TCP connection

Namespace	ClouReaderAPI.CLReader
Function	static bool CreatTcpConn(string tcpParam, IAsynchronousMessage log)
Parameter	tcpParam: TCP connection Parameter, eg:"192.168.1.116:9090" log: Data callback interface, all tags data will be called back from this interface.
Return	True: successful; false: failed
Remark	1. Connection created by this method , "tcpParam" will be this connection channel's ID, which is different with other connection channel. 2. log: Data callback interface, Please refer to 2.7callback interface introductions for more information

2.1.3 Create 485 connection

Namespace	ClouReaderAPI.CLReader
Function	static bool Create485Conn(string _485Param, IAsynchronousMessage log)
Parameter	_485Param: RS485 connection Parameter, eg:"1:COM1:115200", "1" as 485 connection address. log: Data callback interface, all tags data will be called back from this interface.
Return	True: successful; false: failed
Remark	1. Connection created by this method, "485Param" will be this connection channel's ID which is different with other connection channel. 2. log: Data callback interface, Please refer to 2.7callback interface introductions for more information.

2.1.4 Create USB connection

Namespace	ClouReaderAPI.CLReader
Function	static bool CreateUsbConn(string usbParam, IntPtr Handle, IAsynchronousMessage log)
Parameter	usbParam: usb connect parameter。 Handle: window handle or service handle log: Data callback interface, all the tag data will be callback from the interface object
Return	True: successful; false: failed
Remark	1. parameter "usbParam", through same namespace static List<string> GetUsbHidDeviceList() to get connection parameter of USB device list 2. log data callback interface, refer to 2.7callback interface introductions for detailed information. 3. Default name is "UHF READER" after USB connected

2.1.5 Check connection status

Namespace	ClouReaderAPI.CLReader
Function	static bool CheckConnect(string connectID)
Parameter	connectID: connection ID, eg:"192.168.1.116:9090"
Return	True: successful; false: failed
Remark	"connectID" is the connection parameter when creates connection

2.1.6 Close single connection

Namespace	ClouReaderAPI.CLReader
Function	static void CloseConn(string connectID)
Parameter	connectID: connection ID, eg:"192.168.1.116:9090"
Return	None
Remark	"connectID" is the connection parameter when creates connection

2.1.7 Close all connections

Namespace	ClouReaderAPI.CLReader
Function	static void CloseAllConnect()
Parameter	None
Return	None
Remark	This method will close all created connections.

2.1.8 Open TCP server listening

Namespace	ClouReaderAPI.CLReader
Function	static Boolean OpenTcpServer(string serverIP,string serverPort,IAsynchronousMessage log)
Parameter	serverIP: the IP been listening。 serverPort: the port been listening Log: all data callback interface after connected , details refer to: 2.7callback interface introductions
Return	Listen successful or not
Remark	1. Open server listening 2. When device in client mode, will connect related listening port automatically eg.: testing IP and port under client mode of reader as"IP:192.168.1.75 Port:9090" Code example: OpenTcpServer("192.168.1.75","9090",log); 3. The default connecting IP (client mode) is "192.168.1.1", port is "9090"

2.1.9 Close TCP server listening

Namespace	ClouReaderAPI.CLReader
Function	static void CloseTcpServer()
Parameter	None
Return	None
Remark	Close TCP server listening

2.1.10 Get server listening status

Namespace	ClouReaderAPI.CLReader
Function	static bool GetServerStartUp()
Parameter	None
Return	Server listening or not
Remark	True: listening， false: not in listening

2.1.11 API language type set

Namespace	ClouReaderAPI.CLReader
Function	static void SetAPILanguageType(int type)
Parameter	Type: eAPILanguage.Chinese is Chinese, eAPILanguage.English is English
Return	None
Remark	Change the API information prompt language type, non-persistent, default is English

2.1.12 API language type query

Namespace	ClouReaderAPI.CLReader
Function	static Int GetAPILanguageType()
Parameter	None
Return	"0" is English, "1" is Chinese
Remark	The default language is English

2.2 Device configuration

2.2.1 IP configuration

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 Set ReaderNetworkPortParam(String ConnID, String iP, String mask, String gateway)

Parameter	// ConnID: connection identifier
	// iP: IP address, e.g.: "192.168.1.116"
	// mask: Subnet Mask, e.g.: "255.255.255.0"
	// gateway: gateway, e.g.: "192.168.1.1"

Return	0: successful; others: failed
Remark	1. This method will close all created connection.

2.2.2 Query IP configuration

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderNetworkPortParam(String ConnID)
Parameter	// ConnID: connection identifier
Return	IP add | Subnet mask | Gateway, e.g. :"192.168.1.116|255.255.255.0|192.168.1.1"
Remark	

2.2.3 Stop instruction

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 Stop(String ConnID)
Parameter	// ConnID: connection identifier
Return	0: successful; others: failed
Remark	1. This method will make reader stop current workings. 2. Stop inventory reading using this method

2.2.4 Set device time

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderUTC(String ConnID, String param)
Parameter	// ConnID: connection identifier // param: time parameter"yyyy.MM.dd HH:mm:ss", e.g.:"1970.01.01 00:00:00"
Return	0: successful; others: failed
Remark	None

2.2.5 Query device time

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderUTC(String ConnID)
Parameter	// ConnID: connection identifier
Return	Time Parameter"yyyy.MM.dd HH:mm:ss", e.g.:"1970.01.01 00:00:00"
Remark	None

2.2.6 Set serial port parameter

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderSerialPortParam(String ConnID, eBaudrate baudRate)

Parameter	// ConnID: connection identifier
	// baudRate:
	eBaudrate._9600bps,
	eBaudrate._19200bps,
	eBaudrate._115200bps,
	eBaudrate._230400bps,
	eBaudrate._460800bps.

Return	0: successful; others: failed
Remark	None

2.2.7 Get serial port parameter

Namespace	ClouReaderAPI.CLReader._Config
Function	static eBaudrate GetReaderSerialPortParam(String ConnID)
Parameter	// ConnID: connection identifier
Return	eBaudrate._9600bps, eBaudrate._19200bps, eBaudrate._115200bps, eBaudrate._230400bps, eBaudrate._460800bps.
Remark	None

2.2.8 MAC address setting

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderMacParam(String ConnID, String param)
Parameter	// ConnID: connection identifier // param: MAC address format"00-00-00-00-00-00"
Return	0: successful; others: failed
Remark	None

2.2.9 Get MAC address

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderMacParam(String ConnID)
Parameter	// ConnID: connection identifier
Return	MAC address
Remark	MAC address format"00-00-00-00-00-00"

2.2.10 RS485 address setting

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReader485(String ConnID, String param)
Parameter	// ConnID: connection identifier // param: 0~255
Return	0: successful; others: failed
Remark	None

2.2.11 RS485 address Query

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReader485(String ConnID)
Parameter	// ConnID: connection identifier
Return	RS485 address
Remark	None

2.2.12 DHCP configuration

Namespace	ClouReaderAPI.CLReader._Config
Function	static String SetDHCP(String ConnID, Boolean param)
Parameter	// ConnID: connection identifier // param： true:open, false:close
Return	0: successful; others: failed
Remark	None

2.2.13 Query DHCP

Namespace	ClouReaderAPI.CLReader._Config
Function	static Boolean GetDHCP(String ConnID)
Parameter	// ConnID: connection identifier
Return	False:close， true:open
Remark	None

2.2.14 Server/client mode configuration

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderServerOrClient(String ConnID, eWorkMode workMode,String ip,String port)

Parameter	// ConnID: connection identifier
	// workMode: eWorkMode.Server, eWorkMode.Client
	ip: such as"192.168.1.75"
	port: such as "9090"

Return	0: successful; others: failed
Remark	When reader at the server mode, Parameter ip is invalid, can input any character string

2.2.15 Query Server/client mode

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderServerOrClient(String ConnID)
Parameter	// ConnID: connection identifier
Return	Server| "server port"　or　Client | "client port IP" | "client port"
Remark	None

2.2.16 Query device information

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderInformation(String ConnID)
Parameter	// ConnID: connection identifier
Return	Application version | reader name | reader power on time
Remark	Reader power on time unit is "second"

2.2.17 Query Baseband version

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderBaseBandSoftVersion(String ConnID)
Parameter	// ConnID: connection identifier
Return	Baseband version
Remark	None

2.2.18 Query antenna standing wave ratio

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetAntennaStandingWaveRatio(String ConnID，eAntennaNo antNo)
Parameter	// ConnID: connection identification，antNo: Antenna number enumeration
Return	Forward power detection | backward power detection
Remark	If the difference is greater than 25, the antenna is connected, otherwise, the antenna is not connected

2.3 RFID configuration

2.3.1 Factory reset

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderRestoreFactory(String ConnID)
Parameter	// ConnID: connection identifier
Return	0: successful, other: failed
Remark	Restore all setting back to factory status. Please use this interface with caution (MAC address and reader time will not change)

2.3.2 Base band parameter setting

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetEPCBaseBandParam(String ConnID,Int32 basebandMode,Int32 qValue,Int32 session,Int32 searchType)

Parameter	// ConnID: connection identifier
	// basebandMode: EPC Base band speed(0~255, 255 is AUTO)
	(0-Tari=25us, FM0, LHF=40KHz)
	(1-TTari=25us, Miller4, LHF=250KHz)
	(2-Tari=25us, Miller4, LHF=300KHz)
	(3-Tari=6.25us, FM0, LHF=400KHz)
	(255-Auto)
	// qValue: 0~15, reader's initial Q value.
	// session: 0~3
	// searchType: inventory Parameter (0 only use Flag A to inventory, 1 only use Flag B to inventory, 2 use both Flag A and Flag B in turn to inventory).

Return	0: successful; others: failed
Remark	

2.3.3 Query Base band parameter setting

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetEPCBaseBandParam(String ConnID)
Parameter	// ConnID: connection identifier
Return	basebandMode|qValue|session|searchType

Remark	// basebandMode: EPC Base band speed(0~255, 255 is AUTO)
	(0-Tari=25us, FM0, LHF=40KHz)
	(1-TTari=25us, Miller4, LHF=250KHz)
	(2-Tari=25us, Miller4, LHF=300KHz)
	(3-Tari=6.25us, FM0, LHF=400KHz)
	(255-Auto)
	// qValue: 0~15, reader's initial Q value.
	// session: 0~3
	// searchType: inventory Parameter (0 only use Flag A to inventory, 1 only use Flag B to inventory, 2 use both Flag A and Flag B in turn to inventory).

2.3.4 Power setting

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetANTPowerParam(String ConnID, Dictionary<Int32, Int32> dicPower)

Parameter	// ConnID: connection identifier
	//dicPower: antenna number and power level key-value pair
	e.g.: SetANTPowerParam("192.168.1.116:9090",new Dictionary<Int32, Int32>(){{1,30},{2,30}});
	This method will set Ant1 and Ant2 's power to 30

Return	0: successful; others: failed
Remark	Set each antenna's power

2.3.5 Query antenna power set

Namespace	ClouReaderAPI.CLReader._Config
Function	static Dictionary<Int32, Int32> GetANTPowerParam(String ConnID)
Parameter	// ConnID: connection identifier
Return	Can refer to the Dictionary<Int32, Int32> of 2.3.4 antenna power setting
Remark	

2.3.6 Tag data uploading

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetTagUpdateParam(String ConnID, Int32 repeatTimeFilter, Int32 RSSIFilter)

Parameter	// ConnID: connection identifier
	// repeatTimeFilter: repeat tag uploading filtration time
	// RSSIFilter: RSSI filter

Return	0: successful, others: failed
Remark	repeatTimeFilter value range: 0 ~ 65535. RSSIFilter value range: 0 ~ 255

2.3.7 Tag date upload Parameter Query

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetTagUpdateParam(String ConnID)
Parameter	// ConnID: connection identifier
Return	repeatTimeFilter|RSSIFilter。

Remark	// repeatTimeFilter: repeat tag upload filter time (unit: 10ms)
	// RSSIFilter: RSSI filter
	repeatTimeFilter value range: 0 ~ 65535
	RSSIFilter value range: 0 ~ 255

2.3.8 Get Device property

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderProperty(String ConnID)
Parameter	// ConnID: connection identifier
Return	Minimum output power | maximum output power | number of antennas | band list | list of RFID protocols.
Remark	Power unit is dB

2.3.9 RF frequency range Set

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderRF(String ConnID, eRF_Range eRF_Range)

Parameter	// ConnID: connection identifier
	// eRF_Range:
	eRF_Range.GB_920_to_925MHz
	eRF_Range.GB_840_to_845MHz
	eRF_Range.GB_920_to_925MHz_and_GB_840_to_845MHz
	eRF_Range.FCC_902_to_928MHz
	eRF_Range.ETSI_866_to_868MHz
	eRF_Range.JP_916_to_921MHz
	eRF_Range.TW_922_to_927MHz
	eRF_Range.ID_923_to_925MHz
	eRF_Range.RUS_866_to_867MHz

Return	0: successful; others: failed
Remark	None

2.3.10 Get RF frequency range

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderRF(String ConnID)
Parameter	// ConnID: connection identifier
Return	eRF_Range.GB_920_to_925MHz eRF_Range.GB_840_to_845MHz eRF_Range.GB_920_to_925MHz_and_GB_840_to_845MHz eRF_Range.FCC_902_to_928MHz eRF_Range.ETSI_866_to_868MHz eRF_Range.JP_916_to_921MHz eRF_Range.TW_922_to_927MHz eRF_Range.ID_923_to_925MHz eRF_Range.RUS_866_to_867MHz
Remark	None

2.3.11 Set RF working frequency

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderWorkFrequency(String ConnID, eWF_Mode wfMode, List<eGB_840_to_845MHz> ListGB_840_to_845MHz)

Parameter	// ConnID: connection identifier
	//wfMode: eWF_Mode.Specified, eWF_Mode.Auto.
	//eGB_840_to_845MHz:
	eGB_840_to_845MHz._840_625f,
	eGB_840_to_845MHz._840_875f,
	eGB_840_to_845MHz._841_125f,
	eGB_840_to_845MHz._841_375f,
	eGB_840_to_845MHz._841_625f,
	eGB_840_to_845MHz._841_875f,
	eGB_840_to_845MHz._842_125f,
	eGB_840_to_845MHz._842_375f,
	eGB_840_to_845MHz._842_625f,
	eGB_840_to_845MHz._842_875f,
	eGB_840_to_845MHz._843_125f,
	eGB_840_to_845MHz._843_375f,
	eGB_840_to_845MHz._843_625f,
	eGB_840_to_845MHz._843_875f,
	eGB_840_to_845MHz._844_125f,
	eGB_840_to_845MHz._844_375f.

Return	0: successful; others: failed
Remark	This function has many overload，other frequency setting only need change function eGB_840_to_845MHz to GB_920_to_925MHz, GB_840_to_845MHz, GB_920_to_925MHz_and_GB_840_to_845MHz, FCC_902_to_928MHz, ETSI_866_to_868MHz, JP_916_to_921MHz, TW_922_to_927MHz, ID_923_to_925MHz, RUS_866_to_867MHz then you could set related frequency points

2.3.12 Get RF working frequency

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderWorkFrequency(String ConnID)
Parameter	// ConnID: connection identifier
Return	Mode | Frequency range | Frequency points e.g.: Auto|GB_920_to_925MHz|920.625,920.875
Remark	None

2.3.13 Set device Auto-sleep mode

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderAutoSleepParam(String ConnID, bool Switch, String time)
Parameter	// ConnID: connection identifier, Switch: on / off, time: sleep time
Return	0: successful; others: failed
Remark	Switch:true:open, false:close Sleep time unit:10ms

2.3.14 Get device Auto-sleep mode

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderAutoSleepParam(String ConnID)
Parameter	// ConnID: connection identifier
Return	Close or Open| "sleep time"
Remark	Unit is 10ms

2.3.15 Network self-check Setting

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderSelfCheck(String ConnID, bool Switch, String ip)
Parameter	// ConnID: connection identifier, Switch: on / off, ip: ip address
Return	0: successful; others: failed
Remark	Switch:true:open, false:close

2.3.16 Query Network self-check status

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderSelfCheck(String ConnID)
Parameter	// ConnID: connection identifier
Return	Close or Open|"IP address"
Remark	

2.3.17 Set antenna enabled

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderANT(String ConnID, eAntennaNo antNum)
Parameter	// ConnID: connection identifier, antNum: antenna number enumeration
Return	0: successful; others: failed
Remark	Enable ant1 and ant2, e.g.: eAntennaNo._1|eAntennaNo._2

2.3.18 Get antenna enabled

Namespace	ClouReaderAPI.CLReader._Config
Function	static eAntennaNo GetReaderANT(String ConnID)
Parameter	// ConnID: connection identifier
Return	Refer to 2.3.15 - eAntennaNo
Remark	None

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderANT2(String ConnID)
Parameter	// ConnID: connection identifier
Return	Antenna numbers already enabled, numbers apart by "," such as 1,6,8
Remark	None

2.4 GPIO operation

2.4.1 GPI trigger parameter configuration

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderGPIParam(String ConnID, eGPI GPINum, eTriggerStart triggerStart,eTriggerCode triggerCode, eTriggerStop triggerStop, String DelayTime)

Parameter	// ConnID: connection identifier
	// GPINum: eGPI._1,eGPI._2,eGPI._3,eGPI._4;
	
	//triggerStart: eTriggerStart.OFF,eTriggerStart.Low_level,eTriggerStart.High_level,
	eTriggerStart.Rising_edge,eTriggerStart.Falling_edge,eTriggerStart.Any_edge;
	
	//triggerCode: triggerCode.Single_Antenna_read_EPC,
	triggerCode.Single_Antenna_read_EPC_and_TID,
	triggerCode.Double_Antenna_read_EPC,
	triggerCode.Double_Antenna_read_EPC_and_TID,
	triggerCode.Four_Antenna_read_EPC,
	triggerCode.Four_Antenna_read_EPC_and_TID;
	
	//triggerStop: eTriggerStop.OFF,eTriggerStop.Low_level,eTriggerStop.High_level,
	eTriggerStop.Rising_edge,eTriggerStop.Falling_edge,eTriggerStop.Any_edge,
	eTriggerStop.Delay;
	
	e.g.: SetReaderGPIParam("192.168.1.116:9090",eGPI._2,eTriggerStart.Low_level,
	triggerCode.Double_Antenna_read_EPC_and_TID,eTriggerStop.Dealy,"100");

	This sample code is to set a low level trigger start on GPI 2 port and execute trigger code when the GPI 2 be triggered by low level, end after 1000ms delay.
Return	0: successful; others: failed
Remark	Stop delay time: unit is 10ms (when Trigger stop condition is "Delay stop" effective) Related operation performance refers to the RFIDreader demo software Detailed GPI trigger parameter callback, please refer to 2.7 callback interface instruction - GPIControlMsg();

2.4.2 Get GPI trigger Parameter

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderGPIParam(String ConnID,eGPI GPINum)

Parameter	// ConnID: connection identifier
	// GPONum: GPI._1,GPI._2,GPI._3,GPI._4.
	e.g. GetReaderGPIParam("192.168.1.116:9090",GPI._1);

Return	"trigger GPI number|Trigger start condition|Trigger code|Trigger stop condition|end delay time" e.g. "GPI1|Low level|Double Antenna read EPC|Delay|100"
Remark	Stop delay time: unit is 10ms (when Trigger stop condition is" Delay stop" effective) Related operation performance refers to the RFIDreader demo software

2.4.3 Get GPI state

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderGPIState(String ConnID)
Parameter	// ConnID: connection identifier e.g. GetReaderGPIState("192.168.1.116:9090");
Return	e.g."1,Low & 2,High", this return: 1# GPI port is in low level, 2# GPI port in high level
Remark	This method is to get GPI present level state, no business with trigger Detailed GPI trigger parameter callback, please refer to 2.7 callback interface instruction - GPIControlMsg();

2.4.4 GPO level operation

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderGPOState(String ConnID, Dictionary<eGPO, eGPOState> dicState)

Parameter	// ConnID: connection identifier
	// dicState: GPO serial number and level related key-value pair
	e.g. SetReaderGPOState("192.168.1.116:9090",new Dictionary<eGPO, eGPOState> {{1,0},{2,1}});
	this method is to set #1 GPO port into low level (0), and set #2 GPO port into High level (1)

Return	0: successful; others: failed
Remark	None

2.4.5 Set Wiegand parameter

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderWG(String ConnID, eWiegandSwitch wiegandSwitch, eWiegandFormat wiegandFormat, eWiegandDetails param)

Parameter	// ConnID: connection identifier
	// eWiegandSwitch:
	eWiegandSwitch.Close,
	eWiegandSwitch.Open.
	// eWiegandFormat:
	eWiegandFormat.Wiegand26,
	eWiegandFormat.Wiegand34,
	eWiegandFormat.Wiegand66
	// eWiegandDetails:
	eWiegandDetails.end_of_the_EPC_data,
	eWiegandDetails.end_of_the_TID_data.
	
	e.g. SetReaderWG("192.168.1.116:9090",eWiegandSwitch.Open, eWiegandFormat.Wiegand26,eWiegandDetails.end_of_the_TID_data);
	this method is to open Wiegand, communication format is Wiegand 26，appointed transferred data is TID end data

Return	0: successful; others: failed
Remark	None

2.4.6 Get Wiegand parameter

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderWG(String ConnID)
Parameter	// ConnID: connection identifier
Return	// Return: Wiegand switch|Wiegand format|Wiegand transferred data e.g. Return "Open|Wiegand66|end_of_the_EPC_data" this return means switch is open, communication format is Wiegand 66, appointed transferred data is EPC end data
Remark	None

2.5 6C tag operation

2.5.1 Reading

Namespace	ClouReaderAPI.CLReader._Tag6C
Function1	static Int32 GetEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType)

	// only read EPC
	// ConnID: connection identifier
	// antNum: Antenna number enumeration.
	Appoint Antenna 1 and 2 working at same time; e.g.: eAntennaNo._1|eAntennaNo._2
	// ReadType: read type enumeration, Single or Inventory (one-time or cyclically reading)

Function2	static Int32 GetEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType, String accessPassword) // accessPassword: Tag access password

Function3	static Int32 GetEPC_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sEPC)
	// match EPC Read EPC
	// sEPC: should be matched EPC value(Hexadecimal string)

Function4	static Int32 GetEPC_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sEPC, Int32 matchWordStartIndex)
	// match EPC Read EPC
	// matchWordStartIndex: match Data starting index

Function5	static Int32 GetEPC_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sEPC, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password

Function6	static Int32 GetEPC_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sTID)
	// match TID Read EPC
	// sTID: should be matched TID value(Hexadecimal string)

Function7	static Int32 GetEPC_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sTID, Int32 matchWordStartIndex)
	// match TID Read EPC
	// matchWordStartIndex: match Data starting index

Function8	static Int32 GetEPC_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sTID, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password
Function9	static Int32 GetEPC_TID(String ConnID, eAntennaNo antNum, e ReadType ReadType) // Read EPC and TID
Function10	static Int32 GetEPC_TID(String ConnID, eAntennaNo antNum, e ReadType ReadType, String accessPassword) // accessPassword: Tag access password
Function11	static Int32 GetEPC_TID_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sEPC) // match EPC, Read EPC and TID
Function12	static Int32 GetEPC_TID_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sEPC, Int32 matchWordStartIndex) // match EPC, Read EPCand TID
Function13	static Int32 GetEPC_TID_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sEPC, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password
Function14	static Int32 GetEPC_TID_MatchTID(String ConnID, eAntennaNo antNum, e ReadType

	ReadType, String sTID) // match TID, Read EPC and TID
Function15	static Int32 GetEPC_TID_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sTID, Int32 matchWordStartIndex) // match TID, Read EPC and TID
Function16	static Int32 GetEPC_TID_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType, String sTID, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password

Function17	static Int32 GetEPC_TID_UserData(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen)
	// Read EPC,TID and UserData
	// ReadStart reader user area's starting index
	// ReadLen reader user block's length (unit: Word)

Function18	static Int32 GetEPC_TID_UserData(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen, String accessPassword) // accessPassword: Tag access password
Function19	static Int32 GetEPC_TID_UserData_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen, String sEPC) // match EPC, Read EPC, TID and UserData
Function20	static Int32 GetEPC_TID_UserData_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen, String sEPC, Int32 matchWordStartIndex) // match EPC, Read EPC, TID and UserData
Function21	static Int32 GetEPC_TID_UserData_MatchEPC(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen, String sEPC, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password
Function22	static Int32 GetEPC_TID_UserData_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen, String sTID) // match TID, Read EPC, TID and UserData
Function23	static Int32 GetEPC_TID_UserData_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen, String sTID, Int32 matchWordStartIndex) // match TID, Read EPC, TIDand UserData
Function24	static Int32 GetEPC_TID_UserData_MatchTID(String ConnID, eAntennaNo antNum, e ReadType ReadType,Int32 ReadStart, Int32 ReadLen, String sTID, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password
Parameter	refers to each function instruction
Return	True: successful; false: failed
Remark	1. For detailed Return , please kindly follow Appendix A 2. Stop inventory (cycle) reading using "stop" instruction. 3. Difference between inventory and single reading is that single read automatically stops reading after one time reading, but inventory read requires a stop function to stop reading. 4. The same part for inventory and single reading is that after the last tag data is uploaded, they will notify PC side through asynchronous callback that tag upload finished, please refer to 2.7 callback interface instrunction - OutPutTagsOver();

2.5.2 Writing

2.5.2.1 Write EPC

Namespace	ClouReaderAPI.CLReader._Tag6C

Function1	static Int32 WriteEPC(String ConnID, eAntennaNo antNum, String sWriteData)
	// ConnID: connection identifier
	// antNum: Antenna number enumeration.
	// sWriteData: data to be written (Hexadecimal string)
	Appoint antenna 1 and 2 working at same time, e.g.: eAntennaNo._1|eAntennaNo._2

Function2	static Int32 WriteEPC_MatchEPC(String ConnID, eAntennaNo antNum, String sWriteData, String sMatchData, Int32 matchWordStartIndex)
	// match EPC WriteEPC
	// sMatchData EPC data
	// matchWordStartIndex match Data starting index

Function3	static Int32 WriteEPC_MatchEPC(String ConnID, eAntennaNo antNum, String sWriteData, String sMatchData, Int32 matchWordStartIndex, String accessPassword)
	// match EPC Write EPC
	// accessPassword Tag access password

Function4	static Int32 WriteEPC_MatchTID(String ConnID, eAntennaNo antNum, String sWriteData, String sMatchData, Int32 matchWordStartIndex)
	// match TID Write EPC
	// sMatchData to be matched TID data
	// matchWordStartIndex match Data starting index

Function5	static Int32 WriteEPC_MatchTID(String ConnID, eAntennaNo antNum, String sWriteData, String sMatchData, Int32 matchWordStartIndex, String accessPassword) // match TID Write EPC
Parameter	Please refer to Function Remark.
Return	True: successful; others: failed
Remark	1. Suggest to use matched TID to write, means to use "Function4"and "Function5". 2. For detailed Return, refers to Appendix A 3. If the length of EPC data to be written not a multiple of 4, then 0 will be added up, e.g. if we want to write 201807 to EPC area, actually the data 20180700 will be written to tag.

2.5.2.2 Write Userdata

Namespace	ClouReaderAPI.CLReader._Tag6C

Function1	static Int32 WriteUserData(String ConnID, eAntennaNo antNum, String sWriteData,Int32 offset)
	// ConnID: connection identifier
	// antNum: Antenna number enumeration.
	// sWriteData: data to be written (Hexadecimal string)
	//offset: the offset of user area, that is, the number of 0 before writing data
	Appoint antenna 1and 2 working at same time, e.g.: eAntennaNo._1|eAntennaNo._2

Function2	static Int32 WriteUserData_MatchEPC(String ConnID, eAntennaNo antNum, String sWriteData, Int32 offset,String sMatchData, Int32 matchWordStartIndex)

	// match EPC, WriteEPC
	// sMatchData, EPC data to be matched （Hexadecimal string）
	// matchWordStartIndex, match Data starting index

Function3	static Int32 WriteUserData_MatchEPC(String ConnID, eAntennaNo antNum, String sWriteData,Int32 offset,String sMatchData, Int32 matchWordStartIndex, String accessPassword)
	// match EPC, WriteEPC
	// accessPassword　Tag access password

Function4	static Int32 WriteUserData_MatchTID(String ConnID, eAntennaNo antNum, String sWriteData,Int32 offset,String sMatchData, Int32 matchWordStartIndex)
	// match TID, WriteEPC
	// sMatchData ,TID data to be matched （Hexadecimal string）
	// matchWordStartIndex, match Data starting index

Function5	static Int32 WriteUserData_MatchTID(String ConnID, eAntennaNo antNum, String sWriteData, Int32 offset,String sMatchData, Int32 matchWordStartIndex, String accessPassword) // match TID WriteEPC
Parameter	Please refer to FunctionRemark.
Return	True: successful; others: failed
Remark	1. Suggest to use matched TID to write, means to use "Function4" and "Function5". 2. For detailed Return ,refers to Appendix A

2.5.2.3 Write password

Namespace	ClouReaderAPI.CLReader._Tag6C
Function1	static Int32 WriteAccessPassWord(String ConnID, eAntennaNo antNum, String sWriteData) //　Write Tag access password // sWriteData: password content （8 Hexadecimal string data）
Function2	static Int32 WriteAccessPassWord(String ConnID, eAntennaNo antNum, String sWriteData, String accessPassword) //　Write Tag access password // accessPassword: original Tag access password （8 Hexadecimal string data）

Function3	static Int32 WriteAccessPassWord_MatchTID(String ConnID, eAntennaNo antNum, String sWriteData, String sMatchData, Int32 matchWordStartIndex, String accessPassword)
	//　Write Tag access password
	// sMatchData: TID data to be matched
	// matchWordStartIndex: match Data starting index

Function4	static Int32 WriteDestroyPassWord(String ConnID, eAntennaNo antNum, String sWriteData) //　Write the kill password
Function5	static Int32 WriteDestroyPassWord(String ConnID, eAntennaNo antNum, String sWriteData, String accessPassword) //　Write the kill password // accessPassword：original Tag access password （8 Hexadecimal string data）
Function6	static Int32 WriteDestroyPassWord_MatchTID(String ConnID, eAntennaNo antNum, String sWriteData, String sMatchData, Int32 matchWordStartIndex, String accessPassword) //　Write the kill password // sMatchData: TID data to be matched

	// matchWordStartIndex: match Data starting index
Parameter	Please refer to FunctionRemark.
Return	True: successful; others: failed. refers to Appendix A
Remark	

2.5.3 Lock tag

Namespace	ClouReaderAPI.CLReader.Tag_6C

Function1	static Int32 Lock(String ConnID, eAntennaNo antNum, eLockArea lockArea, eLockType lockType)
	// ConnID: connection identifier
	// antNum: antenna number
	// lockArea: lock area enumeration
	// lockType: lock type enumeration

Function2	static Int32 Lock_MatchEPC(String ConnID, eAntennaNo antNum, eLockArea lockArea, eLockType lockType, String sMatchData, Int32 matchWordStartIndex)
	// sMatchData: EPC data to be matched (Hexadecimal string)
	// matchWordStartIndex: match Data starting address, unit is word

Function3	static Int32 Lock_MatchEPC(String ConnID, eAntennaNo antNum, eLockArea lockArea, eLockType lockType, String sMatchData, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password

Function4	static Int32 Lock_MatchTID(String ConnID, eAntennaNo antNum, eLockArea lockArea, eLockType lockType, String sMatchData, Int32 matchWordStartIndex)
	// sMatchData: TID data to be matched(Hexadecimal string)
	// matchWordStartIndex: match Data starting address, unit is word

Function5	static Int32 Lock_MatchTID(String ConnID, eAntennaNo antNum, eLockArea lockArea, eLockType lockType, String sMatchData, Int32 matchWordStartIndex, String accessPassword) // accessPassword: Tag access password
Parameter	// refer to above method Remark
Return	True: successful; false: failed refers to Appendix A
Remark	

2.5.4 Tag killing (destroy)

Namespace	ClouReaderAPI.CLReader.Tag_6C

Function1	static Int32 Destroy(String ConnID, eAntennaNo antNum, String destroyPassword)
	// ConnID: connection identifier
	// antNum: antenna number
	// destroyPassword: kill password (Hexadecimal string)

Function2	static Int32 Destroy_MatchEPC(String ConnID, eAntennaNo antNum, String destroyPassword, String sMatchData, Int32 matchWordStartIndex) // sMatchData: EPC data to be matched(Hexadecimal string)

	// matchWordStartIndex: match Data starting address, unit is word
Function3	static Int32 Destroy_MatchTID(String ConnID, eAntennaNo antNum, String destroyPassword, String sMatchData, Int32 matchWordStartIndex) // sMatchData: TID data to be matched (Hexadecimal string)
Parameter	// refer to above method Remark
Return	True: successful; false: failed refers to Appendix A
Remark	

2.5.5 Get QTParameter

Namespace	ClouReaderAPI.CLReader.Tag_6C

Function1	static String GetQtTagOption (String ConnID, eAntennaNo antNum)
	// ConnID: connection identifier
	// antNum: antenna number

Function2	static String GetQtTagOption (String ConnID, eAntennaNo antNum, String accessPassword) // accessPassword: Access Password (Hexadecimal string)

Function3	static String GetQtTagOption_MatchEPC (String ConnID, eAntennaNo antNum, String sMatchData, Int32 matchWordStartIndex, String accessPassword)
	// sMatchData: matched TID data(Hexadecimal string)
	// matchWordStartIndex: match Data starting address, unit is word

Function4	static String GetQtTagOption_MatchTID (String ConnID, eAntennaNo antNum, String sMatchData, Int32 matchWordStartIndex, String accessPassword)
	// sMatchData: matched TID data(Hexadecimal string)
	// matchWordStartIndex: match Data starting address, unit is word

Parameter	// refers to above method
Return	successfully: qt1 + "|" + qt2 failed: ""
Remark	qt1 value: 1, when OPEN and SECURED state, lower response distance; 0, when OPEN and SECURED state , not lower response distance qt2 value: 1, tag into open mode; 0, tag into private mode

2.5.6 Set QTParameter

Namespace	ClouReaderAPI.CLReader.Tag_6C

Function1	static String SetQtTagOption (String ConnID, eAntennaNo antNum, bool IsNotReduceResponseDistance,bool IsPrivateMode)
	// ConnID: connection identifier
	// antNum: antenna number
	// IsNotReduceResponseDistance: in open and secured state, if lower response distance or not
	// IsPrivateMode: tag into private mode or not

Function2	static String SetQtTagOption (String ConnID, eAntennaNo antNum, bool IsNotReduceResponseDistance, bool IsPrivateMode, String accessPassword) // accessPassword: Access Password (Hexadecimal string)
Function3	static String SetQtTagOption_MatchEPC (String ConnID, eAntennaNo antNum, String

	sMatchData, Int32 matchWordStartIndex, bool IsNotReduceResponseDistance, bool IsPrivateMode, String accessPassword)
	// sMatchData: EPC data need to be matched (Hexadecimal string)
	// matchWordStartIndex: match Data starting address, unit is word

Function4	static String SetQtTagOption_MatchTID (String ConnID, eAntennaNo antNum, String sMatchData, Int32 matchWordStartIndex, bool IsNotReduceResponseDistance, bool IsPrivateMode, String accessPassword)
	// sMatchData: TID data need to be matched (Hexadecimal string)
	// matchWordStartIndex: match Data starting address, unit is word

Parameter	// refers to above method
Return	Successful: successful information; Failed: "" or failed information
Remark	

2.6 6B tag operations (omitted)

If your need to use ISO18000-6B RFID tag for your projects, then ask us to offer 6B development support

2.7 callback interface IAsynchronousMessage Remark

```
// asynchronous callback Message interface
public interface IAsynchronousMessage
{
    void WriteDebugMsg(String msg);
    void WriteLog(String msg);
    void PortConneting(String connID);
    void PortClosing(String connID);
    void OutPutTags(Models.Tag_Model tag);
    void OutPutTagsOver();
    void GPIControlMsg(GPI_Model gpi_model);
}
```

Call back method	Remark
WriteDebugMsg	output debug message
WriteLog	output log message
PortConnecting	Reader at TCP server mode, client-side connection callback When the connection ID is obtained from the callback, the device can be read and written through the connection ID.
PortClosing	When the device is disconnected, the API calls back the connection ID, indicating that the device with the current connection ID is disconnected.

OutPutTags	The output tag information callback, whether it is single read, inventory (circular) read, or getting the tag data in the cache, is the same callback interface. Note: all the tag data is asynchronous callback in API, do not deal with complex logic in the callback to make sure the inside of the API cache cleaning, when complex processing is required, put it into a thread.
OutPutTagsOver	After the last tag is uploaded, a sync end signal is uploaded indicating the end of the current read tag action.
GPIControlMsg	When the GPI trigger parameter is turned on and there is a GPI trigger event, the function will call back the GPI port number where the current event is located, as well as the level status information.

2.8 callback data Tag_Model field introductions

Field	Remark
ReaderName	Reader connection identifier, means which reader are reading data，e.g.: "192.168.1.116:9090"
TagType	Tag type，"6c","6b","gb" 3 types.
EPC	Tag EPC data，Hexadecimal string.
PC	Tag PC value
ANT_NUM	uploaded antenna number
RSSI	RSSI value
TID	Tag TID，Hexadecimal string.
UserData	Tag user area data，Hexadecimal string.
TagetData	Tag reserved area data, include access password and kill password, Hexadecimal string.
Frequency	Tag carrier frequency
Phase	Tag phase
ReadTime	Tag latest reading time

2.9 callback data GPI_Model field introductions

Filed	Remark
ReaderName	Reader connection identifier, means which reader are reading data，e.g.: "192.168.1.116:9090"
GpiIndex	GPI port index, starting at 1, 1 represents GPI1, and so on.
GpiState	0 means low level, 1 means high level
StartOrStop	0 means trigger start, 1 means trigger end
UTC	Sensor trigger UTC time, byte[] type, length is 8, The first 4 bytes is UTC seconds and the last 4 bytes is microseconds
Utc_Time	Sensor trigger UTC time, string type, format is: "yyyy.MM.dd HH:mm:ss.fff"

2.10 Breakpoint continuous transferring

2.10.1 Configuration

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetBreakPointUpload(String ConnID, bool Switch)

Parameter	// ConnID: connection identifier
	// Switch: false:close, true:open
	e.g.: SetBreakPointUpload("192.168.1.116:9090",false);
	this method is for close the breakpoint function (cache function)

Return	0: successful; others: failed
Remark	Start up this function,and the ReadTime field in the Tag Model will be effective。

2.10.2 Get/Query

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetBreakPointUpload(String ConnID)
Parameter	// ConnID: connection identifier e.g.: GetBreakPointUpload("192.168.1.116:9090");
Return	Open or Close。
Remark	None

2.10.3 Query breakpoint cache

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetBreakPointCacheTag(String ConnID)
Parameter	// ConnID: connection identifier
Return	Success: have cache， Null: no cache， Receive Over: data returned completely
Remark	When the PC and the device link layer is interrupted, the data read will be stored in the cache of the reader, (cache support max 5000 times tag reading record, if over this reading times, will use FIFO mode to iterate cache), when this method is called, reader will upload cache data when breakpoint, and the ReadTime field in the Tag Model will be effective.

2.10.4 Clear breakpoint cache

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 ClearBreakPointCache(String ConnID)
Parameter	// ConnID: connection identifier

Return	0: successful; others: failed
Remark	When the PC and the device link layer is interrupted, the data read will be stored in the cache of the reader, When this method is called, the cache when the reader is interrupted will be cleared.

2.11 WiFi operation

2.11.1 Query WiFi switch state

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 GetWiFiSwitchState(String ConnID)
Parameter	// ConnID: connection identifier
Return	0: open； 1: close
Remark	

2.11.2 Set Wi-Fi switch state

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetWiFiSwitchState(String ConnID,bool isOpen)
Parameter	// ConnID: connection identifier // isOpen: Set switch state. True: open； false: close
Return	0: successful; 1: failed
Remark	

2.11.3 Query Wi-Fi adaptor IP

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetReaderWifiIP(String ConnID)
Parameter	// ConnID: connection identifier
Return	IP| Subnet mask | Gateway
Remark	

2.11.4 Set Wi-Fi adaptor IP

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SetReaderWifiIP(String ConnID,String ip,String mask,String gateway)
Parameter	// ConnID: connection identifier // ip: IP address

	//mask: Subnet mask //gateway: network gateway
Return	0: successful; 1: failed
Remark	

2.11.5 Search Wi-Fi hotspot

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 SearchWIFI(String ConnID)
Parameter	// ConnID: connection identifier
Return	0: successful; 1: failed
Remark	

2.11.6 save Wi-Fi searching result

Namespace	ClouReaderAPI.CLReader._Config
Function	static Boolean RequestWiFiDataToTxt(String ConnID,String filepath)
Parameter	// ConnID: connection identifier //filepath: saved txt file path
Return	Ture: successful; false: failed
Remark	

2.11.7connect Wi-Fi hotspot

Namespace	ClouReaderAPI.CLReader._Config
Function	static Int32 ConnectWIFI(String ConnID,string name, string pwd, int authType, int encryption)

Parameter	// ConnID: connection identifier
	// name: wifi name
	//pwd: wifi password
	//authType: Authentication type
	//encryption: Encryption Algorithm

Return	True: successful; false: failed

Remark	e.g.:
	1、when Wi-Fi hotspot need password:
	if (WifiAuth=="WPA")
	{
	authType = 1;
	}
	else if (WifiAuth == "WPA2")
	{

	authType = 2;
	}
	int isConnect = CLReader._Config.ConnectWIFI(ConnID, WifiName, pwd, authType, 2);
	2、when Wi-Fi hotspot without password:
	int ret = CLReader._Config.ConnectWIFI(ConnID, WifiName, "", 0, 2);

2.11.8 Query connecting Wi-Fi name

Namespace	ClouReaderAPI.CLReader._Config
Function	static String GetWiFiConnectInfo(String ConnID)
Parameter	// ConnID: connection identifier
Return	Present connected Wi-Fi name
Remark	

2.13 Antenna number parameter description

- Regarding the tag read, write, lock, kill operation of the antenna number parameter: antNum. The function is to specify whether an antenna or multiple antennas for a reader work.

- While specifying multiple antennas to work with antNum for their total value, for instance:
  Specify antenna 1+ antenna 2 to work: antNum = eAntennaNo._1 | eAntennaNo._2
  Specify antenna1+antenna2+antenna 3 to work: antNum = eAntennaNo._1 | eAntennaNo._2 | eAntennaNo._3
  Specify antenna 1+ antenna 2+ antenna 3+ antenna 4 to work: antNum = eAntennaNo._1 | eAntennaNo._2 | eAntennaNo._3 | eAntennaNo._4

3. Programming example

C# Code: read 6C tag example

```
class Program : ClouReaderAPI.ClouInterface.IAsynchronousMessage
    {
        static void Main(string[] args)
        {
            // ClouReaderAPI.CLReader.CreateSerialConn("COM1:115200",new Program());
               // use serial port to connect RFID reader
            if (ClouReaderAPI.CLReader.CreateTcpConn("192.168.1.116:9090", new Program()))
          // use TCP to connect RFID reader
            {
        ClouReaderAPI.CLReader._Tag6C.GetEPC("192.168.1.116:9090",eAntennaNo._1|eAntennaNo._2|
eAntennaNo._3|eAntennaNo._4, eReadType.Inventory);
          // send reading EPC instruction to the reader using Ant1+ Ant2+Ant3+Ant4 at the same time
            }
            else
            {                }
            Console. ReadKey();
            ClouReaderAPI.CLReader._Tag6C.Stop("192.168.1.116:9090"); // send stop instruction
            ClouReaderAPI.CLReader.CloseConn("192.168.1.116:9090"); // close connection
        }
        #region interface implement
          // Tag CallBack
        public void OutPutTags(ClouReaderAPI.Models.Tag_Model tag)
        {
            Console.WriteLine("EPC:" + tag.EPC + " - TID:" + tag.TID);
        }
        public void WriteDebugMsg(string msg)
        {          }
        public void WriteLog(string msg)
        {          }
        public void PortConneting(string connID)
        {          }
        public void PortClosing(string connID)
        {          }
        public void OutPutTagsOver()
        {          }
        public void GPIControlMsg(GpiModel gpiModel)
        {          }
        #endregion
    }
```

4. FAQ and solution

Question	Solution
Device couldn't work normally	1. Check power light normal or not. 2. If normal, there should be some notice sound when power on.
serial port couldn't work normally	1. Check connection cable connecting normal or not . 2. If conditional, use another device to check this cable normal or not. 3. Try to use RJ45 to communicate. 4. Default baud rate: 115200.
RJ45 couldn't work normally	1. Check LED working normal or not. 2. To use Ping instruction to check cable working normal or not 3. Try serial port connection, inquiry IP correct by Demo 4. Default IP and port: "192.168.1.116:9090"

5. Appendix A: 6C tag operation returned error code

Read tag error codes:

Code	Remark
0	Configuration successful
1	Antenna port Parameter error
2	Read Parameter error
3	TID parameter error
4	user data area parameter error
5	reserved area parameter error
6	Other parameter error

Write tag error codes:

Code	Remark
0	Write successfully
1	Antenna port parameter error
2	Choose parameter error
3	Write parameter error
4	CRC correct error
5	Power not enough
6	Data area overflow
7	Data area locked
8	Access password error
9	Other tag error

10	Tag lost
11	Instruction sent error

Lock tag error codes:

Code	Remark
0	Lock successfully
1	Antenna com port parameter error
2	Choose parameter error
3	Write parameter error
4	CRC correcting error
5	Power no enough
6	data area overflow
7	data area is locked
8	Access password error
9	Other tag error
10	Tag lost
11	instruction sent error

Kill tag error codes:

Code	Remark
0	Kill successfully
1	Antenna port parameter error
2	Choose parameter error
3	CRC correct error
4	Power not enough
5	Password error
6	Other tag error
7	Tag lost
8	Instruction sent error