1. Bluetooth related interface 2

a. Bluetooth status change listener 2

b. Bluetooth status 2

c. Connect a Bluetooth device 2

d. Disconnect the Bluetooth device 2

e. Reader power 2

f. Reader Trigger state change listener 2

2. Uhf related interface 3

a. Get the UHF module model 3

b. Set the Reader buzzer 3

c. Get UHF module temperature 3

d. Get the Reader firmware version number 3

e. Obtain the UHF module firmware version number 3

f. Query the current output power of the reader 3

g. Set the reader's temporary RF output power 4

h. Set the reader RF output power 4

i. Query reader operating frequency range 4

j. Set the reader operating frequency range 4

k. Reader trigger use status 4

l. Initial UHF 4

m. Set the UHF callback interface 5

n. Cancel Tag selection 5

o. Inventory 5

p. R2000 continuous inventory 5

q. Set Tag selected 5

r. Inactivated label 5

s. Read Tag 6

t. Lock Tag 6

u. Write Tag 6

3. Scan related interface 6

a. Initial scan 6

b. Set the scan callback interface 7

c. Continuous scan interface 7

d. Single scan 7

1. Bluetooth related interface
2. Bluetooth status change listener

 /\*\*

 \* Bluetooth status change listener

 \* @param listener

 \*/

public void setBluetoothStateListener(IBluetoothCallback listener)

1. Bluetooth status

 /\*\*

 \* Bluetooth status

 \* @return {@link BluetoothState , BluetoothState.STATE\_CONNECTED;

 \* BluetoothState.STATE\_CONNECTING;

 \* BluetoothState.STATE\_NONE;

 \* BluetoothState.STATE\_LISTEN}

 \*/

public int getBluetoothState()

1. Connect a Bluetooth device

 /\*\*

 \* Connect a Bluetooth device

 \* @param address Bluetooth Mac address

 \*/

public void connect(String address)

1. Disconnect the Bluetooth device

 /\*\*

 \* Disconnect the Bluetooth device

 \*/

public void disconnect()

1. Reader power

 /\*\*

 \* Get Reader power

 \* @param cb Callback interface

 \*/

public void GetReaderPower(IReaderPowerCallback cb)

1. Reader Trigger state change listener

 /\*\*

 \* Trigger state change listener

 \* @param listener Callback interface

 \*/

public void setTriggerStateListener(ITriggerStateChangeCallback listener)

1. Uhf related interface
2. Get the UHF module model

 /\*\*

 \* Get the UHF module model

 \* @return 1 : R2000

 \* 2 : M100

 \*/

public int GetUHFIdentifierID()

1. Set the Reader buzzer

 /\*\*

 \* Set the Reader buzzer, the default buzzer is turned on, the first time the buzzer is turned off, and the buzzer is turned on again.

 \* @param cb Callback interface

 \*/

public void SetReaderBuzzer(IReaderBuzzerCallback cb)

1. Get UHF module temperature

 /\*\*

 \*Get UHF module temperature

 \* @param cb Callback interface

 \*/

public void GetReaderTemperature(IReaderTemperatureCallback cb)

1. Get the Reader firmware version number

 /\*\*

 \* Get the Reader firmware version number

 \* @param cb Callback interface

 \*/

public void GetReaderFirmwareVersion(IGetReaderFirmwareVersionCallback cb)

1. Obtain the UHF module firmware version number

 /\*\*

 \* Obtain the UHF module firmware version number

 \* @param cb Callback interface

 \*/

public void GetFirmwareVersion(IFirmwareVersionCallback cb)

1. Query the current output power of the reader

 /\*\*

 \* Query the current output power of the reader

 \* @param cb Callback interface

 \*/

public void GetOutPutPower(IGetOutPutPowerCallback cb)

1. Set the reader's temporary RF output power

 /\*\*

 \* Set the reader's temporary RF output power

 \* @param cb Callback interface

 \* @param RfPower Temporary output RF output power

 \*/

public void SetTemporaryOutPutPower(ISetTemporaryOutputPowerCallback cb, byte RfPower)

1. Set the reader RF output power

 /\*\*

 \* Set the reader RF output power

 \* @param cb Callback interface

 \* @param RfPower RF output power

 \*/

public void SetOutPutPower(ISetOutPutPowerCallback cb, byte RfPower)

1. Query reader operating frequency range

 /\*\*

 \* Query reader operating frequency range

 \* @param cb Callback interface

 \*/

public void GetFrequencyRegionin(IGetFrequencyRegionCallback cb)

1. Set the reader operating frequency range

 /\*\*

 \* Set the reader operating frequency range

 \* @param cb Callback interface

 \* @param Region RF specification{FCC : 0x01; ETSI : 0x02; CHN : 0x03}

 \* @param StartFreq Frequency starting point

 \* @param EndFreq Frequency end point

 \*/

public void SetFrequencyRegionin(ISetFrequencyRegionCallback cb, byte Region, byte StartFreq, byte EndFreq)

1. Reader trigger use status

 /\*\*

 \* Reader trigger use status

 \* @param available true : can use ; false : can not be used

 \*/

public void setReaderTriggerAvailable(boolean available)

1. Initial UHF

 /\*\*

 \* Initial UHF

 \* @param cb Callback interface

 \* @param triggerAvailable true : can use; false : can not be used

 \*/

public void InitUHF(final IInitCallback cb, final boolean triggerAvailable)

1. Set the UHF callback interface

 /\*\*

 \* Set the UHF callback interface

 \* @param cb Callback interface

 \*/

public void SetInventoryCallback(IInventoryCallback cb)

1. Cancel Tag selection

 /\*\*

 \* Cancel Tag selection

 \* @param cb Callback interface

 \*/

public void ClearTagSelected(ITagSelectCallback cb)

1. Inventory

 /\*\*

 \* Inventory

 \* @param advanced true : advanced ; false : default

 \* @param args default : {0x00}

 \* advanced : {0x01, Session, Target, Repeat}

 \*/

public void Inventory(final boolean advanced, final byte[] args)

1. R2000 continuous inventory

 /\*\*

 \*R2000 continuous inventory

 \* @param cb Callback interface

 \* @param args default : {0x00}

 \* advanced : {0x01, Session, Target, Repeat}

 \* @param continueInventory true : start continue ; false : stop continue

 \*/

public void ContinueInventoryR2000(IContinueInventoryR2000Callback cb, byte[] args, boolean continueInventory)

1. Set Tag selected

 /\*\*

 \* Set Tag selected

 \* @param cb Callback interface

 \* @param Mask Specify tag data

 \*/

public void SetTagSelected(ITagSelectCallback cb, byte[] Mask)

1. Inactivated label

 /\*\*

 \* Inactivated label

 \* @param cb Callback interface

 \* @param PassWord The default password cannot be used, the modified password. 4bytes

 \*/

public void Kill(IKillCallback cb, byte[] PassWord)

1. Read Tag

 /\*\*

 \* Read Tag

 \* @param cb Callback interface

 \* @param MemBank Tag storage area{0x00 : RESERVED; 0x01 : EPC; 0x02 : TID; 0x03 : USER}

 \* @param WordAdd Read data first address. Please refer to the label specifications for the range of values.

 \* @param WordCnt Read data length. Word length, WORD (16 bits) length.

 \* Please refer to the label specification for the range of values.

 \* @param PassWord Tag access password, 4 bytes.

 \*/

public void Read(IReadCallback cb, byte MemBank, byte[] WordAdd, byte WordCnt, byte[] PassWord)

1. Lock Tag

 /\*\*

 \* Lock Tag

 \* @param cb Callback interface

 \* @param PassWord Tag access password, 4 bytes.

 \* @param MemBank Tag storage area{0x00 : RESERVED; 0x01 : EPC; 0x02 : TID; 0x03 : USER, 0x04 : Access Password; 0x05 : Kill Password}

 \* @param LockType Lock operation type.{0x00 : open; 0x01 : lock; 0x02 : Permanently open; 0x03 : Permanent lock}

 \*/

public void Lock(ILockCallback cb, byte[] PassWord, byte MemBank, byte LockType)

1. Write Tag

 /\*\*

 \* Write Tag

 \* @param cb Callback interface

 \* @param PassWord Tag access password, 4 bytes.

 \* @param MemBank Tag storage area{0x00 : RESERVED; 0x01 : EPC; 0x03 : USER}

 \* @param WordAdd Read data first address. Please refer to the label specifications for the range of values.

 \* @param WordCnt Read data length. Word length, WORD (16 bits) length.

 \* Please refer to the label specification for the range of values.

 \* @param Data write data

 \*/

public void Write(IWriteCallback cb, byte[] PassWord, byte MemBank, byte[] WordAdd, byte WordCnt, byte[] Data)

1. Scan related interface
2. Initial scan

 /\*\*

 \* Initial scan

 \* @param cb Callback interface

 \* @param triggerAvailable true : can use; false : can not be used

 \*/

public void InitScan(final IInitCallback cb, final boolean triggerAvailable)

1. Set the scan callback interface

 /\*\*

 \* Set the scan callback interface

 \* @param cb Callback interface

 \*/

public void setScanCallback(IScanCallback cb)

1. Continuous scan interface

 /\*\*

 \* Continuous scan interface

 \* @param cb Callback interface

 \* @param continueScan true : start continue; false : stop continue

 \*/

public void doContinueScan(IContinueScanCallback cb, boolean continueScan)

1. Single scan

 /\*\*

 \* Single scan

 \*/

 public void doScan()