CReader.jar Manual

1. programming interface

Declare: import com.rfid.*;

Interface preview:

```
public CReader(String ipAddr,int Port,int ReaderType,int logswitch)
public int Connect();
public int DisConnect();
public void SetInventoryParameter(ReaderParameter param);
public ReaderParameter GetInventoryPatameter();
public void SetCallBack(TagCallback callback);
public int StartRead();
public void StopRead();
public int SetGPIO(byte GPIO)
public int GetGPIOStatus(byte OutputPin[])
public int GetUHFInformation(byte Version[],byte Power[],byte band[],byte
MaxFre[],byte MinFre[],byte BeepEn[],byte Ant[]);
public int SetRfPower(int Power);
public int SetRegion(int band,int maxfre,int minfre);
public int SetAntenna(byte AntCfg);
public String ReadDataByEPC(String EPCStr,byte Mem,byte WordPtr,byte Num,
```

```
String Password);
public String ReadDataByTID(String TIDStr,byte Mem,byte WordPtr,
byte Num, String Password);
public int WriteDataByEPC(String EPCStr,byte Mem,byte WordPtr, String
Password,String wdata);
public int WriteDataByTID(String TIDStr,byte Mem,byte WordPtr, String
Password,String wdata);
public int WriteEPCByTID(String TIDStr,String EPCStr, String Password);
public int Lock(String EPCStr,byte select,byte setprotect,
String Password);
public int Kill(String EPCStr,String Password);
public int ReadData_G2(byte ENum, byte EPC[], byte Mem,byte WordPtr, byte
Num, byte Password[],byte MaskMem,byte MaskAdr[],byte MaskLen,byte[]
MaskData,byte MaskFlag,byte Data[],byte Errorcode[]);
public int WriteData_G2(byte WNum, byte ENum,byte EPC[],byte Mem, byte
WordPtr, byte Writedata[], byte Password[],byte MaskMem,byte MaskAdr[],byte
MaskLen,byte[]MaskData,byte MaskFlag,byte Errorcode[]);
public int Inventory_G2(byte QValue,byte Session,byte AdrTID,byte
LenTID,byte Target,byte Ant,byte Scantime,byte pOUcharIDList[],int
pOUcharTagNum[],int pListLen[]);
```

2. parameter interface

```
public class ReaderParameter {
      public void SetAddress(byte ComAddr)//Set RFID module address
      public byte GetAddress()//Get RFID module address
      public void SetTidPtr(byte TidPtr);// Set TID start address
   public int GetTidPtr();//Get TID start address
   public void SetTidLen(byte TidLen);// Set the TID length, if it is 0, it
                                         means to read EPC
   public int GetTidLen();//Get TID length
      public void SetSession(int Session);//0~3:S0~S3,255-auto.
      public int GetSession();
      public void SetQValue(int QValue);//0~15
      public int GetQValue()
      public void SetScanTime(int ScanTime);//max scan time of inventory
   public int GetScanTime()
   public void SetAntenna(int Antenna);// The antenna number to be queried,
   which indicates the antenna in bits, such as: binary 00000000 00000001,
   hexadecimal 0x01, indicating antenna 1 query,
   Binary 00000000 00001001, hexadecimal 0x09, indicating that both antennas
   1 and 4 participate in the query,
   public int GetAntenna()
}
```

3. callback interface

```
public interface TagCallback {
      public void tagCallback(ReadTag tag); // Label callback
interface
}
public class ReadTag {
      public String epcId;// EPC or TID
      public int rssi;
      public int antId;//Antenna
}
```

4. Interface details

4.1 Open/Close RFID

4.1.1 CReader

Definition	public CReader(String ipAddr,int Port,int ReaderType,int logswitch)		
Description	Create an object for connection management with the reader.		
Parameters	Name	Type	Notes
	ipAddr	String	Reader ip address
	Port	int	Socket number of tcp
	ReaderType	int	Several devices are currently connected, such as 4, which is connected to a 4-antenna device; 16 means a connected 16-antenna device
	logswitch	int	Log switch,0-close；1-open.
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).		
Sample code			

4.1.2 Connect

Definition	int Connect ();		
Description	Connect with reader.		
Parameters	**Name**	**Type**	**Notes**
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).		
Sample code			

4.1.3 Disconnect

Definition	int Disconnect();		
Description	Disconnect with reader,		
Parameters	**Name**	**Type**	**Notes**
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).		
Sample code			

4.2 18000-6C

4.2.1 StartRead

Definition	Public int StartRead		
Description	Start to read tag		
Parameters	**Name**	**Type**	**Notes**
Return value (int)	Succeed: 0; Failed: non zero; (for detail error definition, refer to reader error code table).		
Sample code			

4.2.2 StopRead

Definition	Public void StopRead();		
Description	Stop to read tag		
Parameters	**Name**	**Type**	**Notes**
Return value (void)			
Sample code			

4.2.3 ReadDataByEPC

Definition	public String ReadDataByEPC(String EPCStr,byte Mem,byte WordPtr,byte Num, String Password);		
Description	Read each memory area data through EPC mask		
Parameters	Name	Type	Notes
	EPCStr	String	The hexadecimal EPC number of the tag
	Mem	byte	memory area to be read, 0- Password area, the first 2 words are the destruction password, the last 2 words are the access password 1- EPC area 2- TID area 3- User area
	WordPtr	byte	The starting word address for reading
	Num	byte	read word length
	Password	String	Tag's hexadecimal access password, 4 bytes
Return value (String)	Success: tag data Failed: null;		
Sample code			

4.2.4 ReadDataByTID

Definition	public String ReadDataByTID(String TIDStr,byte Mem,byte WordPtr,byte Num, String Password);		
Description	Read each memory area data through TID mask		
Parameters	Name	Type	Notes
	TIDStr	String	The hexadecimal EPC number of the tag
	Mem	byte	memory area to be read, 0- Password area, the first 2 words are the destruction password, the last 2 words are the access password 1- EPC area 2- TID area 3- User area
	WordPtr	byte	The starting word address for reading
	Num	byte	read word length
	Password	String	Tag's hexadecimal access password, 4 bytes
Return value (String)	Success: tag data Failed: null;		
Sample code			

4.2.5 WriteDataByEPC

Definition	public int WriteDataByEPC(String EPCStr,byte Mem,byte WordPtr,String Password,String wdata);		
Description	Write data to each memory area through the EPC mask.		
Parameters	Name	Type	Notes
	EPCStr	String	The hexadecimal EPC number of the tag

	Mem	byte	memory area to be write, 0- Password area, the first 2 words are the destruction password, the last 2 words are the access password 1- EPC area 2- TID area 3- User area
	WordPtr	byte	Write start word address
	Password	String	Tag's hexadecimal access password, 4 bytes
	wdata	String	The hexadecimal string of the data to be written, the length must be an integer multiple of 4
Return value (int)	Success:0;		
Sample code			

4.2.6 WriteDataByTID

Definition	public int WriteDataByTID(String TIDStr,byte Mem,byte WordPtr, String Password,String wdata);		
Description	Write data to each memory area through the TID mask.		
Parameters	**Name**	**Type**	**Notes**
	TIDStr	String	The hexadecimal TID number of the tag
	Mem	byte	memory area to be write, 0- Password area, the first 2 words are the destruction password, the last 2 words are the access password 1- EPC area 2- TID area 3- User area
	WordPtr	byte	Write start word address
	Password	String	Tag's hexadecimal access password, 4 bytes
	wdata	String	The hexadecimal string of the data to be written, the length must be an integer multiple of 4
Return value (int)	Success:0;		
Sample code			

4.2.7 WriteEPCByTID

Definition	public int WriteEPCByTID(String TIDStr,String EPCStr,String Password);		
Description	Rewrite the EPC number of the tag through the TID mask. (Requires the TID start address to start from 0)		
Parameters	**Name**	**Type**	**Notes**
	TIDStr	String	The hexadecimal TID number of the tag
	EPCStr	String	The hexadecimal EPC number of the label to be rewritten
	Password	String	Tag's hexadecimal access password, 4 bytes
Return value (int)	Success:0;		
Sample code			

4.2.8 Lock

Definition	public int Lock(String EPCStr,byte select,byte setprotect,String Password);		
Description	Set the protection status of each area of the label		
Parameters	**Name**	**Type**	**Notes**
	EPCStr	String	The hexadecimal EPC number of the tag
	select	byte	1 byte, 0x00 - Controls the Kill password read and write protection settings. 0x01 – Control access password read and write protection settings. 0x02 - Controls EPC memory read and write protection settings. 0x03 - Controls TID memory read and write protection settings. 0x04 - Controls user memory read and write protection settings. Other values are reserved. If the reader/writer receives other values, it will return a parameter error message.
	setprotect	byte	1 byte, When Select is 0x00 or 0x01, the meaning of the SetProtect value is as follows: 0x00 - set to read and write 0x01 - set to always readable and writable 0x02 - set to read and write with password 0x03 - set to never read or write When Select is 0x02, 0x03, 0x04, the meaning of the SetProtect value is as follows: 0x00 - set to writable 0x01 - set to always writable 0x02 - set to writable with password 0x03 - Set to never write.
	Password	String	Tag's hexadecimal access password, 4 bytes
Return value (int)	Success:0;		
Sample code			

4.2.9 Kill

Definition	public int Kill(String EPCStr,String PasswordStr)		
Description	This command is used to destroy the label. Once the tag is destroyed, the reader's commands are never processed again.		
Parameters	**Name**	**Type**	**Notes**
	EPCStr	String	The hexadecimal EPC number of the tag
	PasswordStr	String	Hex string for label kill password
Return value (int)	Success:0;		
Sample code			

4.2.10 ReadData_G2

Definition	public int ReadData_G2(byte ENum, byte EPC[], byte Mem,byte WordPtr, byte Num, byte Password[],byte MaskMem,byte MaskAdr[],byte MaskLen,byte[] MaskData, byte Data[],byte Errorcode[])		
Description	Read data from each memory area		
Parameters	**Name**	**Type**	**Notes**
	ENum	String	1 byte, the length of the EPC number of the label, when Enum=255, the mask method is used.
	EPC	byte []	ENum words, EPC number of the label。
	Mem	byte	memory area to be read, 1- Password area, the first 2 words are the destruction password, the last 2 words are the access password 2- EPC area 3- TID area 3 - User area
	WordPtr	byte	The starting word address for reading
	Num	byte	read word length
	Password	String	Tag's hexadecimal access password, 4 bytes
	MaskMem	byte	1 byte, mask area. 1 – EPC memory area; 2 – TID storage area; 3 – User memory area.
	MaskAdr	byte []	2 bytes, the starting bit address of the mask (unit: Bits). The range is 0~16383.
	MaskLen	byte	1 byte, the bit length of the mask (unit: Bits).
	MaskData	byte []	n bytes, mask data. n is equal to MaskLen/8. If MaskLen is not an integer multiple of 8, then n is [MaskLen/8] rounded up plus 1. If it is not enough, add 0 in the low order.
	Data	byte []	Num words, the read tag data.
	ErrorCode	int []	When the return value is 0xFC, the label error information is returned. Please check the ErrorCode error code table.
Return value (String)	Success: tag data Failed: null;		
Sample code			

4.2.11 WriteData_G2

Definition	public int WriteData_G2(byte WNum, byte ENum,byte EPC[],byte Mem, byte WordPtr, byte Writedata[], byte Password[],byte MaskMem,byte MaskAdr[],byte MaskLen,byte[]MaskData,byte Errorcode[])		
Description	Write data to each memory area.		
Parameters	**Name**	**Type**	**Notes**
	WNum	byte	1 byte, the number of words to write,
	ENum	byte	1 byte, the length of the EPC number of the label. When Enum=255, the mask method is used.
	EPC	byte[]	Enum*2 bytes, the EPC number of the label.

	Mem	byte	memory area to be written, 0- Password area, the first 2 words are the destruction password, the last 2 words are the access password 1- EPC area 2- TID area 3 - User area
	WordPtr	byte	Write start word address
	Writedata	byte[]	Wnum*2 bytes, the data to be written.
	Password	byte[]	4 bytes, the access password for the tag.
	MaskMem	byte	1 byte, mask area. 1 – EPC memory area; 2 – TID storage area; 3 – User memory area.
	MaskAdr	byte []	2 bytes, the starting bit address of the mask (unit: Bits). The range is 0~16383.
	MaskLen	byte	1 byte, the bit length of the mask (unit: Bits).
	MaskData	byte []	n bytes, mask data. n is equal to MaskLen/8. If MaskLen is not an integer multiple of 8, then n is [MaskLen/8] rounded up plus 1. If it is not enough, add 0 in the low order.
	ErrorCode	int []	When the return value is 0xFC, the label error information is returned. Please check the ErrorCode error code table.
Return value (int)	Success:0;		
Sample code			

4.2.12 Inventory_G2

Definition	public int Inventory_G2(byte QValue,byte Session,byte AdrTID,byte LenTID,byte Target,byte Ant,byte Scantime,byte pEPCList [],int pOUcharTagNum[],int pListLen[])		
Description	Check whether there is an electronic label that conforms to the agreement within the valid range		
Parameters	**Name**	**Type**	**Notes**
	QValue	byte	1 byte, 0~15: The initial Q value used when querying the EPC label,
	Session	byte	1 byte, the session value used when querying the EPC tag. 0x00: Session uses S0; 0x01: Session uses S1; 0x02: Session uses S2; 0x03: Session uses S3. 0xff: The reader automatically configures the session (only valid for EPC query) It is recommended to select S0 for single or a small number of label queries.
	AdrTID	byte[]	1 byte, query the starting word address of the TID area.
	LenTID	byte	1 byte, query the number of data words in the TID area, the range is 1~15, 0 means EPC query
	Target	byte	1 byte, the Target value of the query tag. 0 – Target A; 1 – Target B.

	Ant	byte[]	1 byte, the antenna number to be queried this time. 0x80 – Antenna 1; 0x81 – Antenna 2; 0x82 – Antenna 3; 0x83 – Antenna 4; 0x84 – Antenna 5; 0x85 – Antenna 6; 0x86 – Antenna 7; 0x87 – Antenna 8; 0x88 – Antenna 9; 0x89 – Antenna 10; 0x8A – Antenna 11; 0x8B – Antenna 12; 0x8C – Antenna 13; 0x8D – Antenna 14; 0x8A – Antenna 15; 0x8B – Antenna 16; The single-port reader is fixed at 0x80.
	Scantime	byte	1 byte, the maximum operation time for inventory. The valid range of *Scantime* is 0 ~ 255, corresponding to (0 ~ 255)*100ms. For *Scantime* = 0, operation time is not limited.
	pEPCList	byte	The inquired tag data, the data block follows the format stated below: EPC/TID length + EPC/TID No. + RSSI. Data of multiple tags is formed by several identical data blocks in sequence
	pOUcharTagNum	byte []	The total amount of tag inquired during the current inventory.
	pListLen	byte	The total length of the received data stored in *pEPCList*.
Return value (int)	Success:0;		
Sample code			

4.3 Custom command

4.3.1 SetCallBack

Definition	public void SetCallBack(TagCallback callback)		
Description	Set the callback interface after the inventory is started, and the label data is returned through the callback interface		
Parameters	**Name**	**Type**	**Notes**
	callback	TagCallback	Tag's callback interface
Return value (void)			
Sample code			

4.3.2 SetInventoryParameter

Definition	public void SetInventoryPatameter(ReaderParameter param)

Description	Set the query parameters used when inventory is enabled		
Parameters	**Name**	**Type**	**Notes**
	param	ReaderParameter	query parameters
Return value (void)			
Sample code			

4.3.3 GetInventoryParameter

Definition	public ReaderParameter GetInventoryPatameter()		
Description	Get the query parameters used in inventory		
Parameters	**Name**	**Type**	**Notes**
Return value (ReaderParameter**)**	ReaderParameter		
Sample code			

4.3.4 GetUHFInformation

Definition	public int GetUHFInformation(byte Version[],byte Power[],byte band[],byte MaxFre[],byte MinFre[],byte BeepEn[],int Ant[])		
Description	Get basic information of the UHF module.		
Parameters	**Name**	**Type**	**Notes**
	Version	byte []	Output 2 bytes, reader version information. The first byte is the version number, and the second byte is the handle of the subversion number.
	Power	byte []	Output 1 byte, the output power of the reader. The range is 0 to 30 in dBm.
	band	byte[]	Output 1 byte, spectrum band. 1 - "Chinese band2"; 2 - "US band"; 3 - "Korean band"; 4 - "EU band"; 8 - "Chinese band1";
	MaxFre	byte[]	Output 1 byte, indicating the current maximum frequency of the reader.
	MinFre	byte[]	Output 1 byte, indicating the current minimum frequency of the reader.
	BeepEn	byte[]	Output 1 byte, buzzer beeps information.
	Ant	int[]	Output 1 word, antenna configuration information. Each bit represents an antenna number, such as 0x0009, the binary is 00000000 00001001, indicating antenna 1 and antenna 4.
Return value (int**)**	Success:0;		
Sample code			

4.3.5 SetRfPower

Definition	public int SetRfPower(int Power)		
Description	Set the reader power		
Parameters	**Name**	**Type**	**Notes**
	Power	int	The output power of the reader. The range is 0 to 30 in dBm. The highest bit 7 is 1, which means that the power adjustment is not saved; the bit 0 means that the power is saved and saved.
Return value (int)	Success:0;		
Sample code			

4.3.6 SetAntenna

Definition	public int SetAntenna(int SetOnce,int AntCfg)		
Description	Set the effective working antenna of the reader		
Parameters	**Name**	**Type**	**Notes**
	SetOnce	int	1: Indicates that it will not be saved when power off; 0: Indicates power-off save
	AntCfg	byte	Valid antenna number, each bit represents an antenna number, such as 0x0009, binary is 00000000 00001001, indicating antenna 1 and antenna 4.
Return value (int)	Success:0;		
Sample code			

4.3.7 SetGPIO

Definition	public int SetGPIO(byte GPIO)		
Description	Set the state of the GPIO port of the reader		
Parameters	**Name**	**Type**	**Notes**
	GPIO	byte	GPIO port (Out1-Out2 pin) output status. Bit0-Bit1 control the Out1-Out2 pins respectively, and Bit2-Bit7 are reserved.
Return value (int)	Success:0;		
Sample code			

4.3.8 GetGPIOStatus

Definition	public int GetGPIOStatus(byte OutputPin[])		
Description	Read the GPIO port status of the reader		
Parameters	**Name**	**Type**	**Notes**
	OutputPin	byte[]	1 byte, GPIO port output status. Bit0 represents the pin status of IN1, Bit4-Bit5 represent the status of Out1-Out2 respectively, and other bits are reserved.
Return value (int)	Success:0;		
Sample code			

4.3.9 SetRegion

Definition	public int SetRegion(int band,int maxfre,int minfre)		
Description	Set the working frequency band of the reader		
Parameters	**Name**	**Type**	**Notes**
	band	Int	1 byte, spectrum band. 1 - "Chinese band2"; 2 - "US band"; 3 - "Korean band"; 4 - "EU band"; 8 - "Chinese band1";
	maxfre	Int	Indicates the current maximum frequency of the reader's operation.
	minfre	Int	Indicates the minimum frequency at which the current reader works.
Return value (int)	Success:0;		
Sample code			

The calculation formula of each frequency band:

Chinese band2: $Fs = 920.125 + N * 0.25\ (MHz)\ N \in [0, 19];$

US band: $Fs = 902.75 + N * 0.5\ (MHz)\ N \in [0,49];$

Korean band: $Fs = 917.1 + N * 0.2\ (MHz)\ N \in [0, 31];$

EU band: $Fs = 865.1 + N*0.2(MHz)\ N \in [0, 14];$

Chinese band1: $Fs = 840.125 + N * 0.25\ (MHz)\ N \in [0, 19].$

4.3.10 SetBeepNotification

Definition	public int SetBeepNotification(int BeepEn)		
Description	This command is used to set the buzzer switch.		
Parameters	**Name**	**Type**	**Notes**
	BeepEn	int	0x00 – disabled beep; 0x01 – enabled beep.
Return value (int)	Success:0;		
Sample code			

4.3.11 SetRfPowerByAnt

Definition	public int SetRfPowerByAnt(byte Power[])		
Description	Set the power according to the antenna port		
Parameters	**Name**	**Type**	**Notes**
	Power	byte[]	The power of each antenna port, the number of parameter bytes must be consistent with the number of antenna ports, the low byte represents the power of antenna port 1, and so on
Return value (int)	Success:0;		
Sample code			

4.3.12 GetRfPowerByAnt

Definition	public int GetRfPowerByAnt(byte Power[])		
Description	Get the power according to the antenna port		
Parameters	**Name**	**Type**	**Notes**
	Power	byte[]	The power of each antenna port, the number of parameter bytes must be consistent with the number of antenna ports, the low byte represents the power of antenna port 1, and so on
Return value (int)	Success:0;		
Sample code			

4.3.13 ExtSetRegion

Definition	public int ExtSetRegion (int opt,int band,int maxfre, int minfre)		
Description	Set the working frequency band of the reader		
Parameters	**Name**	**Type**	**Notes**
	opt	byte	0-save;1-not save;
	band	byte	frequency band, see table.
	maxfre	byte	maximum frequency point
	minfre	byte	minimum frequency point.
Return value (int)	Success:0;		
Sample code			

Table:

FreBand	Name	Range
0	Full band	Fs = 840 + N * 2 (MHz) ,N∈[0, 60]
1	Chinese band2	Fs = 920.125 + N * 0.25 (MHz) ,N∈[0, 19]
2	US band	Fs = 902.75 + N * 0.5 (MHz) ,N∈[0,49]

3	Korean band	Fs = 917.1 + N * 0.2 (MHz) ,N∈[0, 31]
4	EU band	Fs = 865.1 + N*0.2(MHz) ,N∈[0, 14]
5		
6	Ukraine band	Fs = 868.0 + N*0.1(MHz) ,N∈[0, 6]
7		
8	Chinese band1	Fs = 840.125 + N * 0.25 (MHz),N∈[0, 19]
9	EU3 band	Fs = 865.7 + N * 0.6(MHz) ,N∈[0, 3]
10		
11		
12	US band3	Fs = 902.0 + N * 0.5(MHz),N∈[0, 52]
13		
14		
15		
16	HK band	Fs = 920.25 + N * 0.5(MHz),N∈[0, 9]
17	Taiwan band	Fs = 920.75 + N * 0.5(MHz),N∈[0, 13]
18	ETSI UPPER band	Fs = 916.3 + N * 1.2(MHz),N∈[0,2]
19	Malaysia band	Fs = 919.25 + N * 0.5(MHz),N∈[0,7]
20		
21	Brazil band	Fs = 902.75 + N * 0.5(MHz),N∈[0, 9] Fs = 910.25 + N * 0.5(MHz),N∈[10, 34]
22	Thailand band	Fs = 920.25 + N * 0.5(MHz),N∈[0, 9]
23	Singapore band	Fs = 920.25 + N * 0.5(MHz),N∈[0, 9]
24	Australia band	Fs = 920.25 + N * 0.5(MHz),N∈[0, 9]
25	India band	Fs = 865.1 + N * 0.6(MHz),N∈[0, 3]
26	Uruguay band	Fs = 916.25 + N * 0.5(MHz),N∈[0, 22]
27	Vietnam band	Fs = 918.75 + N * 0.5(MHz),N∈[0, 7]
28	Israel band	Fs = 916.25 + N * 0.5(MHz),N∈[0, 0]
29	Indonesia band	Fs = 917.25 + N * 0.5(MHz),N∈[0, 0] Fs = 919.75 + N * 0.5(MHz),N∈[1, 3]
30	New Zealand band	Fs = 922.25 + N * 0.5(MHz),N∈[0, 9]
31	Japan2 band	Fs = 916.8 + N * 1.2 (MHz),N∈[0, 3]
32	Peru band	Fs = 916.25 + N * 0.5(MHz),N∈[0, 22]
33	Russia band	Fs = 916.2 + N * 1.2 (MHz),N∈[0, 3]
34	South Africa band	Fs = 915.6 + N * 0.2(MHz),N∈[0, 16]
35	Philippines band	Fs = 918.25 + N * 0.5(MHz),N∈[0, 3]

4.4 Return value error code table

Error code	Description
0x00	API is called successfully.
0x01	No find tag
0x05	Access password error.
0x09	Kill password error.
0x0A	All-zero tag killing password is invalid.
0x0B	Command is not support by the tag
0x0C	All-zero tag access password is invalid for such command.
0x0D	Fail to setup read protection for a protection enabled tag.

0x0E	Fail to unlock a protection disabled tag.
0x10	Some bytes stored in the tag are locked.
0x11	Lock operation failed.
0x12	Already locked, lock operation failed.
0x13	Fail to store the value of some preserved parameters. Configuration will still valid before reader shut down.
0x14	Modification failed.
0xF8	Error detected in antenna check.
0xF9	Operation failed.
0xFA	Tag is detected, but fails to complete operation due to poor communication.
0xFB	No tag is detected.
0xFC	Error code returned from tags.
0xFD	Command length error.
0xFE	Illegal command.
0xFF	Parameter error.
0x30	Communication error.