



**OEM UHF – M6X0**  
**Module API for Java**

iDTRONIC GmbH  
Ludwig-Reichling-Straße 4  
67059 Ludwigshafen  
Germany/Deutschland

Phone: +49 621 6690094-0  
Fax: +49 621 6690094-9  
E-Mail: [info@idtronic.de](mailto:info@idtronic.de)  
Web: [idtronic.de](http://idtronic.de)

Issue 1.7  
– 08. July 2025 –

Subject to alteration without prior notice.  
© Copyright iDTRONIC GmbH 2025  
Printed in Germany

## Contents

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
1.1	<b>Enumerations .....</b>	<b>5</b>
1.1.1	READER_ERR Enumeration .....	5
1.1.2	Mtr_Param Enumeration .....	6
1.1.3	SL_TagProtocol Enumeration .....	7
1.1.4	Region_Conf Enumeration .....	7
1.1.5	Lock_Obj Enumeration .....	8
1.1.6	Lock_Type Enumeration .....	8
1.1.7	GpiTrigger_Type Enumeration .....	8
1.2	<b>Helper Classes .....</b>	<b>9</b>
1.2.1	TAGINFO Class .....	9
1.2.2	TagFilter_ST Class .....	9
1.2.3	TagSelector_ST Class .....	9
1.2.4	EmbeddedData_ST Class .....	10
1.2.5	BackReadOption Class .....	10
1.2.6	TagMetaFlags Class .....	11
1.2.7	GpiState_ST Class .....	11
1.2.8	GpiInfo_ST Class .....	11
1.2.9	Inv_Potl Class .....	11
1.2.10	Inv_Potls_ST Class .....	11
1.2.11	AntPortsVSWR class .....	11
1.2.12	AntPower Class .....	12
1.2.13	AntPowerConf Class .....	12
1.2.14	ConnAnts_ST Class .....	12
1.2.15	HoptableData_ST Class .....	12
1.2.16	MultiTagSelectors_ST Class .....	12
1.2.17	GPITrigger Class .....	12
1.2.18	OnGpiTriggerBlock Class .....	12
1.2.19	OnReadingErrorBlock Class .....	13
1.2.20	OnTagReadBlock Class .....	13
1.2.21	CustomParam_ST Class .....	13
1.2.22	Default_Param Class .....	14
1.2.23	Reader_Ip Class .....	14
1.2.24	HardwareDetails Class .....	15
1.2.25	ReadListener Interface .....	15
1.2.26	ReadExceptionListener Interface .....	15
1.2.27	GpiTriggerListener Interface .....	15
1.2.28	GpiTriggerBoundaryListener Interface .....	15
1.3	<b>Reader Class Methods .....</b>	<b>16</b>
<b>2</b>	<b>Reader Life Cycle .....</b>	<b>16</b>
2.1	<b>InitReader_Notype Method .....</b>	<b>16</b>
2.2	<b>CloseReader Method .....</b>	<b>17</b>
<b>3</b>	<b>Reader Parameters .....</b>	<b>17</b>
3.1	<b>Get Parameter .....</b>	<b>17</b>
3.2	<b>Set Parameter .....</b>	<b>17</b>
3.3	<b>Default Configuration .....</b>	<b>17</b>

3.4	<b>Custom Parameter Configuration</b> .....	18
3.5	<b>Parameters</b> .....	18
3.5.1	Read/write parameters .....	18
3.5.2	Read-only Parameters .....	20
4	<b>Synchronous Inventory</b> .....	20
4.1	<b>TagInventory_Raw Method</b> .....	20
4.2	<b>GetNextTag Method</b> .....	21
5	<b>Fast Mode Inventory</b> .....	21
5.1	<b>AsyncStartReading Method</b> .....	21
5.2	<b>AsyncGetTagCount Method</b> .....	22
5.3	<b>AsyncGetNextTag Method</b> .....	22
5.4	<b>AsyncStopReading Method</b> .....	22
6	<b>Asynchronous Inventory</b> .....	23
6.1	<b>Start Inventory</b> .....	24
6.1.1	addReadListener Method .....	24
6.1.2	addReadExceptionListener Method .....	24
6.1.3	addGpiTriggerListener Method .....	25
6.1.4	addGpiTriggerBoundaryListener Method .....	25
6.1.5	StartReading Method .....	25
6.2	<b>Stop Inventory</b> .....	25
6.2.1	StopReading Method .....	26
7	<b>Tag Access</b> .....	26
7.1	<b>GetTagData Method</b> .....	26
7.2	<b>WriteTagData Method</b> .....	27
7.3	<b>WriteTagEpcEx Method</b> .....	27
7.4	<b>LockTag Method</b> .....	28
7.5	<b>KillTag Method</b> .....	29
8	<b>Peripheral Functions</b> .....	29
8.1	<b>GetGPI Method</b> .....	29
8.2	<b>GetGPIEx Method</b> .....	29
8.3	<b>SetGPO Method</b> .....	30
9	<b>String Functions</b> .....	30
9.1	<b>Hex2Str Function</b> .....	30
9.2	<b>Str2Hex Function</b> .....	30
10	<b>Error Handling</b> .....	30
10.1	<b>GetLastDetailError Method</b> .....	31
10.2	<b>Error Codes</b> .....	31

# 1 Introduction

ModuleAPI\_J.jar is the library file of the Java version of ModuleAPI, and ModuleAPI\_J.jar calls the interface function of the C language version of ModuleAPI through the Native interface.

On Windows platforms, ModuleAPIJni.dll is the JNI encapsulation for ModuleAPI\_C.dll. ModuleAPIJni.dll internally references functions in ACE.dll and PCOMM.dll. Please put them in the same catalogue for references as ModuleAPI\_J.jar or add them to the Java library path. Note that on a 32-bit operating system, ModuleAPIJni.dll depends on 32-bit ACE.dll and PCOMM.dll, while on 64-bit operating systems it only depends on 64-bit PCOMM.dll.

On Linux and Android platforms, libModuleAPIJni.so is the library file for JNI encapsulation.

Since the Java development kit basically calls the C language development interface directly through JNI, some of the definitions of methods and some auxiliary classes do not follow the coding conventions for Java language development. All sample code snippets in the documentation assume that `rdr` is an instance of the Reader class.

Import package:

```
1. import com.uhf.api.cls.Reader;
2. import com.uhf.api.cls.Reader.*;
```

## 1.1 Enumerations

There are many enumerations defined as the internal enumeration in Reader class. Many of them are used as arguments to methods in the Reader class.

### 1.1.1 READER\_ERR Enumeration

All API functions use the return value of this enumeration type. If the return value is not `MT_OK_ERR` and not a communication error such as `MT_IO_ERR` or `MT_INVALID_READER_HANDLE`, use the method `GetLastDetailError` to get further information on the error.

Member	Description
MT_OK_ERR	Success
MT_IO_ERR	IO error, usually the reader address does not exist, has restricted access or is occupied
MT_INTERNAL_DEV_ERR	Deprecated error code
MT_CMD_FAILED_ERR	Operation failed, but no fatal error
MT_CMD_NO_TAG_ERR	No tags found, tags must comply with current protocol
MT_M5E_FATAL_ERR	Deprecated error code
MT_OP_NOT_SUPPORTED	Unsupported operation
MT_INVALID_PARA	Invalid parameter
MT_INVALID_READER_HANDLE	Invalid reader handle, usually not initialized successfully
MT_HARDWARE_ALERT_ERR_BY_HIGN_RETURN_LOSS	High return loss, common issues are surrounding metal and loose antenna connections
MT_HARDWARE_ALERT_ERR_BY_TOO_MANY_RESET	RFID engine reset too many times
MT_HARDWARE_ALERT_ERR_BY_NO_ANTENNAS	The reader did not detect the antenna
MT_HARDWARE_ALERT_ERR_BY_HIGH_TEMPERATURE	Reader temperature is too high
MT_HARDWARE_ALERT_ERR_BY_READER_DOWN	No response from the reader
MT_HARDWARE_ALERT_ERR_BY_UNKNOWN_ERR	Unknown error hardware warning
M6E_INIT_FAILED	Deprecated error code
MT_OP_EXECING	The reader is busy and there is an operation being executed

MT_UNKNOWN_READER_TYPE	Unknown reader type, reader may not respond or the API version is too old
MT_OP_INVALID	Invalid operation
MT_HARDWARE_ALERT_BY_FAILED_RESET_MODLUE	Reset RFID engine failed
MT_MAX_ERR_NUM	Enumeration keyword error
MT_UPDFWFROMSP_OPENFILE_FAILED	Failed to open file when upgrade firmware by serial port
MT_UPDFWFROMSP_FILE_FORMAT_ERR	File format error when upgrade firmware by serial port
MT_JNI_INVALID_PARA	Invalid JNI call parameter

### 1.1.2 Mtr\_Param Enumeration

Used by Reader.ParamGet and Reader.ParamSet functions to indicate which parameter will be get or set.

Member	Description
MTR_PARAM_POTL_GEN2_SESSION	Gen2 session
MTR_PARAM_POTL_GEN2_Q	Gen2 Q value
MTR_PARAM_POTL_GEN2_TAGENCODING	Gen2 tag encoding
MTR_PARAM_POTL_GEN2_MAXEPCLEN	Gen2 maximum length of the EPC
MTR_PARAM_RF_ANTPOWER	Tx (output) power of antennas of reader in centi-dbm
MTR_PARAM_RF_MAXPOWER	Maximum tx power in centi-dbm
MTR_PARAM_RF_MINPOWER	Minimum tx power in centi-dbm
MTR_PARAM_TAG_FILTER	Tag filtering rules
MTR_PARAM_TAG_EMBEDDEDATA	Read data of another bank during inventory operation
MTR_PARAM_TAG_INVPOTL	Deprecated - Air protocol of inventory
MTR_PARAM_READER_CONN_ANT	All detected antennas on the reader antenna port
MTR_PARAM_READER_AVAILABLE_ANTPORTS	Number of antenna ports on the reader
MTR_PARAM_READER_IS_CHK_ANT	Antenna detection configuration
MTR_PARAM_READER_VERSION	Reader version number
MTR_PARAM_READER_IP	Reader IP address
MTR_PARAM_FREQUENCY_REGION	Frequency regulatory of reader
MTR_PARAM_FREQUENCY_HOPTABLE	The frequency hopping table of reader
MTR_PARAM_POTL_GEN2_WRITEMODE	Gen2 write mode
MTR_PARAM_POTL_GEN2_TARGET	Gen2 target
MTR_PARAM_TAGDATA_UNIQUEBYANT	Whether the buffer tag list is unique by antenna
MTR_PARAM_TAGDATA_UNIQUEBYEMDDATA	Whether the buffer tag list is unique by additional data
MTR_PARAM_TAGDATA_RECORDHIGHESTRSSI	Whether to record only the highest RSSI
MTR_PARAM_RF_TEMPERATURE	Reader temperature
MTR_PARAM_RF_HOPTIME	Frequency hopping time
MTR_PARAM_RF_LBT_ENABLE	Whether to enable LBT function
MTR_PARAM_POTL_ISO180006B_BLF	ISO-6B-BLF backward link rate
MTR_PARAM_TRANS_TIMEOUT	Transmission time in communication in milliseconds
MTR_PARAM_TAG_EMDSECUREREAD	Embedded encrypted security data
MTR_PARAM_POWERSAVE_MODE	RF power mode
MTR_PARAM_RF_ANTPORTS_VSWR	Antenna port VSWR value
MTR_PARAM_INTERNALSAVECONFIGURATION	Default configuration of the integrated device
MTR_PARAM_CUSTOM	Extended custom parameters
MTR_PARAM_READER_WATCHDOG	Watchdog parameters
MTR_PARAM_READER_ERRORDATA	Error message
MTR_PARAM_RF_HOPANTTIME	Antenna dwell time during fast mode inventory

MTR_PARAM_TAG_MULTISELECTORS	Multiple tag filtering rules (maximum of 16 rules)
MTR_PARAM_SAVEINMODULE	Module power-on default value
MTR_PARAM_SAVEINMODULE_BAUD	Module power-on default baud rate value
MTR_PARAM_TAG_OPPTL	Current tag protocol

### 1.1.3 SL\_TagProtocol Enumeration

Air interface protocol for tags.

Member	Description
SL_TAG_PROTOCOL_NONE	None
SL_TAG_PROTOCOL_ISO180006B	ISO18000-6b
SL_TAG_PROTOCOL_GEN2	GEN2
SL_TAG_PROTOCOL_ISO180006B_UCODE	ISO18000-6B-UCODE
SL_TAG_PROTOCOL_IPX64	IPX64
SL_TAG_PROTOCOL_IPX256	IPX256
SL_TAG_PROTOCOL_GB	GB, national standard

### 1.1.4 Region\_Conf Enumeration

Frequency region in which the reader operates.

Member	Description
RG_NA	North America (902-928 MHz)
RG_EU	Europe, ETSI EN 302 208
RG_EU2	Europe, ETSI EN 300 220
RG_EU3	Europe, revised ETSI EN 302 208 (865-867 MHz)
RG_KR	Korean (917-921 MHz)
RG_PRC	Chinese (920-925 MHz)
RG_OPEN	No regulatory compliance enforced (860-960 MHz)
RG_IN	India (865-867 MHz)
RG_JP	Japan (916-921 MHz, no LBT)
RG_HK	Hong Kong (920-925 MHz)
RG_TAIWAN	Taiwan (920-927 MHz)
RG_MALAYSIA	Malaysia (919-923 MHz)
RG_SOUTH_AFRICA	South Africa (915-918 MHz)
RG_BRAZIL	Brazil (902-927 MHz)
RG_THAILAND	Thailand (920-925 MHz)
RG_SINGAPORE	Singapore (920-925 MHz)
RG_AUSTRALIA	Australia (920 – 925 MHz)
RG_URUGUAY	Uruguay (916-927 MHz)
RG_VIETNAM	Vietnam (918-922 MHz)
RG_ISRAEL	Israel (916 MHz)
RG_PHILIPPINES	Philippines (918-920 MHz)
RG_INDONESIA	Indonesia (917-922 MHz)
RG_NEW_ZEALAND	New Zealand (922-927 MHz)
RG_PERU	Peru (916-927 MHz)
RG_RUSSIA	Russia (916-920 MHz)
RG_JP2_LBT6	Japan (916-921 MHz, with LBT)
RG_JP3_NLBT19	Japan (916-924 MHz, without LBT)

### 1.1.5 Lock\_Obj Enumeration

Used for LockTag function to indicate which memory objects of a tag to lock.

Member	Description
LOCK_OBJECT_KILL_PASSWORD	Kill password
LOCK_OBJECT_ACCESS_PASSWD	Access password
LOCK_OBJECT_BANK1	Bank1 (EPC area in GEN2)
LOCK_OBJECT_BANK2	Bank2 (TID area in GEN2)
LOCK_OBJECT_BANK3	Bank3 (User area in GEN2)

### 1.1.6 Lock\_Type Enumeration

Used for LockTag function to indicate how to lock a tag. After locking, the specified area cannot be read or written. Without unlocking, a password is required for reading and writing. When locking the EPC or User area, the area can be read but not written. The TID area is generally permanently locked at the factory and can only be read.

Member	Description
KILL_PASSWORD_UNLOCK	Unlock kill password
KILL_PASSWORD_LOCK	Temporarily lock kill password
KILL_PASSWORD_PERM_LOCK	Permanently lock kill password
KILL_PASSWORD_PERM_UNLOCK	Permanently unlock the kill password
ACCESS_PASSWD_UNLOCK	Unlock access password
ACCESS_PASSWD_LOCK	Temporarily lock access password
ACCESS_PASSWD_PERM_LOCK	Permanently lock access password
ACCESS_PASSWD_PERM_UNLOCK	Permanently unlock the access password
BANK1_UNLOCK	Unlock bank 1
BANK1_LOCK	Temporarily lock bank 1
BANK1_PERM_LOCK	Permanently lock bank 1
BANK1_PERM_UNLOCK	Permanently unlock bank 1
BANK2_UNLOCK	Unlock bank 2
BANK2_LOCK	Temporarily lock bank 2
BANK2_PERM_LOCK	Permanently lock bank 2
BANK2_PERM_UNLOCK	Permanently unlock bank 2
BANK3_UNLOCK	Unlock bank 3
BANK3_LOCK	Temporarily lock bank 3
BANK3_PERM_LOCK	Permanently lock bank 3
BANK3_PERM_UNLOCK	Permanently unlock bank 3

### 1.1.7 GpiTrigger\_Type Enumeration

Defines the trigger type for GPI triggered inventory.

Member	Description
GPITRIGGER_NONE	Don't use GPI trigger
GPITRIGGER_TRI1START_TRI2STOP	Start triggered by condition 1 and stop triggered by condition 2
GPITRIGGER_TRI1START_TIMEOUTSTOP	Start triggered by condition 1 and stop after timeout
GPITRIGGER_TRI1ORTRI2START_TIMEOUTSTOP	Start triggered by condition 1 or 2 and stop after timeout
GPITRIGGER_TRI1ORTRI2START_TRI1ORTRI2STOP	Start triggered by condition 1 or 2 and stop triggered by condition 1 or 2



## 1.2 Helper Classes

Many of them are used as arguments to methods in the Reader class. The following are commonly used helper classes.

### 1.2.1 TAGINFO Class

The detailed information of the inventoried tag.

Member	Type	Description
ReadCnt	int	The number of times the tag is inventoried.
RSSI	int	Received signal strength of tag.
AntennaID	byte	The antenna ID by which the tag is inventoried.
Frequency	int	The frequency on which the tag was inventoried.
TimeStamp	int	The time the tag was inventoried, relative to the time the command of inventory was issued in millisecond.
EmbeddedDataLen	short	The length of embedded data, in bytes.
EmbeddedData	byte[]	Embedded data, another bank data read when inventory.
Res	byte[]	Reserved
EpcLen	short	The length of EPC code, in bytes.
PC	byte[]	PC field of EPC bank
CRC	byte[]	CRC field of EPC bank
EpcId	byte[]	EPC code
Phase	int	The phase on which the tag was inventoried.
protocol	SL_TagProtocol	Air interface protocol of tag

### 1.2.2 TagFilter\_ST Class

Tag inventory and access operations can only be applied to the tags that match the rules through filtering rules.

Member	Type	Description
bank	int	The memory bank to be matched, legal values are 0,1,2,3,4. 0—3 means gen2 tag's bank0-3; 4 means iso180006b memory.
startaddr	int	The memory bank offset, in bits, at which to begin comparing the fdata.
flen	int	The length, in bits, of the fdata
fdata	byte[]	The data to be compared
isInvert	int	0 means matching the filter criteria, 1 means matching the inverted filter criteria.

**Example:** Filter tag with EPC starting with 0x1234

```
1. TagFilter_ST tf = rdr.new TagFilter_ST();
2. tf.fdata[0] = 0x12;
3. tf.fdata[1] = 0x34;
4. tf.bank = 1;
5. tf.startaddr = 32;
6. tf.flen = 16;
7. tf.isInvert = 0;
```

### 1.2.3 TagSelector\_ST Class

Similar to TagFilter\_ST, with the only difference being that there is no isInvert member.

Member	Type	Description
bank	int	The memory bank to be matched, legal values are 0,1,2,3,4. 0—3 means gen2 tag's bank0-3; 4 means iso180006b memory.
startaddr	int	The memory bank offset, in bits, at which to begin comparing the fdata.
slen	int	The length, in bits, of the sdata

sdata	byte[ ]	The data to be compared
-------	---------	-------------------------

#### 1.2.4 EmbeddedData\_ST Class

Used to specify the details of reading a memory bank during inventory.

Member	Type	Description
bank	int	Specifies which bank to read during inventory, legal values are 0, 1, 2, 3
startaddr	int	The memory bank offset, in blocks, at which to begin reading
bytecnt	int	The number of bytes to read starting at startaddr
accesspwd	byte[ ]	Access password, if no password is required it can be set to null

**Example:** Read two blocks from the USER bank starting at the 4<sup>th</sup> block during inventory. The access password is set to 0x11112222.

```

1. EmbeddedData_ST emd = rdr.new EmbeddedData_ST();
2. emd.accesspwd = new byte[4];
3. emd.ascpwd[0] = 0x11;
4. emd.ascpwd[1] = 0x11;
5. emd.ascpwd[2] = 0x22;
6. emd.ascpwd[3] = 0x22;
7. emd.bank = 3;
8. emd.startaddr = 4;
9. emd.bytecnt = 4;

```

#### 1.2.5 BackReadOption Class

The detailed configuration of asynchronous inventory.

Member	Type	Description
ReadDuration	short	Inventory cycle in milliseconds when using common asynchronous inventory mode. If using high-speed asynchronous inventory mode the attribute does nothing.
ReadInterval	int	The idle interval in milliseconds between two inventory cycles when using common asynchronous inventory mode. If using high-speed asynchronous inventory mode the attribute does nothing.
IsFastRead	Boolean	If 1, it uses high speed asynchronous inventory mode, otherwise it uses common asynchronous inventory mode.
FastReadDutyRation	char	The upper 4 bits must be 0, the lower 4 bits are valid, and the lower 4 bits are used to indicate the proportion of time spent resting during the reader's inventory. If the FastReadDutyRation is 0-9 the proportion is FastReadDutyRation * 5%. If the FastReadDutyRation is 10-15 the proportion is 50%.
TMFlags	TagMetaFlags	Specifies which metadata information the tag carries when using the high-speed asynchronous inventory mode. If using common asynchronous inventory mode the attribute does nothing.
GpiTrigger	GPITrigger	The GPI trigger condition used for starting or stopping asynchronous inventory if IsGPITrigger is true.
IsGPITrigger	boolean	If true, starting and stopping asynchronous inventory is triggered by GPI trigger condition.

### 1.2.6 TagMetaFlags Class

Specifies which metadata information the tag carries when using the high-speed asynchronous inventory mode. Setting the members in the following table to true means asking the reader to return the corresponding information data, otherwise set to false.

Member	Type	Description
IsAntennaID	boolean	The antenna ID on which the tag was inventoried
IsReadCnt	boolean	The read count of the tag
IsRSSI	boolean	Received Signal Strength Indication of the tag
IsFrequency	boolean	The frequency on which the tag was inventoried
IsTimestamp	boolean	The time stamp when the tag was inventoried
IsRFU	boolean	The reserved field
IsEmdData	boolean	The bank data of the tag. During inventory the reader can read tag data of another bank depending on the EmbeddedCmdOfInventory parameter.
IsProtocol	boolean	The protocol

### 1.2.7 GpiState\_ST Class

The status of one GPI pin.

Member	Type	Description
GpiId	int	GPI number, numbered from 1
State	int	GPI state

### 1.2.8 GpiInfo\_ST Class

The status of GPI pins

Member	Type	Description
gpiCount	int	The number of elements of gpiStats
gpiStats	GpiState_ST[]	The status of GPI pins

### 1.2.9 Inv\_Potl Class

The air protocol during inventory.

Member	Description
potl	The air protocol of tag
weight	The proportion of inventory time occupied by the protocol when inventorying tags of multiple protocols, reader currently only supports the Gen2 protocol. Set this field to 100.

### 1.2.10 Inv\_Potls\_ST Class

The air protocol during inventory.

Member	Description
potlcnt	The number of elements of potls
potls	The air protocols of inventory

### 1.2.11 AntPortsVSWR class

Member	Type	Description
andid	int	The antenna ID for detecting standing wave, numbered from 1
power	short	Detected transmission power
region	Region_Conf	Detected frequency band region
frecount	int	Frequency value element group length

vswrs	FrequencyVSWR[ ]	Returned standing wave value
-------	------------------	------------------------------

Method toString() lists the standing wave values corresponding to the frequency point values.

#### 1.2.12 AntPower Class

The Tx power of one antenna.

Member	Type	Description
antid	int	The antenna ID, numbered from 1
readPower	short	Transmit power in centi-dbm when the reader performs a read-related operation.
writePower	short	Transmit power in centi-dbm when the reader performs a write-related operation.

#### 1.2.13 AntPowerConf Class

Tx power configuration of antennas

Member	Type	Description
antcnt	int	The number of elements of Powers
Powers	AntPower[ ]	Tx power configuration of antennas

#### 1.2.14 ConnAnts\_ST Class

The antennas to which the reader is connected

Member	Type	Description
antcnt	int	The number of elements of connected ants
connectedants	int[ ]	The antennas to which the reader is connected

#### 1.2.15 HoptableData\_ST Class

The frequency hopping table of the reader.

Member	Type	Description
lenhtb	int	The number of elements in htb, maximum length is 100
htb	int[ ]	The frequency hopping table, the frequency values must be within the defined frequency band (in KHz)

#### 1.2.16 MultiTagSelectors\_ST Class

Filter rule group can define up to 16 filter rules that are logically related to each other.

Member	Type	Description
tagselectors	TagSelector_ST[ ]	Tag filter rule array
tagselectorcnt	int	Number of elements in tagselectors

#### 1.2.17 GPITrigger Class

Used to set the parameters of the GPI trigger inventory interface.

Member	Type	Description
GpiTrigger1States	GpiInfo_ST	Trigger configuration for condition 1
GpiTrigger2States	GpiInfo_ST	Trigger configuration for condition 2
TriggerType	GpiTrigger_Type	Trigger type
StopTriggerTimeout	int	When the TriggerType has _TIMEOUTSTOP, this defines the timeout in milliseconds

#### 1.2.18 OnGpiTriggerBlock Class

Member	Type	Description
--------	------	-------------

gshandler	GpiStateHandler	Notifies when trigger conditions 1 and 2 are met
gtbhandler	GpiTriggerBoundaryHandler	Notifies the reader to start and stop inventory
cookie	void*	User-defined data

### 1.2.19 OnReadingErrorBlock Class

Asynchronous storage error event handling.

Member	Type	Description
handler	ReadingErrorHandler	Function pointer to the processing function
cookie	void*	User-defined data

### 1.2.20 OnTagReadBlock Class

Member	Type	Description
handler	TagReadHandler	Callback for when a tag is inventoried (asynchronous inventory)
cookie	void*	User data, will be passed to handler as parameter

### 1.2.21 CustomParam\_ST Class

Parameter value type when the keyword is MTR\_PARAM\_CUSTOM.

Member	Type	Description
ParamName	String	<p>"0": No parameter is determined by ParamVal</p> <p>"reader/rdrdetails": Get hardware information of the reader (see method GetSerialNumber)</p> <p>"reader/moduleinfo": Get detailed information of the reader</p> <p>"reader/macaddr": Get MAC address</p> <p>"Reader/Ex10fastmode": Switch between normal fast mode and Ex fast mode</p> <p>"tagcustomcmd/fastid": Whether to enable the fastid function which requires tag support</p> <p>"tagcustomcmd/tagfocus": Whether to enable the tagfocus function</p> <p>"gen2op/multiembeddeddata": Setting multiple additional data items (see method SetInvMultiEmbeddedData)</p> <p>"Reader/Savestandby": Whether to switch the reader to bootloader layer when closing it</p> <p>"R2000/oemregister": Writing to the OEM register</p> <p>"Reader/realtimeinfo": Whether to enable periodic upload of the status when working in fast mode</p> <p>"Reader/dutycycle": Setting duty cycle parameters</p> <p>"Reader/powermode": Setting the reader power mode</p> <p>"Reader/rssifilter": RSSI is filtered internally in the reader</p>
ParamVal	byte[]	<p>When ParamName is "0", the first byte determines the custom operation:</p> <ul style="list-style-type: none"> <li>"0x01": Indicates a carrier test and the parameter value byte sequence is: on or off (1 byte, 1 on, 0 off) + antenna id (1 byte) + power (2 bytes) + frequency (4 bytes)</li> <li>"0x02": Switching the module to the BootLoader layer</li> <li>"0x03": Reading and writing registers: 4-byte address + 4-byte value</li> </ul> <p>When ParamName is not "0":</p> <ul style="list-style-type: none"> <li>"reader/rdrdetails": 16 bytes reserved + 4 bytes hardware version + 12 bytes serial number</li> <li>"reader/moduleinfo": 16 bytes reserved + 16 bytes version information (4 bytes bootloader + 4 bytes hardware version + 4 bytes firmware date + 4 bytes firmware version) + 12 bytes serial number</li> <li>"reader/macaddr": 6 bytes</li> </ul>

		<ul style="list-style-type: none"> <li>“Reader/Ex10fastmode”: mode (1 byte, 1 is Ex fast mode, 0 is normal mode) + fixed byte to 20 + use case (1 byte, 0 is many labels, 1 is small number of labels) + 19 bytes (all 0)</li> <li>“tagcustomcmd/fastid”: 1 byte (1 on, 0 off)</li> <li>“tagcustomcmd/tagfocus”: 1 byte (1 on, 0 off)</li> <li>“gen2op/multiembeddeddata”: number of additional data items (1 byte) + password (4 bytes) + additional 1 bank (1 byte) + additional 1 address (4 bytes) + additional 1 block number (1 byte) + additional 2 bank (1 byte) + ...</li> <li>“Reader/Savestandby”: 1 byte (1 open, 0 closed)</li> <li>“R2000/oemregister”: 4-byte address + 4-byte value</li> <li>“Reader/dutycycle”: duty cycle execution before inventory (2 bytes, in milliseconds) + duty cycle execution base time (2 bytes, in milliseconds)</li> <li>“Reader/powermode”: 1 byte (0x00 full power, 0x01 min saving, 0x02 med saving, 0x03 max saving)</li> <li>“Reader/rssifilter”: when getting parameters 4 bytes all 0, when setting parameters to cancel internal filtering 1 byte 0x01 + 3 bytes all 0, when setting to enable internal filtering 1 byte 0x01 + 1 byte 0xAA + 1 byte signed RSSI value + 1 byte 0x00</li> </ul>
--	--	---

### 1.2.22 Default\_Param Class

Used to set the default parameters for when the module is powered on.

Member	Type	Description
key	Mtr_Param	Defined power-on default parameter keyword: <ul style="list-style-type: none"> <li>MTR_PARAM_SAVEINMODULE_BAUD: val type is int[1] with possible values 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800 or 921600</li> <li>MTR_PARAM_FREQUENCY_REGION: val type is byte[1]</li> <li>MTR_PARAM_RF_ANTPOWER: val type is unsigned short[], serial antenna number 1, read power 1, write power 1, antenna switching time (fixed to 0), antenna number 2, read power 2, ...</li> <li>MTR_PARAM_POTL_GEN2_SESSION: val type is int[1]</li> <li>MTR_PARAM_POTL_GEN2_Q: val type is int[1]</li> <li>MTR_PARAM_POTL_GEN2_TARGET: val type is int[1]</li> <li>MTR_PARAM_SAVEINMODULE: val type is int[1]</li> </ul>
subkey	String	When key is MTR_PARAM_SAVEINMODULE, subkey can be <ul style="list-style-type: none"> <li>“modulesave/hpupload”: Customize HP function and upload actively</li> <li>“modulesave/antport”: The default antenna port is used when powered on</li> <li>“modulesave/default”: Restore the initial power-on default value</li> </ul>
isdefault	boolean	When modifying the custom default configuration, it should be set to false. When set to true, the default value will be used and the value of val will be ignored.
val	Object	Parameter value

### 1.2.23 Reader\_Ip Class

Member	Type	Description
ip	byte[]	Reader IP address
mask	byte[]	Reader network mask
gateway	byte[]	Reader network gateway

### 1.2.24 HardwareDetails Class

Refer to the method `GetHardwareDetails`.

Member	Type	Description
module	Module_Type	Module type of the reader
board	MaindBoard_Type	Mainboard type of the reader
logictype	Reader_Type	Type of the reader
antportnumbers	int	Number of antenna ports for self-test

### 1.2.25 ReadListener Interface

The callback on tags inventoried during asynchronous inventory.

```
1. void tagRead(Reader r, TAGINFO[] t)
```

Parameter	Description
r	The instance of Reader Class
t	The tags inventoried

### 1.2.26 ReadExceptionListener Interface

The callback data on error during asynchronous inventory.

```
1. void tagReadException(Reader r, READER_ERR re)
```

Parameter	Description
r	The instance of Reader Class
re	The error code generated during asynchronous inventory

### 1.2.27 GpiTriggerListener Interface

The callback for notifying GPI states during asynchronous inventory.

```
1. void GpiTrigger(Reader r, GpiInfo_ST gist, int triid)
```

Parameter	Description
r	The instance of Reader Class
gist	The GPI status when triggering start or stop asynchronous inventory
triid	The condition number that triggers the start or stop of the asynchronous inventory

### 1.2.28 GpiTriggerBoundaryListener Interface

This callback will be called to notify the cause of starting or stopping inventory when starting or stopping asynchronous inventory if asynchronous Inventory is configured to start by GPI trigger.

```
1. void GpiTriggerBoundary(Reader r, GpiTriggerBoundaryType btype, GpiTriggerBoundaryReasonType reason)
```

Parameter	Description
r	The instance of Reader Class
btype	GPITriggerBoundary_StartInventory means starting inventory; GPITriggerBoundary_StopInventory means stopping inventory.
reason	GpiTriggerBoundaryReason_ByGpi means starting or stopping inventory by GPI trigger; GpiTriggerBoundaryReason_ByTimeout means stopping inventory by timeout.

### 1.3 Reader Class Methods

All methods of the Reader Class in the development package will return a value of `READER_ERR` type, indicating whether the execution of the method was successful or why it failed. Many of the methods of the Reader class have the parameter `timeout`, this parameter specifies the maximum time in milliseconds for the API method to execute or block. If the method completes the operation earlier than this timeout, it will return earlier. The valid timeout range is 50-65535.

Method	Description
<code>Reader.InitReader_Notype</code>	Initialize the reader
<code>Reader.CloseReader</code>	Free the internal resource of Reader Class
<code>Reader.ParamGet</code>	Get parameters of the reader
<code>Reader.ParamSet</code>	Set parameters of the reader
<code>Reader.TagInventory_Raw</code>	Synchronous inventory operation
<code>Reader.GetNextTag</code>	Get one tag from the tags inventoried by the <code>TagInventory_Raw</code> method
<code>Reader.addReadListener</code>	Add the <code>ReadListener</code> callback
<code>Reader.addReadExceptionListener</code>	Add the <code>ReadExceptionListener</code> callback
<code>Reader.addGpiTriggerListener</code>	Add the <code>GpiTriggerListener</code> callback
<code>Reader.addGpiTriggerBoundaryListener</code>	Add the <code>GpiTriggerBoundaryListener</code> callback
<code>Reader.StartReading</code>	Start asynchronous inventory operation
<code>Reader.StopReading</code>	Stop asynchronous inventory operation
<code>Reader.GetTagData</code>	Read data block of the storage area of the tag
<code>Reader.WriteTagData</code>	Write data block to storage area of the tag
<code>Reader.WriteTagEpcEx</code>	Write the EPC code of a tag
<code>Reader.LockTag</code>	Lock the storage areas of a tag
<code>Reader.KillTag</code>	Kill a tag
<code>Reader.GetGPI</code>	Get the state of one gpi pin
<code>Reader.GetGPIEx</code>	Get states of multiple gpi pins
<code>Reader.SetGPO</code>	Set state of one gpo pin
<code>Reader.GetLastDetailError</code>	Get more detailed error information

## 2 Reader Life Cycle

The Reader class contains all the methods that the reader will use throughout the application lifecycle.

Firstly, you should call the `Reader.InitReader_Notype` method to initialize the reader. After that you may call the `Reader.ParamSet` or `Reader.ParamGet` methods to configure the reader. Now you can call tag operations and peripheral methods of the Reader class. Call the `Reader.CloseReader` method when you no longer use the reader.

### 2.1 InitReader\_Notype Method

Initialize the reader.

```
1. public READER_ERR InitReader_Notype(String src, int antscnt)
```

Parameter	Description
<code>src</code>	IP address or serial port number of the reader e.g. com1.
<code>antscnt</code>	The number of physical ports / antennas of the reader.

#### Example

```
1. Reader.READER_ERR err = rdr.InitReader_Notype("192.168.1.100", 4);
```



## 2.2 CloseReader Method

Close the reader and release related resources.

```
1. public void CloseReader()
```

## 3 Reader Parameters

The Reader.ParamGet and Reader.ParamSet methods are to get and set the readers' parameters. Every parameter of the reader is represented by a key value which is of Reader.Mtr\_Param Enumeration type. The parameters can be divided into read only and read/write parameters. A parameter of the reader will be in effect until it is set to another value.

### 3.1 Get Parameter

The Reader.ParamGet method is for getting the parameters of a reader. You can call this method at any time during the lifetime of a reader.

```
1. public Reader_ERR ParamGet(Mtr_Param key, Object val)
```

Parameter	Description
key	The key of a parameter
val	Parameter value, used as a return parameter

**Example:** Get the Gen2Session parameter.

```
1. int[] sess = new int[1];
2. Reader.READER_ERR err = rdr.ParamGet(Reader.Mtr_Param.MTR_PARAM_POTL_GEN2_SESSION, sess);
```

### 3.2 Set Parameter

The Reader.ParamSet method is for setting the parameters of a reader. You can call this method at any time during the lifetime of a reader. Once a parameter was set, it will be in effect during the whole lifetime of the reader unless it is modified.

```
1. public Reader_ERR ParamSet(Mtr_Param key, Object val)
```

Parameter	Description
key	The key of a parameter
val	Parameter value

**Example:** Set the Gen2Session of the reader as Session 0.

```
1. Reader.READER_ERR err = rdr.ParamSet(Reader.Mtr_Param.MTR_PARAM_POTL_GEN2_SESSION, new int[] {0});
```

### 3.3 Default Configuration

When the reader is powered on and initialized, the module reads the parameters stored in the internal registers and uses the values to configure the reader after powering on. Modifying the default configuration is permanent, meaning the parameter values won't be lost due to power failure of the reader.

**Example:** Set the serial baud rate of the reader to 921600.

```
1. Default_Param dp = rdr.new Default_Param();
2. dp.isdefault = false;
3. dp.key = Mtr_Param.MTR_PARAM_SAVEINMODULE_BAUD;
4. dp.val = 921600;
5. err = Jreader.ParamSet(Mtr_Param.MTR_PARAM_SAVEINMODULE, dp);
```

**Example:** Set Gen2 Session.

```
1. Default_Param dp = rdr.new Default_Param();
```

```

2. dp.isdefault = false;
3. dp.key = Mtr_Param.MTR_PARAM_POTL_GEN2_SESSION;
4. dp.val = 1;
5. err = rdr.ParamSet(Mtr_Param.MTR_PARAM_SAVEINMODULE, dp);

```

### 3.4 Custom Parameter Configuration

Mainly for adapting to the expansion of reader functions and related differences between different products.

**Example:** Enable fastid.

```

1. CustomParam_ST cpara = rdr.new CustomParam_ST();
2. cpara.ParamName = "tagcustomcmd/fastid";
3. byte[] vals = new byte[1];
4. vals[0] = 1;
5. cpara.ParamVal = vals;
6. err = rdr.ParamSet(Mtr_Param.MTR_PARAM_CUSTOM, cpara);

```

### 3.5 Parameters

The parameters can be divided into read only and read/write parameters. A parameter of the reader will be in effect until it is set to another value.

#### 3.5.1 Read/write parameters

Key	Value Type	Description
MTR_PARAM_POTL_GEN2_SESSION	int[1]	Legal values: 0,1,2,3. For few but fast-moving tags use Session0; for many but slow-moving tags use Session1; Session2 and 3 for when tags are read once and won't be read again for a long time.
MTR_PARAM_POTL_GEN2_Q	int[1]	Legal values: -1 to 15. (-1: automatically adjust Q value; 0 to 15: static Q value)
MTR_PARAM_POTL_GEN2_TAGENCODING	int[1]	R2000 chip supports profile1-profile5: value (0x10-0x15) The E series code value supports 203, 111 (RF_MODE_11), 220, 101 (RF_MODE_1), 45, 115 (RF_MODE_15), 112 (RF_MODE_12), 103 (RF_MODE_3), 105 (RF_MODE_5), 107 (RF_MODE_7), 113 (RF_MODE_13) The SFM series code value, 6C protocol (100-109), GB protocol (110-118), 6B protocol (119), GB2 protocol (120-132)
MTR_PARAM_RF_ANTPOWER	AntPowerConf	Note: the default power may not be the maximum transmitting power of reader.
MTR_PARAM_TAG_FILTER	TagFilter_ST	When you don't use filter criteria you can set to null.
MTR_PARAM_TAG_EMBEDDEDATA	EmbeddedData_ST	To disable this feature please set it to null.

MTR_PARAM_READER_IS_CHK_ANT	int[1]	1 is enabled, 0 is disabled. After enabling, the antenna is detected before transmitting power, if no antenna is connected, an error is returned
MTR_PARAM_READER_IP	Reader_Ip	IP address information
MTR_PARAM_FREQUENCY_REGION	Region_Conf	Operating frequency band
MTR_PARAM_FREQUENCY_HOPTABLE	HoptableData_ST	The frequency point must be within the spectrum of region regulatory.
MTR_PARAM_POTL_GEN2_WRITEMODE	int[1]	Legal values: 0, 1 (0: write in words; 1: write in blocks)
MTR_PARAM_POTL_GEN2_TARGET	int[1]	Legal values: 0,1,2,3; 0: A; 1: B; 2: A->B; 3: B->A
MTR_PARAM_TAGDATA_UNIQUEBYANT	int[1]	Legal values: 0,1 (0: no matter how many antennas read the tag there would be only one tag record; 1: different tag records when the same tag was read by different antennas)
MTR_PARAM_TAGDATA_UNIQUEBYEMDDATA	int[1]	Legal values: 0,1 (0: regard as one tag record; 1: regard as multiply tag records)
MTR_PARAM_TAGDATA_RECORDHIGHESTRSSI	int[1]	Legal value: 0,1 (0: record the highest RSSI value; 1 : not to record the highest RSSI value)
MTR_PARAM_RF_HOPTIME	int[1]	Frequency hopping time in milliseconds
MTR_PARAM_TRANS_TIMEOUT	int[1]	Transmission time in milliseconds
MTR_PARAM_TAG_EMDSECUREREAD	EmbededSecureRead_ST	The member tagtype of EmbededSecureRead_ST indicates the tag type, 1: Alien Higgs3; 2: Impinj Monza 4, other values are illegal; pwdtype indicates the password type, 1: fixed password; 2: calculated password, other values are illegal; ApIndexStartBitsInEpc indicates the starting epc bit position as the password index; ApIndexBitsNumInEpc indicates the number of bits as the password index; bank indicates which bank to read; address indicates the starting address in the bank (in blocks); blkcnt indicates the number of blocks to read; accesspwd indicates the access password (if fixed password mode is used)
MTR_PARAM_POWERSAVE_MODE	int[1]	Power saving level, 0 no power saving, level 3 highest power saving
MTR_PARAM_RF_ANTPORTS_VSWR	AntPortsVSWR	VSWR value reflects the compatibility of the antenna port and the antenna. The smaller the value the better
MTR_PARAM_CUSTOM	CustomParam_ST	Extended custom parameters

MTR_PARAM_READER_WATCHDOG	byte[]	Watchdog parameters
MTR_PARAM_READER_ERRORDATA	byte[]	Error message
MTR_PARAM_RF_HOPANTTIME	int[1]	Antenna dwell time during fast mode inventory
MTR_PARAM_TAG_MULTISELECTORS	MultiTagSelectors_ST	Multi-tag filtering rules, max. 16 rules
MTR_PARAM_SAVEINMODULE	Default_Param	Reader power-on default value
MTR_PARAM_TAG_OPOTL	int[1]	Current tag protocol
MTR_PARAM_TAG_INVPOTL	Inv_Potls_ST	This parameter must be set before calling the StartReading and TagInventory_Raw methods. Then the reader only supports the Gen2 protocol.

### 3.5.2 Read-only Parameters

Key	Value Type	Description
MTR_PARAM_READER_CONN_ANTs	ConnAnts_ST	Not all kinds of antennas can be detected.
MTR_PARAM_READER_AVAILABLE_ANTPORTS	int[1]	The maximum number of antennas that can be connected to the reader
MTR_PARAM_RF_MAXPOWER	short[]	The maximum power in centi-dBm
MTR_PARAM_RF_MINPOWER	short[]	The minimum power in centi-dBm
MTR_PARAM_READER_VERSION	Reader_Ver	Reader version number
MTR_PARAM_RF_TEMPERATURE	int[1]	Reader temperature

## 4 Synchronous Inventory

Synchronous inventory is implemented using the `Reader.TagInventory_Raw` and `Reader.GetNextTag` methods. `Reader.TagInventory_Raw` blocks until the end of the inventory. When the user needs to perform a short period of inventory operation, and the number of tags is not a lot, the synchronous inventory is a good choice.

Special Note: the `Reader.Mtr_Param.MTR_PARAM_TAG_INVPOTL` parameter must be set before calling the above methods, otherwise it will return an error.

### 4.1 TagInventory\_Raw Method

Implements synchronous inventory operation. After calling this method users should immediately call the `Reader.GetNextTag` method to get the inventoried tags.

```
1. public READER_ERR TagInventory_Raw(int[] ants,int antcnt, short timeout, int[] tagcnt)
```

Parameter	Description
ants	The antenna ids used for this operation
antcnt	The number of antennas in the ants array
timeout	The time period for the operation in milliseconds. Method blocks until the timeout is expired
tagcnt	Output parameter, the number of read tags

**Example:** TagInventory with antenna 1, 3 and 4

```
1. int[] ants = new int[]{1, 3, 4};
2. int[] tagnum = new int[1];
3. Reader.READER_ERR err = rdr.TagInventory_Raw(ants, 3, 1000, tagnum);
```

## 4.2 GetNextTag Method

Get the next tag. Users could get the number of tags inventoried by calling `Reader.TagInventory_Raw` method and then execute `Reader.GetNextTag` method as many times as the number of tags to get all tags inventoried.

```
1. public READER_ERR GetNextTag(TAGINFO TI)
```

Parameter	Description
TI	Output parameter, used to store the tag data

**Example:**

```
1. Reader.TAGINFO tagInfo = rdr.new TAGINFO();
2. Reader.READER_ERR err = rdr.GetNextTag(tagInfo);
```

**Example: Inventory**

```
1. public void inventory() {
2.     for (int i = 0; i < 10; ++i) {
3.         err = rdr.TagInventory_Raw(ants, ants.length, (short) timeout, tagcnt);
4.         if (err == READER_ERR.MT_OK_ERR) {
5.             System.out.println("tagcnt: " + tagcnt[0]);
6.             for (int j = 0; j < tagcnt[0]; ++j) {
7.                 TAGINFO tag_ = rdr.new TAGINFO();
8.                 err = rdr.GetNextTag(tag_);
9.                 if (err != READER_ERR.MT_OK_ERR)
10.                    break;
11.                 System.out.println("ecpid: " + Reader.bytes_Hexstr(tag_.EpcId) + " ant: " +
tag_.AntennaID + "fre: " + tag_.Frequency + " rssi: " + tag_.RSSI);
12.             }
13.         }
14.         if (err != READER_ERR.MT_OK_ERR)
15.             System.out.println("inventory tags err: " + err);
16.     }
17. }
```

## 5 Fast Mode Inventory

The fast mode inventory does not buffer the read tags inside the reader but uploads them while reading and is therefore real-time. Once the fast mode is started using the method `AsyncStartReading`, the reader will always be in working state. The `AsyncGetTagCount` method can operate in intervals of 50 ms to 1000 ms. Therefore, a timer or thread is required in the program design to maintain this frequency. When the number of tags returned is greater than 0, the method `AsyncGetNextTag` can be called to retrieve the tags. The method returns data of one tag each time. To stop the fast mode, the method `AsyncStopReading` method must be called.

The fast mode is divided into normal fast mode and EX special fast mode. The later must be supported by the E series reader. The default is the normal fast mode and can be switched by custom parameter configurations.

### 5.1 AsyncStartReading Method

After starting fast mode inventory, the reader is always in inventory state. It does not stop unless there is an exception or the `AsyncStopReading` method was called.

```
1. public READER_ERR AsyncStartReading(int[] ants, int antcnt, int option)
```

Parameter	Description
ants	Operating antennas

antcnt	Number of elements in ants
option	<p>The first byte controls whether to enable the antenna group polling notification method. If enabled (0x01), a virtual tag will be generated after polling the antenna once and its epc length is 1.</p> <p>The second and third byte are metaflags as explained in the example below.</p> <p>The lower four bits of the fourth byte are the pause percentage (0 is 0%, 1 is 5%, 2 is 10% ... 10 is 50%, 11 is 55 %, 12 is 60 %, 13 is 70%, 14 is 80% and 15 is 90 %). The highest bit of the fourth byte controls whether to enable the method of sending heartbeat packets to the host every 15 seconds, and one bit lower (bit6) controls whether to enable the function of automatically stopping inventory without new tags.</p>

**Example:**

```

1. // Return all data items
2. int metaflag = 0;
3. metaflag |= 0x0001; // return the number of reads
4. metaflag |= 0x0002; // return the RSSI signal
5. metaflag |= 0x0004; // Return the antenna number
6. metaflag |= 0x0008; // Return the frequency
7. metaflag |= 0x0010; // Return the timestamp
8. metaflag |= 0x0020; // Return the phase value
9. metaflag |= 0x0040; // Return the tag protocol value
10. metaflag |= 0x0080; // Return additional data
11. // Pause ratio: 50% (ratio of read time and interval)
12. int option = (metaflag << 8) | 10;
13. // enable heartbeat
14. option |= 0x80;
15. // enable automatic stop function
16. option |= 0x40;
17. // enable antenna polling notification function
18. option |= (0x01 << 24);

```

**5.2 AsyncGetTagCount Method**

Get the number of tags read in the buffer in fast mode. It is recommended that the query time interval is more than 50 ms.

```
1. public READER_ERR AsyncGetTagCount(int[] tagcnt)
```

Parameter	Description
tagcnt	Number of tags read

**5.3 AsyncGetNextTag Method**

Gets the buffered tag data in fast mode.

```
1. public READER_ERR AsyncGetNextTag(TAGINFO pTInfo)
```

Parameter	Description
pTInfo	Stores a tag and its data item value

**5.4 AsyncStopReading Method**

Stop fast mode inventory.

```
1. public READER_ERR AsyncStopReading()
```

**Example:**

```

1. public void fastmodeinventory() {
2.     err = rdr.AsyncStartReading(ants, ants.length, option);
3.     if (err == READER_ERR.MT_OK_ERR) {
4.         System.out.println("AsyncStartReading successful");
5.         startt = System.currentTimeMillis();
6.         endt = startt;
7.     } else {
8.         System.out.println("AsyncStartReading failed. Err: " + err);
9.         return;
10.    }
11.    while (true) {
12.        endt = System.currentTimeMillis();
13.        if ((endt - startt) >= testseconds || isbreak) {
14.            rdr.AsyncStopReading();
15.            break;
16.        } else {
17.            err = rdr.AsyncGetTagCount(tagcount);
18.            if (err == READER_ERR.MT_OK_ERR) {
19.                for (int i = 0; i < tagcount[0]; ++i) {
20.                    TAGINFO pTInfo = rdr.new TAGINFO();
21.                    err = rdr.AsyncGetNextTag(pTInfo);
22.                    if (err == READER_ERR.MT_OK_ERR)
23.                        System.out.println("epc: " + Reader.bytes_Hexstr(pTInfo.EpcId));
24.                    else {
25.                        isbreak = true;
26.                        break;
27.                    }
28.                }
29.            } else
30.                isbreak = true;
31.        }
32.        try
33.            Thread.sleep(1000);
34.        catch (InterruptedException e)
35.            e.printStackTrace();
36.    }
37. }

```

## 6 Asynchronous Inventory

The SDK can implement asynchronous inventory in two ways. One is to internally use a thread to call the synchronous inventory Reader.TagInventory\_Raw method. This asynchronous inventory is called common asynchronous inventory mode. The other is a true asynchronous inventory, where the reader is in a continuous inventory state. This way, the inventory performance of the reader is optimal, and applications with higher inventory performance requirements should use this mode of asynchronous inventory. This asynchronous inventory is called high-speed asynchronous inventory mode or short: fast mode inventory. The BackReadOption switches between normal inventory and fast mode.

Attention: Not all devices support asynchronous high-speed inventory mode. Only those readers built with R2000, E series and sfm series chips can support this inventory mode.

During asynchronous inventory, if tags are inventoried, the `ReadListener` callback will be called to pass the tag data to the user program; if the reader has an abnormal condition, the `ReadExceptionListener` callback will be called to pass the exception details to the user program; if the asynchronous inventory is configured to start or stop with a GPI trigger, the `GpiTriggerListener` and `GpiTriggerBoundaryListener` callbacks are called when the GPI condition is met.

Asynchronous Inventory Methods:

Method	Description
<code>Reader.StartReading</code>	Start asynchronous inventory
<code>Reader.StopReading</code>	Stop asynchronous inventory

Asynchronous Inventory Callback Interfaces

Callback interface	Description
<code>ReadListener</code>	Called when tags are inventoried
<code>ReadExceptionListener</code>	Called when exception occurs
<code>GpiTriggerListener</code>	Called when gpi trigger condition is met if asynchronous Inventory is configured to start by <code>GpiTrigger</code> .
<code>GpiTriggerBoundaryListener</code>	Called when starting or stopping asynchronous inventory if asynchronous inventory is configured to start by <code>GpiTrigger</code> .

## 6.1 Start Inventory

Call the `Reader.StartReading` method to start the asynchronous inventory. Before calling the `Reader.StartReading` method, you must ensure that some callbacks and a parameter have been set correctly.

- `Reader.Mtr_Param.MTR_PARAM_TAG_INVPOTL` parameter must be set, otherwise it will return an error.
- The `ReadListener` callback must be added, otherwise the inventoried tags cannot be passed to the user program.
- The `ReadExceptionListener` callback must be added, otherwise the reader exception state cannot be passed to the user program.

If the reader is currently performing an asynchronous inventory, then calling the `Reader.StartReading` method will return an error. If you are not sure about the current state of the reader, you can call the `Reader.StopReading` method first before starting the asynchronous inventory.

### 6.1.1 addReadListener Method

Add the `ReadListener` callback interface.

```
1. public void addReadListener(ReadListener listener)
```

Parameter	Description
listener	Tag event listener

### 6.1.2 addReadExceptionListener Method

Add the `ReadExceptionListener` callback interface

```
1. public void addReadExceptionListener(ReadExceptionListener listener)
```

Parameter	Description
listener	Exception event listener



### 6.1.3 addGpiTriggerListener Method

Add the GpiTriggerListener callback interface.

```
1. public void addGpiTriggerListener(GpiTriggerListener listener)
```

Parameter	Description
listener	GPI trigger event listener

### 6.1.4 addGpiTriggerBoundaryListener Method

Add the GpiTriggerBoundaryListener callback interface.

```
1. public void addGpiTriggerBoundaryListener(GpiTriggerBoundaryListener listener)
```

Parameter	Description
listener	Boundary trigger event listener

### 6.1.5 StartReading Method

Start the asynchronous inventory

```
1. public Reader_ERR StartReading(int[] ants, int antcnt, BackReadOption pBRO)
```

Parameter	Description
ants	Store operating antennas in this array
antcnt	The number of antennas in the ants array
pBRO	The detailed configuration of asynchronous inventory, please see BackReadOption class.

**Example:** Use antenna 1, 3 and 4 and start asynchronous inventory with common asynchronous inventory mode

```
1. int[] ants = new int[]{1, 3, 4};
2. BackReadOption Brop = new BackReadOption();
3. Brop.ReadDuration = 200;
4. Brop.ReadInterval = 10;
5. Brop.IsFastRead = 0;
6. Brop.IsGPITrigger = 0;
7. Reader.READER_ERR err = rdr.StartReading(ants, 3, Brop);
```

Use antenna 1, 3 and 4 and start asynchronous inventory with high-speed asynchronous inventory mode

```
1. int[] ants = new int[]{1, 3, 4};
2. BackReadOption Brop = new BackReadOption();
3. Brop.IsFastRead = 1;
4. Brop.IsGPITrigger = 0;
5. Brop.TMFlags.IsAntennaID = true;
6. Brop.TMFlags.IsReadCnt = true;
7. Brop.TMFlags.IsRSSI = true;
8. Brop.TMFlags.IsFrequency = true;
9. Brop.TMFlags.IsRFU = false;
10. Brop.TMFlags.IsTimestamp = false;
11. Brop.TMFlags.IsEmdData = false;
12. Reader.READER_ERR err = rdr.StartReading(ants, 3, Brop);
```

## 6.2 Stop Inventory

Call the Reader.StopReading method to stop the asynchronous inventory.

### 6.2.1 StopReading Method

Stop the asynchronous inventory

```
1. READER_ERR StopReading()
```

**Example:**

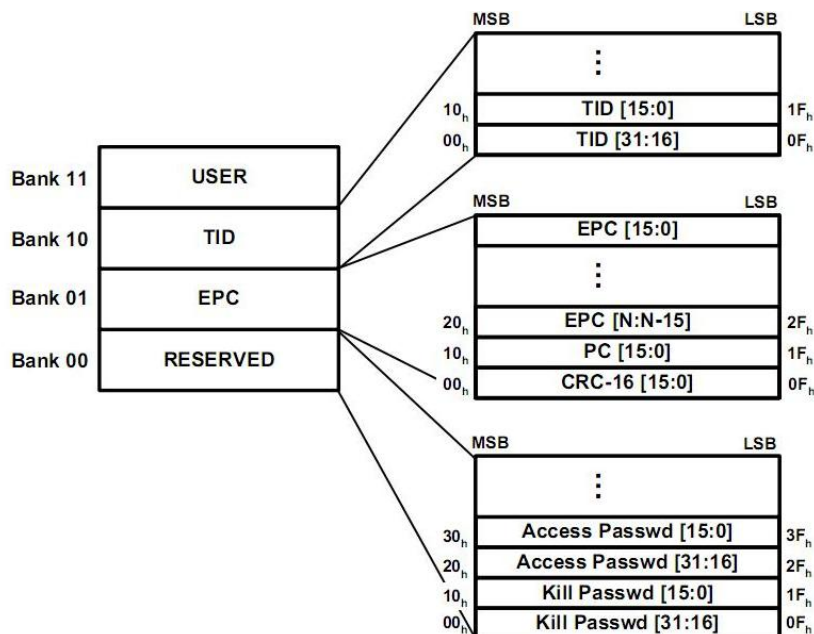
```
1. Reader.READER_ERR err = rdr.StopReading();
```

## 7 Tag Access

Tag access operations include operations such as reading, writing, writing EPC, locking and killing operations. Tag access operations are applied to a single tag. If there are multiple tags in the antenna field, it is undetermined which tag the operation ultimately acts on. In this case, if you need to accurately act on a tag, you do it by setting the `Reader.Mtr_Param.MTR_PARAM_TAG_FILTER` parameter. The tag access operations can only be performed on one antenna at a time. Each tag operation method can specify a timeout, the maximum duration the method will block.

Method	Description
<code>Reader.GetTagData</code>	Read data block of the storage area of the tag
<code>Reader.WriteTagData</code>	Write data block to storage area of the tag
<code>Reader.WriteTagEpcEx</code>	Write the EPC code of a tag
<code>Reader.LockTag</code>	Lock the storage areas of a tag
<code>Reader.KillTag</code>	Kill a tag

A Gen2 tag is divided into four banks which are bank 0, bank 1, bank 2 and bank 3. Bank 0 is also called the reserved bank which contains the access password and the kill password where each password has 32 bits. Bank 1 is called EPC bank, which contains a CRC field (16 bits), PC field (Protocol Control, 16 bits) and EPC field (Electronic Product Code, maximum length is 496 bits and the common length is 96 bits). Bank 2 also known as TID bank, which contains tag- and vendor-specific data (for example, a tag serial number). Bank 3 is called user bank which allows user-specific data storage. Lots of tags do not have bank 3.



### 7.1 GetTagData Method

Read memory area of a tag.

```
1. public READER_ERR GetTagData(int ant, char bank, int address, int blkcnt,
```

```
2. byte[] data, byte[] accesspasswd, short timeout)
```

Parameter	Description
ant	The operating antenna
bank	The memory bank to read, value range 0 to 4; 0 – 3 for Gen2 tag, 4 for ISO18000-6b tag
address	Starting address in bank, in blocks
blkcnt	The number of blocks to read
data	Output parameter, to store the read data
accesspasswd	If needed set access password (4 bytes), else the parameter is null
timeout	The timeout period for the operation

**Example:** Read two blocks of data from the second block of the bank 0 (i.e. access password) with antenna 1 and access password 0x12345678.

```
1. byte[] data = new byte[4];
2. byte[] pwd = new byte[4];
3. pwd[0] = 0x12;
4. pwd[1] = 0x34;
5. pwd[2] = 0x56;
6. pwd[3] = 0x78;
7. Reader.READER_ERR err = GetTagData(1, 0, 2, 2, data, pwd, 1000);
```

## 7.2 WriteTagData Method

Write data into the memory area of a tag.

```
1. public Reader_ERR WriteTagData(int ant, char bank, int address,
2. byte[] data, int datalen, byte[] accesspasswd, short timeout)
```

Parameter	Description
ant	The operating antenna
bank	The memory bank to write, value range 0 to 4; 0 – 3 for Gen2 tag, 4 for ISO18000-6b tag
address	Starting address in bank, in blocks
data	The data to write
datalen	The length of the data in bytes (must be a multiple of 2)
accesspasswd	If needed set access password (4 bytes), else the parameter is null
timeout	The timeout period for the operation

**Example:** Write data 0x111122223333 to memory bank 3 starting at the second block, no access password needed.

```
1. byte[] data = new byte[6];
2. data[0] = 0x11;
3. data[1] = 0x11;
4. data[2] = 0x22;
5. data[3] = 0x22;
6. data[4] = 0x33;
7. data[5] = 0x33;
8. Reader.READER_ERR err = rdr.WriteTagData(1, 3, 2, data, 6, null, 1000);
```

## 7.3 WriteTagEpcEx Method

Write the EPC code of a tag.

```
1. public Reader_ERR WriteTagEpcEx(int ant, byte[] Epc, int epclen, byte[] accesspasswd, short timeout)
```

Parameter	Description
ant	The operating antenna
Epc	Epc data to write
epclen	The length of the EPC data in bytes (must be a multiple of 2)
accesspasswd	If needed set access password (4 bytes), else the parameter is null
timeout	The timeout period for the operation

**Example:** Write EPC code 0x111122223333111122223333 to the EPC memory bank with access password 0x12345678.

```

1. byte[] epccdata = new byte[12];
2. byte[] pwd = new byte[4];
3. pwd[0] = 0x12;
4. pwd[1] = 0x34;
5. pwd[2] = 0x56;
6. pwd[3] = 0x78;
7. epccdata[0] = 0x11;
8. epccdata[1] = 0x11;
9. epccdata[2] = 0x22;
10. epccdata[3] = 0x22;
11. epccdata[4] = 0x33;
12. epccdata[5] = 0x33;
13. epccdata[6] = 0x11;
14. epccdata[7] = 0x11;
15. epccdata[8] = 0x22;
16. epccdata[9] = 0x22;
17. epccdata[10] = 0x33;
18. epccdata[11] = 0x33;
19. Reader.READER_ERR err = rdr.WriteTagEpcEx(1, epccdata, 12, pwd, 1000);

```

The EPC code can also be written through the WriteTagData method. The difference with WriteTagEpcEx is that it also changes the EPC length field of the PC field while writing the EPC code.

## 7.4 LockTag Method

Lock the tag's storage areas. The following objects can be locked: kill password, access password, EPC bank, TID bank, USER bank. The accesspasswd parameter cannot be null. This method could lock multiple objects of one tag at the same time. The lock type can be unlock, temporarily lock or permanently lock. This method only supports gen2 tags.

```

1. public Reader_ERR LockTag(int ant, byte lockobjects, short locktypes, byte[] accesspasswd, short timeout)

```

Parameter	Description
ant	The operating antenna
lockobjects	The objects to lock. In case of multiple Reader.Lock_Obj values, use OR operator
locktypes	The lock type. In case of multiple Reader.Lock_Type values, use OR operator
accesspasswd	Access password (4 bytes)
timeout	The timeout period for the operation

**Example:** Temporarily lock bank 1 and bank 2 and unlock the access password. The access password is 0x12345678.

```

1. String pwd= "12345678";
2. byte[] pwdb=new byte[4];
3. rdr.Str2Hex(pwd, pwd.length(), pwdb);
4. Reader.READER_ERR err = rdr.LockTag(1, (byte)(Reader.Lock_Obj.LOCK_OBJECT_ACCESS_PASSWD.value() |
5.         (byte)Reader.Lock_Obj.LOCK_OBJECT_BANK1.value() |
6.         (byte)Reader.Lock_Obj.LOCK_OBJECT_BANK3.value()),

```

```

7.         (short)Reader.Lock_Type.KILL_PASSWORD__UNLOCK.value() |
8.         (short)Reader.Lock_Type.BANK1_LOCK.value() |
9.         (short)Reader.Lock_Type.BANK3_LOCK.value(), pwdb, 1000);

```

There must be a corresponding `Reader.Lock_Type` enumeration value for each `Reader.Lock_Obj`.

## 7.5 KillTag Method

Kill a tag.

```
1. public Reader_ERR KillTag(int ant, byte[] killpasswd, short timeout)
```

Parameter	Description
ant	The operating antenna
killpasswd	Kill password (4 bytes)
timeout	The timeout period for the operation

**Example:** Destroy a tag, its kill password is 0x43215678.

```

1. byte[] kpwd = new byte[4];
2. kpwd[0] = 0x43;
3. kpwd[1] = 0x21;
4. kpwd[2] = 0x56;
5. kpwd[3] = 0x78;
6. Reader.READER_ERR err = rdr.KillTag(1, kpwd, 1000);

```

## 8 Peripheral Functions

Peripheral functions currently only support getting GPI status and setting GPO status.

There are two methods to get the GPI state, one is to get only one GPI pin's state at a time, the other is to get all the reader's GPI pins states at a time.

### 8.1 GetGPI Method

Get the state of one GPI pin.

```
1. public Reader_ERR GetGPI(int gpuid, int[] val)
```

Parameter	Description
gpuid	GPI ID, one-based numbering
val	Output parameter containing the state of the GPI pin

**Example:** Get the state of GPI pin 1

```

1. int[] state = new int[1];
2. Reader.READER_ERR err = GetGPI(1, state);

```

### 8.2 GetGPIEx Method

Get the states of all the GPI pins of the reader.

```
1. public Reader_ERR GetGPIEx(GpiInfo_ST gpist)
```

Parameter	Description
gpist	Output parameter containing the states of all GPI pins

**Example:**

```
1. GpiInfo_ST ginfo = new GpiInfo_ST();
```

```
2. Reader.READER_ERR err = rdr.GetGPIEx(ginfo);
```

### 8.3 SetGPO Method

Set the state of a GPO pin.

```
1. public Reader_ERR SetGPO(int gpoid, int val)
```

Parameter	Description
gpoid	GPO ID, one-based numbering
val	The state to set, 0 or 1

**Example:** Set the state of GPO pin 1 to 1.

```
1. Reader.READER_ERR err = rdr.SetGPO (1, 1);
```

## 9 String Functions

Convert byte data into hexadecimal string and vice versa.

### 9.1 Hex2Str Function

```
1. public void Hex2Str(byte[] buf, int len, char[] out);
```

Parameter	Description
buf	Byte data
len	Number of bytes in buf
out	Output buffer to store the resulting hex string

**Example:**

```
1. byte[] hexbuf = new byte[]{1, 2, 3, 4};
2. char[] str = new char[2*4+1];
3. Hex2Str(hexbuf, 4, str);
4. // Content of str = "01020304";
```

### 9.2 Str2Hex Function

```
1. public void Str2Hex(String buf, int len, byte[] hexbuf);
```

Parameter	Description
buf	Hexadecimal string
len	Length of the string in buf, must be a multiple of 2
hexbuf	Output parameter to store the binary byte data

**Example:**

```
1. String str = "12345678abcd";
2. byte[] out = new byte[100];
3. Str2Hex(str, str.length(), out);
```

## 10 Error Handling

All the API methods with a return value will return `Reader.READER_ERR.MT_OK_ERR` on success. If the return value is `Reader.READER_ERR.MT_IO_ERR` it means, there is something wrong with the network connection or serial port connection. You should call `Reader.CloseReader` and check these connections, then try to call

`Reader.InitReader_Notype`. As for `Reader.READER_ERR.MT_CMD_FAILED_ERR`, it just means the failure of the method, it is not a fatal error, and you can do any operation next. As for `Reader.READER_ERR.MT_CMD_NO_TAG_ERR`, it means no tag was found.

The errors starting with `Reader.READER_ERR.MT_HARDWARE_ALERT_ERR_BY` are serious errors which cannot be ignored. There are some improper operations which can cause these exception and hardware damage. These operations include:

- Transmit power through antenna ports without antenna connection,
- use of unqualified antennas,
- high ambient temperature and
- high return loss caused by metal plate in front of the antennas.

It would be best to turn off the reader and check the working condition and working environment for these errors.

There is another way to do error troubleshooting. You can call the `Reader.GetLastError` method immediately after an error occurs to get a more detailed error code and an error message represented by a string.

All functions of the current version of the SDK (except for `InitReader_Notype`) are not thread-safe for the same reader instance handle. Users should ensure that there is no race condition.

## 10.1 GetLastError Method

Get the more detailed error information. After `InitReader_Notype` returns failure, this method should not be called as the handle of the reader is invalid.

```
1. public Reader_ERR GetLastError(ErrorInfo ei)
```

Parameter	Description
ei	The error information including an internal error code and an error string

**Example:**

```
1. ErrInfo eif = new ErrInfo();
2. Reader.READER_ERR err = rdr.GetLastError(eif);
```

## 10.2 Error Codes

The following is a table of error codes returned by calling the `GetLastError` function.

Status Code (Hex)	Status Code (Dec)	Description
0x0000	0	Success.
0x0100	256	Sending data length error. The reader firmware version may not match the API version.
0x0101	257	Command is not available. Make sure the reader supports the command and the power supply of the reader, as it may cause soft resets if the power supply is unstable.
0x0105	261	Parameter value is not available.
0x010A	266	Baud rate is not available.
0x010B	267	The current operating frequency band is unavailable.
0x0200	512	The CRC of the firmware layer is incorrect. Try upgrading the reader firmware again.
0x0302	770	Flash undefined error, flash write failed.
0x0400	1024	Tag not found, usually due to insufficient power or filter mismatch.
0x0402	1026	Protocol is not available. Check the reader firmware and if the API supports the protocol.
0x0404	1028	When the read after write function is enabled, the write succeeded but the read failed.
0x040A	1034	General tag error (read/write lock, kill command), can be re-executed.
0x040B	1035	The read memory area exceeds the limit (only 96 blocks can be read at a time).

0x040C	1036	Unusable kill password, usually the kill password is not written.
0x0420	1056	Gen2 protocol error.
0x0423	1059	The storage area is out of bounds.
0x0424	1060	The storage area being operated on is locked.
0x042B	1067	Insufficient energy.
0x042F	1071	Non-specific error.
0x0430	1072	Unknown error.
0x0500	1280	Frequency value not available.
0x0504	1284	Temperature is too high.
0x0505	1285	Excessive return loss. Check for metal in the near environment.
0x500F	20495	The detected temperature exceeds the measurement range.
0x50FF	20735	Unstable temperature detected.
0x7F00	32512	The RF hardware circuit failed to start.
0xAA02	43522	Failed to write OEM register.
0xAA03	43523	Failed to read OEM register.
0xAA04	43524	Command execution failed.
0xAA2A	43562	OEM format failed.
0xAA31	43569	Carrier start or stop failed (no antenna connected).
0xAA40	43584	Save configuration command failed to write the OEM register.
0xAA4A	43594	Standing wave detection failed.
0xAA4B	43595	Reset or power on/off operation failed.
0xAA4C	43596	No valid setting for multi-tag matching filter data.
0xAA4D	43597	Setting SESSION2/SESSION3 target to A failed.
0xAA55	43605	User-defined storage area command, read and write execution failed.
0xAA56	43606	User-defined command configuration or read execution failed.
0xEE01	60929	The chip did not reach the application layer.
0xEE02	60930	Chip firmware startup upgrade failed.
0xEE03	60931	Chip firmware upgrade continues to fail.
0xEE04	60932	Chip firmware upgrade failed.
0xFF11	65297	Flash initialization failed.
0xFF12	65298	GPIO initialization failed.
0xFF13	65299	Timer initialization failed.
0xFF14	65300	SPI initialization failed.
0xFF15	65301	Antenna control initialization failed.
0xFF16	65302	Frequency band initialization failed.
0xFF17	65303	EX10 initialization failed.
0xFF1F	65311	Firmware abnormality.
0xFFFF	65535	Hardware version number error.
Non 0	Non 0	Command execution failed.