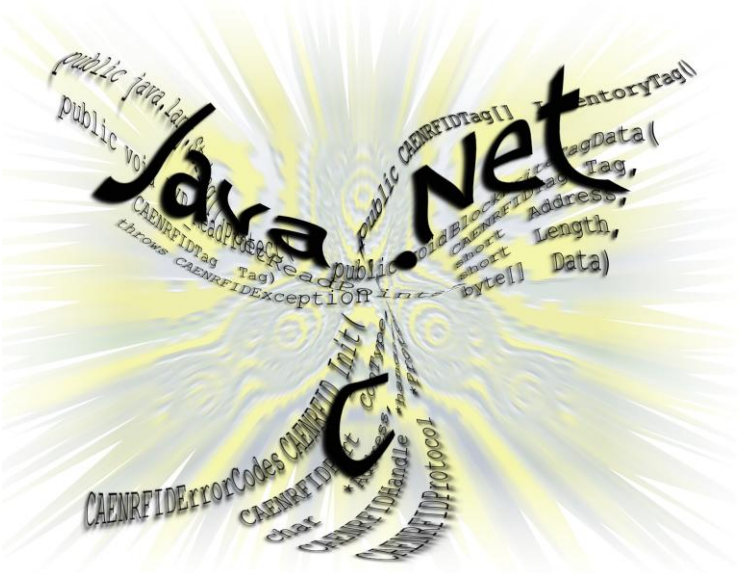
CAEN RFID API

Reference Manual



CAENRFID

Reference Manual

Revision n. 03

22/03/2011

Scope of Manual

This manual documents the API used by C, Java and .Net programmers who want to write applications for controlling and using CAEN RFID readers.

Change Document Record

Date	Revision	Changes
29 Jun 2010	01	Initial release.
14 Jan 2011	02	Added BRAZIL radiofrequency regulation in the CAENRFIDRFRegulations Enumeration.
		Corrected GetTimeStamp Method's return value.
		Added Federal Communications Commission (FCC) Notice
22 Mar 2011	03	Added R1260U Reader in the declaration of Federal Communications Commission (FCC) note.

CAEN RFID srl

Via Vetraia, 11 55049 Viareggio (LU) - ITALY
Tel. +39.0584.388.398 Fax +39.0584.388.959
info@caenrfid.it
www.caenrfid.it

© CAEN RFID srl – 2010

Disclaimer

No part of this manual may be reproduced in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of CAEN RFID.

The information contained herein has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies. CAEN RFID reserves the right to modify its products specifications without giving any notice; for up to date information please visit www.caenrfid.it.

Federal Communications Commission (FCC) Notice (Preliminary) [1]

This device was tested and found to comply with the limits set forth in Part 15 of the FCC Rules. Operation is subject to the following conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received including interference that may cause undesired operation. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment.

This device generates, uses, and can radiate radio frequency energy. If not installed and used in accordance with the instruction manual, the product may cause harmful interference to radio communications. Operation of this product in a residential area is likely to cause harmful interference, in which case, the user is required to correct the interference at their own expense. The authority to operate this product is conditioned by the requirements that no modifications be made to the equipment unless the changes or modifications are expressly approved by CAEN RFID.

[1] This declaration only applies to FCC readers A828US, A829US, A528, R1230CB, R1260I, R1260U.

Index

Scope of Manual 2
Change Document Record 2
Index 3
List of Tables 5
1 Introduction 6
Overview on SDK 7
Functions and methods names 7
Error Handling 7
Managing connections with the readers 7
Return data mechanism 8
Passing parameters to methods and functions 8
2 CAEN RFID API Structure 9
CAENRFID Classes 10
CAENRFID Enumerations 14
3 Classes Description 15
CAENRFIDException Class 16
getError Method 16
CAENRFIDLogicalSource Class 17
AddReadPoint Method 17
BlockWriteTagData Method 18
CustomCommand_EPC_C1G2 Method 20
EventInventoryTag Method 21
Fujitsu_BurstErase Method 22
Fujitsu_BurstWrite Method 24
Fujitsu_ChgBlockGroupPassword Method 26
Fujitsu_ChgBlockLock Method 28
Fujitsu_ChgWordLock Method 30
Fujitsu_ReadBlockLock Method 32
Fujitsu_Refresh Method 34
GetBufferedData Method 35
GetDESB_ISO180006B Method 35
GetName Method 36
GetQ_EPC_C1G2 Method 36
GetReadCycle Method 37
GetSelected_EPC_C1G2 Method 37
GetSession_EPC_C1G2 Method 38
GetTarget_EPC_C1G2 Method 38
GroupSelUnsel Method 39
Hitachi_BlockLock Method 40
Hitachi_BlockReadLock Method 42
Hitachi_GetSystemInformation Method 44
Hitachi_ReadLock Method 45
Hitachi_SetAttenuate Method 46
Hitachi_WriteMultipleWords Method 48
InventoryTag Method 50
isReadPointPresent Method 57
KillTag_EPC_C1G1 Method 57
KillTag_EPC_C1G2 Method 58
LockBlockPermaLock_EPC_C1G2 Method 60
LockTag_EPC_C1G2 Method 61
LockTag_ISO180006B Method 64
NXP_ChangeEAS Method 64
NXP_EAS_Alarm Method 65
NXP_ReadProtect Method 65
NXP_ResetReadProtect Method 66
ProgramID_EPC_C1G1 Method 67
ProgramID_EPC_C1G2 Method 67
ProgramID_EPC119 Method 68
Query_EPC_C1G2 Method 69

QueryAck_EPC_C1G2 Method 69
ReadBLockPermalock_EPC_C1G2 Method 70
ReadTagData Method 71
ReadTagData_EPC_C1G2 Method 72
RemoveReadPoint Method 76
ResetSession_EPC_C1G2 Method 76
SetDESB_ISO180006B Method 77
SetQ_EPC_C1G2 Method 77
SetReadCycle Method 78
SetSelected_EPC_C1G2 Method 78
SetSession_EPC_C1G2 Method 79
SetTarget_EPC_C1G2 Method 79
WriteTagData Method 80
WriteTagData_EPC_C1G2 Method 81
CAENRFIDNotify Class 85
getDate Method 85
getReadPoint Method 85
getRSSI Method 86
getStatus Method 86
getTagID Method 86
getTagLength Method 86
getTagSource Method 87
getTagType Method 87
getTID Method 87
CAENRFIDReader Class 88
Connect Method 88
Init Function 88
Disconnect Method 89
End 89
GetBitRate Method 89
GetFirmwareRelease Method 90
GetIO Method 90
GetIODirection Method 91
GetLBTMode Method 91
GetPower Method 92
GetProtocol Method 92
GetReaderInfo Method 93
GetReadPoints Method 93
GetReadPointStatus Method 94
GetRFChannel Method 94
GetRFRegulation Method 95
GetSource Method 95
GetSourceNames Method 96
GetSources Method 96
InventoryAbort Method 96
RFControl Method 97
SetBitRate Method 97
SetDateTime Method 98
SetIO Method 98
SetIODIRECTION Method 99
SetLBTMode Method 99
SetNetwork Method 100
SetPower Method 100
SetProtocol Method 101
SetRFChannel Method 101
SetRFRegulation Method 102
SetRS232 Method 103
CAENRFIDReaderInfo Class 104
GetModel Method 104
GetSerialNumber Method 104
CAENRFIDTag Class 105
GetId Method 105
GetLength Method 105
GetReadPoint Method 106

GetRSSI Method 106
GetSource Method 106
GetTID Method 107
GetTimeStamp Method 107
GetType Method 107
4 Event Handling 108
Event Handling 109
EventInventoryTag Method 110
InventoryAbort Method 111
C# Event Handling 112
CAENRFIDEventArgs Class 112
CAENRFIDEventHandler Delegate 112
CAENRFIDEvent Event 112
JAVA Event Handling 113
CAENRFIDEvent Class 113
CAENRFIDEventListener Interface 113
addCAENRFIDEventListener 113
removeCAENRFIDEventListener 113
C Event Handling 114
CAENRFID_INVENTORY_CALLBACK 114
5 Enumerations Description 115
CAENRFIDBitRate Enumeration 116
CAENRFIDLogicalSourceConstants Enumeration 117
CAENRFIDPort Enumeration 118
CAENRFIDProtocol Enumeration 118
CAENRFIDReadPointStatus Enumeration 119
CAENRFIDRFRegulations Enumeration 119
CAENRFIDRS232Constants Enumeration 120
CAENRFIDSelUnselOptions Enumeration 120
6 CAENRFID Obsolete Methods 121
C# Obsolete Methods 122
C# Obsolete Members 123
JAVA Obsolete Methods 123
C Obsolete Functions 125
C Obsolete Data Types 126

List of Tables

Tab. 2.1: CAENRFID classes 10
Tab. 2.2: CAENRFID methods 13
Tab. 2.3: CAENRFID Enumerations 14
Tab. 6.1: C# Obsolete Methods 122
Tab. 6.2: C# Obsolete Members 123
Tab. 6.3: JAVA Obsolete Methods 124
Tab. 6.4: C Obsolete Functions 126
Tab. 6.5: C Obsolete Data Types 126

1 Introduction

This Chapter gives basic information about CAENRFID Software Development Kit (SDK). It contains these topics:

- Overview on SDK
- Functions and methods names
- Error Handling
- Managing connections with the readers
- Return data mechanism
- Passing parameters to methods and functions

Overview on SDK

CAEN RFID provides a Software Development Kit (SDK) aimed to facilitate the software developers in interfacing with its readers. The SDK provides Application Program Interfaces (API) for three programming languages: C, Java and J#/C#/Visual Basic .NET.

The functionalities and the behaviors exported by the libraries are exactly the same for all the languages but, due to the syntax differences between them, there are differences in the implementation of functions and methods. Java and .NET implementation are very similar because they are both Object Oriented environments while the C implementation differs more.

The Object Oriented implementation (Java and .NET) defines a set of classes that models the devices characteristics, the main one are the CAENRFIDReader class and the CAENRFIDLogicalSource class. The first one implements the main methods used to configure general readers' parameters like the output power, the link interface and so on, the latter provides the methods to be used in order to communicate with the RFID tags (tags detection, read and write commands and so on).

The C implementation, on the contrary, implements a set of data types (defined into the CAENRFIDTypes.h header file) and a list of functions (defined into the CAENRFIDLib.h header file) in order to obtain the same functionalities as the Java and .NET classes.

In the Object Oriented languages (C# and Java) there are some methods that return objects, these methods have no correspondent in C language.

Further details on .NET and Java APIs can be found into the CAEN RFID API User Manual.

The following paragraphs will denote the differences in functionality for the topics listed below:

- Functions and methods names
- Error Handling
- Managing connections with the readers
- Return data mechanism
- Passing parameters to methods and functions

Functions and methods names

The functions and methods with the same functionalities have the same name in all languages. The only exceptions are due to the absence of the overloading feature in the C language: methods that are overloaded in Java and .NET are translated in a corresponding set of different functions in C.

Note: some methods and functions have changed name in the last revision of the API but older names are still functional to preserve backward compatibility (see § CAENRFID Obsolete Methods pag.121).

Error Handling

Java and .NET language API handle error conditions using the exceptions mechanism: when a method encounters an error, an exception is thrown to the calling code. The API defines a proper class for the exception generated by its methods (CAENRFIDException) the origin of the error is represented inside the CAENRFIDException object as a string.

C language does not provide the exception mechanism so the errors are handled using the return value of the functions. Each C function returns a numeric error code that can be interpreted using the CAENRFIDErrorCodes enumeration. Since no exceptions are generated, the execution flow of the program is not interrupted by the errors so it is always suggested to check for error conditions in the code before to call other functions.

Managing connections with the readers

Java and .NET languages allow to initiate and terminate the communication with the reader by means of two specific methods of the CAENRFIDReader objects. So, after an object of the class CAENRFIDReader is instantiated, the Connect method permits to start the communication with a reader while the Disconnect method permits to terminate the communication.

C language is not object oriented and the handling of the communication state is implemented using two functions. CAENRFID_Init is used to start the communication with a reader and to initialize all the library's internal data structures

needed in order to maintain the communication active. The function returns an "handle" (very similar to the handles used in managing files) that have to be used in any subsequent function calls relative to that reader. At the end of the operation, a call to the CAENRFID_End function permits to close the communication link and to free the internal data structures.

Return data mechanism

As seen in the Error Handling paragraph, all the C functions return a numeric error codes. Due to that reason, functions that need to return data to the caller use output parameters. Output parameters for the C functions are highlighted in this reference manual by the underlined name in the formal parameter list.

Java and .NET languages use exception for the error handling so, typically, the data is returned to the caller using the return value of the methods.

Passing parameters to methods and functions

There are differences in the parameters' lists between Java/.NET methods and C functions. Many of those differences are due to the implicit reference of the methods to their objects. This characteristic of object oriented languages is emulated in C functions using an additional explicit parameter. Methods belonging to CAENRFIDLogicalSource objects, for example, are emulated in C functions that accept SourceName parameters.

Other differences are due to the better handling of complex data types in Java and .NET languages. Arrays, for example, have implicit size in Java/.NET that permit to pass a single parameter to methods requiring this data type. In C functions, passing an array as a parameter, need to specify both the memory address of the array and its size explicitly.

2 CAEN RFID API Structure

This chapter describes CAEN RFID API Structure. It contains these topics:

- CAENRFID Classes
- CAENRFID Enumerations

CAENRFID Classes

In .NET (henceforth C#) and Java languages, CAENRFID methods are divided into the following classes:

Class	Description
CAENRFIDEventArgs2	This class defines the CAENRFID event arguments.
CAENRFIDException	This class defines the CAEN RFID exceptions.
CAENRFIDLogicalSource	The CAENRFIDLogicalSource class is used to create logical source objects. Logical source objects represent an aggregation of read points (antennas). Operations on the tags are performed using the logical source methods. In addition to the methods used to operate on the tags, the logical source class exports methods to configure the anticollision algorithm and to configure the composition of the logical source itself.
CAENRFIDNotify	This class defines the structure of a notification message.
CAENRFIDReader	The CAENRFIDReader class is used to create reader objects which permit to access to CAEN RFID readers' configuration and control commands.
CAENRFIDReaderInfo	The CAENRFIDReaderInfo class is used to create reader info objects. Reader info objects represent the information about the reader device (model and serial number).
CAENRFIDTag	This class is used to define objects representing the tags. These objects are used as return value for the inventory methods and as arguments for many tag access methods.

Tab. 2.1: CAENRFID classes

Each class contains the following methods:

Methods	Description
CAENRFIDEventArgs Class	
getData	Returns the event object value.
CAENRFIDException Class	
getError	Gets the error string associated to the exception.
CAENRFIDLogicalSource Class	
AddReadPoint	Adds a read point to the logical source.
BlockWriteTagData	Overloaded. This method can be used to write a portion of the user memory in a ISO18000-6B tag using blocks of four bytes for each command.
CustomCommand_EPC_C1G2	Overloaded. This method can be used to issue a generic Custom command as defined by the EPC Class1 Gen2 protocol specification. The parameters are used to specify the type of the custom command and its parameters.
EventInventoryTag	A call to this method will start a sequence of read cycle on each read point linked to the logical source. The readings will be notified to the controller via event generation.
Fujitsu_BurstErase	Overloaded. This method can be used to issue a BurstErase custom command as defined by the Fujitsu datasheet.
Fujitsu_BurstWrite	Overloaded. This method can be used to issue a BurstWrite custom command as defined by the Fujitsu datasheet.
Fujitsu_ChgBlockGroupPassword	Overloaded. This method can be used to issue a ChgBlockGroupPassword custom command as defined by the Fujitsu datasheet.
Fujitsu_ChgBlockLock	Overloaded. This method can be used to issue a ChgBlockLock custom command as defined by the Fujitsu datasheet.
Fujitsu_ChgWordLock	Overloaded. This method can be used to issue a ChgWordLock custom command as defined by the Fujitsu datasheet.
Fujitsu_ReadBlockLock	Overloaded. This method can be used to issue a ReadBlockLock custom command as defined by the Fujitsu datasheet.
Fujitsu_Refresh	Overloaded. This method can be used to issue a Refresh custom command as defined by the Fujitsu datasheet.

[2] For the description of this class, see § Event Handling pag.107

Methods	Description
GetBufferedData	The function returns all the Tags stored in reader's memory using all the ReadPoints belonging to the Source.
GetDESB_ISO180006B	This method can be used to retrieve the Data Exchange Status Bit setting (see ISO18000-6B protocol specification) used by the anticollision algorithm when called on this logical source.
GetName	Gets a string representing the name of the logical source.
GetQ_EPC_C1G2	This method can be used to retrieve the current setting for the initial Q value (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
GetReadCycle	Gets the current setting for the number of read cycles performed by the logical source during the inventory algorithm execution.
GetSelected_EPC_C1G2	This method can be used to retrieve the Selected flag (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
GetSession_EPC_C1G2	This method can be used to retrieve the Session setting (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
GetTarget_EPC_C1G2	This method can be used to retrieve the Target setting (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
GroupSelUnsel	This method can be used to send a Group Select/Unselect command to the tag (see ISO18000-6B protocol specification).
Hitachi_BlockLock	Overloaded. This method can be used to issue a BlockLock custom command as defined by the Hitachi Hibiki datasheet.
Hitachi_BlockReadLock	Overloaded. This method can be used to issue a BlockReadLock custom command as defined by the Hitachi Hibiki datasheet.
Hitachi_GetSystemInformation	Overloaded. This method can be used to issue a GetSystemInformation custom command as defined by the Hitachi Hibiki datasheet.
Hitachi_ReadLock	Overloaded. This method can be used to issue a ReadLock custom command as defined by the Hitachi Hibiki datasheet.
Hitachi_SetAttenuate	Overloaded. This method can be used to issue a SetAttenuate custom command as defined by the Hitachi Hibiki datasheet.
Hitachi_WriteMultipleWords	Overloaded. This method can be used to issue a WriteMultipleWords custom command as defined by the Hitachi Hibiki datasheet.
InventoryTag	Overloaded. A call to this method will execute a read cycle on each read point linked to the logical source. Depending on the air protocol setting it will execute the appropriate anticollision algorithm.
isReadPointPresent	Checks if a read point is present in the logical source.
KillTag_EPC_C1G1	This method can be used to kill an EPC Class 1 Gen 1 tag.
KillTag_EPC_C1G2	Overloaded. This method can be used to kill an EPC of an EPC Class 1 Gen 2 tag.
LockBlockPermaLock_EPC_C1G2	This method implements the BLockPermaLock with ReadLock=1 as specified in EPCC1G2 rev. 1.2.0 protocol.
LockTag_EPC_C1G2	Overloaded. This method can be used to lock a memory bank of an EPC Class 1 Gen 2 tag.
LockTag_ISO180006B	This method can be used to lock a byte in the memory of a ISO18000-6B tag.
NXP_ChangeEAS	This method can be used to issue a ChangeEAS custom command as defined by the NXP G2XM and G2XL datasheet after having put it in Secured state using the Access command.
NXP_EAS_Alarm	This method can be used to issue an EAS_Alarm custom command as defined by the NXP G2XM and G2XL datasheet.
NXP_ReadProtect	Overloaded. This method can be used to issue a ReadProtect custom command as defined by the NXP G2XM and G2XL datasheet.
NXP_ResetReadProtect	This method can be used to issue a ResetReadProtect custom command as defined by the NXP G2XM and G2XL datasheet.
ProgramID_EPC_C1G1	This method can be used to write the EPC of an EPC Class 1 Gen 1 tag.
ProgramID_EPC_C1G2	Overloaded. This method can be used to write the EPC of an EPC Class 1 Gen 2 tag.
ProgramID_EPC119	This method can be used to write the UID of an EPC 1.19 tag.
Query_EPC_C1G2	This method make the reader generate an EPC Class1 Gen2 Query command.

Methods	Description
QueryAck_EPC_C1G2	This method make the reader generate a sequence of EPC Class1 Gen2 Query and Ack commands. It can be used to read a single tag under the field. If there are more than one tag under the field the method fails.
ReadBLockPermalock_EPC_C1G2	This method implements the BLockPermaLock with ReadLock=0 as specified in EPCC1G2 rev. 1.2.0 protocol.
ReadTagData	This method can be used to read a portion of the user memory in a ISO18000-6B tag.
ReadTagData_EPC_C1G2	Overloaded. This method can be used to read a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag.
RemoveReadPoint	Removes a read point from the logical source.
ResetSession_EPC_C1G2	This method can be used to reset the Session status for EPC Class1 Gen2 tags. After the execution of this method all the tags in the field of the antennas belonging to this logical source are back in the default Session.
SetDESB_ISO180006B	This method can be used to set the Data Exchange Status Bit (see ISO18000-6B protocol specification) used by the anticollision algorithm when called on this logical source.
SetQ_EPC_C1G2	This method can be used to set the initial Q value (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
SetReadCycle	Sets the number of read cycles to be performed by the logical source during the inventory algorithm execution.
SetSelected_EPC_C1G2	This method can be used to set the Session (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
SetSession_EPC_C1G2	This method can be used to set the Session (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
SetTarget_EPC_C1G2	This method can be used to set the Target setting (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.
WriteTagData	This method can be used to write a portion of the user memory in a ISO18000-6B tag.
WriteTagData_EPC_C1G2	Overloaded. This method can be used to write a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag.
CAENRFIDNotify Class	
getDate	Returns a timestamp representing the time at which the event was generated.
getReadPoint	Returns the read point that has detected the tag.
getRSSI	Returns the RSSI value measured for the tag.
getStatus	Returns the event type associated to the tag.
getTagID	Returns the tag's ID (the EPC code in Gen2 tags).
getTagLength	Returns the tag's ID length.
getTagSource	Returns the name of the logical source that has detected the tag.
getTagType	Returns the air protocol of the tag.
getTID	Returns the TID field value in a EPC Class 1 Gen 2 Tag
CAENRFIDReader Class	
Connect	Starts the communication with the reader. It must be called before any other call to method of the CAENRFIDReader object.
Disconnect	Closes the connection with the CAEN RFID Reader releasing all the allocated resources.
GetBitRate	Gets the current setting of the RF bit rate.
GetFirmwareRelease	Permits to read the release of the firmware loaded into the device.
GetIO	Gets the current digital Input and Output lines status.
GetIODirection	Gets the current I/O direction setting as a bitmask. Each bit represents a I/O line, a value of 0 means that the line is configured as an input, 1 as an output. This setting has a meaning only for those readers with configurable I/O lines.
GetLBTMode	Gets the current LBT mode setting. If the current regulation is based on the frequency hopping mechanism it returns the FH status.
GetPower	Gets the current setting of the RF power expressed in mW.
GetProtocol	Gets the current air protocol of the Reader.

Methods	Description
GetReaderInfo	Permits to read the reader information loaded into the device.
GetReadPoints	Gets the names of the read points (antennas) available in the reader.
GetReadPointStatus	Gets the CAENRFIDReadPointStatus object rapresenting the status of a read point (antenna).
GetRFChannel	Gets the index of the RF channel currently in use. The index value meaning change for different country regulations.
GetRFRegulation	Gets the current RF regulation setting value.
GetSource	Gets a CAENRFIDLogicalSource object given its name
GetSourceNames	Gets the names of the logical sources available in the reader.
GetSources	Gets the CAENRFIDLogicalSource objects available on the reader.
InventoryAbort	Stops the EventInventoryTag execution.
RFControl Method	Permits to control the RF CW (Carrier Wave) signal generation.
SetBitRate	Sets the RF bit rate to use.
SetDateTime	Sets the Date/Time of the reader.
SetIO	Sets the Output lines value.
SetIODIRECTION	Sets the current I/O direction setting as a bitmask. Each bit represents a I/O line, a value of 0 means that the line is configured as an input, 1 as an output. This setting has a meaning only for those readers with configurable I/O lines.
SetLBTMode	Sets/Resets the LBT/FH mode.
SetNetwork	Permits to configure the network settings of the reader. In order to apply the changes the reader must be restarted.
SetPower	Sets the conducted RF power of the Reader.
SetProtocol	Set the air protocol of the reader.
SetRFChannel	Sets the RF channel to use. This method fixes the RF channel only when the listen before talk or the frequency hopping feature is disabled.
SetRFRegulation	Sets the RF regulation setting value.
SetRS232	Permits to change the serial port settings. Valid settings values depend on the reader model.
CAENRFIDReaderInfo Class	
GetModel	Gets the reader's model.
GetSerialNumber	Gets the reader's serial number.
CAENRFIDTag Class	
GetId	Returns the tag's ID (the EPC code in Gen2 tags).
GetLength	Returns the tag's ID length.
GetReadPoint	Returns the read point that has detected the tag.
GetRSSI	Returns the RSSI value measured for the tag.
GetSource	Returns the name of the logical source that has detected the tag.
GetTID	Returns the tag's TID (valid only for EPC Class 1 Gen 2 tags).
GetTimeStamp	Gets the Tag's TimeStamp.
GetType	Returns the air protocol of the tag.

Tab. 2.2: CAENRFID methods

CAENRFID Enumerations

The following enumerations are present in C# language They correspond to classes in Java language and to enumerations and data types in C language:

Enumerations	Description
BitRate	Gives a list of the supported radiofrequency profiles.
LogicalSourceConstants	Gives a list of constants used for the configuration of the logical sources. Detailed explanation of the settings can be found in the EPC Class 1 Gen 2 and ISO 18000-6B specification documents.
Port	Gives a list of the communication ports supported by the CAEN RFID readers.
Protocol	Gives a list of the air protocol supported by the CAEN RFID readers.
ReadPointStatus	Gives a list of the possible ReadPoint status values.
RFRegulations	Gives a list of country radiofrequency regulations.
RS232Constants	Gives a list of settings for the serial port configuration.
SelUnselOptions	Gives a list of operations supported by the Group Select/Unselect command (valid only for the ISO18000-6B air protocol).

Tab. 2.3: CAENRFID Enumerations

3 Classes Description

This chapter gives a description of CAENRFID methods divided into classes. It contains these topics:

- CAENRFIDException Class
- CAENRFIDLogicalSource Class
- CAENRFIDNotify Class
- CAENRFIDReader Class
- CAENRFIDReaderInfo Class
- CAENRFIDTag Class

CAENRFIDException Class

The CAENRFIDException class defines the CAEN RFID exceptions.

getError Method

Description:
This method gets the error string associated to the exception.

Return value:
The string representing the error.

Syntax:

C# representation:

```
public string                getError()
```

JAVA representation:

```
public java.lang.String      getError()
```

Remarks:
This function does not exist in C language, see § Error Handling pag. 7 for more informations.

CAENRFIDLogicalSource Class

The CAENRFIDLogicalSource class is used to create logical source objects. Logical source objects represent an aggregation of read points (antennas). Operations on the tags are performed using methods belonging to the logical source. In addition to the methods used to operate on the tags, the logical source class exports methods to configure the anticollision algorithm and to configure the composition of the logical source itself.

AddReadPoint Method

Description:
This method adds a read point to the logical source.

Parameters:

Name	Description
ReadPoint	A string representing the name of the read point (antenna).

Syntax:

C# representation:

```
public void         AddReadPoint(
                              string              ReadPoint)
```

JAVA representation:

```
public void         AddReadPoint(
                              java.lang.String    ReadPoint)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_AddReadPoint(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              char                *ReadPoint);
```

BlockWriteTagData Method

BlockWriteTagData Method (CAENRFIDTag, Int16, Int16, Byte[])

Description:

This method can be used to write a portion of the user memory in a ISO18000-6B tag using blocks of four bytes for each command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	The data to be written into the tag's user memory.

Syntax:

C# representation:

```
public void         BlockWriteTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Length,
                              byte[]              Data)
```

JAVA representation:

```
public void         BlockWriteTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Length,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_BlockWriteTagData(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 Address,
                              int                 Length,
                              void                *Data);
```

BlockWriteTagData Method (CAENRFIDTag, Int16, Int16, Int16, Byte[])

Description:

This method can be used to write a portion of the user memory in a ISO18000-6B tag using blocks of four bytes for each command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
Address	The address where to start writing the data.
Mask	A bitmask that permit to select which of the four bytes have to be written (i.e. mask 0x05 write the bytes on position Address + 1 and Address + 3).
Length	The number of byte to be written.
Data	The data to be written into the tag's user memory.

Syntax:

C# representation:

```
public void         BlockWriteTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Mask,
                              short               Length,
                              byte[]              Data)
```

JAVA representation:

```
public void         BlockWriteTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Mask,
                              short               Length,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_FilterBlockWriteTagData(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *ID,
                              int                 Address,
                              short               Mask,
                              int                 Length,
                              void                *Data);
```

CustomCommand_EPC_C1G2 Method

CustomCommand_EPC_C1G2 Method (CAENRFIDTag, Byte, Int16, Byte[], Int16)

Description:

This method can be used to issue a generic Custom command as defined by the EPC Class1 Gen2 protocol specification. The parameters are used to specify the type of the custom command and its parameters.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to which send the Custom command.
SubCmd	The SubCommand field of the Custom command.
TxLen	The length of the data to be sent to the tag.
Data	The data to be sent to the tag.
RxLen	The length of the data to be received by the tag.

Return value:

An array of bytes representing the reply from the tag as specified by the custom command.

Syntax:

C# representation:

```
public byte[]       CustomCommand_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              byte                SubCmd,
                              short               TxLen,
                              byte[]              Data,
                              short               RxLen)
```

JAVA representation:

```
public byte[]       CustomCommand_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              byte                SubCmd,
                              short               TxLen,
                              byte[]              Data,
                              short               RxLen)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_CustomCommand_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              unsigned char       SubCmd,
                              int                 TxLen,
                              void                *Data,
                              int                 RxLen,
                              void                *TRData);
```

CustomCommand_EPC_C1G2 Method (CAENRFIDTag, Byte, Int16, Byte[], Int16, Int32)

Description:

This method can be used to issue a generic Custom command as defined by the EPC Class1 Gen2 protocol specification. The parameters are used to specify the type of the custom command and its parameters. The Custom command is executed after an Access command to switch the tag in the Secured state using the provided password.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to select.
SubCmd	The SubCommand field of the Custom command.
TxLen	The length of the data to be sent to the tag.
Data	The data to be sent to the tag.
RxLen	The length of the data to be received by the tag.
AccessPassword	The access password.

Return value:

An array of bytes representing the reply from the tag as specified by the custom command.

Syntax:

C# representation:

```
public byte[]       CustomCommand_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              byte                SubCmd,
                              short               TxLen,
                              byte[]              Data,
                              short               RxLen,
                              int                 AccessPassword)
```

JAVArepresentation:

```
public byte[]       CustomCommand_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              byte                SubCmd,
                              short               TxLen,
                              byte[]              Data,
                              short               RxLen,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureCustomCommand_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              unsigned char       SubCmd,
                              int                 TxLen,
                              void                *Data,
                              int                 RxLen,
                              int                 AccessPassword,
                              void                *TRData);
```

EventInventoryTag Method

For the description of this method, see § Event Handling pag.108.

Fujitsu_BurstErase Method

Fujitsu_BurstErase Method (CAENRFIDTag, Byte, Int16, Byte)

Description:

This method can be used to issue a BurstErase custom command as defined by the Fujitsu datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
MemBank	The memory bank where to erase the data.
Address	The address where to erase the data.
Length	The number of byte to be erased.

Return value:

A byte representing the "Write-locked" response of the tag.

Syntax:

C# representation:

```
public byte         Fujitsu_BurstErase(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length)
```

JAVA representation:

```
public byte         Fujitsu_BurstErase(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_BurstErase(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                MemBank,
                              short               Address,
                              char                Length,
                              char                *TRData);
```

Fujitsu_BurstErase Method (CAENRFIDTag, Byte, Int16, Byte, Int32)

Description:

This method can be used to issue a BurstErase custom command as defined by the Fujitsu datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
MemBank	The memory bank where to erase the data.
Address	The address where to erase the data.
Length	The number of byte to be erased.
AccessPassword	The access password.

Return value:

A byte representing the "Write-locked" response of the tag.

Syntax:

C# representation:

```
public byte         Fujitsu_BurstErase(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length,
                              int                 AccessPassword)
```

JAVA representation:

```
public byte         Fujitsu_BurstErase(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_SecureBurstErase(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                MemBank,
                              short               Address,
                              char                Length,
                              char                *TRData,
                              int                 AccessPassword);
```

Fujitsu_BurstWrite Method

Fujitsu_BurstWrite Method (CAENRFIDTag, Byte, Int16, Byte, Byte[])

Description:

This method can be used to issue a BurstWrite custom command as defined by the Fujitsu datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.

Return value:

A byte representing the "Write-locked" response of the tag.

Syntax:

C# representation:

```
public byte         Fujitsu_BurstWrite(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length,
                              byte[]              Data)
```

JAVA representation:

```
public byte         Fujitsu_BurstWrite(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_BurstWrite(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                MemBank,
                              short               Address,
                              char                Length,
                              char                *Data,
                              char                *TRData);
```

Fujitsu_BurstWrite Method (CAENRFIDTag, Byte, Int16, Byte, Byte[], Int32)

Description:

This method can be used to issue a BurstWrite custom command as defined by the Fujitsu datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.
AccessPassword	The access password.

Return value:

A byte representing the "Write-locked" response of the tag.

Syntax:

C# representation:

```
public byte         Fujitsu_BurstWrite(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length,
                              byte[]              Data,
                              int                 AccessPassword)
```

JAVA representation:

```
public byte         Fujitsu_BurstWrite(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               Address,
                              byte                Length,
                              byte[]              Data,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_SecureBurstWrite(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                MemBank,
                              short               Address,
                              char                Length,
                              char                *Data,
                              char                *TRData,
                              int                 AccessPassword);
```

Fujitsu_ChgBlockGroupPassword Method

Fujitsu_ChgBlockGroupPassword Method (CAENRFIDTag, Byte, Int32, Int32)

Description:

This method can be used to issue a ChgBlockGroupPassword custom command as defined by the Fujitsu datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
BlockGroupPtr	The BlockGroupPtr parameter as defined by the Fujitsu datasheet.
NewPassword	The new password to set.
OldPassword	The current password.

Syntax:

C# representation:

```
public void         Fujitsu_ChgBlockGroupPassword(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 NewPassword,
                              int                 OldPassword)
```

JAVA representation:

```
public void         Fujitsu_ChgBlockGroupPassword(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 NewPassword,
                              int                 OldPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_ChgBlockGroupPassword(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockGroupPtr,
                              int                 NewPassword,
                              int                 OldPassword);
```

Fujitsu_ChgBlockGroupPassword Method (CAENRFIDTag, Byte, Int32, Int32, Int32)

Description:

This method can be used to issue a ChgBlockGroupPassword custom command as defined by the Fujitsu datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
BlockGroupPtr	The BlockGroupPtr parameter as defined by the Fujitsu datasheet.
NewPassword	The new password to set.
OldPassword	The current password.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Fujitsu_ChgBlockGroupPassword(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 NewPassword,
                              int                 OldPassword,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Fujitsu_ChgBlockGroupPassword(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 NewPassword,
                              int                 OldPassword,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_SecureChgBlockGroupPassword(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockGroupPtr,
                              int                 NewPassword,
                              int                 OldPassword,
                              int                 AccessPassword);
```

Fujitsu_ChgBlockLock Method

Fujitsu_ChgBlockLock Method (CAENRFIDTag, Byte, Int32, Int32)

Description:

This method can be used to issue a ChgBlockLock custom command as defined by the Fujitsu datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
BlockGroupPtr	The BlockGroupPtr parameter as defined by the Fujitsu datasheet.
Payload	The Payload parameter as defined by the Fujitsu datasheet.
UserPassword	The Password parameter as defined by the Fujitsu datasheet.

Syntax:

C# representation:

```
public void         Fujitsu_ChgBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 Payload,
                              int                 UserPassword)
```

JAVA representation:

```
public void         Fujitsu_ChgBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 Payload,
                              int                 UserPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_ChgBlockLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockGroupPtr,
                              int                 Payload,
                              int                 UserPassword);
```

Fujitsu_ChgBlockLock Method (CAENRFIDTag, Byte, Int32, Int32, Int32)

Description:

This method can be used to issue a ChgBlockLock custom command as defined by the Fujitsu datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
BlockGroupPtr	The BlockGroupPtr parameter as defined by the Fujitsu datasheet.
Payload	The Payload parameter as defined by the Fujitsu datasheet.
UserPassword	The Password parameter as defined by the Fujitsu datasheet.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Fujitsu_ChgBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 Payload,
                              int                 UserPassword,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Fujitsu_ChgBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 Payload,
                              int                 UserPassword,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_SecureChgBlockLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockGroupPtr,
                              int                 Payload,
                              int                 UserPassword,
                              int                 AccessPassword);
```

Fujitsu_ChgWordLock Method

Fujitsu_ChgWordLock Method (CAENRFIDTag, Byte, Int16, Byte, Int32)

Description:

This method can be used to issue a ChgWordLock custom command as defined by the Fujitsu datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
MemBank	The memory bank (only user memory is valid).
WordPtr	The WordPtr parameter as defined by the Fujitsu datasheet.
Payload	The Payload parameter as defined by the Fujitsu datasheet.
UserPassword	The Password parameter as defined by the Fujitsu datasheet.

Syntax:

C# representation:

```
public void         Fujitsu_ChgWordLock(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               WordPtr,
                              byte                Payload,
                              int                 UserPassword)
```

JAVA representation:

```
public void         Fujitsu_ChgWordLock(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               WordPtr,
                              byte                Payload,
                              int                 UserPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_ChgWordLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                MemBank,
                              short               WordPtr,
                              char                Payload,
                              int                 UserPassword);
```

Fujitsu_ChgWordLock Method (CAENRFIDTag, Byte, Int16, Byte, Int32, Int32)

Description:

This method can be used to issue a ChgWordLock custom command as defined by the Fujitsu datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
MemBank	The memory bank (only user memory is valid).
WordPtr	The WordPtr parameter as defined by the Fujitsu datasheet.
Payload	The Payload parameter as defined by the Fujitsu datasheet.
UserPassword	The Password parameter as defined by the Fujitsu datasheet.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Fujitsu_ChgWordLock(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               WordPtr,
                              byte                Payload,
                              int                 UserPassword,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Fujitsu_ChgWordLock(
                              CAENRFIDTag         Tag,
                              byte                MemBank,
                              short               WordPtr,
                              byte                Payload,
                              int                 UserPassword,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_SecureChgWordLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                MemBank,
                              short               WordPtr,
                              char                Payload,
                              int                 UserPassword,
                              int                 AccessPassword);
```

Fujitsu_ReadBlockLock Method

Fujitsu_ReadBlockLock Method (CAENRFIDTag, Byte)

Description:

This method can be used to issue a ReadBlockLock custom command as defined by the Fujitsu datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
BlockGroupPtr	The BlockGroupPtr parameter as defined by the Fujitsu datasheet.

Return value:

An array of bytes representing the BlockLockFlag of the tag.

Syntax:

C# representation:

```
public byte[]       Fujitsu_ReadBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr)
```

JAVA representation:

```
public byte[]       Fujitsu_ReadBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_ReadBlockLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockGroupPtr,
                              char                *TRData);
```

Fujitsu_ReadBlockLock Method (CAENRFIDTag, Byte, Int32)

Description:

This method can be used to issue a ReadBlockLock custom command as defined by the Fujitsu datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
BlockGroupPtr	The BlockGroupPtr parameter as defined by the Fujitsu datasheet.
AccessPassword	The access password.

Return value:

An array of bytes representing the BlockLockFlag of the tag.

Syntax:

C# representation:

```
public byte[]       Fujitsu_ReadBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 AccessPassword)
```

JAVA representation:

```
public byte[]       Fujitsu_ReadBlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockGroupPtr,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_SecureReadBlockLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockGroupPtr,
                              char                *TRData,
                              int                 AccessPassword);
```

Fujitsu_Refresh Method

Fujitsu_Refresh Method (CAENRFIDTag, Byte)

Description:

This method can be used to issue a Refresh custom command as defined by the Fujitsu datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
Option	The option parameter as defined by the Fujitsu datasheet.

Syntax:

C# representation:

```
public void         Fujitsu_Refresh(
                              CAENRFIDTag         Tag,
                              byte                Option)
```

JAVA representation:

```
public void         Fujitsu_Refresh(
                              CAENRFIDTag         Tag,
                              byte                Option)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_Refresh(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                Option);
```

Fujitsu_Refresh Method (CAENRFIDTag, Byte, Int32)

Description:

This method can be used to issue a Refresh custom command as defined by the Fujitsu datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
Option	The option parameter as defined by the Fujitsu datasheet.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Fujitsu_Refresh(
                              CAENRFIDTag         Tag,
                              byte                Option,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Fujitsu_Refresh(
                              CAENRFIDTag         Tag,
                              byte                Option,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Fujitsu_SecureRefresh(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                Option,
                              int                 AccessPassword);
```

GetBufferedData Method

Description:
This method returns all the Tags stored in reader's buffer using all the ReadPoints belonging to the Source. Only on A828BT.

Return value:
An array of CAENRFIDTag objects detected.

Syntax:

C# representation:

```
public CAENRFIDTag[]     GetBufferedData()
```

JAVA representation:

```
public CAENRFIDTag[]     GetBufferedData()
                               throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes       CAENRFID_GetBufferedData(
                               CAENRFIDHandle      handle,
                               char                *source,
                               CAENRFIDTag         **Receive,
                               int                 *Size);
```

GetDESB_ISO180006B Method

Description:
This method can be used to retrieve the Data Exchange Status Bit setting (see ISO18000-6B protocol specification) used by the anticollision algorithm when called on this logical source.

Return value:
The current DESB setting value.

Syntax:

C# representation:

```
public CAENRFIDLogicalSourceConstants GetDESB_ISO180006B()
```

JAVA representation:

```
public CAENRFIDLogicalSourceConstants GetDESB_ISO180006B()
                                      throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes                    GetDESB_ISO180006B(
                                      CAENRFIDHandle      handle,
                                      unsigned short      *Status);
```

GetName Method

Description:
This method gets a string representing the name of the logical source.

Return value:
A string representing the name of the logical source.

Syntax:

C# representation:

```
public string           GetName()
```

JAVA representation:

```
public java.lang.String  GetName()
```

Remarks:
This function does not exist in C language, see § Overview on SDK pag. 7 for more informations.

GetQ_EPC_C1G2 Method

Description:
This method can be used to retrieve the current setting for the initial Q value (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Return value:
The current initial Q value setting.

Syntax:

C# representation:

```
public int          GetQ_EPC_C1G2()
```

JAVA representation:

```
public int          GetQ_EPC_C1G2()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetQValue_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              int                 *Q);
```

GetReadCycle Method

Description:
This method gets the current setting for the number of read cycles performed by the logical source during the inventory algorithm execution.
ReadCycle affects only inventory performed with continuos mode (see EventInventoryTag method).

Return value:
The number of read cycles.

Syntax:

C# representation:

```
public int          GetReadCycle()
```

JAVA representation:

```
public int          GetReadCycle()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetReadCycle(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              int                 *value);
```

GetSelected_EPC_C1G2 Method

Description:
This method can be used to retrieve the Selected flag (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Return value:
The current Selected value

Syntax:

C# representation:

```
public CAENRFIDLogicalSourceConstants      GetSelected_EPC_C1G2()
```

JAVA representation:

```
public CAENRFIDLogicalSourceConstants      GetSelected_EPC_C1G2()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetSelected_EPC_C1G2(
                              CAENRFIDHandle                  handle,
                              char                            *SourceName,
                              CAENRFIDLogicalSourceConstants  *value);
```

GetSession_EPC_C1G2 Method

Description:
This method can be used to retrieve the Session setting (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Return value:
The current Session value setting.

Syntax:

C# representation:

```
public CAENRFIDLogicalSourceConstants      GetSession_EPC_C1G2()
```

JAVA representation:

```
public CAENRFIDLogicalSourceConstants      GetSession_EPC_C1G2()
                                        throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetSession_EPC_C1G2(
                                CAENRFIDHandle                  handle,
                                char                            *SourceName,
                                CAENRFIDLogicalSourceConstants  *value);
```

GetTarget_EPC_C1G2 Method

Description:
This method can be used to retrieve the Target setting (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Return value:
The current Target value setting.

Syntax:

C# representation:

```
public CAENRFIDLogicalSourceConstants      GetTarget_EPC_C1G2()
```

JAVA representation:

```
public CAENRFIDLogicalSourceConstants      GetTarget_EPC_C1G2()
                                        throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetTarget_EPC_C1G2(
                                CAENRFIDHandle                  handle,
                                char                            *SourceName,
                                CAENRFIDLogicalSourceConstants  *value);
```

GroupSelUnsel Method

Description:
This method can be used to send a Group Select/Unselect command to the tag (see ISO18000-6B protocol specification).

Parameters:

Name	Description
Code	The operation code as defined by the protocol.
Address	The Address from which start the comparison.
BitMask	The bit mask to use.
Data	The data to be compared.

Return value:
The selected tag.

Syntax:

C# representation:

```
public CAENRFIDTag      GroupSelUnsel(
                              CAENRFIDSelUnselOptions Code,
                              short               Address,
                              short               BitMask,
                              byte[]              Data)
```

JAVA representation:

```
public CAENRFIDTag      GroupSelUnsel(
                              CAENRFIDSelUnselOptions Code,
                              short               Address,
                              short               BitMask,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes      CAENRFID_GroupSelUnsel(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              CAENRFID_SelUnsel_Op Code,
                              int                 Address,
                              int                 BitMask,
                              void                *Data,
                              CAENRFIDTag         *Tag);
```

Hitachi_BlockLock Method

Hitachi_BlockLock Method (CAENRFIDTag, Byte, Int32, Byte)

Description:

This method can be used to issue a BlockLock custom command as defined by the Hitachi Hibiki datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to select.
BlockNum	The index of the memory block to be locked.
UserPassword	The user password.
Payload	The Payload parameter for the BlockLock custom command as defined by Hitachi Hibiki datasheet.

Syntax:

C# representation:

```
public void         Hitachi_BlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload)
```

JAVA representation:

```
public void         Hitachi_BlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_BlockLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockNum,
                              int                 UserPassword,
                              char                Payload);
```

Hitachi_BlockLock Method (CAENRFIDTag, Byte, Int32, Byte, Int32)

Description:

This method can be used to issue a BlockLock custom command as defined by the Hitachi Hibiki datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to select.
BlockNum	The index of the memory block to be locked.
UserPassword	The user password.
Payload	The Payload parameter for the BlockLock custom command as defined by Hitachi Hibiki datasheet.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Hitachi_BlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Hitachi_BlockLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_SecureBlockLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                BlockNum,
                              int                 UserPassword,
                              char                Payload,
                              int                 AccessPassword);
```

Hitachi_BlockReadLock Method

Hitachi_BlockReadLock Method (CAENRFIDTag, Byte, Int32, Byte)

Description:

This method can be used to issue a BlockReadLock custom command as defined by the Hitachi Hibiki datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to select.
BlockNum	The index of the memory block to be locked.
UserPassword	The user password.
Payload	The Payload parameter for the BlockReadLock custom command as defined by Hitachi Hibiki datasheet.

Syntax:

C# representation:

```
public void         Hitachi_BlockReadLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload)
```

JAVA representation:

```
public void         Hitachi_BlockReadLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_BlockReadLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload);
```

Hitachi_BlockReadLock Method (CAENRFIDTag, Byte, Int32, Byte, Int32)

Description:

This method can be used to issue a BlockReadLock custom command as defined by the Hitachi Hibiki datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to select.
BlockNum	The index of the memory block to be locked.
UserPassword	The user password.
Payload	The Payload parameter for the BlockReadLock custom command as defined by Hitachi Hibiki datasheet.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Hitachi_BlockReadLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Hitachi_BlockReadLock(
                              CAENRFIDTag         Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_SecureBlockReadLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              byte                BlockNum,
                              int                 UserPassword,
                              byte                Payload,
                              int                 AccessPassword);
```

Hitachi_GetSystemInformation Method

Hitachi_GetSystemInformation Method (CAENRFIDTag)

Description:

This method can be used to issue a GetSystemInformation custom command as defined by the Hitachi Hibiki datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.

Return value:

The command response as defined by the Hitachi Hibiki datasheet.

Syntax:

C# representation:

```
public byte[]           Hitachi_GetSystemInformation(
                                CAENRFIDTag         Tag)
```

JAVA representation:

```
public HitachiSysInfo   Hitachi_GetSystemInformation(
                                CAENRFIDTag         Tag)
                                throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes      CAENRFID_Hitachi_GetSystemInformation(
                                CAENRFIDHandle      handle,
                                CAENRFIDTag         *Tag,
                                char                *TRData);
```

Hitachi_GetSystemInformation Method (CAENRFIDTag, Int32)

Description:

This method can be used to issue a GetSystemInformation custom command as defined by the Hitachi Hibiki datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
AccessPassword	The access password.

Return value:

The command response as defined by the Hitachi Hibiki datasheet.

Syntax:

C# representation:

```
public byte[]           Hitachi_GetSystemInformation(
                              CAENRFIDTag         Tag,
                              int                 AccessPassword)
```

JAVA representation:

```
public HitachiSysInfo   Hitachi_GetSystemInformation(
                              CAENRFIDTag         Tag,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes      CAENRFID_Hitachi_SecureGetSystemInformation(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                *TRData,
                              int                 AccessPassword);
```

Hitachi_ReadLock Method

Hitachi_ReadLock Method (CAENRFIDTag, Int16)

Description:

This method can be used to issue a ReadLock custom command as defined by the Hitachi Hibiki datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to select.
Payload	The Payload parameter for the ReadLock custom command as defined by Hitachi Hibiki datasheet.

Syntax:

C# representation:

```
public void         Hitachi_ReadLock(
                              CAENRFIDTag         Tag,
                              short               Payload)
```

JAVA representation:

```
public void         Hitachi_ReadLock(
                              CAENRFIDTag         Tag,
                              short               Payload)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_ReadLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               Payload);
```

Hitachi_ReadLock Method (CAENRFIDTag, Int16, Int32)

Description:

This method can be used to issue a ReadLock custom command as defined by the Hitachi Hibiki datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to select.
Payload	The Payload parameter for the ReadLock custom command as defined by Hitachi Hibiki datasheet.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Hitachi_ReadLock(
                              CAENRFIDTag         Tag,
                              short               Payload,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Hitachi_ReadLock(
                              CAENRFIDTag         Tag,
                              short               Payload,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_SecureReadLock(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               Payload,
                              int                 AccessPassword);
```

Hitachi_SetAttenuate Method

Hitachi_SetAttenuate Method (CAENRFIDTag, Byte, Boolean)

Description:

This method can be used to issue a SetAttenuate custom command as defined by the Hitachi Hibiki datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
Level	The level of attenuation.
LevelLock	A flag that permit to lock permanently the attenuation level.

Syntax:

C# representation:

```
public void         Hitachi_SetAttenuate(
                              CAENRFIDTag         Tag,
                              byte                Level,
                              bool                LevelLock)
```

JAVA representation:

```
public void         Hitachi_SetAttenuate(
                              CAENRFIDTag         Tag,
                              byte                Level,
                              boolean             LevelLock)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_SetAttenuate(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                Level,
                              char                LevelLock);
```

Hitachi_SetAttenuate Method (CAENRFIDTag, Byte, Boolean, Int32)

Description:

This method can be used to issue a SetAttenuate custom command as defined by the Hitachi Hibiki datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be selected.
Level	The level of attenuation.
LevelLock	A flag that permit to lock permanently the attenuation level.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Hitachi_SetAttenuate(
                              CAENRFIDTag         Tag,
                              byte                Level,
                              bool                LevelLock,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Hitachi_SetAttenuate(
                              CAENRFIDTag         Tag,
                              byte                Level,
                              boolean             LevelLock,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_SecureSetAttenuate(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                Level,
                              char                LevelLock,
                              int                 AccessPassword);
```

Hitachi_WriteMultipleWords Method

Hitachi_WriteMultipleWords Method (CAENRFIDTag, Int16, Int16, Int16, Byte[])

Description:

This method can be used to issue a WriteMultipleWords custom command as defined by the Hitachi Hibiki datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to select.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.

Syntax:

C# representation:

```
public void         Hitachi_WriteMultipleWords(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data)
```

JAVA representation:

```
public void         Hitachi_WriteMultipleWords(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              byte                Length,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_WriteMultipleWords(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              char                *Data);
```

Hitachi_WriteMultipleWords Method (CAENRFIDTag, Int16, Int16, Int16, Byte[], Int32)

Description:

This method can be used to issue a WriteMultipleWords custom command as defined by the Hitachi Hibiki datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to select.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         Hitachi_WriteMultipleWords(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         Hitachi_WriteMultipleWords(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              byte                Length,
                              byte[]              Data,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Hitachi_SecureWriteMultipleWords(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              char                *Data,
                              int                 AccessPassword);
```

InventoryTag Method

InventoryTag Method ()

Description:
A call to this method will execute a read cycle on each read point linked to the logical source. Depending on the air protocol setting it will execute the appropriate anticollision algorithm.

Return value:
An array containing the CAENRFIDTag objects representing the tags read from the read points.

Syntax:

C# representation:

```
public CAENRFIDTag[]     InventoryTag()
```

JAVA representation:

```
public CAENRFIDTag[]     InventoryTag()
                               throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes       CAENRFID_InventoryTag (
                               CAENRFIDHandle      handle,
                               char                *SourceName,
                               CAENRFIDTag         **Receive,
                               int                 *Size);
```

InventoryTag Method (Byte[], Int16, Int16)

Description:

A call to this method will execute a read cycle on each read point linked to the logical source.

Parameters:

Name	Description
Mask	A byte array representing the bitmask to apply.
MaskLength	A value representing the bit-oriented length of the bitmask.
Position	A value representing the first bit of ID where the match will start.

Return value:

An array containing the CAENRFIDTag objects representing the tags read from the read points.

Syntax:

C# representation:

```
public CAENRFIDTag[]     InventoryTag(
                               byte[]              Mask,
                               short               MaskLength,
                               short               Position)
```

JAVA representation:

```
public CAENRFIDTag[]     InventoryTag(
                               byte[]              Mask,
                               short               MaskLength,
                               short               Position)
                               throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_FilteredInventoryTag(
                               CAENRFIDHandle      handle,
                               char                *SourceName,
                               char                *Mask,
                               unsigned char       MaskLength,
                               unsigned char       Position,
                               CAENRFIDTag         **Receive,
                               int                 *Size);
```

Remarks:

Depending on the air protocol setting it will execute the appropriate anticollision algorithm. This version of the method permits to specify a bitmask for filtering tag's populations as described by the EPC Class1 Gen2 (ISO18000-6C) air protocol. The filtering will be performend on the memory bank specified by bank parameter, starting at the bit indicated by the Position index and for a MaskLength length. The method will return only the tags that matches the given Mask. Passing a zero value for MaskLength it performs as the non-filtering InventoryTag method.

InventoryTag Method (Byte[], Int16, Int16, Int16)

Description:
A call to this method will execute a read cycle on each read point linked to the logical source.

Parameters:

Name	Description
Mask	A byte array representing the bitmask to apply.
MaskLength	A value representing the bit-oriented length of the bitmask.
Position	A value representing the first bit of ID where the match will start.
Flag	A bitmask representing the InventoryTag options.

Return value:
An array containing the CAENRFIDTag objects representing the tags read from the read points.

Syntax:

C# representation:

```
public CAENRFIDTag[]     InventoryTag(
                               byte[]              Mask,
                               short               MaskLength,
                               short               Position,
                               short               Flag)
```

JAVA representation:

```
public CAENRFIDTag[]     InventoryTag(
                               byte[]              Mask,
                               short               MaskLength,
                               short               Position,
                               short               Flag)
                               throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_FlagInventoryTag (
                               CAENRFIDHandle      handle,
                               char                *SourceName,
                               char                *Mask,
                               unsigned char       MaskLength,
                               unsigned char       Position,
                               unsigned char       Flag,
                               CAENRFIDTag         **Receive,
                               int                 *Size);
```

Remarks:
Depending on the air protocol setting it will execute the appropriate anticollision algorithm. This version of the method permits to specify a bitmask for filtering tag's populations as described by the EPC Class1 Gen2 (ISO18000-6C) air protocol. The filtering will be performend on the memory bank specified by bank parameter, starting at the bit indicated by the Position index and for a MaskLength length. The method will return only the tags that matches the given Mask. Passing a zero value for MaskLength it performs as the non-filtering InventoryTag method. The Flags parameter permits to set InventoryTag method's options.In this case bit 1 and 2 of the flag (continuos and framed mode) are ignored.

Flag value meaning	
Bit 0	RSSI: a 1 value indicates the reader will transmit the RSSI (Return Signal Strength Indicator) in the response.
Bit 1	Framed data: a 1 value indicates that the tag's data will be transmitted by the reader to the PC as soon as the tag is detected, a 0 value means that all the tags detected are buffered in the reader and trasmitted all together at the end of the inventory cycle.
Bit 2	Continuous acquisition: a 1 value indicates that the inventory cycle is repeated by the reader depending on the SetReadCycle setting value, a 0 value means that only one inventory cycle will be performed. If the continuous mode is selected a 0 value in the ReadCycle setting will instruct the reader to repeat the inventory cycle until an InventoryAbort method is invoked, a value X different from 0 means that the inventory cycle will be performed X times by the reader.
Bit 3	Compact data: a 1 value indicates that only the EPC of the tag will be returned by the reader, a 0 value indicates that the complete data will be returned. In case that the compact option is enabled all the other data will be populated by this library with fakes values.
Bit 4	TID reading: a 1 value indicates that also the TID of the tag will be returned by the reader together with the other informations.

InventoryTag Method (Int16, Byte[], Int16, Int16)

Description:
A call to this method will execute a read cycle on each read point linked to the logical source.

Parameters:

Name	Description
bank	A value representing the memory bank where apply the filter.
Mask	A byte array representing the bitmask to apply.
MaskLength	A value representing the bit-oriented length of the bitmask.
Position	A value representing the first bit of ID where the match will start.

Return value:
An array containing the CAENRFIDTag objects representing the tags read from the read points.

Syntax:

C# representation:

```
public CAENRFIDTag[]     InventoryTag(
                               short               bank,
                               byte[]              Mask,
                               short               MaskLength,
                               short               Position)
```

JAVA representation:

```
public CAENRFIDTag[]     InventoryTag(
                               short               bank,
                               byte[]              Mask,
                               short               MaskLength,
                               short               Position)
                               throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_BankFilteredInventoryTag (
                               CAENRFIDHandle      handle,
                               char                *SourceName,
                               short               bank,
                               short               Position,
                               short               MaskLength,
                               char                *Mask,
                               CAENRFIDTag         **Receive,
                               int                 *Size);
```

Remarks:
Depending on the air protocol setting it will execute the appropriate anticollision algorithm. This version of the method permits to specify a bitmask for filtering tag's populations as described by the EPC Class1 Gen2 (ISO18000-6C) air protocol. The filtering will be performend on the memory bank specified by bank parameter, starting at the bit indicated by the Position index and for a MaskLength length. The method will return only the tags that matches the given Mask. Passing a zero value for MaskLength it performs as the non-filtering InventoryTag method.

InventoryTag Method (Int16, Byte[], Int16, Int16, Int16)

Description:

A call to this method will execute a read cycle on each read point linked to the logical source.

Parameters:

Name	Description
bank	A value representing the memory bank where apply the filter.
Mask	A byte array representing the bitmask to apply.
MaskLength	A value representing the bit-oriented length of the bitmask.
Position	A value representing the first bit of ID where the match will start.
Flag	A bitmask representing the InventoryTag options.

Return value:

An array containing the CAENRFIDTag objects representing the tags read from the read points.

Syntax:

C# representation:

```
public CAENRFIDTag[]     InventoryTag(
                                short               bank,
                                byte[]              Mask,
                                short               MaskLength,
                                short               Position,
                                short               Flag)
```

JAVA representation:

```
public CAENRFIDTag[]     InventoryTag(
                                short               bank,
                                byte[]              Mask,
                                short               MaskLength,
                                short               Position,
                                short               Flag)
                                throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_BankFilteredFlagInventoryTag (
                                CAENRFIDHandle      handle,
                                char                *SourceName,
                                short               bank,
                                short               Position,
                                short               MaskLength,
                                char                *Mask,
                                unsigned char       Flag,
                                CAENRFIDTag         **Receive,
                                int                 *Size);
```

Remarks:
Depending on the air protocol setting it will execute the appropriate anticollision algorithm. This version of the method permits to specify a bitmask for filtering tag's populations as described by the EPC Class1 Gen2 (ISO18000-6C) air protocol. The filtering will be performend on the memory bank specified by bank parameter, starting at the bit indicated by the Position index and for a MaskLength length. The method will return only the tags that matches the given Mask. Passing a zero value for MaskLength it performs as the non-filtering InventoryTag method. The Flags parameter permits to set InventoryTag method's options.In this case bit 1 and 2 of the flag (continuos and framed mode) are ignored.

Flag value meaning	
Bit 0	RSSI: a 1 value indicates the reader will transmit the RSSI (Return Signal Strength Indicator) in the response.
Bit 1	Framed data: a 1 value indicates that the tag's data will be transmitted by the reader to the PC as soon as the tag is detected, a 0 value means that all the tags detected are buffered in the reader and trasmitted all together at the end of the inventory cycle.
Bit 2	Continuous acquisition: a 1 value indicates that the inventory cycle is repeated by the reader depending on the SetReadCycle setting value, a 0 value means that only one inventory cycle will be performed. If the continuous mode is selected a 0 value in the ReadCycle setting will instruct the reader to repeat the inventory cycle until an InventoryAbort method is invoked, a value X different from 0 means that the inventory cycle will be performed X times by the reader.
Bit 3	Compact data: a 1 value indicates that only the EPC of the tag will be returned by the reader, a 0 value indicates that the complete data will be returned. In case that the compact option is enabled all the other data will be populated by this library with fakes values.
Bit 4	TID reading: a 1 value indicates that also the TID of the tag will be returned by the reader together with the other informations.

FreeTagsMemory

Description:
The function permits to free the allocated memory by CAENRFID_InventoryTag.
Unlike the C#/Java languages where objects are automatically destroyed by the Runtime Environment, in C language it is necessary to esplicitly deallocate the memory allocated by the identified tags. To do that, the FreeTagsMemory function is available, passing the pointer to the identified tags list.

Parameters:

Name	Description
Tags	tags array returned by one of the inventory family function.

Syntax:

C representation:

```
void                CAENRFID_FreeTagsMemory(
                                CAENRFIDTag         **Tags);
```

isReadPointPresent Method

Description:
This method checks if a read point is present in the logical source.

Parameters:

Name	Description
ReadPoint	A string representing the name of the read point (antenna).

Return value:
A boolean value representing the presence of a read point in the logical source (true means that it is present, false if it is not present).

Syntax:

C# representation:

```
public bool         isReadPointPresent(
                              string              ReadPoint)
```

JAVA representation:

```
public boolean      isReadPointPresent(
                              java.lang.String    ReadPoint)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_isReadPointPresent(
                              CAENRFIDHandle      handle,
                              char                *ReadPoint,
                              char                *SourceName,
                              short               *isPresent);
```

KillTag_EPC_C1G1 Method

Description:
This method can be used to kill a EPC Class 1 Gen 1 tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be killed.
Password	The tag's kill password.

Syntax:

C# representation:

```
public void         KillTag_EPC_C1G1(
                              CAENRFIDTag         Tag,
                              short               Password)
```

JAVA representation:

```
public void         KillTag_EPC_C1G1(
                              CAENRFIDTag         Tag,
                              short               Password)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_KillTag_EPC_C1G1(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                Password);
```

KillTag_EPC_C1G2 Method

KillTag_EPC_C1G2 Method (CAENRFIDTag, Int32)

Description:

This method can be used to kill a EPC Class 1 Gen 2 tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be killed.
Password	The tag's kill password.

Syntax:

C# representation:

```
public void         KillTag_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              int                 Password)
```

JAVA representation:

```
public void         KillTag_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              int                 Password)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_KillTag_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 Password);
```

KillTag_EPC_C1G2 Method (Int16, Int16, Int16, Byte[], Int32)

Description:

This method can be used to kill a EPC Class 1 Gen 2 tag.

Parameters:

Name	Description
BankMask	Memory bank for tag identificantion.
PositionMask	Bit position (from the start of the selected bank) where apply the mask to match.
LengthMask	Length of the mask.
Mask	Mask of byte.
Password	The tag's kill password.

Syntax:

C# representation:

```
public void         KillTag_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              int                 Password)
```

JAVA representation:

```
public void         KillTag_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              int                 Password)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_BankFilteredKillTag_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              char                *Mask,
                              int                 Password);
```

LockBlockPermaLock_EPC_C1G2 Method

Description:
This method implements the BLockPermaLock with ReadLock=1 as specified in EPC C1G2 rev. 1.2.0 protocol.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
MemBank	The memory bank where to write the data.
BlockPtr	The address where to start writing the data.
BlockRange	The number of word of the mask.
Mask	A bitmask that permit to select which of the four bytes have to be locked (i.e. mask 0x05 write the bytes on position Address + 1 and Address + 3).
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         LockBlockPermaLock_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               BlockPtr,
                              short               BlockRange,
                              byte[]              Mask,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         LockBlockPermaLock_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               BlockPtr,
                              short               BlockRange,
                              byte[]              Mask,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_LockBlockPermaLock_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              short               BlockPtr,
                              short               BlockRange,
                              byte[]              Mask,
                              int                 AccessPassword);
```

LockTag_EPC_C1G2 Method

LockTag_EPC_C1G2 Method (CAENRFIDTag, Int32)

Description:

This method can be used to lock a memory bank of a EPC Class 1 Gen 2 tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be locked.
Payload	The Payload parameter for the lock command as defined by the EPC Class 1 Gen 2 protocol specification.

Syntax:

C# representation:

```
public void         LockTag_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              int                 Payload)
```

JAVA representation:

```
public void         LockTag_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              int                 Payload)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_LockTag_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 Payload);
```

LockTag_EPC_C1G2 Method (CAENRFIDTag, Int32, Int32)

Description:

This method can be used to lock a memory bank of a EPC Class 1 Gen 2 tag after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be locked.
Payload	The Payload parameter for the lock command as defined by the EPC Class 1 Gen 2 protocol specification.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         LockTag_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              int                 Payload,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         LockTag_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              int                 Payload,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureLockTag_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 Payload,
                              int                 AccessPassword);
```

LockTag_EPC_C1G2 Method (Int16, Int16, Int16, Byte[], Int32)

Description:

This method can be used to lock a memory bank of a EPC Class 1 Gen 2 tag.

Parameters:

Name	Description
BankMask	Memory bank for tag identificantion.
PositionMask	Bit position (from the start of the selected bank) where apply the mask to match.
LengthMask	Length of the mask.
Mask	Mask of byte.
Payload	The Payload parameter for the lock command as defined by the EPC Class 1 Gen 2 protocol specification.

Syntax:

C# representation:

```
public void         LockTag_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              int                 Payload)
```

JAVA representation:

```
public void         LockTag_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              int                 Payload)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_BankFilteredLockTag_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              char                *Mask,
                              int                 Payload);
```

LockTag_EPC_C1G2 Method (Int16, Int16, Int16, Byte[], Int32, Int32)

Description:

This method can be used to lock a memory bank of a EPC Class 1 Gen 2 tag after having put it in Secured state using the Access command.

Parameters:

Name	Description
BankMask	Memory bank for tag identification.
PositionMask	Bit position (from the start of the selected bank) where apply the mask to match.
LengthMask	Length of the mask.
Mask	Mask of byte.
Payload	The Payload parameter for the lock command as defined by the EPC Class 1 Gen 2 protocol specification.
AccessPassword	Access password.

Syntax:

C# representation:

```
public void         LockTag_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              int                 Payload,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         LockTag_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              int                 Payload,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureBankFilteredLockTag_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              char                *Mask,
                              int                 Payload,
                              int                 AccessPassword);
```

LockTag_ISO180006B Method

Description:
This method can be used to lock a byte in the memory of a ISO18000-6B tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be locked.
Address	The byte's address to lock.

Syntax:

C# representation:

```
public void         LockTag_ISO180006B(
                              CAENRFIDTag         Tag,
                              short               Address)
```

JAVA representation:

```
public void         LockTag_ISO180006B(
                              CAENRFIDTag         Tag,
                              short               Address)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_LockTag_ISO180006B(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               Address);
```

NXP_ChangeEAS Method

Description:
This method can be used to issue a ChangeEAS custom command as defined by the NXP G2XM and G2XL datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to select.
EAS	A boolean representing the EAS state to set.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         NXP_ChangeEAS(
                              CAENRFIDTag         Tag,
                              bool                EAS,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         NXP_ChangeEAS(
                              CAENRFIDTag         Tag,
                              boolean             EAS,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_NXP_SecureChangeEAS(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                EAS,
                              int                 AccessPassword);
```

NXP_EAS_Alarm Method

Description:
This method can be used to issue a EAS_Alarm custom command as defined by the NXP G2XM and G2XL datasheet.

Return value:
An array of bytes representing the EAS Code.

Syntax:

C# representation:

```
public byte[]       NXP_EAS_Alarm()
```

JAVA representation:

```
public byte[]       NXP_EAS_Alarm()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_NXP_EAS_Alarm(
                              CAENRFIDHandle      handle,
                              char                *TRData);
```

NXP_ReadProtect Method

NXP_ReadProtect Method (CAENRFIDTag)

Description:
This method can be used to issue a ReadProtect custom command as defined by the NXP G2XM and G2XL datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to select.

Syntax:

C# representation:

```
public void         NXP_ReadProtect(
                              CAENRFIDTag         Tag)
```

JAVA representation:

```
public void         NXP_ReadProtect(
                              CAENRFIDTag         Tag)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_NXP_ReadProtect(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag);
```

NXP_ReadProtect Method (CAENRFIDTag, Int32)

Description:

This method can be used to issue a ReadProtect custom command as defined by the NXP G2XM and G2XL datasheet after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to select.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         NXP_ReadProtect(
                              CAENRFIDTag         Tag,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         NXP_ReadProtect(
                              CAENRFIDTag         Tag,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_NXP_SecureReadProtect(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 AccessPassword);
```

NXP_ResetReadProtect Method

Description:

This method can be used to issue a ResetReadProtect custom command as defined by the NXP G2XM and G2XL datasheet.

Parameters:

Name	Description
Tag	The CAENRFIDTag object representing the tag to reset the read protection.
Password	The ReadProtect password.

Syntax:

C# representation:

```
public void         NXP_ResetReadProtect(
                              CAENRFIDTag         Tag,
                              int                 Password)
```

JAVA representation:

```
public void         NXP_ResetReadProtect(
                              CAENRFIDTag         Tag,
                              int                 Password)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_NXP_ResetReadProtect(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 Password);
```

ProgramID_EPC_C1G1 Method

Description:
This method can be used to write the EPC of a EPC Class 1 Gen 1 tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be programmed, the ID contained in this object will be programmed into the tag.
Password	The password needed in order to write into the tag.
Lock	A flag used to lock the EPC in the tag (1 if the EPC have to be locked).

Syntax:

C# representation:

```
public void         ProgramID_EPC_C1G1(
                              CAENRFIDTag         Tag,
                              short               Password,
                              bool                Lock)
```

JAVA representation:

```
public void         ProgramID_EPC_C1G1(
                              CAENRFIDTag         Tag,
                              short               Password,
                              boolean             Lock)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_ProgramID_EPC_C1G1(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                Password,
                              unsigned short      Lock);
```

ProgramID_EPC_C1G2 Method

ProgramID_EPC_C1G2 Method (CAENRFIDTag, Int16)

Description:
This method can be used to write the EPC of a EPC Class 1 Gen 2 tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be programmed, the ID contained in this object will be programmed into the tag.
NSI	The Numbering System Identifier as defined in EPC Class 1 Gen 2 protocol specifications.

Syntax:

C# representation:

```
public void         ProgramID_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               NSI)
```

JAVA representation:

```
public void         ProgramID_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               NSI)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_ProgramID_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              unsigned short      NSI);
```

ProgramID_EPC_C1G2 Method (CAENRFIDTag, Int16, Int32)

Description:
This method can be used to write the EPC of a EPC Class 1 Gen 2 tag after having put it in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be programmed, the ID contained in this object will be programmed into the tag.
NSI	The Numbering System Identifier as defined in EPC Class 1 Gen 2 protocol specifications.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         ProgramID_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               NSI,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         ProgramID_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               NSI,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureProgramID_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              unsigned short      NSI,
                              int                 AccessPassword);
```

ProgramID_EPC119 Method

Description:
This method can be used to write the UID of a EPC 1.19 tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be programmed.
NewID	An array of bytes representing the new UID for the tag.

Syntax:

C# representation:

```
public void         ProgramID_EPC119(
                              CAENRFIDTag         Tag,
                              byte[]              NewID)
```

JAVA representation:

```
public void         ProgramID_EPC119(
                              CAENRFIDTag         Tag,
                              byte[]              NewID)
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_ProgramID_EPC119(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              char                *NewID);
```

Query_EPC_C1G2 Method

Description:
This method makes the reader generate a EPC Class1 Gen2 Query command.

Return value:
True on successfull completion.

Syntax:

C# representation:

```
public bool         Query_EPC_C1G2()
```

JAVA representation:

```
public boolean      Query_EPC_C1G2()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Query_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               *isPresent);
```

QueryAck_EPC_C1G2 Method

Description:
This method make the reader generate a sequence of EPC Class1 Gen2 Query and Ack commands. It can be used to read a single tag under the field. If there are more than one tag under the field the method fails.

Return value:
An array of bytes representing the EPC of the tag

Syntax:

C# representation:

```
public byte[]       QueryAck_EPC_C1G2()
```

JAVA representation:

```
public byte[]       QueryAck_EPC_C1G2()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  QueryAck_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              byte                *Tag);
```

ReadBLockPermalock_EPC_C1G2 Method

Description:
This method implements the BLockPermaLock with ReadLock=0 as specified in EPCC1G2 rev. 1.2.0 protocol.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be read.
MemBank	The memory bank where to read the data.
Blockptr	The address where to start reading the data.
BlockRange	The number of word to be read.
AccessPassword	The access password.

Return value:
An array of bytes representing the data read from the tag.

Syntax:

C# representation:

```
public byte[]       ReadBLockPermalock_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Blockptr,
                              short               BlockRange,
                              int                 AccessPassword)
```

JAVA representation:

```
public byte[]       ReadBLockPermalock_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Blockptr,
                              short               BlockRange,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_ReadBLockPermalock_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              short               Blockptr,
                              short               BlockRange,
                              int                 AccessPassword)
```

ReadTagData Method

Description:
This method can be used to read a portion of the user memory in a ISO18000-6B tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be read.
Address	The address where to start reading the data.
Length	The number of byte to be read.

Return value:
An array of bytes representing the data read from the tag.

Syntax:

C# representation:

```
public byte[]       ReadTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Length)
```

JAVA representation:

```
public byte[]       ReadTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Length)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_ReadTagData(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 Address,
                              int                 Length,
                              void                *Data);
```

ReadTagData_EPC_C1G2 Method

ReadTagData_EPC_C1G2 Method (CAENRFIDTag, Int16, Int16, Int16)

Description:

This method can be used to read a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be read.
MemBank	The memory bank where to read the data.
Address	The address where to start reading the data.
Length	The number of byte to be read.

Return value:

An array of bytes representing the data read from the tag.

Syntax:

C# representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length)
```

JAVA representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_ReadTagData_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              void                *Data);
```

ReadTagData_EPC_C1G2 Method (CAENRFIDTag, Int16, Int16, Int16, Int32)

Description:

This method can be used to read a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag after having put the tag in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be read.
MemBank	The memory bank where to read the data.
Address	The address where to start reading the data.
Length	The number of byte to be read.
AccessPassword	The access password.

Return value:

An array of bytes representing the data read from the tag.

Syntax:

C# representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              int                 AccessPassword)
```

JAVA representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureReadTagData_EPC_C1G2 (
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              int                 AccessPassword,
                              void                *Data);
```

ReadTagData_EPC_C1G2 Method (Int16, Int16, Int16, Byte[], Int16, Int16, Int16)

Description:

This method can be used to read a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag. In this case the target tag is identified by 'LenghtMask' bytes of passed mask placed in a memory bank 'BankMask' at 'PositionMask' byte from bank starting address byte.

Parameters:

Name	Description
BankMask	Memory bank for tag identificantion.
PositionMask	Bit position (from the start of the selected bank) where apply the mask to match.
LengthMask	Length of the mask.
Mask	Mask of byte.
MemBank	Memory bank where read.
Address	Address where starts reading.
Length	Number of byte to read.

Return value:

An array of bytes representing the data read from the tag.

Syntax:

C# representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length)
```

JAVA representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_BankFilteredReadTagData_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              char                *Mask,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              void                *Data);
```

ReadTagData_EPC_C1G2 Method (Int16, Int16, Int16, Byte[], Int16, Int16, Int16, Int32)

Description:

This method can be used to read a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag. In this case the target tag is identified by 'LenghtMask' bytes of passed mask placed in a memory bank 'BankMask' at 'PositionMask' byte from bank starting address byte. This is the secure version using the Access command.

Parameters:

Name	Description
BankMask	Memory bank for tag identificantion.
PositionMask	Bit position (from the start of the selected bank) where apply the mask to match.
LengthMask	Length of the mask.
Mask	Mask of byte.
MemBank	Memory bank where read.
Address	Address where starts reading.
Length	Number of byte to read.
AccessPassword	Access Password.

Return value:

An array of bytes representing the data read from the tag.

Syntax:

C# representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              int                 AccessPassword)
```

JAVA representation:

```
public byte[]       ReadTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureBankFilteredReadTagData_EPC_C1G2 (
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              void                *Data,
                              int                 AccessPassword);
```

RemoveReadPoint Method

Description:
This method removes a read point from the logical source.

Parameters:

Name	Description
ReadPoint	A string representing the name of the read point (antenna).

Syntax:

C# representation:

```
public void         RemoveReadPoint(
                              string              ReadPoint)
```

JAVA representation:

```
public void         RemoveReadPoint(
                              java.lang.String    ReadPoint)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_RemoveReadPoint(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              char                *ReadPoint);
```

ResetSession_EPC_C1G2 Method

Description:
This method can be used to reset the Session status for EPC Class1 Gen2 tags. After the execution of this method all the tags in the field of the antennas belonging to this logical source are back in the default Session.

Syntax:

C# representation:

```
public void         ResetSession_EPC_C1G2()
```

JAVA representation:

```
public void         ResetSession_EPC_C1G2()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_ResetSession_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName);
```

SetDESB_ISO180006B Method

Description:
This method can be used to set the Data Exchange Status Bit (see ISO18000-6B protocol specification) used by the anticollision algorithm when called on this logical source.

Parameters:

Name	Description
Value	The DESB setting value.

Syntax:

C# representation:

```
public void         SetDESB_ISO180006B(
                              CAENRFIDLogicalSourceConstants   Value)
```

JAVA representation:

```
public void         SetDESB_ISO180006B(
                              CAENRFIDLogicalSourceConstants   Value)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetDESB_ISO180006B(
                              CAENRFIDHandle                   handle,
                              unsigned int                     Value);
```

SetQ_EPC_C1G2 Method

Description:
This method can be used to set the initial Q value (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Parameters:

Name	Description
Value	The initial Q value setting.

Syntax:

C# representation:

```
public void         SetQ_EPC_C1G2(
                              int                  Value)
```

JAVA representation:

```
public void         SetQ_EPC_C1G2(
                              int                  Value)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetQValue_EPC_C1G2(
                              CAENRFIDHandle       handle,
                              char                 *SourceName,
                              int                  Value);
```

SetReadCycle Method

Description:
This method sets the number of read cycles to be performed by the logical source during the inventory algorithm execution.

Parameters:

Name	Description
value	The number of read cycles.

Syntax:

C# representation:

```
public void         SetReadCycle(
                              int                 value)
```

JAVA representation:

```
public void         SetReadCycle(
                              int                 value)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetReadCycle(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              int                 value);
```

SetSelected_EPC_C1G2 Method

Description:
This method can be used to set the Selected flag (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Parameters:

Name	Description
Value	The Selected flag value.

Syntax:

C# representation:

```
public void         SetSelected_EPC_C1G2(
                              CAENRFIDLogicalSourceConstants   Value)
```

JAVA representation:

```
public void         SetSelected_EPC_C1G2(
                              CAENRFIDLogicalSourceConstants   Value)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetSelected_EPC_C1G2(
                              CAENRFIDHandle                   handle,
                              char                             *SourceName,
                              CAENRFIDLogicalSourceConstants   Value);
```

SetSession_EPC_C1G2 Method

Description:
This method can be used to set the Session (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Parameters:

Name	Description
Value	The Session value.

Syntax:

C# representation:

```
public void         SetSession_EPC_C1G2(
                              CAENRFIDLogicalSourceConstants   Value)
```

JAVA representation:

```
public void         SetSession_EPC_C1G2(
                              CAENRFIDLogicalSourceConstants   Value)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetSession_EPC_C1G2(
                              CAENRFIDHandle                   handle,
                              char                             *SourceName,
                              CAENRFIDLogicalSourceConstants   Value);
```

SetTarget_EPC_C1G2 Method

Description:
This method can be used to set the Target setting (see EPC Class1 Gen2 protocol specification) used by the anticollision algorithm when called on this logical source.

Parameters:

Name	Description
Value	The Target value.

Syntax:

C# representation:

```
public void         SetTarget_EPC_C1G2(
                              CAENRFIDLogicalSourceConstants   Value)
```

JAVA representation:

```
public void         SetTarget_EPC_C1G2(
                              CAENRFIDLogicalSourceConstants   Value)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetTarget_EPC_C1G2(
                              CAENRFIDHandle                   handle,
                              char                             *SourceName,
                              CAENRFIDLogicalSourceConstants   Value);
```

WriteTagData Method

Description:
This method can be used to write a portion of the user memory in a ISO18000-6B tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	The data to be written into the tag's user memory.

Syntax:

C# representation:

```
public void         WriteTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Length,
                              byte[]              Data)
```

JAVA representation:

```
public void         WriteTagData(
                              CAENRFIDTag         Tag,
                              short               Address,
                              short               Length,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_WriteTagData(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              int                 Address,
                              int                 Length,
                              void                *Data);
```

WriteTagData_EPC_C1G2 Method

WriteTagData_EPC_C1G2 Method (CAENRFIDTag, Int16, Int16, Int16, Byte[])

Description:

This method can be used to write a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.

Syntax:

C# representation:

```
public void         WriteTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data)
```

JAVA representation:

```
public void         WriteTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_WriteTagData_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              void                *Data);
```

WriteTagData_EPC_C1G2 Method (CAENRFIDTag, Int16, Int16, Int16, Byte[], Int32)

Description:

This method can be used to write a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag after having put the tag in Secured state using the Access command.

Parameters:

Name	Description
Tag	The CAENRFIDTag representing the tag to be written.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         WriteTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         WriteTagData_EPC_C1G2(
                              CAENRFIDTag         Tag,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureWriteTagData_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              CAENRFIDTag         *Tag,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              void                *Data,
                              int                 AccessPassword);
```

WriteTagData_EPC_C1G2 Method (Int16, Int16, Int16, Byte[], Int16, Int16, Int16, Byte[])

Description:

This method can be used to write a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag.

Parameters:

Name	Description
BankMask	Memory bank for tag identificantion.
PositionMask	Bit position (from the start of the selected bank) where apply the mask to match.
LengthMask	Length of the mask.
Mask	Mask of byte.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.

Syntax:

C# representation:

```
public void         WriteTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data)
```

JAVA representation:

```
public void         WriteTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_BankFilteredWriteTagData_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              char                *Mask,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              void                *Data);
```

WriteTagData_EPC_C1G2 Method (Int16, Int16, Int16, Byte[], Int16, Int16, Int16, Byte[], Int32)

Description:

This method can be used to write a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag after having put the tag in Secured state using the Access command.

Parameters:

Name	Description
BankMask	Memory bank for tag identificantion.
PositionMask	Bit position (from the start of the selected bank) where apply the mask to match.
LengthMask	Length of the mask.
Mask	Mask of byte.
MemBank	The memory bank where to write the data.
Address	The address where to start writing the data.
Length	The number of byte to be written.
Data	An array of bytes representing the data to be written into the tag.
AccessPassword	The access password.

Syntax:

C# representation:

```
public void         WriteTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data,
                              int                 AccessPassword)
```

JAVA representation:

```
public void         WriteTagData_EPC_C1G2(
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              byte[]              Mask,
                              short               MemBank,
                              short               Address,
                              short               Length,
                              byte[]              Data,
                              int                 AccessPassword)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SecureBankFilteredWriteTagData_EPC_C1G2(
                              CAENRFIDHandle      handle,
                              char                *SourceName,
                              short               BankMask,
                              short               PositionMask,
                              short               LengthMask,
                              char                *Mask,
                              short               MemBank,
                              int                 Address,
                              int                 Length,
                              void                *Data,
                              int                 AccessPassword);
```

CAENRFIDNotify Class

The CAENRFIDNotify class defines the structure of a notification message.

In both Java and C# lanuguage this class is composed by methods while in C language is present as a struct (for more informations see § Overview on SDK pag.7):

C representation:

```
typedef struct {
                    byte             ID[MAX_ID_LENGTH];
                    short            Length;
                    int              EvtType;
                    CAENRFIDProtocol Type;
                    char             LogicalSource[MAX_LOGICAL_SOURCE_NAME];
                    struct timeval   TimeStamp;
                    short            RSSI;
                    char             ReadPoint[MAX_READPOINT_NAME];
                    byte             TID[MAX_TID_SIZE];
                    short            TIDLen;
                    } CAENRFIDNotify;
```

getDate Method

Description:
This method returns a timestamp representing the time at which the event was generated.

Return value:
The timestamp value.

Syntax:

C# representation:

```
public DateTime              getDate()
```

JAVA representation:

```
public java.util.Date        getDate()
```

getReadPoint Method

Description:
This method returns the read point that has detected the tag.

Return value:
The name of the read point that has detected the Tag.

Syntax:

C# representation:

```
public string                getReadPoint()
```

JAVA representation:

```
public java.lang.String      getReadPoint()
```

getRSSI Method

Description:
This method returns the RSSI value measured for the tag.

Return value:
The tag's RSSI.

Syntax:

C# representation:

```
public short                    getRSSI()
```

JAVA representation:

```
public short                    getRSSI()
```

getStatus Method

Description:
This method returns the event type associated to the tag.

Return value:
The event type associated to the Tag.

Syntax:

C# representation:

```
public CAENRFIDTagEventType    getStatus()
```

JAVA representation:

```
public CAENRFIDTagEventType    getStatus()
```

getTagID Method

Description:
This method returns the tag's ID (the EPC code in Gen2 tags).

Return value:
An array of bytes representing the tag's ID (the EPC code in EPC Class 1 Gen 2 tags).

Syntax:

C# representation:

```
public byte[]                   getTagID()
```

JAVA representation:

```
public byte[]                   getTagID()
```

getTagLength Method

Description:
This method returns the tag's ID length.

Return value:
The tag's length.

Syntax:

C# representation:

```
public short                    getTagLength()
```

JAVA representation:

```
public short                    getTagLength()
```

getTagSource Method

Description:
This method returns the name of the logical source that has detected the tag.

Return value:
The name of the logical source that has detected the tag.

Syntax:

C# representation:

```
public string                getTagSource()
```

JAVA representation:

```
public java.lang.String      getTagSource()
```

getTagType Method

Description:
This method returns the air protocol of the tag.

Return value:
The air protocol of the tag.

Syntax:

C# representation:

```
public short                 getTagType()
```

JAVA representation:

```
public CAENRFIDProtocol      getTagType()
```

getTID Method

Description:
This method returns the TID field value in a EPC Class 1 Gen 2 Tag

Return value:
The bytes of the TID field.

Syntax:

C# representation:

```
public byte[]                getTID()
```

JAVA representation:

```
public java.lang.String      getAntenna()
```

CAENRFIDReader Class

The CAENRFIDReader class is used to create reader objects which permit to access to CAEN RFID readers' configuration and control commands.

Connect Method

Description:
In C# and Java languages, this method starts the communication with the reader. It must be called before any other call to method of the CAENRFIDReader object. See § Managing connections with the readers pag. 7 for more informations.

Parameters:

Name	Description
ConType	The communication link to use for the connection.
Address	Depending on ConType parameter: IP address for TCP/IP communications ("xxx.xxx.xxx.xxx"), COM port for RS232 communications ("COMx"), An index for USB communications (not yet supported).

Syntax:

C# representation:

```
public void         Connect(
                              CAENRFIDPort        ConType,
                              string              Address)
```

JAVA representation:

```
public void         Connect(
                              CAENRFIDPort        ConType,
                              java.lang.String    Address)
                              throws CAENRFIDException
```

Init Function

Description:
In C language, this function generates an opaque handle to identify a module attached to the PC. See § Managing connections with the readers pag. 7 for more informations.

Parameters:

Name	Description
ConType	The communication link to use for the connection.
Address	Communication address (i.e.: "COM1" for RS232, "USB0" for USB of IP address for TCP/IP etc.).
handle	The handle that identifies the device.

Syntax:

C representation:

```
CAENRFIDErrorCodes  CAENRFID_Init(
                              CAENRFIDPort        ConType,
                              char                *Address,
                              CAENRFIDHandle      *handle,
                              CAENRFIDProtocol    *Protocol);
```

Disconnect Method

Description:
In C# and Java languages, this method closes the connection with the CAEN RFID Reader releasing all the allocated resources. See § Managing connections with the readers pag. 7 for more informations.

Syntax:

C# representation:

```
public void         Disconnect()
```

JAVA representation:

```
public void         Disconnect()
                              throws CAENRFIDException
```

End

Description:
In C language, this function closes the connection with the CAEN RFID Reader releasing all the allocated resources. See § Managing connections with the readers pag. 7 for more informations.

Parameters:

Name	Description
handle	The handle that identifies the device.

Syntax:

C representation:

```
CAENRFIDErrorCodes  CAENRFID_End(
                              CAENRFIDHandle      handle);
```

GetBitRate Method

Description:
This method gets the current setting of the RF bit rate.

Return value:
The current RF bit rate value.

Syntax:

C# representation:

```
public CAENRFIDBitRate   GetBitRate()
```

JAVA representation:

```
public CAENRFIDBitRate   GetBitRate()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes       CAENRFID_GetBitrate(
                              CAENRFIDHandle      handle,
                              CAENRFID_Bitrate    *Bitrate);
```

GetFirmwareRelease Method

Description:
This method permits to read the release of the firmware loaded into the device.

Return value:
A string representing the firmware release of the device.

Syntax:

C# representation:

```
public string           GetFirmwareRelease()
```

JAVA representation:

```
public java.lang.String GetFirmwareRelease()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes      CAENRFID_GetFirmwareRelease(
                              CAENRFIDHandle     handle,
                              char               *FWRel);
```

GetIO Method

Description:
This method gets the current digital Input and Output lines status.

Return value:
A bitmask representing the I/O lines status. The format and the meaning of the bits depends on the Reader's model. Please refer to the corresponding user manual available on http://www.caen.it/rfid

Syntax:

C# representation:

```
public int          GetIO()
```

JAVA representation:

```
public int          GetIO()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetIO(
                              CAENRFIDHandle     handle,
                              unsigned int       *IORegister);
```

GetIODirection Method

Description:
This method gets the current I/O direction setting as a bitmask. Each bit represent a I/O line, a value of 0 means that the line is configured as an input, 1 as an output. This setting has a meaning only for those readers with configurable I/O lines.

Return value:
A bitmask representing the I/O setting.

Syntax:

C# representation:

```
public int          GetIODirection()
```

JAVA representation:

```
public int          GetIODirection()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetIODirection(
                              CAENRFIDHandle      handle,
                              unsigned int        *IODirection);
```

GetLBTMode Method

Description:
This method gets the current LBT mode setting. If the current regulation is based on the frequency hopping mechanism it returns the FH status.

Return value:
A zero value if the LBT/FH is disabled, non-zero value if it is enabled.

Syntax:

C# representation:

```
public short        GetLBTMode()
```

JAVA representation:

```
public short        GetLBTMode()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetLBTMode(
                              CAENRFIDHandle      handle,
                              unsigned short      *LBTMode);
```

GetPower Method

Description:
This method gets the current setting of the RF power expressed in mW.

Return value:
The current conducted RF power expressed in mW.

Syntax:

C# representation:

```
public int          GetPower()
```

JAVA representation:

```
public int          GetPower()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetPower(
                              CAENRFIDHandle      handle,
                              unsigned int        *Power);
```

GetProtocol Method

Description:
This method gets the current air protocol of the Reader.

Return value:
A CAENRFIDProtocol representing the current air protocol set on the reader.

Syntax:

C# representation:

```
public CAENRFIDProtocol  GetProtocol()
```

JAVA representation:

```
public CAENRFIDProtocol  GetProtocol()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes       CAENRFID_GetProtocol(
                              CAENRFIDHandle      handle,
                              CAENRFIDProtocol    *Protocol);
```

GetReaderInfo Method

Description:
This method permits to read the reader information loaded into the device.

Return value:
The reader information of the device.

Syntax:

C# representation:

```
public CAENRFIDReaderInfo     GetReaderInfo()
```

JAVA representation:

```
public CAENRFIDReaderInfo     GetReaderInfo()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes            CAENRFID_GetReaderInfo(
                                    CAENRFIDHandle      handle,
                                    char                *Model,
                                    char                *SerialNum);
```

GetReadPoints Method

Description:
This method gets the names of the read points (antennas) available in the reader.

Return value:
An array containing the read points (antennas) names available in the reader.

Syntax:

C# representation:

```
public string[]               GetReadPoints()
```

JAVA representation:

```
public java.lang.String[]     GetReadPoints()
```

C representation:

```
CAENRFIDErrorCodes            CAENRFID_GetReadPoints(
                                    CAENRFIDHandle      handle,
                                    char                **AntNames [],
                                    int                 *AntNumber);
```

GetReadPointStatus Method

Description:
This method gets the CAENRFIDReadPointStatus object representing the status of a read point (antenna).

Parameters:

Name	Description
ReadPoint	The name of the read point to check.

Return value:
The CAENRFIDReadPointStatus object rapresenting the current status of the read point.

Syntax:

C# representation:

```
public CAENRFIDReadPointStatus      GetReadPointStatus(
                                    string              ReadPoint)
```

JAVA representation:

```
public CAENRFIDReadPointStatus      GetReadPointStatus(
                                    java.lang.String    ReadPoint)
                                    throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes                  CAENRFID_GetReadPointStatus(
                                    CAENRFIDHandle      handle,
                                    char                *ReadPoint,
                                    CAENRFIDReadPointStatus *Status);
```

GetRFChannel Method

Description:
This method gets the index of the RF channel currently in use. The index value meaning change for different country regulations.

Return value:
The RF channel index.

Syntax:

C# representation:

```
public short        GetRFChannel()
```

JAVA representation:

```
public short        GetRFChannel()
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_GetRFChannel(
                              CAENRFIDHandle      handle,
                              unsigned short      *RFChannel);
```

Remarks
This method is only used for testing applications.

GetRFRegulation Method

Description:
This method gets the current RF regulation setting value.

Return value:
The RF regulation value.

Syntax:

C# representation:

```
public CAENRFIDRFRegulations  GetRFRegulation()
```

JAVA representation:

```
public CAENRFIDRFRegulations  GetRFRegulation()
                                    throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes            CAENRFID_GetRFRegulation(
                                    CAENRFIDHandle            handle,
                                    CAENRFIDRFRegulations *RFRegulation);
```

GetSource Method

Description:
This method gets a CAENRFIDLogicalSource object given its name.

Parameters:

Name	Description
Source	The name of the logical source.

Return value:
The CAENRFIDLogicalSource object corresponding to the requested name.

Syntax:

C# representation:

```
public CAENRFIDLogicalSource  GetSource(
                                    string              Source)
```

JAVA representation:

```
public CAENRFIDLogicalSource  GetSource(
                                    java.lang.String    Source)
                                    throws CAENRFIDException
```

Remarks:
This function does not exist in C language, see § Overview on SDK pag. 7 for more informations.

GetSourceNames Method

Description:
This method gets the names of the logical sources available in the reader.

Return value:
An array containing the logical source names available in the reader.

Syntax:

C# representation:

```
public static string[]              GetSourceNames()
```

JAVA representation:

```
public static java.lang.String[]    GetSourceNames()
```

C representation:

```
CAENRFIDErrorCodes                  CAENRFID_GetSourceNames(
                                    CAENRFIDHandle      handle,
                                    char                **SrcNames[],
                                    int                 *SrcNumber);
```

GetSources Method

Description:
This method gets the CAENRFIDLogicalSource objects available on the reader.

Return value:
An array of the logical source objects available in the Reader.

Syntax:

C# representation:

```
public CAENRFIDLogicalSource[]      GetSources()
```

JAVA representation:

```
public CAENRFIDLogicalSource[]      GetSources()
```

Remarks:
This function does not exist in C language, see § Overview on SDK pag. 7 for more informations.

InventoryAbort Method

For the description of this method, see § Event Handling pag.108.

RFControl Method

Description:
Permits to control the RF CW (Carrier Wave) signal generation.

Parameters:

Name	Description
OnOff	The value to set. 1 generates the CW , 0: stops the CW generation.

Syntax:

C# representation:

```
public void         RFControl(
                              int                 OnOff)
```

JAVA representation:

```
public void         RFControl(
                              int                 OnOff)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_RFControl(
                              CAENRFIDHandle      handle,
                              int                 OnOff);
```

Remarks
This method is only used for testing applications.

SetBitRate Method

Description:
This method sets the RF bit rate to use.

Parameters:

Name	Description
BitRate	The RF bit rate value to be set.

Syntax:

C# representation:

```
public void         SetBitRate(
                              CAENRFIDBitRate     BitRate)
```

JAVA representation:

```
public void         SetBitRate(
                              CAENRFIDBitRate     BitRate)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetBitRate(
                              CAENRFIDHandle      handle,
                              CAENRFID_Bitrate    BitRate);
```

SetDateTime Method

Description:
This method sets the Date/Time of the reader.

Parameters:

Name	Description
DateTime	The Date/Time to be set on the reader as a string in the format: "yyyy-mm-dd hh:mm:ss".

Syntax:

C# representation:

```
public void         SetDateTime(
                              string              DateTime)
```

JAVA representation:

```
public void         SetDateTime(
                              java.lang.String    DateTime)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetDateTime(
                              CAENRFIDHandle      handle,
                              char                *DateTime);
```

SetIO Method

Description:
This method sets the Output lines value.

Parameters:

Name	Description
IOValue	A bitmask representing the I/O lines value. The format and the meaning of the bits depends on the reader's model. Please refer to the corresponding user manual available on http://www.caen.it/rfid.

Syntax:

C# representation:

```
public void         SetIO(
                              int                 IOValue)
```

JAVA representation:

```
public void         SetIO(
                              int                 IOValue)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetIO(
                              CAENRFIDHandle      handle,
                              unsigned int        IOValue);
```

SetIODIRECTION Method

Description:
This method sets the current I/O direction setting as a bitmask. Each bit represent a I/O line, a value of 0 means that the line is configured as an input, 1 as an output. This setting has a meaning only for those readers with configurable I/O lines.

Parameters:

Name	Description
IODirection	The IODirection value to set.

Syntax:

C# representation:

```
public void         SetIODIRECTION(
                              int                 IODirection)
```

JAVA representation:

```
public void         SetIODIRECTION(
                              int                 IODirection)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetIODirection(
                              CAENRFIDHandle      handle,
                              unsigned int        IODirection);
```

SetLBTMode Method

Description:
This method sets/resets the LBT/FH mode.

Parameters:

Name	Description
LBTMode	The LBT/FH setting value to be set. 1 set LBT/FH Mode to ON, 0: resets LBT/FH Mode to OFF.

Syntax:

C# representation:

```
public void         SetLBTMode(
                              short               LBTMode)
```

JAVA representation:

```
public void         SetLBTMode(
                              short               LBTMode)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetLBTMode(
                              CAENRFIDHandle      handle,
                              unsigned short      LBTMode);
```

SetNetwork Method

Description:
This method permits to configure the network settings of the reader. In order to apply the changes the reader must be restarted.

Parameters:

Name	Description
IPAddress	The IP address to set on the reader network interface.
NetMask	The netmask to set on the reader network interface.
Gateway	The gateway to set on the reader network interface.

Syntax:

C# representation:

```
public void         SetNetwork(
                              string              IPAddress,
                              string              NetMask,
                              string              Gateway)
```

JAVA representation:

```
public void         SetNetwork(
                              java.lang.String    IPAddress,
                              java.lang.String    NetMask,
                              java.lang.String    Gateway)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetNetwork(
                              CAENRFIDHandle      handle,
                              char                *IPAddress,
                              char                *NetMask,
                              char                *Gateway);
```

SetPower Method

Description:
This method sets the conducted RF power of the Reader.

Parameters:

Name	Description
power	The conducted RF power value expressed in mW.

Syntax:

C# representation:

```
public void         SetPower(
                              int                 power)
```

JAVA representation:

```
public void         SetPower(
                              int                 power)
                                    throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetPower(
                              CAENRFIDHandle      handle,
                              unsigned int        Power);
```

SetProtocol Method

Description:
This method sets the air protocol of the reader.

Parameters:

Name	Description
Protocol	The CAENRFIDProtocol representing the air protocol to be set.

Syntax:

C# representation:

```
public void         SetProtocol(
                              CAENRFIDProtocol    Protocol)
```

JAVA representation:

```
public void         SetProtocol(
                              CAENRFIDProtocol    Protocol)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetProtocol(
                              CAENRFIDHandle      handle,
                              CAENRFIDProtocol    Protocol);
```

SetRFChannel Method

Description:
This method sets the RF channel to use. This method fixes the RF channel only when the listen before talk or the frequency hopping feature is disabled.

Parameters:

Name	Description
Channel	The RF channel index to be set.

Syntax:

C# representation:

```
public void         SetRFChannel(
                              short               Channel)
```

JAVA representation:

```
public void         SetRFChannel(
                              short               Channel)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetRFChannel(
                              CAENRFIDHandle      handle,
                              unsigned short      Channel);
```

Remarks
This method is only used for testing applications.

SetRFRegulation Method

Description:
This method sets the RF regulation setting value.

Parameters:

Name	Description
RFRegulation	The RF regulation value to set.

Syntax:

C# representation:

```
public void         SetRFRegulation(
                              CAENRFIDRFRegulations     RFRegulation)
```

JAVA representation:

```
public void         SetRFRegulation(
                              CAENRFIDRFRegulations     RFRegulation)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetRFRegulation(
                              CAENRFIDHandle            handle,
                              CAENRFIDRFRegulations     RFRegulation);
```

SetRS232 Method

Description:
This method permits to change the serial port settings. Valid settings values depend on the reader model.

Parameters:

Name	Description
baud	The baud rate value to set.
datab	The number of data bits to set.
stopb	The number of stop bits to set.
parity	The parity value to set.
flowc	The flow control value to set.

Syntax:

C# representation:

```
public void         SetRS232(
                              int                         baud,
                              int                         datab,
                              int                         stopb,
                              CAENRFIDRS232Constants      parity,
                              CAENRFIDRS232Constants      flowc)
```

JAVA representation:

```
public void         SetRS232(
                              int                         baud,
                              int                         datab,
                              int                         stopb,
                              CAENRFIDRS232Constants      parity,
                              CAENRFIDRS232Constants      flowc)
                              throws CAENRFIDException
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_SetRS232(
                              CAENRFIDHandle              handle,
                              unsigned long               baud,
                              unsigned long               datab,
                              unsigned long               stopb,
                              CAENRFID_RS232_Parity       parity,
                              CAENRFID_RS232_FlowControl  flowc);
```

CAENRFIDReaderInfo Class

The CAENRFIDReaderInfo class is used to create reader info objects. Reader info objects represent the information about the reader device (model and serial number).

GetModel Method

Description:
This method gets the reader's model.

Return value:
The reader's model.

Syntax:

C# representation:

```
public string                GetModel()
```

JAVA representation:

```
public java.lang.String      GetModel()
```

Remarks:
This method does not exist in C language. It is possible to use the *GetReaderInfo Method* pag. 93 instead. In fact *GetReaderInfo Method* (in the C language) returns the reader's model and the serial number.

GetSerialNumber Method

Description:
This method gets the reader's serial number.

Return value:
The reader's serial number.

Syntax:

C# representation:

```
public string                GetSerialNumber()
```

JAVA representation:

```
public java.lang.String      GetSerialNumber()
```

Remarks:
This method does not exist in C language. It is possible to use the *GetReaderInfo Method* pag. 93 instead. In fact *GetReaderInfo Method* (in the C language) returns the reader's model and the serial number.

CAENRFIDTag Class

The CAENRFIDTag class is used to define objects representing the tags. These objects are used as return values for the inventory methods and as arguments for many tag access methods.

In both Java and C# lanuguage this class is composed by methods while in C language the following struct is present (for more informations see § Overview on SDK pag.7):

C representation:

```
typedef struct {
                    byte             ID[MAX_ID_LENGTH];
                    short            Length;
                    char             LogicalSource[MAX_LOGICAL_SOURCE_NAME];
                    char             ReadPoint[MAX_READPOINT_NAME];
                    CAENRFIDProtocol Type;
                    short            RSSI;
                    byte             TID[MAX_TID_SIZE];
                    short            TIDLen;
                    } CAENRFIDTag;
```

GetId Method

Description:
This method returns the tag's ID (the EPC code in Gen2 tags).

Return value:
An array of bytes representing the tag's ID (the EPC code in EPC Class 1 Gen 2 tags).

Syntax:

C# representation:

```
public byte[]           GetId()
```

JAVA representation:

```
public byte[]           GetId()
```

GetLength Method

Description:
This method returns the tag's ID length.

Return value:
The tag's length.

Syntax:

C# representation:

```
public short            GetLength()
```

JAVA representation:

```
public short            GetLength()
```

GetReadPoint Method

Description:
This method returns the read point that has detected the tag.

Return value:
The name of the read point that has detected the Tag

Syntax:

C# representation:

```
public string                GetReadPoint()
```

JAVA representation:

```
public java.lang.String      GetReadPoint()
                             throws CAENRFIDException
```

GetRSSI Method

Description:
This method returns the RSSI value measured for the tag.

Return value:
The tag's RSSI.

Syntax:

C# representation:

```
public short                 GetRSSI()
```

JAVA representation:

```
public short                 GetRSSI()
```

GetSource Method

Description:
This method returns the name of the logical source that has detected the tag.

Return value:
The name of the logical source that has detected the tag.

Syntax:

C# representation:

```
public CAENRFIDLogicalSource        GetSource()
```

JAVA representation:

```
public CAENRFIDLogicalSource        GetSource()
```

GetTID Method

Description:
This method returns the tag's TID (valid only for EPC Class 1 Gen 2 tags).

Return value:
An array of bytes representing the tag's TID.

Syntax:

C# representation:

```
public byte[]            GetTID()
```

JAVA representation:

```
public byte[]            GetTID()
```

GetTimeStamp Method

Description:
This method gets the Tag's TimeStamp.

Return value:
The Tags's Unix TimeStamp.

Syntax:

C# representation:

```
public DateTime              GetTimeStamp()
```

JAVA representation:

```
public java.util.Date        GetTimeStamp()
```

GetType Method

Description:
This method returns the air protocol of the tag.

Return value:
The air protocol of the tag.

Syntax:

C# representation:

```
public new CAENRFIDProtocol          GetType()
```

JAVA representation:

```
public CAENRFIDProtocol              GetType()
```

4 Event Handling

This chapter gives a description of CAENRFID event handling. It contains these topics:

- Event Handling
- C# Event Handling
- JAVA Event Handling
- C Event Handling

Event Handling

Standard tag's detection method (InventoryTag) is based on a polling mechanism: a call to the InventoryTag method/function results in a single read cycle and the detected tags in that cycle are returned.

An useful variant ("continuous mode") uses an event mechanism to notify detected tags: a call to the EventInventoryTag method/function starts a continuous tags' detection algorithm (multiple read cycles) and an event is generated for each read cycle to notify the detected tags (see the CAEN RFID API User Manual.for further information).

The user of the library can define an event handler method/function that is called automatically when the event raises; the data related to the event is passed to the handler as a parameter.

The user can define the number of read cycles that the EventInventoryTag have to perform using the ReadCycle parameter of the relevant LogicalSource. If ReadCycle is equal to 0 the EventInventoryTag method loops indefinetely.

The "continuous mode" can be interrupted using the InventoryAbort method function.

The event handling is implemented using the standard event handling mechanism in .NET and Java while in C it is simulated using the callback mechanism.

For further information on the use of the EventInventoryTag, please refer to the CAEN RFID API User Manual.

EventInventoryTag Method

Description:
A call to this method will start a sequence of read cycle on each read point linked to the logical source. The readings will be notified to the controller via event generation.

Parameters:

Name	Description
Mask	A byte array representing the bitmask to apply.
MaskLength	A value representing the bit-oriented length of the bitmask.
Position	A value representing the first bit where the match will start.
Flag	A bitmask representing the InventoryTag options.
pCallBack	The user defined handler called by EventInventoryTag (only in C language).

Return value:
A boolean value that represents the status of the command: true if the reader has accepted the command; false otherwise.

Syntax:

C# representation:

```
public bool         EventInventoryTag(
                        byte[]                          Mask,
                        short                           MaskLength,
                        short                           Position,
                        short                           Flag)
```

JAVA representation:

```
public boolean      EventInventoryTag(
                        byte[]                          Mask,
                        short                           MaskLength,
                        short                           Position,
                        short                           Flag)
                        throws CAENRFIDException
```

C representation:

```
typedef struct {
                        char                            *SourceName;
                        char                            *Mask;
                        unsigned char                   MaskLength;
                        unsigned char                   Position;
                        CAENRFID_INVENTORY_CALLBACK     pCallBack;
                        short                           flag;}
                        CAENRFID_EventInventoryParams;

CAENRFIDErrorCodes  CAENRFID_EventInventoryTag (
                            CAENRFIDHandle              handle,
                            CAENRFID_EventInventoryParams    InvParams);
```

Remarks:
Depending on the air protocol setting it will execute the appropriate anticollision algorithm. This version of the method permits to specify a bitmask for filtering tag's populations as described by the EPC Class1 Gen2 (ISO18000-6C) air protocol. The filtering will be performend on the memory bank specified by bank parameter, starting at the bit indicated by the Position index and for a MaskLength length. The method will return only the tags that matches the given Mask. Passing a zero value for MaskLength it performs as the non-filtering InventoryTag method. The Flags parameter permits to set InventoryTag method's options.In this case bit 1 and 2 of the flag (continuos and framed mode) are ignored.

Flag value meaning	
Bit 0	RSSI: a 1 value indicates the reader will transmit the RSSI (Return Signal Strength Indicator) in the response.
Bit 1	Framed data: a 1 value indicates that the tag's data will be transmitted by the reader to the PC as soon as the tag is detected, a 0 value means that all the tags detected are buffered in the reader and trasmitted all together at the end of the inventory cycle.
Bit 2	Continuous acquisition: a 1 value indicates that the inventory cycle is repeated by the reader depending on the SetReadCycle setting value, a 0 value means that only one inventory cycle will be performed. If the continuous mode is selected a 0 value in the ReadCycle setting will instruct the reader to repeat the inventory cycle until an InventoryAbort method is invoked, a value X different from 0 means that the inventory cycle will be performed X times by the reader.
Bit 3	Compact data: a 1 value indicates that only the EPC of the tag will be returned by the reader, a 0 value indicates that the complete data will be returned. In case that the compact option is enabled all the other data will be populated by this library with fakes values.
Bit 4	TID reading: a 1 value indicates that also the TID of the tag will be returned by the reader together with the other informations.

InventoryAbort Method

Description:
This method stops the EventInventoryTag execution.

Syntax:

C# representation:

```
public void         InventoryAbort()
```

JAVA representation:

```
public void         InventoryAbort()
```

C representation:

```
CAENRFIDErrorCodes  CAENRFID_InventoryAbort(
                                CAENRFIDHandle      handle);
```

C# Event Handling

CAENRFIDEventArgs Class

The CAENRFIDEventArgs class defines the CAENRFID event arguments.

getData Method

Description:
This method returns the event object value.

Return value:
The value of the event object.

Syntax:

C# representation:

```
public CAENRFIDNotify[]        getData()
```

CAENRFIDEventHandler Delegate

CAENRFIDEventHandler delegate declaration.

Parameters:

Name	Description
Event	the Data Event.

Syntax:

C# representation:

```
public delegate void     CAENRFIDEventHandler(
                                              object              Sender,
                                              CAENRFIDEventArgs   Event)
```

CAENRFIDEvent Event

The CAEN RFID event is generated by the library each time tag data arrives from the reader. The event is generated only when the EventInventoryTag method is used. It is an event of the Reader Class.

Syntax:

C# representation:

```
public event CAENRFIDEventHandler          CAENRFIDEvent
```

Event Data

The event handler receives an argument of type CAENRFIDEventArgs containing data related to this event. The following CAENRFIDEventArgs property provides information specific to this event.

Property	Description
Data	Represents the event object value.

JAVA Event Handling

CAENRFIDEvent Class

The CAENRFIDEvent class defines the CAENRFID event arguments.

getData Method

Description:
This method returns the event object value.

Return value:
The value of the event object.

Syntax:

JAVA representation:

```
public java.util.ArrayList    getData()
```

CAENRFIDEventListener Interface

The listener interface for receiving CAEN RFID events.

CAENRFIDTagNotify

Description:
This method is invoked when an action occurs.

Parameters:

Name	Description
evt	The CAENRFIDEvent contains the Data Event.

Syntax:

JAVA representation:

```
void                    CAENRFIDTagNotify(
                                    CAENRFIDEvent       evt)
```

addCAENRFIDEventListener

This is a Reader Class method. It adds the specified CAENRFIDEvent listener to receive CAENRFIDEvent events from this CAENRIFDReader.

Parameters:

Name	Description
listener	listener - the CAENRFIDEvent listener.

Syntax:

JAVA representation:

```
public void         addCAENRFIDEventListener(
                            CAENRFIDEventListener     listener)
```

removeCAENRFIDEventListener

This is a Reader Class method. It Removes the specified CAENRFIDEvent listener so that it no longer receives CAENRFID events from this CAENRIFDReader.

Parameters:

Name	Description
listener	listener - the CAENRFIDEvent listener.

Syntax:

JAVA representation:

```
public void         removeCAENRFIDEventListener(
                            CAENRFIDEventListener     listener)
```

C Event Handling

CAENRFID_INVENTORY_CALLBACK

This function prototype defines the type of the user defined event handler (see the CAEN RFID API User Manual. for further information)

Syntax:

C representation:

```
typedef CAENRFIDErrorCodes (__stdcall *CAENRFID_INVENTORY_CALLBACK)
                           (const CAENRFIDNotify* Tags, const int Size);
```

5 Enumerations Description

This chapter gives a description of CAENRFID enumerations. It contains these topics:

- CAENRFIDBitRate Enumeration
- CAENRFIDLogicalSourceConstants Enumeration
- CAENRFIDPort Enumeration
- CAENRFIDProtocol Enumeration
- CAENRFIDReadPointStatus Enumeration
- CAENRFIDRFRegulations Enumeration
- CAENRFIDRS232Constants Enumeration
- CAENRFIDSelUnselOptions Enumeration

CAENRFIDBitRate Enumeration

The CAENRFIDBitRate Enumeration gives a list of the supported radiofrequency profiles.

Syntax:

C# representation:

```
public enum             CAENRFIDBitRate
```

JAVA representation:

```
public final class      CAENRFIDBitRate
```

C representation:

```
typedef enum            CAENRFID_Bitrate;
```

In the following table, the CAENRFIDBitRate Enumeration members are listed:

Member	Description
DSB_ASK_FM0_TX10RX40	DSB-ASK transmission modulation, FM0 return link encoding, 10 Kbit in transmission, 40 Kbit in reception.
DSB_ASK_FM0_TX40RX40	DSB-ASK transmission modulation, FM0 return link encoding, 40 Kbit in transmission, 40 Kbit in reception.
DSB_ASK_FM0_TX40RX160	DSB-ASK transmission modulation, FM0 return link encoding, 40 Kbit in transmission, 160 Kbit in reception.
DSB_ASK_FM0_TX160RX400	DSB-ASK transmission modulation, FM0 return link encoding, 160 Kbit in transmission, 400 Kbit in reception.
DSB_ASK_M2_TX40RX160	DSB-ASK transmission modulation, Miller (M=2) return link encoding, 40 Kbit in transmission, 160 Kbit in reception.
PR_ASK_M2_TX40RX250	PR-ASK transmission modulation, Miller (M=2) return link encoding, 40 Kbit in transmission, 250 Kbit in reception.
PR_ASK_M4_TX40RX250	PR-ASK transmission modulation, Miller (M=4) return link encoding, 40 Kbit in transmission, 250 Kbit in reception.
PR_ASK_M4_TX40RX300	PR-ASK transmission modulation, Miller (M=4) return link encoding, 40 Kbit in transmission, 300 Kbit in reception.

CAENRFIDLogicalSourceConstants Enumeration

The CAENRFIDLogicalSourceConstants Enumeration gives a list of constants used for the configuration of the logical sources. Detailed explanation of the settings can be found in the EPC Class 1 Gen 2 and ISO 18000-6B specification documents.

Syntax:

C# representation:

```
public enum             CAENRFIDLogicalSourceConstants
```

JAVA representation:

```
public final class      CAENRFIDLogicalSourceConstants
```

C representation:

```
typedef enum            CAENRFIDLogicalSourceConstants;
```

In the following table, the CAENRFIDLogicalSourceConstants Enumeration members are listed:

Member	Description
EPC_C1G2_SESSION_S0	Session 0 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_SESSION_S1	Session 1 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_SESSION_S2	Session 2 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_SESSION_S3	Session 3 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_TARGET_A	Target A is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_TARGET_B	Target B is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_SELECTED_YES	Only the tags with the SL flag set to true are considered in the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_SELECTED_NO	Only the tags with the SL flag set to false are considered in the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
EPC_C1G2_ALL_SELECTED	All the tags are considered in the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).
ISO18006B_DESB_ON	The Data Exchange Status Bit feature is used for the anticollision algorithm execution on the logical source (valid only for the ISO18000-6B air protocol).
ISO18006B_DESB_OFF	The Data Exchange Status Bit feature is not used for the anticollision algorithm execution on the logical source (valid only for the ISO18000-6B air protocol).

CAENRFIDPort Enumeration

The CAENRFIDPort Enumeration gives a list of the communication ports supported by the CAEN RFID readers.

Syntax:

C# representation:

```
public enum             CAENRFIDPort
```

JAVA representation:

```
public final class      CAENRFIDPort
```

C representation:

```
typedef enum            CAENRFIDPort;
```

Remarks:
In order to align the three libraries, the members name in C language have changed, now reporting the CAENRFID_ suffix, but the value of the members is the same of the previous library version.

In the following table, the CAENRFIDPort Enumeration members are listed:

Member	Description
CAENRFID_RS232	Serial port communication link.
CAENRFID_TCP	TCP/IP network communication link.
CAENRFID_USB	USB communication link.

CAENRFIDProtocol Enumeration

The CAENRFIDProtocol Enumeration gives a list of the air protocol supported by the CAEN RFID readers.

Syntax:

C# representation:

```
public enum             CAENRFIDProtocol
```

JAVA representation:

```
public final class      CAENRFIDProtocol
```

C representation:

```
typedef enum            CAENRFIDProtocol;
```

Remarks:
In order to align the three libraries, the members name in C language have changed, now reporting the CAENRFID_ suffix, but the value of the members is the same of the previous library version.

In the following table, the CAENRFIDProtocol Enumeration members are listed:

Member	Description
CAENRFID_ISO18000_6b	ISO18000-6B air protocol.
CAENRFID_EPC119	EPC 1.19 air protocol.
CAENRFID_EPC_C1G1	EPCGlobal Class1 Gen1 air protocol.
CAENRFID_ISO18000_6a	ISO18000-6A air protocol.
CAENRFID_EPC_C1G2	EPCGlobal Class1 Gen2 (aka ISO18000-6C) air protocol.
CAENRFID_MULTIPROTOCOL	This value permits to use all the supported air protocol at the same time. Suggested setting only for demo purposes.

CAENRFIDReadPointStatus Enumeration

The CAENRFIDReadPointStatus gives a list of the possible ReadPoint status values.

Syntax:

C# representation:

```
public enum              CAENRFIDReadPointStatus
```

JAVA representation:

```
public final class       CAENRFIDReadPointStatus
```

C representation:

```
typedef enum             CAENRFIDReadPointStatus;
```

Remarks:
In order to align the three libraries, the members name in C language have changed, now reporting the STATUS_ suffix, but the value of the members is the same of the previous library version.

In the following table, the CAENRFIDReadPointStatus Enumeration members are listed:

Member	Description
STATUS_BAD	Bad antenna connection.
STATUS_GOOD	Good antenna connection.
STATUS_POOR	Poor antenna connection.

CAENRFIDRFRegulations Enumeration

The CAENRFIDRFRegulations gives a list of country radiofrequency regulations.

Syntax:

C# representation:

```
public enum              CAENRFIDRFRegulations
```

JAVA representation:

```
public final class       CAENRFIDRFRegulations
```

C representation:

```
typedef enum             CAENRFIDRFRegulations;
```

Remarks:
In order to align the three libraries, the regulations, previously declared as #define, are now members of an enumeration, but the value of the members is the same of the previous library version.

In the following table, the CAENRFIDRFRegulations Enumeration members are listed:

Member	Description
ETSI_302208	ETSI_302208 radiofrequency regulation.
ETSI_300220	ETSI_300220 radiofrequency regulation.
FCC_US	FCC_US radiofrequency regulation.
MALAYSIA	MALAYSIA radiofrequency regulation.
JAPAN	JAPAN radiofrequency regulation.
KOREA	KOREA radiofrequency regulation.
AUSTRALIA	AUSTRALIA radiofrequency regulation.
CHINA	CHINA radiofrequency regulation.
TAIWAN	TAIWAN radiofrequency regulation.
SINGAPORE	SINGAPORE radiofrequency regulation.
BRAZIL	BRAZIL radiofrequency regulation.

CAENRFIDRS232Constants Enumeration

The CAENRFIDRS232Constants gives a list of settings for the serial port configuration.

Syntax:

C# representation:

```
public enum              CAENRFIDRS232Constants
```

JAVA representation:

```
public final class       CAENRFIDRS232Constants
```

C representation:

```
typedef enum             CAENRFID_RS232_Parity;
typedef enum             CAENRFID_RS232_FlowControl;
```

In the following table, the CAENRFIDRS232Constants Enumeration members are listed:

Member	Description
CAENRS232_Parity_None	No parity bit is sent at all.
CAENRS232_Parity_Odd	Odd parity.
CAENRS232_Parity_Even	Even parity.
CAENRFID_RS232_FlowControl_XonXoff	Software flow control.
CAENRFID_RS232_FlowControl_Hardware	Hardware flow control.
CAENRFID_RS232_FlowControl_None	No flow control.

CAENRFIDSelUnselOptions Enumeration

The CAENRFIDSelUnselOptions gives a list of operations supported by the Group Select/Unselect command (valid only for the ISO18000-6B air protocol).

Syntax:

C# representation:

```
public enum              CAENRFIDSelUnselOptions
```

JAVA representation:

```
public final class       CAENRFIDSelUnselOptions
```

C representation:

```
typedef enum             CAENRFID_SelUnsel_Op;
```

In the following table, the CAENRFIDSelUnselOptions Enumeration members are listed:

Member	Description
SEL_EQUAL	select equal to.
SEL_NOT_EQUAL	select not equal to.
SEL_GREATER_THAN	select greater than.
SEL_LOWER_THAN	select lower than.
UNS_EQUAL	unselect equal to.
UNS_NOT_EQUAL	unselect not equal to.
UNS_GREATER_THAN	unselect greater than.
UNS_LOWER_THAN	unselect lower than.

6 CAENRFID Obsolete Methods

This chapter gives a list of CAENRFID obsolete methods, functions, members and data types. It contains these topics:

- C# Obsolete Methods
- C# Obsolete Members
- JAVA Obsolete Methods
- C Obsolete Functions
- C Obsolete Data Types

Below it is available a list of obsolete methods, functions, members and data types for the three different program languages.
It is recommended not to use these methods since they will not be available in new reader's firmware release.
Some of these obsolete methods have been replaced by new ones as specified in the table below.

C# Obsolete Methods

Method	Description
Channel Class	
AddSource	This method is now obsolete.
AddTrigger	This method is now obsolete.
GetChannelStatus	This method is now obsolete.
GetChannelType	This method is now obsolete.
GetName	This method is now obsolete.
IsSourcePresent	This method is now obsolete.
IsTriggerPresent	This method is now obsolete.
RemoveSource	This method is now obsolete.
RemoveTrigger	This method is now obsolete.
LogicalSource Class	
AddTrigger	This method is now obsolete.
GetLostThreshold	This method is now obsolete.
GetObservedThreshold	This method is now obsolete.
Inventory	This method is now obsolete.
KillTag	This method is now obsolete.
LockTag	This method is now obsolete.
NXP_Calibrate	This method is now obsolete.
ProgramID	This method is now obsolete.
RemoveTrigger	This method is now obsolete.
SetLostThreshold	This method is now obsolete.
SetObservedThreshold	This method is now obsolete.
Reader Class	
ConnectRS232	This method is now obsolete.
CreateChannel	This method is now obsolete.
CreateTrigger	This method is now obsolete.
FWUpgradeTFTP	This method is now obsolete.
GetAllocatedChannels	This method is now obsolete.
GetAllocatedTriggers	This method is now obsolete.
GetChannelData	This method is now obsolete.
GetDESB	This method is now obsolete.
GetEventMode	This method is now obsolete.
RemoveChannel	This method is now obsolete.
RemoveTrigger	This method is now obsolete.
SetDESB	This method is now obsolete.
SetEventMode	This method is now obsolete.
SetReaderOPtions	This method is now obsolete.
Receiver Class	
KillServer	This method is now obsolete.
Trigger Class	
GetIOLineValue	This method is now obsolete.
GetName	This method is now obsolete.
GetTimerValue	This method is now obsolete.
IsLinkedToChannel	This method is now obsolete.
IsLinkedToSource	This method is now obsolete.

Tab. 6.1: C# Obsolete Methods

C# Obsolete Members

Member	Description
BitRate Enumeration	
TX10RX40	This member is now obsolete.
TX40RX40	This member is now obsolete.
TX40RX160	This member is now obsolete.
EventMode Enumeration	
READCYCLE_MODE	This member is now obsolete.
TIME_MODE	This member is now obsolete.
NOEVENT_MODE	This member is now obsolete.
TagEventType Enumeration	
TAG_GLIMPSED	This member is now obsolete.
TAG_LOST	This member is now obsolete.
TAG_OBSERVED	This member is now obsolete.
TAG_UNKNOWN	This member is now obsolete.

Tab. 6.2: C# Obsolete Members

JAVA Obsolete Methods

Method	Description
BitRate Class	
TX10RX40	This method is now obsolete.
TX40RX40	This method is now obsolete.
TX40RX160	This method is now obsolete.
Channel Class	
AddSource	This method is now obsolete.
AddTrigger	This method is now obsolete.
GetChannelStatus	This method is now obsolete.
GetChannelType	This method is now obsolete.
GetName	This method is now obsolete.
IsSourcePresent	This method is now obsolete.
IsTriggerPresent	This method is now obsolete.
RemoveSource	This method is now obsolete.
RemoveTrigger	This method is now obsolete.
Event Class	
Data	This method is now obsolete. Use getData instead.
EventMode Class	
READCYCLE_MODE	This method is now obsolete.
TIME_MODE	This method is now obsolete.
NOEVENT_MODE	This method is now obsolete.
LogicalSource Class	
AddTrigger	This method is now obsolete.
GetLostThreshold	This method is now obsolete.
GetObservedThreshold	This method is now obsolete.
Hitachi_GetSystemInfo	This method is now obsolete. Use Hitachi_GetSystemInformation instead.
Inventory	This method is now obsolete.
NXP_Calibrate	This method is now obsolete.
NXP_ChangeEAS (only non secure version)	This method is now obsolete.
NXP_EAS_Alarm (only secure version)	This method is now obsolete.
NXP_ResetReadProtect (only secure version)	This method is now obsolete.
RemoveTrigger	This method is now obsolete.
SetLostThreshold	This method is now obsolete.
SetObservedThreshold	This method is now obsolete.
Notify Class	
getAntenna	This method is now obsolete. Use getReadPoint instead.
Reader Class	
CreateChannel	This method is now obsolete.
CreateTrigger	This method is now obsolete.

Method	Description
FWUpgradeTFTP	This method is now obsolete.
GetAllocatedChannels	This method is now obsolete.
GetAllocatedTriggers	This method is now obsolete.
GetChannelData	This method is now obsolete.
GetEventMode	This method is now obsolete.
RemoveChannel	This method is now obsolete.
RemoveTrigger	This method is now obsolete.
SetEventMode	This method is now obsolete.
SetReaderOPtions	This method is now obsolete.
Receiver Class	
KillServer	This method is now obsolete.
TagEventType Class	
TAG_GLIMPSED	This method is now obsolete.
TAG_LOST	This method is now obsolete.
TAG_OBSERVED	This method is now obsolete.
TAG_UNKNOWN	This method is now obsolete.
Trigger Class	
GetIOLineValue	This method is now obsolete.
GetName	This method is now obsolete.
GetTimerValue	This method is now obsolete.
IsLinkedToChannel	This method is now obsolete.
IsLinkedToSource	This method is now obsolete.
HitachiSysInfo	
GetBankLock	This method is now obsolete.
GetBlockReadLock	This method is now obsolete.
GetBlockReadWriteLock	This method is now obsolete.
GetBlockWriteLock	This method is now obsolete.
GetInfoFlag	This method is now obsolete.
getReserved	This method is now obsolete.
getSetAttenuateLevel	This method is now obsolete.
getTID	This method is now obsolete.
getUII	This method is now obsolete.
getUser	This method is now obsolete.

Tab. 6.3: JAVA Obsolete Methods

C Obsolete Functions

Function	Description
AddNotifyTrigger	This function is now obsolete.
AddReadTrigger	This function is now obsolete.
AddSourceToChannel	This function is now obsolete.
AllocateChannel	This function is now obsolete.
AllocateTrigger	This function is now obsolete.
CustomCmd_C1G2	This function is now obsolete. Use CustomCommand_EPC_C1G2 instead.
DeallocateChannel	This function is now obsolete.
DeallocateTrigger	This function is now obsolete.
ExtendedInventoryTag	This function is now obsolete.
FirmwareUpgrade	This function is now obsolete.
FreeNotifyMemory	This function is now obsolete.
GetAllocatedChannels	This function is now obsolete.
GetAllocatedTriggers	This function is now obsolete.
GetChannelData	This function is now obsolete.
GetChannelInTrigger	This function is now obsolete.
GetChannelStatus	This function is now obsolete.
GetDE_SB	This function is now obsolete. Use GetDESB_ISO180006B instead.
GetEventMode	This function is now obsolete.
GetFWRelease	This function is now obsolete. Use GetFirmwareRelease instead.
GetModulation	This function is now obsolete.
GetNotification	This function is now obsolete.
GetQ_C1G2	This function is now obsolete. Use GetQValue_EPC_C1G2 instead.
GetQ_EPC_C1G2	This function is now obsolete. Use GetQValue_EPC_C1G2 instead.
GetReadPointInSource	This function is now obsolete. Use isReadPointPresent instead.
GetSourceConfiguration	This function is now obsolete.
GetSourceInChannel	This function is now obsolete.
GetSourceInTrigger	This function is now obsolete.
GetSWRelease	This function is now obsolete.
GetTriggerInChannel	This function is now obsolete.
Inventory	This function is now obsolete.
KillTag	This function is now obsolete. Use KillTag_EPC_C1G1 instead.
KillTag_C1G2	This function is now obsolete. Use KillTag_EPC_C1G2 instead.
Lock	This function is now obsolete. Use LockTag_ISO180006B instead.
Lock_C1G2	This function is now obsolete. Use LockTag_EPC_C1G2 instead.
NXP_Calibrate	This function is now obsolete.
NXP_ChangeEAS	This function is now obsolete.
NXP_SecureCalibrate	This function is now obsolete.
NXP_SecureEAS_Alarm	This function is now obsolete.
NXP_SecureResetReadProtect	This function is now obsolete.
ProgramID	This function is now obsolete. Use ProgramID_EPC_C1G1 instead.
ProgramID_C1G2	This function is now obsolete. Use ProgramID_EPC_C1G2 instead.
QueryAck_C1G2	This function is now obsolete. Use QueryAck_EPC_C1G2 instead.
QueryTag_C1G2	This function is now obsolete. Use Query_EPC_C1G2 instead.
Read	This function is now obsolete. Use ReadTagData instead.
Read_C1G2	This function is now obsolete. Use ReadTagData_EPC_C1G2 instead.
RemoveNotifyTrigger	This function is now obsolete.
RemoveReadTrigger	This function is now obsolete.
RemoveSourceFromChannel	This function is now obsolete.
SecureCustomCmd_C1G2	This function is now obsolete.

Function	Description
	Use SecureCustomCommand_EPC_C1G2 instead.
SecureLock_C1G2	This function is now obsolete. Use SecureLockTag_EPC_C1G2 instead.
SecureProgramID_C1G2	This function is now obsolete. Use SecureProgramID_EPC_C1G2 instead.
SecureRead_C1G2	This function is now obsolete. Use SecureReadTagData_EPC_C1G2 instead.
SecureWrite_C1G2	This function is now obsolete. Use SecureWriteTagData_EPC_C1G2 instead.
SetBitrate	This function is now obsolete. Use CAENRFID_SetBitRate instead.
SetDE_SB	This function is now obsolete. Use SetDESB_ISO180006B instead.
SetEventMode	This function is now obsolete.
SetModulation	This function is now obsolete.
SetQ_C1G2	This function is now obsolete. Use SetQValue_EPC_C1G2 instead.
SetQ_EPC_C1G2	This function is now obsolete. Use SetQValue_EPC_C1G2 instead.
SetSourceConfiguration	This function is now obsolete.
Write	This function is now obsolete. Use WriteTagData instead.
Write_C1G2	This function is now obsolete. Use WriteTagData_EPC_C1G2 instead.

Tab. 6.4: C Obsolete Functions

C Obsolete Data Types

Data Type	Description
ETSI302208 ETSI300220 FCCUS MALAYSIA JAPAN KOREA AUSTRALIA CHINA TAIWAN SINGAPORE	This data type is now obsolete. Use enum CAENRFIDRFRegulations instead.
CAENRFID_SOURCE_Parameter	
CONFIG_READCYCLE	This data type is now obsolete. Use Get/SetReadCycle Method instead.
CONFIG_OBSERVEDTHRESHOLD	This data type is now obsolete.
CONFIG_LOSTTHRESHOLD	This data type is now obsolete.
CONFIG_G2_Q_VALUE	This data type is now obsolete. Use Get/SetQ_EPC_C1G2 Method instead.
CONFIG_G2_SESSION	This data type is now obsolete. Use Get/SetSession_EPC_C1G2 Method instead.
CONFIG_G2_TARGET	This data type is now obsolete. Use Get/SetTarget_EPC_C1G2 Method instead.
CONFIG_G2_SELECTED	This data type is now obsolete. Use Get/SetSelected_EPC_C1G2 Method instead.
CONFIG_ISO18006B_DESB	This data type is now obsolete. Use Get/SetDESB_ISO180006B Method instead.
CAENRFID_EventMode	
READCYCLE_MODE	This data type is now obsolete.
TIME_MODE	This data type is now obsolete.
NOEVENT_MODE	This data type is now obsolete.
CAENRFID_FWUpgradeType	
RFID_TFTP	This data type is now obsolete.
CAENRFID_ExtendedInventoryParams	This data type is now obsolete.

Tab. 6.5: C Obsolete Data Types