Visit [SDK- Software Development Kit](#) Web Page and you will find the latest revision of manuals and software. All you need to start using your SDK software in a few clicks!

**Scope of Manual**

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

**Change Document Record**

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>29 Jun 2010</td>
<td>01</td>
<td>Initial release</td>
<td>-</td>
</tr>
<tr>
<td>14 Jan 2011</td>
<td>02</td>
<td>Corrected GetTimeStamp Method’s return value</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Federal Communications Commission (FCC) Notice</td>
<td>4</td>
</tr>
<tr>
<td>22 Mar 2011</td>
<td>03</td>
<td>Added R1260U Reader in the declaration of Federal Communications</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Commission (FCC) note</td>
<td></td>
</tr>
<tr>
<td>06 Sep 2011</td>
<td>04</td>
<td>Added XPC field information</td>
<td>13, 14, 31, 34, 74, 77, 74, 95, 98, 101</td>
</tr>
<tr>
<td>08 Aug 2012</td>
<td>05</td>
<td>Added R4300P reader in the Federal Communications Commission (FCC)</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Notice (Preliminary)</td>
<td></td>
</tr>
<tr>
<td>29 Nov 2012</td>
<td>06</td>
<td>Added bit 5 (event trigger), bit 7 (match tag) and bit 8 (PC) in the</td>
<td>31, 34, 101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tables Flag value meaning</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed bit 1 and bit 2 description in the table Flag value meaning of</td>
<td>31, 34, 101</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the InventoryTag Method and EventInventoryTag Method</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added the following members in the CAENRFIDBitRate Enumeration:</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSB_ASK_M4_TX40RX256, PR_ASK_FMO_TX40RX640,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PR_ASK_M4_TX40RX256, PR_ASK_M4_TX40RX320,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PR_ASK_M4_TX80RX320</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added NXP_ChangeConfig Method in the LogicalSource Class</td>
<td>10, 45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Method, SL900A_GetMeasurementSetup Method, SL900A_StartLog Mode Method,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SL900A_StartLog Method in the LogicalSource Class</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added methods representation in Android language</td>
<td>15+121</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed the following methods: Hitachi_BlockLock,</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hitachi_BlockReadLock, Hitachi_GetSystemInformation,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hitachi_ReadLock, Hitachi_SetAttenuate,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hitachi_WriteMultipleWords. See § CAENRFID OBSOLETE METHODS chapter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Removed the following methods: Fujitsu_BurstErase, Fujitsu_BurstWrite,</td>
<td>115</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fujitsu_ChgBlockGroupPassword, Fujitsu_ChgBlockLock, Fujitsu_ChgWordLock,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fujitsu_ReadBlockLock, Fujitsu_Refresh. See § CAENRFID OBSOLETE METHODS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>chapter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added CAENRFIDLogicalSource.InventoryFlag Enumeration</td>
<td>14, 108</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added overloaded Connect Method.</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added IDSTagData Class</td>
<td>10, 14, 15+16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added PC field information</td>
<td>10, 31, 34, 74, 96, 101</td>
</tr>
<tr>
<td>19 Apr 2013</td>
<td>07</td>
<td>Added PC field in C representation of CAENRFIDTag Class</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added AS288 Reader in the declaration of Federal Communications</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Commission (FCC) note</td>
<td></td>
</tr>
<tr>
<td>11 Sep 2013</td>
<td>08</td>
<td>Added CAENRFIDRFRegulations Enumeration</td>
<td>14, 111</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added CAENRFIDTag.MemBanks Enumeration</td>
<td>14, 114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added information about SDK Web Page</td>
<td>2</td>
</tr>
<tr>
<td>03 Jul 2014</td>
<td>09</td>
<td>Added R1270 Quark Up and R1170I qIDmini Reader in the declaration</td>
<td>4</td>
</tr>
</tbody>
</table>
### Changes

<table>
<thead>
<tr>
<th>Date</th>
<th>Added/Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 Jun 2015</td>
<td>R1250i Tile in the declaration of Federal Communications Commission (FCC) note</td>
</tr>
<tr>
<td>10 Sep 2015</td>
<td>Added R4301P Ion in the declaration of Federal Communications Commission (FCC) note</td>
</tr>
<tr>
<td>27 Jul 2015</td>
<td>Added overloaded MatchReadPointImpedance Method in the CAEN RFID Reader Class</td>
</tr>
<tr>
<td>87</td>
<td>Added PrintScreen Method in the CAEN RFID Reader Class</td>
</tr>
<tr>
<td>111</td>
<td>Added Peru and South Africa radiofrequency regulation in the CAENRFIDFRFRegulations Enumeration</td>
</tr>
<tr>
<td>78</td>
<td>Modified Address parameter description in Connect Method and Init Function</td>
</tr>
<tr>
<td>14 Nov 2016</td>
<td>Added CHILE radiofrequency regulation in the CAENRFIDFRFRegulations Enumeration</td>
</tr>
<tr>
<td>111</td>
<td>Added SL900A_GetMeasurementSetup Method and SL900A_SetLogLimits Method in the CAENRFIDLogicalSource Class</td>
</tr>
<tr>
<td>78</td>
<td>Added Connect Method in the CAENRFIDReader Class</td>
</tr>
<tr>
<td>106</td>
<td>Added DSB_ASK_M8_TX40RX256 member in the CAENRFIDBitRate Enumeration</td>
</tr>
<tr>
<td>80</td>
<td>Added GetBatteryLevel Method in the CAENRFIDReader Class</td>
</tr>
<tr>
<td>23</td>
<td>Modified GetBufferedData Method description.</td>
</tr>
<tr>
<td>23</td>
<td>Added overloaded GetBufferedData Method in the CAENRFIDLogicalSource Class</td>
</tr>
<tr>
<td>18, 20, 24, 25, 56, 57, 68</td>
<td>Added new methods in the CAENRFIDLogicalSource Class: Authenticate_EPC_C1G2 Method, ClearBuffer Method, GetBufferSize Method, GetInventoryCounts Method, GetInventoryDwellTime Method, GetInventoryQuietTime Method, SetInventoryCounts Method, SetInventoryDwellTime Method, SetInventoryQuietTime Method, Untraceable_EPC_C1G2 Method</td>
</tr>
<tr>
<td>102</td>
<td>Added ForceAbort Method in Event Handling chapter</td>
</tr>
<tr>
<td>115</td>
<td>Removed the GetLBTMode method. See § CAENRFID OBSOLETE METHODS chapter</td>
</tr>
</tbody>
</table>

---

**CAEN RFID srl**

Via Vetraia, 11 55049 Viareggio (LU) - ITALY

Tel. +39.0584.388.398 Fax +39.0584.388.959

[info@caenrfid.com](mailto:info@caenrfid.com)

[www.caenrfid.com](http://www.caenrfid.com)

---

**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.com](http://www.caenrfid.com).
Federal Communications Commission (FCC) Notice

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, R4300P, A528B, R12401, R1270, R1170I (Mod. WR1170IUAPLP and WR1170IUHIDP), R1250I (Mod. WR1250IUXAAA, WR1250IUXAFI, WR1250IUXBAA, WR1250IUXBFL), R4301P (Mod. WR4301PXAAAA, WR4301PXGPRS and WR4301PXWIFI).
Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope of Manual</td>
<td>2</td>
</tr>
<tr>
<td>Change Document Record</td>
<td>2</td>
</tr>
<tr>
<td>Index</td>
<td>5</td>
</tr>
<tr>
<td>List of Tables</td>
<td>7</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>8</td>
</tr>
<tr>
<td>Overview on SDK</td>
<td>8</td>
</tr>
<tr>
<td>Functions and methods names</td>
<td>8</td>
</tr>
<tr>
<td>Error Handling</td>
<td>8</td>
</tr>
<tr>
<td>Managing connections with the readers</td>
<td>9</td>
</tr>
<tr>
<td>Return data mechanism</td>
<td>9</td>
</tr>
<tr>
<td>Passing parameters to methods and functions</td>
<td>9</td>
</tr>
<tr>
<td>2 CAEN RFID API STRUCTURE</td>
<td>10</td>
</tr>
<tr>
<td>CAENRFID Classes</td>
<td>10</td>
</tr>
<tr>
<td>CAENRFID Enumerations</td>
<td>14</td>
</tr>
<tr>
<td>3 CLASSES DESCRIPTION</td>
<td>15</td>
</tr>
<tr>
<td>CAENRFIDException Class</td>
<td>15</td>
</tr>
<tr>
<td>getError Method</td>
<td>15</td>
</tr>
<tr>
<td>IDSTagData Class</td>
<td>15</td>
</tr>
<tr>
<td>getADError Method</td>
<td>15</td>
</tr>
<tr>
<td>getRangeLimit Method</td>
<td>16</td>
</tr>
<tr>
<td>getSensorValue Method</td>
<td>16</td>
</tr>
<tr>
<td>CAENRFIDLogicalSource Class</td>
<td>17</td>
</tr>
<tr>
<td>AddReadPoint Method</td>
<td>17</td>
</tr>
<tr>
<td>Authenticate_EPC_C1G2 Method</td>
<td>18</td>
</tr>
<tr>
<td>BlockWriteTagData Method</td>
<td>19</td>
</tr>
<tr>
<td>ClearBuffer Method</td>
<td>20</td>
</tr>
<tr>
<td>CustomCommand_EPC_C1G2 Method</td>
<td>21</td>
</tr>
<tr>
<td>EventInventoryTag Method</td>
<td>22</td>
</tr>
<tr>
<td>GetBufferedData Method</td>
<td>23</td>
</tr>
<tr>
<td>GetBufferSize Method</td>
<td>24</td>
</tr>
<tr>
<td>GetDESB_ISO180006B Method</td>
<td>24</td>
</tr>
<tr>
<td>GetInventoryCounts Method</td>
<td>24</td>
</tr>
<tr>
<td>GetInventoryDwellTime Method</td>
<td>25</td>
</tr>
<tr>
<td>GetInventoryQuietTime Method</td>
<td>25</td>
</tr>
<tr>
<td>GetName Method</td>
<td>26</td>
</tr>
<tr>
<td>GetQ_EPC_C1G2 Method</td>
<td>26</td>
</tr>
<tr>
<td>GetReadCycle Method</td>
<td>26</td>
</tr>
<tr>
<td>GetSelected_EPC_C1G2 Method</td>
<td>27</td>
</tr>
<tr>
<td>GetSession_EPC_C1G2 Method</td>
<td>27</td>
</tr>
<tr>
<td>GetTarget_EPC_C1G2 Method</td>
<td>28</td>
</tr>
<tr>
<td>GroupSelUnsel Method</td>
<td>28</td>
</tr>
<tr>
<td>InventoryTag Method</td>
<td>29</td>
</tr>
<tr>
<td>isReadPointPresent Method</td>
<td>35</td>
</tr>
<tr>
<td>KillTag_EPC_C1G1 Method</td>
<td>35</td>
</tr>
<tr>
<td>KillTag_EPC_C1G2 Method</td>
<td>36</td>
</tr>
<tr>
<td>LockBlockPermaLock_EPC_C1G2 Method</td>
<td>37</td>
</tr>
<tr>
<td>LockTag_EPC_C1G2 Method</td>
<td>38</td>
</tr>
<tr>
<td>LockTag_ISO180006B Method</td>
<td>41</td>
</tr>
<tr>
<td>NXP_ChangeEAS Method</td>
<td>41</td>
</tr>
<tr>
<td>NXP_ChangeConfig Method</td>
<td>42</td>
</tr>
<tr>
<td>NXP_EAS_Alarm Method</td>
<td>43</td>
</tr>
<tr>
<td>NXP_ReadProtect Method</td>
<td>43</td>
</tr>
<tr>
<td>NXP_ResetReadProtect Method</td>
<td>44</td>
</tr>
<tr>
<td>NXP_ChangeConfig Method</td>
<td>45</td>
</tr>
<tr>
<td>ProgramID_EPC_C1G1 Method</td>
<td>46</td>
</tr>
<tr>
<td>ProgramID_EPC_C1G2 Method</td>
<td>46</td>
</tr>
<tr>
<td>ProgramID_EPC119 Method</td>
<td>47</td>
</tr>
<tr>
<td>Query_EPC_C1G2 Method</td>
<td>48</td>
</tr>
<tr>
<td>QueryAck_EPC_C1G2 Method</td>
<td>48</td>
</tr>
<tr>
<td>ReadBlockPermaLock_EPC_C1G2 Method</td>
<td>49</td>
</tr>
<tr>
<td>ReadTagData Method</td>
<td>50</td>
</tr>
</tbody>
</table>
SetPower Method ..................................................................................................................... 91
SetProtocol Method ................................................................................................................ 92
SetRFChannel Method ............................................................................................................ 92
SetRS232 Method .................................................................................................................... 93
CAENRFIDReaderInfo Class .................................................................................................. 94
GetModel Method .................................................................................................................. 94
GetSerialNumber Method ...................................................................................................... 94
CAENRFIDTag Class .............................................................................................................. 95
GetId Method .......................................................................................................................... 95
GetLength Method ................................................................................................................ 95
GetPC Method ....................................................................................................................... 96
GetReadPoint Method ........................................................................................................... 96
GetRSSI Method .................................................................................................................... 96
GetSource Method ................................................................................................................ 97
GetTID Method ..................................................................................................................... 97
GetTimeStamp Method ......................................................................................................... 97
GetType Method ..................................................................................................................... 97
GetXPC Method ..................................................................................................................... 98

4 EVENT HANDLING .............................................................................................................. 99
Event Handling ....................................................................................................................... 99
EventInventoryTag Method .................................................................................................. 100
InventoryAbort Method ........................................................................................................ 101
ForceAbort Method .............................................................................................................. 102
C# Event Handling ................................................................................................................ 103
CAENRFIDEventArgs Class .................................................................................................. 103
CAENRFIDEventDelegate ...................................................................................................... 103
CAENRFIDEvent ................................................. ................................................................. 103
Java and Android Event Handling ......................................................................................... 104
CAENRFIDEvent Class .......................................................................................................... 104
CAENRFIDEventListener Interface ....................................................................................... 104
addCAENRFIDEventListener ............................................................................................... 104
removeCAENRFIDEventListener ........................................................................................ 104
C Event Handling ................................................................................................................. 105
CAENRFID_INVENTORY_CALLBACK ................................................................................ 105

5 ENUMERATIONS DESCRIPTION ....................................................................................... 106
CAENRFIDBitRate Enumeration ............................................................................................ 106
CAENRFIDLogicalSourceConstants Enumeration ........................................................... 107
CAENRFIDLogicalSource.InventoryFlag Enumeration ..................................................... 108
CAENRFIDPort Enumeration ............................................................................................... 109
CAENRFIDProtocol Enumeration ......................................................................................... 109
CAENRFIDReadPointStatus Enumeration ........................................................................ 110
CAENRFIDRFRegulations Enumeration .......................................................................... 111
CAENRFIDRS232Constants Enumeration .......................................................................... 112
CAENRFIDSelUnselOptions Enumeration ........................................................................ 113
CAENRFIDTag.MemBanks Enumeration ......................................................................... 114

6 CAENRFID OBSOLETE METHODS ................................................................................. 115
C# Obsolete Methods ........................................................................................................... 115
C# Obsolete Members ........................................................................................................... 116
Java and Android Obsolete Methods .................................................................................. 116
C Obsolete Functions ........................................................................................................... 119
C Obsolete Data Types ........................................................................................................ 121

List of Tables

Tab. 2.1: CAENRFID classes ................................................................................................. 10
Tab. 2.2: CAENRFID methods ............................................................................................ 14
Tab. 2.3: CAENRFID Enumerations .................................................................................... 14
Tab. 6.1: C# Obsolete Methods ........................................................................................ 116
Tab. 6.2: C# Obsolete Members ......................................................................................... 116
Tab. 6.3: Java and Android Obsolete Methods .................................................................. 118
Tab. 6.4: C Obsolete Functions .......................................................................................... 120
Tab. 6.5: C Obsolete Data Types ......................................................................................... 121
1 INTRODUCTION

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 page 115).

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 a "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 CAENRFID LogicalSource 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.
CAENRFID Classes

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

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAENRFIDEventArgs</td>
<td>This class defines the CAENRFID event arguments.</td>
</tr>
<tr>
<td>CAENRFIDException</td>
<td>This class defines the CAEN RFID exceptions.</td>
</tr>
<tr>
<td>IDSTagData</td>
<td>This class represents data returned by tags based on IDS Chip SL900A.</td>
</tr>
<tr>
<td>CAENRFIDLogicalSource</td>
<td>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.</td>
</tr>
<tr>
<td>CAENRFIDNotify</td>
<td>This class defines the structure of a notification message.</td>
</tr>
<tr>
<td>CAENRFIDReader</td>
<td>The CAENRFIDReader class is used to create reader objects which permit to access to CAEN RFID readers’ configuration and control commands.</td>
</tr>
<tr>
<td>CAENRFIDReaderInfo</td>
<td>The CAENRFIDReaderInfo class is used to create reader info objects. Reader info objects represent the information about the reader device (model and serial number).</td>
</tr>
<tr>
<td>CAENRFIDTag</td>
<td>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.</td>
</tr>
</tbody>
</table>

Tab. 2.1: CAENRFID classes

Each class contains the following methods:

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAENRFIDEventArgs Class</td>
<td></td>
</tr>
<tr>
<td>getData</td>
<td>Returns the event object value.</td>
</tr>
<tr>
<td>CAENRFIDException Class</td>
<td></td>
</tr>
<tr>
<td>getError</td>
<td>Gets the error string associated to the exception.</td>
</tr>
<tr>
<td>CAENRFID IDSTagData Class</td>
<td></td>
</tr>
<tr>
<td>getA DError</td>
<td>Gets the error status of the A/D.</td>
</tr>
<tr>
<td>getRangeLimit</td>
<td>Gets the range limit parameter.</td>
</tr>
<tr>
<td>getSensorValue</td>
<td>Gets the value obtained by the sensor.</td>
</tr>
<tr>
<td>CAENRFIDLogicalSource Class</td>
<td></td>
</tr>
<tr>
<td>AddReadPoint</td>
<td>Adds a read point to the logical source.</td>
</tr>
<tr>
<td>Authenticate_EPC_C1G2</td>
<td>This method allows an interrogator to perform tag, interrogator or mutual authentication. The generic nature of the authenticate command allows it to support a variety of cryptographic suites. The number of authenticate commands required to implement an authentication depends on the authentication type and on the chosen cryptographic suite.</td>
</tr>
<tr>
<td>BlockWriteTagData</td>
<td>Overloaded. This method can be used to write a portion of the user memory in an ISO18000-6B tag using blocks of four bytes for each command.</td>
</tr>
<tr>
<td>ClearBuffer</td>
<td>This method deletes all items stored in the internal buffer.</td>
</tr>
<tr>
<td>CustomCommand_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>EventInventoryTag</td>
<td>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</td>
</tr>
</tbody>
</table>

7 For the description of this class, see § EVENT HANDLING page 95.
<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBufferedData</td>
<td>Overloaded. The function returns all the Tags stored in reader’s memory using all the ReadPoints belonging to the Source.</td>
</tr>
<tr>
<td>GetBufferSize</td>
<td>This method gets the current number of records (tags) stored in the reader’s internal buffer.</td>
</tr>
<tr>
<td>GetDESBI_SO180006B</td>
<td>This method can be used to retrieve the Data Exchange Status Bit setting (see ISO18000-68 protocol specification) by the anticollision algorithm when called on this logical source.</td>
</tr>
<tr>
<td>GetInventoryCounts</td>
<td>This method can be used to get the current setting for the number of inventory counts performed by the logical source after pressing the TRIGGER during the inventory algorithm execution.</td>
</tr>
<tr>
<td>GetInventoryDwellTime</td>
<td>This method can be used to get the inventory execution time (msec) used by the logical source during the inventory algorithm execution.</td>
</tr>
<tr>
<td>GetInventoryQuietTime</td>
<td>This method can be used to get the inventory quiet time (msec) used by the logical source during the inventory algorithm execution.</td>
</tr>
<tr>
<td>GetName</td>
<td>Gets a string representing the name of the logical source.</td>
</tr>
<tr>
<td>GetQ_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>GetReadCycle</td>
<td>Gets the current setting for the number of read cycles performed by the logical source during the inventory algorithm execution.</td>
</tr>
<tr>
<td>GetSelected_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>GetSession_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>GetTarget_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>GroupSelUnsel</td>
<td>This method can be used to send a Group Select/Unselect command to the tag (see ISO18000-68 protocol specification).</td>
</tr>
<tr>
<td>IsReadPointPresent</td>
<td>Checks if a read point is present in the logical source.</td>
</tr>
<tr>
<td>KillTag_EPC_C1G1</td>
<td>This method can be used to kill an EPC Class 1 Gen 1 tag.</td>
</tr>
<tr>
<td>KillTag_EPC_C1G2</td>
<td>Overloaded. This method can be used to kill an EPC of an EPC Class 1 Gen 2 tag.</td>
</tr>
<tr>
<td>LockBlockPermaLock_EPC_C1G2</td>
<td>This method implements the BlockPermaLock with ReadLock=1 as specified in EPCC1G2 rev. 1.2.0 protocol.</td>
</tr>
<tr>
<td>LockTag_EPC_C1G2</td>
<td>Overloaded. This method can be used to lock a memory bank of an EPC Class 1 Gen 2 tag.</td>
</tr>
<tr>
<td>LockTag_ISO180006B</td>
<td>This method can be used to lock a byte in the memory of a ISO18000-68 tag.</td>
</tr>
<tr>
<td>NXP_ChangeEAS</td>
<td>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.</td>
</tr>
<tr>
<td>NXP_ChangeConfig</td>
<td>Overloaded. This method can be used to issue a NXP_ChangeConfig custom command as defined in the NXP UCODE G2/M and G2/M+ datasheet.</td>
</tr>
<tr>
<td>NXP_EAS_Alarm</td>
<td>This method can be used to issue an EAS_Alarm custom command as defined by the NXP G2XM and G2XL datasheet.</td>
</tr>
<tr>
<td>NXP_ReaderProtect</td>
<td>Overloaded. This method can be used to issue a ReaderProtect custom command as defined by the NXP G2XM and G2XL datasheet.</td>
</tr>
<tr>
<td>NXP_ResetReaderProtect</td>
<td>This method can be used to issue a ResetReaderProtect custom command as defined by the NXP G2XM and G2XL datasheet.</td>
</tr>
<tr>
<td>ProgramID_EPC_C1G1</td>
<td>This method can be used to write the EPC of an EPC Class 1 Gen 1 tag.</td>
</tr>
<tr>
<td>ProgramID_EPC_C1G2</td>
<td>Overloaded. This method can be used to write the EPC of an EPC Class 1 Gen 2 tag.</td>
</tr>
<tr>
<td>ProgramID_EPC119</td>
<td>This method can be used to write the UID of an EPC 1.19 tag.</td>
</tr>
<tr>
<td>Query_EPC_C1G2</td>
<td>This method makes the reader generate an EPC Class1 Gen2 Query command. 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.</td>
</tr>
<tr>
<td>QueryAck_EPC_C1G2</td>
<td>This method makes 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.</td>
</tr>
</tbody>
</table>
### Methods Description

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadBlockPermalock_EPC_C1G2</td>
<td>This method implements the BLockPermaLock with ReadLock=0 as specified in EPC1G2 rev. 1.2.0 protocol.</td>
</tr>
<tr>
<td>ReadTagData</td>
<td>This method can be used to read a portion of the user memory in a ISO18000-6B tag.</td>
</tr>
<tr>
<td>ReadTagData_EPC_C1G2</td>
<td>Overloaded. This method can be used to read a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag.</td>
</tr>
<tr>
<td>RemoveReadPoint</td>
<td>Removes a read point from the logical source.</td>
</tr>
<tr>
<td>ResetSession_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>SetDESB_ISO180006B</td>
<td>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.</td>
</tr>
<tr>
<td>SetInventoryCounts</td>
<td>This method can be used to set the current setting for the number of inventory counts performed by the logical source after pressing the TRIGGER during the inventory algorithm execution.</td>
</tr>
<tr>
<td>SetInventoryDwellTime</td>
<td>This method can be used to set the inventory execution time (msec) used by the logical source during the inventory algorithm execution.</td>
</tr>
<tr>
<td>SetInventoryQuietTime</td>
<td>This method can be used to set the inventory quiet time (msec) used by the logical source during the inventory algorithm execution.</td>
</tr>
<tr>
<td>SetQ_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>SetReadCycle</td>
<td>Sets the number of read cycles to be performed by the logical source during the inventory algorithm execution.</td>
</tr>
<tr>
<td>SetSelected_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>SetSession_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>SetTarget_EPC_C1G2</td>
<td>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.</td>
</tr>
<tr>
<td>SL900A_EndLog</td>
<td>This method can be used to issue an IDS SL900A EndLog custom command as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>SL900A_GetLogState</td>
<td>This method can be used to issue an IDS SL900A GetLogState custom command as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>SL900A_GetMeasurementSetup</td>
<td>This method gets the current system setup as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>SL900A_GetSensorValue</td>
<td>This method can be used to issue an IDS SL900A GetSensorValue custom command as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>SL900A_Initialize</td>
<td>This method can be used to issue an IDS SL900A Initialize custom command as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>SL900A_SetLogLimits</td>
<td>This method sets 4 limits which can be used for logging measurement data as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>SL900A_SetLogMode</td>
<td>This method can be used to issue an IDS SL900A SetLogMode custom command as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>SL900A_StartLog</td>
<td>This method can be used to issue an IDS SL900A StartLog custom command as defined in the IDS SL900A datasheet.</td>
</tr>
<tr>
<td>Untraceable_EPC_C1G2</td>
<td>This method allows an interrogator with an asserted untraceable privilege to instruct a Tag to (a) alter the L and U bits in EPC memory, (b) hide memory from interrogators with a deasserted Untraceable privilege and/or (c) reduce its operating range for all interrogators.</td>
</tr>
<tr>
<td>WriteTagData</td>
<td>This method can be used to write a portion of the user memory in a ISO18000-6B tag.</td>
</tr>
<tr>
<td>WriteTagData_EPC_C1G2</td>
<td>Overloaded. This method can be used to write a portion of memory in a ISO18000-6C (EPC Class1 Gen2) tag.</td>
</tr>
</tbody>
</table>

### CAENRFIDNotify Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getDate</td>
<td>Returns a timestamp representing the time at which the event was generated.</td>
</tr>
<tr>
<td>getPC</td>
<td>Returns the tag’s PC code</td>
</tr>
<tr>
<td>getReadPoint</td>
<td>Returns the read point that has detected the tag.</td>
</tr>
<tr>
<td>Methods</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>getRSSI</td>
<td>Returns the RSSI value measured for the tag.</td>
</tr>
<tr>
<td>getStatus</td>
<td>Returns the event type associated to the tag.</td>
</tr>
<tr>
<td>getTagID</td>
<td>Returns the tag’s ID (the EPC code in Gen2 tags).</td>
</tr>
<tr>
<td>getTagLength</td>
<td>Returns the tag’s ID length.</td>
</tr>
<tr>
<td>getTagSource</td>
<td>Returns the name of the logical source that has detected the tag.</td>
</tr>
<tr>
<td>getTagType</td>
<td>Returns the air protocol of the tag.</td>
</tr>
<tr>
<td>getTID</td>
<td>Returns the TID field value in a EPC Class 1 Gen 2 Tag</td>
</tr>
<tr>
<td>getXPC</td>
<td>Returns the tag’s XPC words.</td>
</tr>
</tbody>
</table>

**CAENRFIDReader Class**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>Overloaded. Starts the communication with the reader. It must be called before any other call to method of the CAENRFIDReader object.</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Closes the connection with the CAEN RFID Reader releasing all the allocated resources.</td>
</tr>
<tr>
<td>GetBatteryLevel</td>
<td>This method gets the current battery charge.</td>
</tr>
<tr>
<td>GetBitRate</td>
<td>Gets the current setting of the RF bit rate.</td>
</tr>
<tr>
<td>GetFirmwareRelease</td>
<td>Permits to read the release of the firmware loaded into the device.</td>
</tr>
<tr>
<td>GetIO</td>
<td>Gets the current digital Input and Output lines status.</td>
</tr>
<tr>
<td>GetIODirection</td>
<td>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.</td>
</tr>
<tr>
<td>GetFHSSMode</td>
<td>This method gets the current Frequency Hopping status.</td>
</tr>
<tr>
<td>GetPower</td>
<td>Gets the current setting of the RF power expressed in mW.</td>
</tr>
<tr>
<td>GetProtocol</td>
<td>Gets the current air protocol of the Reader.</td>
</tr>
<tr>
<td>GetReaderInfo</td>
<td>Permits to read the reader information loaded into the device.</td>
</tr>
<tr>
<td>GetReadPoints</td>
<td>Gets the names of the read points (antennas) available in the reader.</td>
</tr>
<tr>
<td>GetReadPointStatus</td>
<td>Gets the CAENRFIDReadPointStatus object representing the status of a read point (antenna).</td>
</tr>
<tr>
<td>GetRFChannel</td>
<td>Gets the index of the RF channel currently in use. The index value meaning change for different country regulations.</td>
</tr>
<tr>
<td>GetRFRegulation</td>
<td>Gets the current RF regulation setting value.</td>
</tr>
<tr>
<td>GetSource</td>
<td>Gets a CAENRFIDLogicalSource object given its name.</td>
</tr>
<tr>
<td>GetSourceNames</td>
<td>Gets the names of the logical sources available in the reader.</td>
</tr>
<tr>
<td>GetSources</td>
<td>Gets the CAENRFIDLogicalSource objects available on the reader.</td>
</tr>
<tr>
<td>InventoryAbort</td>
<td>Stops the EventInventoryTag execution.</td>
</tr>
<tr>
<td>MatchReadPointImpedance</td>
<td>Overloaded. This method matches the antenna impedance passed in ReadPoint.</td>
</tr>
<tr>
<td>PrintScreen</td>
<td>Print ASCII text on the reader’s screen (only for readers with display, e.g. R1170I qIDmini).</td>
</tr>
<tr>
<td>RFControlMethod</td>
<td>Permits to control the RF CW (Carrier Wave) signal generation.</td>
</tr>
<tr>
<td>SetBitRate</td>
<td>Sets the RF bit rate to use.</td>
</tr>
<tr>
<td>SetDateTime</td>
<td>Sets the Date/Time of the reader.</td>
</tr>
<tr>
<td>SetIO</td>
<td>Sets the Output lines value.</td>
</tr>
<tr>
<td>SetIODIRECTION</td>
<td>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.</td>
</tr>
<tr>
<td>SetNetwork</td>
<td>Permits to configure the network settings of the reader. In order to apply the changes the reader must be restarted.</td>
</tr>
<tr>
<td>SetPower</td>
<td>Sets the conducted RF power of the Reader.</td>
</tr>
<tr>
<td>SetProtocol</td>
<td>Set the air protocol of the reader.</td>
</tr>
<tr>
<td>SetRFChannel</td>
<td>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.</td>
</tr>
<tr>
<td>SetRS232</td>
<td>Permits to change the serial port settings. Valid settings values depend on the reader model.</td>
</tr>
</tbody>
</table>

**CAENRFIDReaderInfo Class**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetModel</td>
<td>Gets the reader’s model.</td>
</tr>
<tr>
<td>GetSerialNumber</td>
<td>Gets the reader’s serial number.</td>
</tr>
</tbody>
</table>

**CAENRFIDTag Class**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetId</td>
<td>Returns the tag’s ID (the EPC code in Gen2 tags).</td>
</tr>
<tr>
<td>GetLength</td>
<td>Returns the tag’s ID length.</td>
</tr>
<tr>
<td>GetPC</td>
<td>Returns the tag’s PC code</td>
</tr>
<tr>
<td>GetReadPoint</td>
<td>Returns the read point that has detected the tag.</td>
</tr>
</tbody>
</table>

**CAEN RFID API STRUCTURE - CAEN RFID API Reference Manual** 13
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRSSI</td>
<td>Returns the RSSI value measured for the tag.</td>
</tr>
<tr>
<td>GetSource</td>
<td>Returns the name of the logical source that has detected the tag.</td>
</tr>
<tr>
<td>GetTID</td>
<td>Returns the tag’s TID (valid only for EPC Class 1 Gen 2 tags).</td>
</tr>
<tr>
<td>GetTimeStamp</td>
<td>Gets the Tag’s TimeStamp.</td>
</tr>
<tr>
<td>GetType</td>
<td>Returns the air protocol of the tag.</td>
</tr>
<tr>
<td>GetXPC</td>
<td>Returns the tag’s XPC words.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Enumerations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitRate</td>
<td>Gives a list of the supported radiofrequency profiles.</td>
</tr>
<tr>
<td>LogicalSourceConstants</td>
<td>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.</td>
</tr>
<tr>
<td>CAENRFIDLogicalSource.InventoryFlag</td>
<td>Gives a list of constants used for the configuration of the inventory function.</td>
</tr>
<tr>
<td>Port</td>
<td>Gives a list of the communication ports supported by the CAEN RFID readers.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Gives a list of the air protocol supported by the CAEN RFID readers.</td>
</tr>
<tr>
<td>ReadPointStatus</td>
<td>Gives a list of the possible ReadPoint status values.</td>
</tr>
<tr>
<td>CAENRFIDRFRegulations</td>
<td>The CAENRFIDRFRegulations gives a list of country radiofrequency regulations.</td>
</tr>
<tr>
<td>RS232Constants</td>
<td>Gives a list of settings for the serial port configuration.</td>
</tr>
<tr>
<td>SelUnselOptions</td>
<td>Gives a list of operations supported by the Group Select/Unselect command (valid only for the ISO18000-6B air protocol).</td>
</tr>
</tbody>
</table>

Tab. 2.3: CAENRFID Enumerations
CAENRFIDException Class

The CAENRFIDException class defines the CAEN RFID exceptions.

**getErrorCode Method**

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

*Return value:*
The string representing the error.

*Syntax:
C# representation:*
```csharp
public string getErrorCode()
```

*Java and Android representation:*
```java
public java.lang.String getErrorCode()
```

*Remarks:*
This function does not exist in C language, see § *Error Handling* page 8 for more information.

IDSTagData Class

This class represents data returned by tags based on IDS Chip SL900A.

In Java, Android and C# languages this class is composed by methods while in C language is represented by a struct (for more information see § *Overview on SDK* page 8):

*C representation:*
```c
typedef struct {
    BOOL ADError_i;
    unsigned int RangeLimit_i;
    unsigned int SensorValue_i;
} CAENRFID_IDSTagData;
```

**getADSError Method**

*Description:*
This method returns if an A/D error is raised.

*Return value:*
True if an A/D error occurs, false otherwise.

*Syntax:*
*C# representation:*
```csharp
public bool ADError {
    get;
}
```

*Java and Android representation:*
```java
public boolean getADSError()
```
getRangeLimit Method

Description:
This method returns the range limit set on sensor.

Return value:
A bitmask representing the range limit.

Syntax:
C# representation:
```csharp
public uint RangeLimit {
    get;
}
```

Java and Android representation:
```java
public int getRangeLimit()
```

getsensorValue Method

Description:
This method returns the sensor value.

Return value:
A bitmask representing the value obtained by the sensor.

Syntax:
C# representation:
```csharp
public uint SensorValue {
    get;
}
```

Java and Android representation:
```java
public int getSensorValue()
```
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:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadPoint</td>
<td>A string representing the name of the read point (antenna).</td>
</tr>
</tbody>
</table>

*Syntax:*

**C# representation:**

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

**Java and Android representation:**

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

**C representation:**

```c
CAENRFIDErrorCodes CAENRFID_AddReadPoint(
    CAENRFIDHandle handle,
    char *SourceName,
    char *ReadPoint);
```
Authenticate_EPC_C1G2 Method

Description:
This method allows an interrogator to perform tag, interrogator or mutual authentication. The generic nature of the authenticate command allows it to support a variety of cryptographic suites. The number of authenticate commands required to implement an authentication depends on the authentication type and on the chosen cryptographic suite.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be authenticated.</td>
</tr>
<tr>
<td>senRep</td>
<td>Specifies whether a tag backscatters its response or first stores the response in its ResponseBuffer and then returns the data from there.</td>
</tr>
<tr>
<td>incRepLen</td>
<td>Specifies whether a tag omits or includes the response length in its reply</td>
</tr>
<tr>
<td>csi</td>
<td>Selects the cryptographic suite that tag and interrogator use for the authentication.</td>
</tr>
<tr>
<td>challenge</td>
<td>It includes parameters and data for authentication.</td>
</tr>
<tr>
<td>repLen</td>
<td>Specify the byte’s length of the tag’s response.</td>
</tr>
<tr>
<td>password</td>
<td>The access password</td>
</tr>
</tbody>
</table>

Return value:
A byte array containing the tag’s response to the authenticate command.

Syntax:
C# representation:
```csharp
public void Authenticate_EPC_C1G2(CAENRFIDTag Tag, bool senRep, bool incRepLen, char csi, byte[] challenge, short repLen, uint password)
```

Java and Android representation:
```java
public void Authenticate_EPC_C1G2(CAENRFIDTag Tag, boolean senRep, boolean incRepLen, char csi, byte[] challenge, short repLen, int password)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_Authentica te_EPC_C1G2(CAENRFIDHandle handle, char* SourceName, CAENRFIDTag *Tag, BOOL senRep, BOOL incRepLen, char csi, byte* challenge, short challengeLen, short repLen, int password, byte**authResponse);
```
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be written.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start writing the data.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be written.</td>
</tr>
<tr>
<td>Data</td>
<td>The data to be written into the tag’s user memory.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void BlockWriteTagData(       
    CAENRFIDTag Tag,       
    short Address,       
    short Length,       
    byte[] Data)        
```

**Java and Android representation:**
```java
public void BlockWriteTagData(       
    CAENRFIDTag Tag,       
    short Address,       
    short Length,       
    byte[] Data)        
    throws CAENRFIDException
```

**C representation:**
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be written.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start writing the data.</td>
</tr>
<tr>
<td>Mask</td>
<td>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).</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be written.</td>
</tr>
<tr>
<td>Data</td>
<td>The data to be written into the tag's user memory.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:****

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

**Java and Android representation:**

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

**C representation:**

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

---

**ClearBuffer Method**

**Description:**
This method deletes all items stored in the internal buffer.

**Syntax:**

**C# representation:**

```csharp
public int ClearBuffer
```

**Java and Android representation:**

```java
public int ClearBuffer() throws CAENRFIDException
```

**C representation:**

```c
CAENRFIDErrorCodes CAENRFID_ClearBuffer(CAENRFIDHandle handle);
```
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to which send the Custom command.</td>
</tr>
<tr>
<td>SubCmd</td>
<td>The SubCommand field of the Custom command.</td>
</tr>
<tr>
<td>TxLen</td>
<td>The length of the data to be sent to the tag.</td>
</tr>
<tr>
<td>Data</td>
<td>The data to be sent to the tag.</td>
</tr>
<tr>
<td>RxLen</td>
<td>The length of the data to be received by the tag.</td>
</tr>
</tbody>
</table>

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

Syntax:

**C# representation:**
```csharp
public byte[] CustomCommand_EPC_C1G2(
    CAENRFIDTag Tag,
    byte SubCmd,
    short TxLen,
    byte[] Data,
    short RxLen)
```

**Java and Android representation:**
```java
public byte[] CustomCommand_EPC_C1G2(
    CAENRFIDTag Tag,
    byte SubCmd,
    short TxLen,
    byte[] Data,
    short RxLen)
    throws CAENRFIDException
```

**C representation:**
```c
CAENRFIDCustomCommand_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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
<tr>
<td>SubCmd</td>
<td>The SubCommand field of the Custom command.</td>
</tr>
<tr>
<td>TxLen</td>
<td>The length of the data to be sent to the tag.</td>
</tr>
<tr>
<td>Data</td>
<td>The data to be sent to the tag.</td>
</tr>
<tr>
<td>RxLen</td>
<td>The length of the data to be received by the tag.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

**Return value:**
An array of bytes representing the reply from the tag as specified by the custom command.

**Syntax:**

**C# representation:**
```csharp
public byte[] CustomCommand_EPC_C1G2(
    CAENRFIDTag Tag,
    byte SubCmd,
    short TxLen,
    byte[] Data,
    short RxLen,
    int AccessPassword)
```

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

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

EventInventoryTag Method

For the description of this method, see § EVENT HANDLING page 99.
### GetBufferedData Method

**GetBufferedData ()**

**Description:**
This method returns all the Tags stored in reader’s buffer using all the ReadPoints belonging to the Source.

**Return value:**
An array of CAENRFIDTag objects detected.

**Syntax:**

- **C# representation:**
  ```csharp
  public CAENRFIDTag[] GetBufferedData()
  ```

- **Java and Android representation:**
  ```java
  public CAENRFIDTag[] GetBufferedData()
  ```

- **C representation:**
  ```c
  CAENRFIDEErrorCodes CAENRFID_GetBufferedData(
      CAENRFIDHandle handle,
      char *source,
      CAENRFIDTag **Receive,
      int *Size);
  ```

### GetBufferedData (Int32, Int32)

**Description:**
This method returns all the Tags stored in reader’s buffer using all the ReadPoints belonging to the Source.

**Return value:**
An array of CAENRFIDTag objects detected.

**Syntax:**

- **C# representation:**
  ```csharp
  public CAENRFIDTag[] GetBufferedData (int Address, int Length)
  ```

- **Java and Android representation:**
  ```java
  public CAENRFIDTag[] GetBufferedData (int Address, int Length)
  ```

- **C representation:**
  ```c
  CAENRFIDEErrorCodes CAENRFID_GetBufferedDataRange(
      CAENRFIDHandle handle,
      char *source,
      int address, int length
      CAENRFIDTag **Receive,
      int *Size);
  ```
GetBufferSize Method

Description:
This method gets the current number of records (tags) stored in the reader's internal buffer.

Return value:
The current number of items stored in the internal's reader buffer.

Syntax:
C# representation:
public int GetBufferSize()

Java and Android representation:
public int GetBufferSize()
throws CAENRFIDException

C representation:
CAENRFIDErrorCodes CAENRFID_GetBufferSize(
CAENRFIDHandle handle,
unsigned 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 and Android representation:
public CAENRFIDLogicalSourceConstants GetDESB_ISO180006B()
throws CAENRFIDException

C representation:
CAENRFIDErrorCodes GetDESB_ISO180006B(
CAENRFIDHandle handle,
unsigned short *Status);

GetInventoryCounts Method

Description:
This method gets the current setting for the number of inventory counts performed by the logical source after pressing the TRIGGER during the inventory algorithm execution.

Return value:
The number of read cycles.

Syntax:
C# representation:
public int GetInventoryCounts()

Java and Android representation:
public int GetInventoryCounts()
throws CAENRFIDException

C representation:
CAENRFIDErrorCodes CAENRFID_GetInventoryCounts(
CAENRFIDHandle handle,
char *SourceName,
int *value);
GetInventoryDwellTime Method

Description:
This method gets the inventory execution time (msec) used by the logical source during the inventory algorithm execution.

Return value:
The inventory execution time (msec).

Syntax:
C# representation:
```csharp
public int GetInventoryDwellTime();
```

Java and Android representation:
```java
public int GetInventoryDwellTime() throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_GetInventoryDwellTime(
    CAENRFIDHandle handle,
    char *SourceName,
    int *Value);
```

GetInventoryQuietTime Method

Description:
This method can be used to get the inventory quiet time (msec) used by the logical source during the inventory algorithm execution.

Return value:
The quite time expressed in ms.

Syntax:
C# representation:
```csharp
public int GetInventoryQuietTime();
```

Java and Android representation:
```java
public int GetInventoryQuietTime() throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_GetInventoryQuietTime(
    CAENRFIDHandle handle,
    char *SourceName,
    int *Value);
```
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 and Android representation:
public java.lang.String GetName()

Remarks:
This function does not exist in C language, see § Overview on SDK page 8 for more information.

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 and Android 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 continuous mode (see § EventInventoryTag Method page 22).

Return value:
The number of read cycles.

Syntax:
C# representation:
public int GetReadCycle()

Java and Android 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:
```csharp
public CAENRFIDLogicalSourceConstants GetSelected_EPC_C1G2()
```

Java and Android representation:
```java
public CAENRFIDLogicalSourceConstants GetSelected_EPC_C1G2()
```

C representation:
```c
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:
```csharp
public CAENRFIDLogicalSourceConstants GetSession_EPC_C1G2()
```

Java and Android representation:
```java
public CAENRFIDLogicalSourceConstants GetSession_EPC_C1G2()
```

C representation:
```c
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:
```csharp
public CAENRFIDLogicalSourceConstants GetTarget_EPC_C1G2()
```

Java and Android representation:
```java
public CAENRFIDLogicalSourceConstants GetTarget_EPC_C1G2()
throws CAENRFIDException
```

C representation:
```c
CAENRFIDEErrorCodes 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>The operation code as defined by the protocol.</td>
</tr>
<tr>
<td>Address</td>
<td>The Address from which start the comparison.</td>
</tr>
<tr>
<td>BitMask</td>
<td>The bit mask to use.</td>
</tr>
<tr>
<td>Data</td>
<td>The data to be compared.</td>
</tr>
</tbody>
</table>

Return value:
The selected tag.

Syntax:
C# representation:
```csharp
public CAENRFIDTag GroupSelUnsel(
    CAENRFIDSelUnselOptions Code,
    short Address,
    short BitMask,
    byte[] Data)
```

Java and Android representation:
```java
public CAENRFIDTag GroupSelUnsel(
    CAENRFIDSelUnselOptions Code,
    short Address,
    short BitMask,
    byte[] Data)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDEErrorCodes CAENRFID_GroupSelUnsel(
    CAENRFIDHandle handle,
    char *SourceName,
    CAENRFID_SelUnsel_Op Code,
    int Address,
    int BitMask,
    void *Data,
    CAENRFIDTag *Tag);
```
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 and Android representation:
public CAENRFIDTag[] InventoryTag()
throws CAENRFIDException

C representation:
CAENRFILErrorCodes 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask</td>
<td>A byte array representing the bitmask to apply.</td>
</tr>
<tr>
<td>MaskLength</td>
<td>A value representing the bit-oriented length of the bitmask.</td>
</tr>
<tr>
<td>Position</td>
<td>A value representing the first bit of ID where the match will start.</td>
</tr>
</tbody>
</table>

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 and Android representation:
public CAENRFIDTag[] InventoryTag(
byte[] Mask,
short MaskLength,
short Position)
throws CAENRFIDException

C representation:
CAENRFILErrorCodes 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 performed 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 match 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:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask</td>
<td>A byte array representing the bitmask to apply.</td>
</tr>
<tr>
<td>MaskLength</td>
<td>A value representing the bit-oriented length of the bitmask.</td>
</tr>
<tr>
<td>Position</td>
<td>A value representing the first bit of ID where the match will start.</td>
</tr>
<tr>
<td>Flag</td>
<td>A bitmask representing the InventoryTag options.</td>
</tr>
</tbody>
</table>

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

Syntax:

C# representation:
```csharp
public CAENRFIDTag[] InventoryTag(
    byte[] Mask,
    short MaskLength,
    short Position,
    short Flag)
```

Java and Android representation:
```java
public CAENRFIDTag[] InventoryTag(
    byte[] Mask,
    short MaskLength,
    short Position,
    short Flag)
```

C representation:
```c
#include "CAENRFID.h"

CAENRFIDTag* CAENRFID_FlagInventoryTag (CAENRFIDHandle handle, const char *SourceName, const 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 performed 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 match 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 (continuous and framed mode) are ignored.

<table>
<thead>
<tr>
<th>Flag value meaning</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>RSSI: a 1 value indicates that the reader will transmit the RSSI (Return Signal Strength Indicator) in the response.</td>
</tr>
<tr>
<td>Bit 1</td>
<td>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 transmitted all together at the end of the inventory cycle.</td>
</tr>
<tr>
<td>Bit 2</td>
<td>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.</td>
</tr>
<tr>
<td>Bit 3</td>
<td>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.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>TID reading: a 1 value indicates that also the TID of the tag will be returned by the reader together with the other information.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Event trigger: when this flag is set together with the continuous acquisition flag, the inventory cycle is performed in the same way of the continuous mode with the only difference that the inventory command is sent only by pressing the left key of the A828BT reader.</td>
</tr>
<tr>
<td>Bit 6</td>
<td>XPC: a 1 value allows the reader to get the XPC word if backscattered by a tag. Tags that do not backscatter the XPC words will return an XPC array with all the 4 bytes set to 0</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Match tag: a 1 value enables the matching of read tags with a tag present in the memory (A828BT reader only).</td>
</tr>
<tr>
<td>Bit 8</td>
<td>PC: a 1 value allows the reader to return the PC of a Gen2 tag in addition to the ID (A828BT reader only).</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bank</td>
<td>A value representing the memory bank where apply the filter.</td>
</tr>
<tr>
<td>Mask</td>
<td>A byte array representing the bitmask to apply.</td>
</tr>
<tr>
<td>MaskLength</td>
<td>A value representing the bit-oriented length of the bitmask.</td>
</tr>
<tr>
<td>Position</td>
<td>A value representing the first bit of ID where the match will start.</td>
</tr>
</tbody>
</table>

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 and Android 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 performed 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 match 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bank</td>
<td>A value representing the memory bank where apply the filter.</td>
</tr>
<tr>
<td>Mask</td>
<td>A byte array representing the bitmask to apply.</td>
</tr>
<tr>
<td>MaskLength</td>
<td>A value representing the bit-oriented length of the bitmask.</td>
</tr>
<tr>
<td>Position</td>
<td>A value representing the first bit of ID where the match will start.</td>
</tr>
<tr>
<td>Flag</td>
<td>A bitmask representing the InventoryTag options.</td>
</tr>
</tbody>
</table>

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

Syntax:
C# representation:
```csharp
public CAENRFIDTag[] InventoryTag(
    short bank,
    byte[] Mask,
    short MaskLength,
    short Position,
    short Flag)
```

Java and Android representation:
```java
public CAENRFIDTag[] InventoryTag(
    short bank,
    byte[] Mask,
    short MaskLength,
    short Position,
    short Flag)
```

C representation:
```c
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 performed 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 match 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 (continuous and framed mode) are ignored.

**Flag value meaning**

| Bit 0 | RSSI: a 1 value indicates that 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 transmitted 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 information. |
| Bit 5 | Event trigger: when this flag is set together with the continuous acquisition flag, the inventory cycle is performed in the same way of the continuous mode with the only difference that the inventory command is sent only by pressing the left key of the A828BT reader. |
| Bit 6 | XPC: a 1 value allows the reader to get the XPC word if backscattered by a tag.Tags that do not backscatter the XPC words will return an XPC array with all the 4 bytes set to 0. |
| Bit 7 | Match tag: a 1 value enables the matching of read tags with a tag present in the memory (A828BT reader only). |
| Bit 8 | PC: a 1 value allows the reader to return the PC of a Gen2 tag in addition to the ID (A828BT reader only). |

**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 explicitly 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:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tags</td>
<td>tags array returned by one of the inventory family function.</td>
</tr>
</tbody>
</table>

*Syntax:*

**C representation:**

```c
void CAENRFID_FreeTagsMemory(
    CAENRFIDTag **Tags);
```
isReadPointPresent Method

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

Parameters:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadPoint</td>
<td>A string representing the name of the read point (antenna).</td>
</tr>
</tbody>
</table>

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

Java and Android representation:
```java
public boolean isReadPointPresent(
    java.lang.String ReadPoint)
throws CAENRFIDException
```

C representation:
```c
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:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be killed.</td>
</tr>
<tr>
<td>Password</td>
<td>The tag’s kill password.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void KillTag_EPC_C1G1(
    CAENRFIDTag Tag,
    short Password)
```

Java and Android representation:
```java
public void KillTag_EPC_C1G1(
    CAENRFIDTag Tag,
    short Password)
throws CAENRFIDException
```

C representation:
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be killed.</td>
</tr>
<tr>
<td>Password</td>
<td>The tag’s kill password.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void KillTag_EPC_C1G2(
    CAENRFIDTag Tag,
    int Password
)
```

**Java and Android representation:**
```java
public void KillTag_EPC_C1G2(
    CAENRFIDTag Tag,
    int Password
) throws CAENRFIDException
```

**C representation:**
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankMask</td>
<td>Memory bank for tag identification.</td>
</tr>
<tr>
<td>PositionMask</td>
<td>Bit position (from the start of the selected bank) where apply the mask to match.</td>
</tr>
<tr>
<td>LengthMask</td>
<td>Length of the mask.</td>
</tr>
<tr>
<td>Mask</td>
<td>Mask of byte.</td>
</tr>
<tr>
<td>Password</td>
<td>The tag’s kill password.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void KillTag_EPC_C1G2(
    short BankMask,
    short PositionMask,
    short LengthMask,
    byte[] Mask,
    int Password
)
```

**Java and Android representation:**
```java
public void KillTag_EPC_C1G2(
    short BankMask,
    short PositionMask,
    short LengthMask,
    byte[] Mask,
    int Password
) throws CAENRFIDException
```

**C representation:**
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be written.</td>
</tr>
<tr>
<td>MemBank</td>
<td>The memory bank where to write the data.</td>
</tr>
<tr>
<td>BlockPtr</td>
<td>The address where to start writing the data.</td>
</tr>
<tr>
<td>BlockRange</td>
<td>The number of word of the mask.</td>
</tr>
<tr>
<td>Mask</td>
<td>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).</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void LockBlockPermaLock_EPC_C1G2(
    CAENRFIDTag Tag,
    short MemBank,
    short BlockPtr,
    short BlockRange,
    byte[] Mask,
    int AccessPassword)
```

Java and Android representation:
```java
public void LockBlockPermaLock_EPC_C1G2(
    CAENRFIDTag Tag,
    short MemBank,
    short BlockPtr,
    short BlockRange,
    byte[] Mask,
    int AccessPassword)
```

C representation:
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be locked.</td>
</tr>
<tr>
<td>Payload</td>
<td>The Payload parameter for the lock command as defined by the EPC Class 1 Gen 2 protocol specification.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void LockTag_EPC_C1G2(
    CAENRFIDTag Tag,
    int Payload)
```

Java and Android representation:
```java
public void LockTag_EPC_C1G2(
    CAENRFIDTag Tag,
    int Payload)
    throws CAENRFIDException
```

C representation:
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be locked.</td>
</tr>
<tr>
<td>Payload</td>
<td>The Payload parameter for the lock command as defined by the EPC Class 1 Gen 2 protocol specification.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void LockTag_EPC_C1G2(
    CAENRFIDTag Tag,
    int Payload,
    int AccessPassword)
```

Java and Android representation:
```java
public void LockTag_EPC_C1G2(
    CAENRFIDTag Tag,
    int Payload,
    int AccessPassword)
    throws CAENRFIDException
```

C representation:
```c
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 an EPC Class 1 Gen 2 tag.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankMask</td>
<td>Memory bank for tag identification.</td>
</tr>
<tr>
<td>PositionMask</td>
<td>Bit position (from the start of the selected bank) where apply the mask to match.</td>
</tr>
<tr>
<td>LengthMask</td>
<td>Length of the mask.</td>
</tr>
<tr>
<td>Mask</td>
<td>Mask of byte.</td>
</tr>
<tr>
<td>Payload</td>
<td>The Payload parameter for the lock command as defined by the EPC Class 1 Gen 2 protocol specification.</td>
</tr>
</tbody>
</table>

Syntax:

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

Java and Android 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:*

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

*Syntax:*

**C# representation:**

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

**Java and Android representation:**

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

**C representation:**

```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be locked.</td>
</tr>
<tr>
<td>Address</td>
<td>The byte's address to lock.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```
public void LockTag_ISO180006B(
    CAENRFIDTag Tag,
    short Address)
```

Java and Android representation:
```
public void LockTag_ISO180006B(
    CAENRFIDTag Tag,
    short Address)
throws CAENRFIDException
```

C representation:
```
CAENRFIDEErrorCodes 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
<tr>
<td>EAS</td>
<td>A boolean representing the EAS state to set.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```
public void NXP_ChangeEAS(
    CAENRFIDTag Tag,
    bool EAS,
    int AccessPassword)
```

Java and Android representation:
```
public void NXP_ChangeEAS(
    CAENRFIDTag Tag,
    boolean EAS,
    int AccessPassword)
throws CAENRFIDException
```

C representation:
```
CAENRFIDEErrorCodes CAENRFID_NXP_SecureChangeEAS(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    char EAS,
    int AccessPassword);
```
**NXP_ChangeConfig Method**

**NXP_ChangeConfig Method (CAENRFIDTag, UInt16)**

**Description:**
This method can be used to issue a NXP_ChangeConfig custom command as defined in the NXP UCODE G2IM and G2IM+ datasheet.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
<tr>
<td>ConfigWord</td>
<td>The configuration word.</td>
</tr>
</tbody>
</table>

**Syntax:**

- **C# representation:**
  ```csharp
  public void NXP_ChangeConfig(CAENRFIDTag Tag, ushort ConfigWord)
  ```

- **Java and Android representation:**
  ```java
  public void NXP_ChangeConfig(CAENRFIDTag Tag, short ConfigWord)
  ```

- **C representation:**
  ```c
  CAENRFIDErrorCodes CAENRFID_NXP_ChangeConfig(
      CAENRFIDHandle handle,
      CAENRFIDTag *Tag,
      short ConfigWord,
      char *TRData);
  ```

**NXP_ChangeConfig Method (CAENRFIDTag, UInt16, Int32)**

**Description:**
This method can be used to issue a NXP_ChangeConfig custom command as defined in the NXP UCODE G2IM and G2IM+ datasheet after having put it in Secured state using the Access Password.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
<tr>
<td>ConfigWord</td>
<td>The configuration word.</td>
</tr>
<tr>
<td>Password</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

**Syntax:**

- **C# representation:**
  ```csharp
  public void NXP_ChangeConfig(CAENRFIDTag Tag, ushort ConfigWord, int Password)
  ```

- **Java and Android representation:**
  ```java
  public void NXP_ChangeConfig(CAENRFIDTag Tag, short ConfigWord, int Password)
  ```

- **C representation:**
  ```c
  CAENRFIDErrorCodes CAENRFID_NXP_SecureChangeConfig(
      CAENRFIDHandle handle,
      CAENRFIDTag *Tag,
      short ConfigWord,
      char *TRData,
      int SecurePassword);
  ```
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:
```csharp
public byte[] NXP_EAS_Alarm()
```
Java and Android representation:
```java
public byte[] NXP_EAS_Alarm()
    throws CAENRFIDException
```
C representation:
```c
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:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void NXP_ReadProtect(CAENRFIDTag Tag)
```
Java and Android representation:
```java
public void NXP_ReadProtect(CAENRFIDTag Tag)
    throws CAENRFIDException
```
C representation:
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void NXP_ReadProtect(
    CAENRFIDTag Tag,
    int AccessPassword)
```

**Java and Android representation:**
```java
public void NXP_ReadProtect(
    CAENRFIDTag Tag,
    int AccessPassword)
```

**C representation:**
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to reset the read protection.</td>
</tr>
<tr>
<td>Password</td>
<td>The ReadProtect password.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void NXP_ResetReadProtect(
    CAENRFIDTag Tag,
    int Password)
```

**Java and Android representation:**
```java
public void NXP_ResetReadProtect(
    CAENRFIDTag Tag,
    int Password)
```

**C representation:**
```c
CAENRFIDErrorCodes CAENRFID_NXP_ResetReadProtect(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    int Password);
```
NXP_ChangeConfig Method

NXP_ChangeConfig Method (CAENRFIDTag, UInt16)

Description:
This method can be used to issue a NXP_ChangeConfig custom command as defined in the NXP UCODE G2iM and G2iM+ datasheet.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
<tr>
<td>ConfigWord</td>
<td>The Configuration word.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void NXP_ChangeConfig(
    CAENRFIDTag Tag,
    ushort ConfigWord)
```

Java and Android representation:
```java
public void NXP_ChangeConfig(
    CAENRFIDTag Tag,
    short ConfigWord)
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_NXP_ChangeConfig(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    short ConfigWord,
    char *TRData);
```

NXP_ChangeConfig Method (CAENRFIDTag, UInt16, Int32)

Description:
This method can be used to issue a NXP_ChangeConfig custom command as defined in the NXP UCODE G2iM and G2iM+ datasheet after having put it in Secured state using the Access Password.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag object representing the tag to select.</td>
</tr>
<tr>
<td>ConfigWord</td>
<td>The Configuration word.</td>
</tr>
<tr>
<td>Password</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void NXP_ChangeConfig(
    CAENRFIDTag Tag,
    ushort ConfigWord,
    int Password)
```

Java and Android representation:
```java
public void NXP_ChangeConfig(
    CAENRFIDTag Tag,
    short ConfigWord,
    int Password)
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_NXP_SecureChangeConfig(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    short ConfigWord,
    char *TRData,
    int SecurePassword);
```
ProgramID_EPC_C1G1 Method

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

Parameters:

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

Syntax:

C# representation:
```csharp
public void ProgramID_EPC_C1G1(
    CAENRFIDTag Tag,
    short Password,
    bool Lock)
```

Java and Android representation:
```java
public void ProgramID_EPC_C1G1(
    CAENRFIDTag Tag,
    short Password,
    boolean Lock)
throws CAENRFIDException
```

C representation:
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be programmed, the ID contained in this object will be programmed into the tag.</td>
</tr>
<tr>
<td>NSI</td>
<td>The Numbering System Identifier as defined in EPC Class 1 Gen 2 protocol specifications.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void ProgramID_EPC_C1G2(
    CAENRFIDTag Tag,
    short NSI)
```

Java and Android representation:
```java
public void ProgramID_EPC_C1G2(
    CAENRFIDTag Tag,
    short NSI)
throws CAENRFIDException
```

C representation:
```c
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:**

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

**Syntax:**

**C# representation:**
```csharp
public void ProgramID_EPC_C1G2(
    CAENRFIDTag Tag,
    short NSI,
    int AccessPassword
)
```

**Java and Android representation:**
```java
public void ProgramID_EPC_C1G2(
    CAENRFIDTag Tag,
    short NSI,
    int AccessPassword
) throws CAENRFIDException
```

**C representation:**
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be programmed.</td>
</tr>
<tr>
<td>NewID</td>
<td>An array of bytes representing the new UID for the tag.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void ProgramID_EPC119(
    CAENRFIDTag Tag,
    byte[] NewID
)
```

**Java and Android representation:**
```java
public void ProgramID_EPC119(
    CAENRFIDTag Tag,
    byte[] NewID
)
```

**C representation:**
```c
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 successful completion.

Syntax:
C# representation:
public bool Query_EPC_C1G2()

Java and Android 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 makes 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 and Android 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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be read.</td>
</tr>
<tr>
<td>MemBank</td>
<td>The memory bank where to read the data.</td>
</tr>
<tr>
<td>Blockptr</td>
<td>The address where to start reading the data.</td>
</tr>
<tr>
<td>BlockRange</td>
<td>The number of word to be read.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

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

**Syntax:**

**C# representation:**
```csharp
public byte[] ReadBLockPermalock_EPC_C1G2(
    CAENRFIDTag Tag,
    short MemBank,
    short Blockptr,
    short BlockRange,
    int AccessPassword)
```

**Java and Android representation:**
```java
public byte[] ReadBLockPermalock_EPC_C1G2(
    CAENRFIDTag Tag,
    short MemBank,
    short Blockptr,
    short BlockRange,
    int AccessPassword
) throws CAENRFIDException
```

**C representation:**
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be read.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start reading the data.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be read.</td>
</tr>
</tbody>
</table>

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

Syntax:

**C# representation:**
```csharp
public byte[] ReadTagData(
    CAENRFIDTag Tag,
    short Address,
    short Length)
```

**Java and Android representation:**
```java
public byte[] ReadTagData(
    CAENRFIDTag Tag,
    short Address,
    short Length)
throws CAENRIDException
```

**C representation:**
```c
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 an ISO18000-6C (EPC Class1 Gen2) tag.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be read.</td>
</tr>
<tr>
<td>MemBank</td>
<td>The memory bank where to read the data.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start reading the data.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be read.</td>
</tr>
</tbody>
</table>

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

**Syntax:**

**C# representation:**
```csharp
public byte[] ReadTagData_EPC_C1G2(CAENRFIDTag Tag, short MemBank, short Address, short Length)
```

**Java and Android representation:**
```java
public byte[] ReadTagData_EPC_C1G2(CAENRFIDTag Tag, short MemBank, short Address, short Length)
```

**C representation:**
```c
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 an ISO18000-6C (EPC Class1 Gen2) tag after having put the tag in Secured state using the Access command.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be read.</td>
</tr>
<tr>
<td>MemBank</td>
<td>The memory bank where to read the data.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start reading the data.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be read.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

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

Syntax:

C# representation:
```csharp
public byte[] ReadTagData_EPC_C1G2(
    CAENRFIDTag Tag,
    short MemBank,
    short Address,
    short Length,
    int AccessPassword)
```

Java and Android representation:
```java
public byte[] ReadTagData_EPC_C1G2(
    CAENRFIDTag Tag,
    short MemBank,
    short Address,
    short Length,
    int AccessPassword)
    throws CAENRFIDException
```

C representation:
```c
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 an ISO18000-6C (EPC Class1 Gen2) tag. In this case the target tag is identified by 'LengthMask' bytes of passed mask placed in a memory bank 'BankMask' at 'PositionMask' byte from bank starting address byte.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankMask</td>
<td>Memory bank for tag identification.</td>
</tr>
<tr>
<td>PositionMask</td>
<td>Bit position (from the start of the selected bank) where apply the mask to match.</td>
</tr>
<tr>
<td>LengthMask</td>
<td>Length of the mask.</td>
</tr>
<tr>
<td>Mask</td>
<td>Mask of byte.</td>
</tr>
<tr>
<td>MemBank</td>
<td>Memory bank where read.</td>
</tr>
<tr>
<td>Address</td>
<td>Address where starts reading.</td>
</tr>
<tr>
<td>Length</td>
<td>Number of bytes to read.</td>
</tr>
</tbody>
</table>

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

Syntax:

**C# representation:**
```csharp
public byte[] ReadTagData_EPC_C1G2(
    short BankMask,
    short PositionMask,
    short LengthMask,
    byte[] Mask,
    short MemBank,
    short Address,
    short Length)
```

**Java and Android representation:**
```java
public byte[] ReadTagData_EPC_C1G2(
    short BankMask,
    short PositionMask,
    short LengthMask,
    byte[] Mask,
    short MemBank,
    short Address,
    short Length)
    throws CAENRFIDException
```

**C representation:**
```c
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 an ISO18000-6C (EPC Class1 Gen2) tag. In this case the target tag is identified by 'LengthMask' 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BankMask</td>
<td>Memory bank for tag identification.</td>
</tr>
<tr>
<td>PositionMask</td>
<td>Bit position (from the start of the selected bank) where apply the mask to match.</td>
</tr>
<tr>
<td>LengthMask</td>
<td>Length of the mask.</td>
</tr>
<tr>
<td>Mask</td>
<td>Mask of byte.</td>
</tr>
<tr>
<td>MemBank</td>
<td>Memory bank where read.</td>
</tr>
<tr>
<td>Address</td>
<td>Address where starts reading.</td>
</tr>
<tr>
<td>Length</td>
<td>Number of byte to read.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>Access Password.</td>
</tr>
</tbody>
</table>

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

Syntax:

C# representation:
```csharp
public byte[] ReadTagData_EPC_C1G2(
    short BankMask,
    short PositionMask,
    short LengthMask,
    byte[] Mask,
    short MemBank,
    short Address,
    short Length,
    int AccessPassword)
```

Java and Android representation:
```java
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:
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadPoint</td>
<td>A string representing the name of the read point (antenna).</td>
</tr>
</tbody>
</table>

**Syntax:**

- **C# representation:**
  
  ```csharp
  public void RemoveReadPoint(string ReadPoint)
  ```

- **Java and Android representation:**
  
  ```java
  public void RemoveReadPoint(java.lang.String ReadPoint)
  ```

- **C representation:**
  
  ```c
  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:**
  
  ```csharp
  public void ResetSession_EPC_C1G2()
  ```

- **Java and Android representation:**
  
  ```java
  public void ResetSession_EPC_C1G2()
  ```

- **C representation:**
  
  ```c
  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:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The DESB setting value.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void SetDESB_ISO180006B(
    CAENRFIDLogicalSourceConstants Value)
```

Java and Android representation:
```java
public void SetDESB_ISO180006B(
    CAENRFIDLogicalSourceConstants Value)
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_SetDESB_ISO180006B(
    CAENRFIDHandle handle,
    unsigned int Value);
```

SetInventoryCounts Method

Description:
This method can be used to set the current setting for the number of inventory counts performed by the logical source after pressing the TRIGGER during the inventory algorithm execution.

Parameters:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The number of read cycles.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void SetInventoryCounts(
    int value);
```

Java and Android representation:
```java
public void SetInventoryCounts(
    int value)
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_SetInventoryCounts(
    CAENRFIDHandle handle,
    char *SourceName,
    int *value);
```
SetInventoryDwellTime Method

Description:
This method can be used to sets the inventory execution time (msec) used by the logical source during the inventory algorithm execution.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The execution time in ms</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
public int SetInventoryDwellTime(
    int value);

Java and Android representation:
public int SetInventoryDwellTime(
    int value)
    throws CAENRFIDException

C representation:
CAENRFIDErrorCodes CAENRFID_SetInventoryDwellTime(
    CAENRFIDHandle handle,
    char *SourceName,
    int value);

SetInventoryQuietTime Method

Description:
This method can be used to sets the inventory quiet time (msec) used by the logical source during the inventory algorithm execution.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The quiet time in ms</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
public int SetInventoryQuietTime(
    int value);

Java and Android representation:
public int SetInventoryQuietTime(
    int value)
    throws CAENRFIDException

C representation:
CAENRFIDErrorCodes CAENRFID_SetInventoryQuietTime(
    CAENRFIDHandle handle,
    char *SourceName,
    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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The initial Q value setting.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void SetQ_EPC_C1G2(
    int Value)
```
Java and Android representation:
```java
public void SetQ_EPC_C1G2(
    int Value)
throws CAENRFIDException
```
C representation:
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The number of read cycles.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void SetReadCycle(
    int value)
```
Java and Android representation:
```java
public void SetReadCycle(
    int value)
throws CAENRFIDException
```
C representation:
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The Selected flag value.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void SetSelected_EPC_C1G2(
    CAENRFIDLogicalSourceConstants Value)
```

**Java and Android representation:**
```java
public void SetSelected_EPC_C1G2(
    CAENRFIDLogicalSourceConstants Value)
```

**C representation:**
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The Session value.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void SetSession_EPC_C1G2(
    CAENRFIDLogicalSourceConstants Value)
```

**Java and Android representation:**
```java
public void SetSession_EPC_C1G2(
    CAENRFIDLogicalSourceConstants Value)
```

**C representation:**
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The Target value.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void SetTarget_EPC_C1G2(
    CAENRFIDLogicalSourceConstants Value)
```

Java and Android representation:
```java
public void SetTarget_EPC_C1G2(
    CAENRFIDLogicalSourceConstants Value)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_SetTarget_EPC_C1G2(
    CAENRFIDHandle handle,
    char *SourceName,
    CAENRFIDLogicalSourceConstants Value);
```

SL900A_EndLog Method

Description:
This method can be used to issue an IDS SL900A EndLog custom command as defined in the IDS SL900A datasheet.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The tag where stop the log</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void SL900A_EndLog(
    CAENRFIDTag Tag)
```

Java and Android representation:
```java
public void SL900A_EndLog(
    CAENRFIDTag Tag)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_IDS_SL900A_EndLog(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag);
```
SL900A_GetLogState Method

**Description:**
This method can be used to issue an IDS SL900A Get Log State custom command as defined in the IDS SL900A datasheet.

**Parameters:**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The tag selected</td>
</tr>
<tr>
<td>ShelfLife</td>
<td>This parameter is used to inform the reader if the shelf life flag is set in the tag's EEPROM</td>
</tr>
</tbody>
</table>

**Return Value:**
This method returns the status of the logging process. The structure of the byte array is the following:
- byte[0]: Limite Counter.
- byte[2]: System status.
- byte[4]: Shelf Life Block (only if the ShelfLife parameter is true).
- byte[12]: Current Shelf Life (only if the ShelfLife parameter is true).
- byte[15]: Status Flags (if ShelfLife parameter is false this byte follows immediately the System status word).

**Syntax:**
- **C# representation:**
  ```csharp
  public byte[] SL900A_GetLogState(
      CAENRFIDTag Tag,
      bool ShelfLife)
  ```
- **Java and Android representation:**
  ```java
  public byte[] SL900A_GetLogState(
      CAENRFIDTag Tag,
      boolean ShelfLife)
  throws CAENRFIDException
  ```
- **C representation:**
  ```c
  CAENRFIDErrorCodes IDS_SL900A_GetLogState(
      CAENRFIDHandle handle,
      CAENRFIDTag *Tag,
      BOOL ShelfLife,
      char *TRData);
  ```
**SL900A_GetMeasurementSetup Method**

*Description:*
This method gets the current system setup as defined in the IDS SL900A datasheet.

*Parameters:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The target tag from which read the current setup.</td>
</tr>
</tbody>
</table>

*Syntax:*

**C# representation:**
```
public byte[] SL900A_GetMeasurementSetup(
    CAENRFIDTag Tag)
```

**Java and Android representation:**
```
public byte[] SL900A_GetMeasurementSetup(
    CAENRFIDTag Tag)
    throws CAENRFIDException;
```

**C representation:**
```
CAENRFIDErrorCodes CAENRFID_IDS_SL900A_GetMeasurementSetup(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    byte **SetupData,
    int *SetupDataLength);
```

*Remarks:*
According to the IDS SL900A datasheet, the current setup is returned as:
SL900A_GetSensorValue Method

Description:
This method can be used to issue an IDS SL900A Get Sensor Value custom command as defined in the IDS SL900A datasheet.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The tag to extract sensor data.</td>
</tr>
<tr>
<td>SensorType</td>
<td>Describes which sensor to choose. (see remark)</td>
</tr>
</tbody>
</table>

Return Value:
Returns an IDSTagData object containing all the data read from the tag's selected sensor.

Syntax:

C# representation:
```csharp
public IDSTagData SL900A_GetSensorValue(
    CAENRFIDTag Tag,
    byte SensorType)
```

Java and Android representation:
```java
public IDSTagData SL900A_GetSensorValue(
    CAENRFIDTag Tag,
    byte SensorType)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_IDS_SL900A_GetSensorValue(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    byte SensorType,
    CAENRFID_IDSTagData *IDSTagData);
```

Remarks:
According to the IDS SL900A datasheet, the Sensor Type byte is composed as:
bit 07..02: Extreme Lower
bit 01..00: Sensor Type.
Sensor type bits can be:
00: Temperature sensor
01: External sensor 1.
10: External sensor 2.
11: Battery Voltage.
SL900A_Initialize Method

**Description:**
This method can be used to issue an IDS SL900A Initialize custom command as defined in the IDS SL900A datasheet.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The tag to initialize</td>
</tr>
<tr>
<td>DelayTime</td>
<td>The DelayTime parameter. See the IDS SL900A datasheet for further details.</td>
</tr>
<tr>
<td>ApplicationData</td>
<td>The Application data. See the IDS SL900A datasheet for further details.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void SL900A_Initialize(
    CAENRFIDTag Tag,
    ushort DelayTime,
    ushort ApplicationData)
```

**Java and Android representation:**
```java
public void SL900A_Initialize(
    CAENRFIDTag Tag,
    short DelayTime,
    short ApplicationData)
throws CAENRFIDException
```

**C representation:**
```c
CAENRFIDErrorCodes CAENRFID_IDS_SL900A_Initialize(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    unsigned short DelayTime,
    unsigned short ApplicationData);
```

**Remarks:**
According to the IDS SL900A datasheet, the DelayTime parameter is composed as:
- bit 15..4: Delay time (expressed in seconds)
- bit 3..2: RFU
- bit 1: Delay mode (0 : Internal timer, 1 : External switch)
- bit 0: IRQ + Timer Enable

According to the IDS SL900A datasheet, the Application Data parameter is composed as:
- bit 15..7: Application Area size (in words)
- bit 6..3: RFU bit 2..0 : Broken word pointer
**SL900A_SetLogLimits Method**

**Description:**
This method sets 4 limits which can be used for logging measurement data as defined in the IDS SL900A datasheet.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The target tag where apply the limits.</td>
</tr>
<tr>
<td>LogLimits</td>
<td>An array of 4 bytes containing the forty bits representing limit values. See the IDS SL900A datasheet for further details.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void SL900A_SetLogLimits(
    CAENRFIDTag Tag,
    byte[] LogLimit)
```

**Java and Android representation:**
```java
public void SL900A_SetLogLimits(
    CAENRFIDTag Tag,
    byte[] LogLimits)
    throws CAENRFIDException
```

**C representation:**
```c
CAENRFIDErrorCodes CAENRFID_IDS_SL900A_SetLogLimits(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    byte *LogLimits,
    int LogLimitsLength);
```

**Remarks:**

According to the IDS SL900A datasheet, each of the 4 limits is 10 bit long. The LogLimits parameter is composed as:

- bits 39..30: Extreme Lower
- bits 29..20: Lower
- bits 19..10: Upper
- bits 9..0: Extreme Upper
SL900A_SetLogMode Method

Description:
This method can be used to issue an IDS SL900A Set Log Mode custom command as defined in the IDS SL900A datasheet.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The tag to set log mode on.</td>
</tr>
<tr>
<td>LogMode</td>
<td>The LogMode parameter. See the IDS SL900A datasheet for further details.</td>
</tr>
</tbody>
</table>

Syntax:

**C# representation:**

```csharp
public void SL900A_SetLogMode(
    CAENRFIDTag Tag,
    uint LogMode)
```

**Java and Android representation:**

```java
public void SL900A_SetLogMode(
    CAENRFIDTag Tag,
    int LogMode)
```

**C representation:**

```c
CAENRFIDErrorCodes CAENRFID_IDS_SL900A_SetLog(
    CAENRFIDHandle handle,
    CAENRFIDTag *Tag,
    unsigned int LogMode);
```

Remarks:
According to the IDS SL900A datasheet, the DelayTime parameter is composed as:

- bit 31..24: RFU.
- bit 23..21: Logging Form.
- bit 20: Storage Rule.
- bit 19: Ext1 sensor enable.
- bit 18: Ext2 sensor enable.
- bit 17: Temperature sensor enable.
- bit 16: Battery Check enable.
- bit 15..0: Log Interval.
- bit 0: RFU.
SL900A_StartLog Method

Description:
This method can be used to issue an IDS SL900A Start Log custom command as defined in the IDS SL900A datasheet.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The Tag where start logging.</td>
</tr>
<tr>
<td>StartTime</td>
<td>The start time. See remark for structures.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void SL900A_StartLog(CAENRFIDTag Tag, uint StartTime)
```

Java and Android representation:
```java
public void SL900A_StartLog(CAENRFIDTag Tag, int StartTime)
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_IDS_SL900A_StartLog(CAENRFIDHandle handle, CAENRFIDTag *Tag, unsigned int StartTime);
```

Remarks:
According to the IDS SL900A datasheet, the StartTime parameter is composed as:

- bit 31..26: Year
- bit 25..21: Month
- bit 15..11: Hour
- bit 10..6: Minute
- bit 5..0: Second.
Untraceable_EPC_C1G2 Method

Description:
This method allows an interrogator with an asserted Untraceable privilege to instruct a Tag to (a) alter the L and U bits in EPC memory, (b) hide memory from interrogators with a deasserted Untraceable privilege and/or (c) reduce its operating range for all interrogators.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be untraced</td>
</tr>
<tr>
<td>u</td>
<td>A boolean value for the U bit in XPC_W1 word.</td>
</tr>
<tr>
<td>hideEPC</td>
<td>Specify whether a Tag untraceably hides part of EPC memory</td>
</tr>
<tr>
<td>hideTID</td>
<td>Specify whether a Tag untraceably hides part of TID memory (Allowed values: 0,1,2)</td>
</tr>
<tr>
<td>hideUSER</td>
<td>Specify whether a Tag untraceably hides part of USER memory</td>
</tr>
<tr>
<td>range</td>
<td>A value specifying a tag’s operating range in terms of reading distance (allowed values: 0,1,2).</td>
</tr>
<tr>
<td>newEPCLen</td>
<td>A value specifying a new EPC length field</td>
</tr>
<tr>
<td>password</td>
<td>The access password</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void Untraceable_EPC_C1G2(
    CAENRFIDTag Tag,
    bool u,
    bool hideEPC,
    byte hideTID,
    bool hideUser,
    ushort range,
    ushort newEPCLen,
    uint password )
```

Java and Android representation:
```java
public void Untraceable_EPC_C1G2(
    CAENRFIDTag Tag,
    boolean u,
    boolean hideEPC,
    boolean hideTID,
    boolean hideUser,
    short range,
    short newEPCLen,
    int password )
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_Untraceable_EPC_C1G2(
    CAENRFIDHandle *handle,
    char* SourceName,
    CAENRFIDTag *Tag,
    BOOL u,
    BOOL hideEPC,
    byte hideTID,
    BOOL hideUser,
    unsigned short range,
    unsigned short newEPCLen,
    int password);
```
WriteTagData Method

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

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be written.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start writing the data.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be written.</td>
</tr>
<tr>
<td>Data</td>
<td>The data to be written into the tag's user memory.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void WriteTagData(
    CAENRFIDTag Tag,
    short Address,
    short Length,
    byte[] Data)
```

Java and Android representation:
```java
public void WriteTagData(
    CAENRFIDTag Tag,
    short Address,
    short Length,
    byte[] Data)
     throws CAENRFIDException
```

C representation:
```c
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 an ISO18000-6C (EPC Class1 Gen2) tag.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be written.</td>
</tr>
<tr>
<td>MemBank</td>
<td>The memory bank where to write the data.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start writing the data.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be written.</td>
</tr>
<tr>
<td>Data</td>
<td>An array of bytes representing the data to be written into the tag.</td>
</tr>
</tbody>
</table>

Syntax:

**C# representation:**

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

**Java and Android representation:**

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

**C representation:**

```c
CAENFIDErrorCodes CAENFID_WriteTagData_EPC_C1G2(
    CAENFIDHandle 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 an ISO18000-6C (EPC Class1 Gen2) tag after having put the tag in Secured state using the Access command.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>The CAENRFIDTag representing the tag to be written.</td>
</tr>
<tr>
<td>MemBank</td>
<td>The memory bank where to write the data.</td>
</tr>
<tr>
<td>Address</td>
<td>The address where to start writing the data.</td>
</tr>
<tr>
<td>Length</td>
<td>The number of byte to be written.</td>
</tr>
<tr>
<td>Data</td>
<td>An array of bytes representing the data to be written into the tag.</td>
</tr>
<tr>
<td>AccessPassword</td>
<td>The access password.</td>
</tr>
</tbody>
</table>

**Syntax:**

C# representation:

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

Java and Android representation:

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

C representation:

```c
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 an ISO18000-6C (EPC Class1 Gen2) tag.

Parameters:

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

Syntax:

C# representation:
```csharp
public void WriteTagData_EPC_C1G2(
    short BankMask,
    short PositionMask,
    short LengthMask,
    byte[] Mask,
    short MemBank,
    short Address,
    short Length,
    byte[] Data)
```

Java and Android representation:
```java
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:
```c
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 an ISO18000-6C (EPC Class1 Gen2) tag after having put the tag in Secured state using the Access command.

**Parameters:**

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

**Syntax:**

**C# representation:**

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

**Java and Android representation:**

```java
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:**

```c
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 Java, Android and C# languages this class is composed by methods while in C language is present as a struct (for more information see § Overview on SDK page 8):

C representation:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>byte</code></td>
<td>ID[MAX_ID_LENGTH]</td>
</tr>
<tr>
<td><code>short</code></td>
<td>Length</td>
</tr>
<tr>
<td><code>char</code></td>
<td>LogicalSource[MAX_LOGICAL_SOURCE_NAME]</td>
</tr>
<tr>
<td><code>char</code></td>
<td>ReadPoint[MAX_READPOINT_NAME]</td>
</tr>
<tr>
<td><code>CAENRFIDProtocol</code></td>
<td>Type</td>
</tr>
<tr>
<td><code>short</code></td>
<td>RSSI</td>
</tr>
<tr>
<td><code>byte</code></td>
<td>TID[MAX_TID_SIZE]</td>
</tr>
<tr>
<td><code>short</code></td>
<td>TIDLen</td>
</tr>
<tr>
<td><code>byte</code></td>
<td>XPC[XPC_LENGTH]</td>
</tr>
<tr>
<td><code>byte</code></td>
<td>PC[PC_LENGTH]</td>
</tr>
</tbody>
</table>

```c
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;
    byte          XPC[XPC_LENGTH];
    byte          PC[PC_LENGTH];
} 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:**
```c
public DateTime getDate()
```

**Java and Android representation:**
```java
public java.util.Date getDate()
```

getPC Method

**Description:**
This method represents the PC code in the tag.

**Return value:**
The tag's Protocol Control code.

**Syntax:**

**C# representation:**
```c
public byte[] getPC()
```

**Java and Android representation:**
```java
public byte[] getPC()
```
**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:**
```csharp
public string getReadPoint()
```

**Java and Android representation:**
```java
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:**
```csharp
public short getRSSI()
```

**Java and Android representation:**
```java
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:**
```csharp
public CAENRFIDTagEventType getStatus()
```

**Java and Android representation:**
```java
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:**
```csharp
public byte[] getTagID()
```

**Java and Android representation:**
```java
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 and Android 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 and Android 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 and Android 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 and Android representation:
```
public java.lang.String getAntenna()
```
getXPC Method

Description:
This method returns the tag’s XPC words.

Return value:
The tag’s XPC words.

Syntax:
C# representation:  
public byte[] getXPC()  

Java and Android representation:  
public byte[] getXPC()  

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

Connect Method (CAENRFIDPort, string)

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 page 9 for more information.
For android Bluetooth connection see below § Connect Method (BluetoothSocket)

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConType</td>
<td>The communication link to use for the connection.</td>
</tr>
<tr>
<td>Address</td>
<td>Depending on ConType parameter: IP address for TCP/IP communications (&quot;xxx.xxx.xxx.xxx&quot;), COM port for RS232 communications (&quot;COMx&quot;), an index for USB communications (not yet supported). To specify a TCP port separate address and port by a semi-colon (ex: &quot;192.168.0.1:2300&quot;).</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
public void Connect(CAENRFIDPort ConType, string Address)

Java and Android representation:
public void Connect(CAENRFIDPort ConType, java.lang.String Address) throws CAENRFIDException

Connect Method (BluetoothSocket)

Description:
Start the android SPP bluetooth communication with the CAEN RFID Reader. This method must be called before any other methods of the Reader object.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTSock</td>
<td>The BluetoothSocket to read/write data.</td>
</tr>
</tbody>
</table>

Syntax:
Android representation:
public void Connect(BluetoothSocket BTSock) throws CAENRFIDException

Remarks
The BTSock parameter must be obtained through a createRfcomSocketToServiceRecord(UUID uuid) call.
The standard UUID for the Serial Port Profile is 00001101-0000-1000-8000-00805F9B34FB.
Connect Method (VCPSerialPort)

Description:
Start the android VCP communication with the CAEN RFID Reader. This method must be called before any other methods of the Reader object.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>The vcp port used to communicate with reader.</td>
</tr>
</tbody>
</table>

Syntax:

Android representation:

```java
public void Connect(VCPSerialPort port)
throws CAENRFIDException
```

Remarks:
To find the VCP Port of the CAENRFID USB readers attached to the Android device, use the VCPSerialPort.findVCPDevice(Context) class method where the Context parameter is an UsbManager object, and pick the one (usually the first) related to the usb port of the reader.

To use a VCPSerialPort and get an UsbManager objects, the user application should obtain the permission to use the USB System Service of the Android OS.
See the Android USB topics on the developer.android.com site for further details.

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 page 9 for more information.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConType</td>
<td>The communication link to use for the connection.</td>
</tr>
<tr>
<td>Address</td>
<td>Depending on ConType parameter: IP address for TCP/IP communications (&quot;xxx.xxx.xxx.xxx&quot;), COM port for RS232 communications (&quot;COMx&quot;), an index for USB communications (not yet supported). To specify a TCP port separate address and port by a semi-colon (ex: &quot;192.168.0.1:2300&quot;).</td>
</tr>
<tr>
<td>handle</td>
<td>The handle that identifies the device.</td>
</tr>
</tbody>
</table>

Syntax:

C representation:

```c
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 page 9 for more information.

Syntax:

C# representation:

```c
public void Disconnect()
```

Java and Android representation:

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

End Function

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 page 9 for more information.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>handle</td>
<td>The handle that identifies the device.</td>
</tr>
</tbody>
</table>

**Syntax:**

C representation:

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

### GetBatteryLevel Method

**Description:**

This method gets the current battery charge.

**Return value:**

The current charge level expressed in %.

**Syntax:**

C# representation:

```c
public int GetBatteryLevel()
```

Java and Android representation:

```java
public int GetBatteryLevel() throws CAENRFIDException
```

C representation:

```c
CAENRFIDErrorCodes CAENRFID_GetBatteryLevel (    CAENRFIDHandle handle,
    unsigned int *Charge);
```

### 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:

```c
public CAENRFIDBitRate GetBitRate()
```

Java and Android representation:

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

C representation:

```c
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:
```csharp
public string GetFirmwareRelease()
```
Java and Android representation:
```java
public java.lang.String GetFirmwareRelease()
```

C representation:
```c
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 depend on the Reader’s model. Please refer to the corresponding user manual available at [www.caenrfid.com](http://www.caenrfid.com).

Syntax:
C# representation:
```csharp
public int GetIO()
```
Java and Android representation:
```java
public int GetIO()
```

C representation:
```c
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 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.

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

Syntax:
C# representation:
```csharp
public int GetIODirection()
```
Java and Android representation:
```java
public int GetIODirection()
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_GetIODirection(CAENRFIDHandle handle, unsigned int *IODirection);
```
GetFHSSMode Method

Description:
This method gets the current Frequency Hopping status.

Return value:
A zero value if the FHSS is disabled, non-zero value if it is enabled.

Syntax:
C# representation:
```csharp
public short GetFHSSMode()
```

Java and Android representation:
```java
public short GetFHSSMode()
    throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_GetFHSSMode(
    CAENRFIDHandle handle,
    unsigned short *FHSSMode);
```

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:
```csharp
public int GetPower()
```

Java and Android representation:
```java
public int GetPower()
    throws CAENRFIDException
```

C representation:
```c
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:**
```csharp
public CAENRFIDProtocol GetProtocol()
```

**Java and Android representation:**
```java
public CAENRFIDProtocol GetProtocol()
```
```java
throws CAENRFIDException
```

**C representation:**
```c
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:**
```csharp
public CAENRFIDReaderInfo GetReaderInfo()
```

**Java and Android representation:**
```java
public CAENRFIDReaderInfo GetReaderInfo()
```
```java
throws CAENRFIDException
```

**C representation:**
```c
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:**
```csharp
public string[] GetReadPoints()
```

**Java and Android representation:**
```java
public java.lang.String[] GetReadPoints()
```

**C representation:**
```c
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadPoint</td>
<td>The name of the read point to check.</td>
</tr>
</tbody>
</table>

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

Syntax:

C# representation:
```
public CAENRFIDReadPointStatus GetReadPointStatus(
    string ReadPoint)
```

Java and Android 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 changes for different country regulations.

Return value:
The RF channel index.

Syntax:

C# representation:
```
public short GetRFChannel()
```

Java and Android 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:
```csharp
public CAENRFIDRFRegulations GetRFRegulation()
```

Java and Android representation:
```java
public CAENRFIDRFRegulations GetRFRegulation() throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_GetRFRegulation(CAENRFIDHandle handle, CAENRFIDRFRegulations *RFRegulation);
```

GetSource Method

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

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>The name of the logical source.</td>
</tr>
</tbody>
</table>

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

Syntax:
C# representation:
```csharp
public CAENRFIDLogicalSource GetSource(string Source)
```

Java and Android representation:
```java
public CAENRFIDLogicalSource GetSource(java.lang.String Source) throws CAENRFIDException
```

Remarks:
This function does not exist in C language, see § Overview on SDK page 8 for more information.
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 and Android 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 and Android representation:
public CAENRFIDLogicalSource[] GetSources()

Remarks:
This function does not exist in C language, see § Overview on SDK page 8 for more information.

InventoryAbort Method

For the description of this method, see § EVENT HANDLING page 99.
## MatchReadPointImpedance Method

### MatchReadPointImpedance (String)

**Description:**
MatchReadPointImpedance matches the antenna impedance passed in ReadPoint.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadPoint</td>
<td>The antenna to be matched</td>
</tr>
</tbody>
</table>

**Return value:**
A real number greater than one, that represents the return status of the matching operation.

**Syntax:**

**C# representation:**
```csharp
public float MatchReadPointImpedance(
    string ReadPoint)
```

**Java and Android representation:**
```java
public float MatchReadPointImpedance(
    String ReadPoint
) throws CAENRFIDException
```

**C representation:**
```c
CAENRFIDMatchReadPointImpedance (
    CAENRFIDHandle handle,
    char *ReadPoint,
    float *Value);
```

### MatchReadPointImpedance (String, CAENRFIDMatchingParams, Int16)

**Description:**
MatchReadPointImpedance matches the antenna impedance passed in ReadPoint.

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadPoint</td>
<td>The antenna to be matched</td>
</tr>
<tr>
<td>MatchParam</td>
<td>A CAENRFIDMatchingParams parameters for matching operation</td>
</tr>
<tr>
<td>MatchParamValue</td>
<td>The value of the MatchParam</td>
</tr>
</tbody>
</table>

**Return value:**
A real number greater than one, that represents the return status of the matching operation.

**Syntax:**

**C# representation:**
```csharp
public float MatchReadPointImpedance(
    string ReadPoint,
    CAENRFIDMatchingParams MatchParam,
    short MatchParamValue)
```

**Java and Android representation:**
```java
public float MatchReadPointImpedance(
    String ReadPoint,
    CAENRFIDMatchingParams MatchParam,
    short MatchParamValue
) throws CAENRFIDException
```

**C representation:**
```c
CAENRFIDErrorCodes CAENRFID_MatchReadPointImpedance (
    CAENRFIDHandle handle,
    *ReadPoint,
    *MatchParam,
    *MatchParamValue,
    *Value);
```
PrintScreen Method

*Description:*
Print ASCII text on the reader's screen (only for readers with display, e.g. R1170i qIDmini).

*Parameters:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>An arbitrary ASCII string.</td>
</tr>
<tr>
<td>TerminalType</td>
<td>RFU parameter, default is 0 (VT100).</td>
</tr>
</tbody>
</table>

*Syntax:*

**C# representation:**

```csharp
public void PrintScreen(
    string Text,
    short TerminalType)
```

**Java and Android representation:**

```java
public void PrintScreen(
    string Text,
    int TerminalType)
```

**C representation:**

```c
CAENRFIDErrorCodes CAENRFID_PrintScreen(
    CAENRFIDHandle handle,
    char *Text,
    unsigned short TerminalType);
```

RFControl Method

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

*Parameters:*

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnOff</td>
<td>The value to set. 1 generates the CW, 0: stops the CW generation.</td>
</tr>
</tbody>
</table>

*Syntax:*

**C# representation:**

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

**Java and Android representation:**

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

**C representation:**

```c
CAENRFIDEErrorCodes 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitRate</td>
<td>The RF bit rate value to set.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void SetBitRate(CAENRFIDBitRate BitRate)
```

Java and Android representation:
```java
public void SetBitRate(CAENRFIDBitRate BitRate)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_SetBitRate(
    CAENRFIDHandle handle,
    CAENRFID_Bitrate BitRate);
```

SetDateTime Method

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

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateTime</td>
<td>The Date/Time to be set on the reader as a string in the format: &quot;yyyy-mm-dd hh:mm:ss&quot;.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
public void SetDateTime(string DateTime)
```

Java and Android representation:
```java
public void SetDateTime(java.lang.String DateTime)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_SetDateTime(
    CAENRFIDHandle handle,
    char *DateTime);
```
SetIO Method

Description:
This method sets the Output lines value.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOValue</td>
<td>A bitmask representing the I/O lines value. The format and the meaning of the bits depend on the reader’s model. Please refer to the corresponding user manual available on <a href="http://www.caenrfid.com">www.caenrfid.com</a></td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void SetIO(int IOValue)
```

Java and Android representation:
```java
public void SetIO(int IOValue)
throws CAENRFIDException
```

C representation:
```c
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 represents an 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IODirection</td>
<td>The IODirection value to set.</td>
</tr>
</tbody>
</table>

Syntax:

C# representation:
```csharp
public void SetIODIRECTION(int IODirection)
```

Java and Android representation:
```java
public void SetIODIRECTION(int IODirection)
throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_SetIODirection(CAENRFIDHandle handle, unsigned int IODirection);
```
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPAddress</td>
<td>The IP address to set on the reader network interface.</td>
</tr>
<tr>
<td>NetMask</td>
<td>The netmask to set on the reader network interface.</td>
</tr>
<tr>
<td>Gateway</td>
<td>The gateway to set on the reader network interface.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void SetNetwork(string IPAddress, string NetMask, string Gateway)
```

**Java and Android representation:**
```java
```

**C representation:**
```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>The conducted RF power value expressed in mW.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**
```csharp
public void SetPower(int power)
```

**Java and Android representation:**
```java
public void SetPower(int power)
```

**C representation:**
```c
CAENRFIDErrorCodes CAENRFID_SetPower(CAENRFIDHandle handle, unsigned int Power);
```
SetProtocol Method

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

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>The CAENRFIDProtocol representing the air protocol to be set.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**

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

**Java and Android representation:**

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

**C representation:**

```c
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:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>The RF channel index to be set.</td>
</tr>
</tbody>
</table>

**Syntax:**

**C# representation:**

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

**Java and Android representation:**

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

**C representation:**

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

**Remarks**
This method is only used for testing applications.
## SetRS232 Method

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

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baud</td>
<td>The baud rate value to set.</td>
</tr>
<tr>
<td>datab</td>
<td>The number of data bits to set.</td>
</tr>
<tr>
<td>stopb</td>
<td>The number of stop bits to set.</td>
</tr>
<tr>
<td>parity</td>
<td>The parity value to set.</td>
</tr>
<tr>
<td>flowc</td>
<td>The flow control value to set.</td>
</tr>
</tbody>
</table>

**Syntax:**

- **C# representation:**
  ```csharp
  public void SetRS232(int baud, int datab, int stopb, CAENRFIDRS232Constants parity, CAENRFIDRS232Constants flowc);
  ```

- **Java and Android representation:**
  ```java
  public void SetRS232(int baud, int datab, int stopb, CAENRFIDRS232Constants parity, CAENRFIDRS232Constants flowc)
  throws CAENRFIDException;
  ```

- **C representation:**
  ```c
  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:
  ```csharp
  public string GetModel()
  ```

  Java and Android representation:
  ```java
  public java.lang.String GetModel()
  ```

- **Remarks:**
  This method does not exist in C language. It is possible to use the GetReaderInfo Method page 83 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:
  ```csharp
  public string GetSerialNumber()
  ```

  Java and Android representation:
  ```java
  public java.lang.String GetSerialNumber()
  ```

- **Remarks:**
  This method does not exist in C language. It is possible to use the GetReaderInfo Method page 83 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# language this class is composed by methods while in C language the following struct is present (for more information see § Overview on SDK page 8):

C representation:
```c
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;
    byte XPC[XPC_LENGTH];
    byte PC[PC_LENGTH];
} 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:
```c
public byte[] GetId()
```

Java and Android representation:
```c
public byte[] GetId()
```

GetLength Method

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

Return value:
The tag’s length.

Syntax:
C# representation:
```c
public short GetLength()
```

Java and Android representation:
```c
public short GetLength()
```
GetPC Method

Description:
This method returns the Protocol Control(PC) word code of the tag.

Return value:
The tag's Protocol Control code.

Syntax:
C# representation:
```csharp
public byte[] GetPC()
```
Java and Android representation:
```java
public byte[] GetPC()
```

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:
```csharp
public string GetReadPoint()
```
Java and Android representation:
```java
public java.lang.String GetReadPoint()
```
```java
throws CAENRFIDException
```

GetRSSI Method

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

Return value:
The tag's RSSI.

Syntax:
C# representation:
```csharp
public short GetRSSI()
```
Java and Android representation:
```java
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 and Android 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 and Android 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 and Android 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 and Android representation:
public CAENRFIDProtocol GetType()
GetXPC Method

*Description:*
This method returns the tag's XPC words.

*Return value:*
The tag's XPC words.

*Syntax:*

**C# representation:**
```csharp
public byte[] GetXPC()
```

**Java and Android representation:**
```java
public byte[] GetXPC()
```
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 indefinitely.

The continuous mode is obtained by setting to 1 both framed (bit 1) and continuous (bit 2) flags.

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

In readers equipped with button (like the qID R1240IE/IU and qIDmini R1170I readers), if the event trigger flag (bit 5) is enabled and the continuous mode is enabled (bit 1 and bit 2), the event handler is recalled every time the button is pressed.

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

No other methods can be invoked on logical source and reader, during the continuous mode, nor inside the event handler. The only operation allowed is an inventory abort, that must be used to stop a reader which is working in continuous mode.

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:

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

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

Java and Android representation:
```java
public boolean EventInventoryTag(
    byte[] Mask,
    short MaskLength,
    short Position,
    short Flag)
```

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

CAENRFIDErrorCodes CAENRFID_EventInventoryTag (CAENRFID_Handle 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 performed 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 match 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.

<table>
<thead>
<tr>
<th>Flag value meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>RSSI: a 1 value indicates the reader will transmit the RSSI (Return Signal Strength Indicator) in the response.</td>
</tr>
<tr>
<td>Bit 1</td>
<td>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 transmitted all together at the end of the inventory cycle.</td>
</tr>
<tr>
<td>Bit 2</td>
<td>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.</td>
</tr>
<tr>
<td>Bit 3</td>
<td>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.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>TID reading: a 1 value indicates that also the TID of the tag will be returned by the reader together with the other information.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Event trigger: when this flag is set together with the continuous mode (continuous acquisition flag + framed data flag), the inventory cycle is performed in the same way of the continuous mode with the only difference that the inventory command is performed only by pressing the button of the qiD R1240I and qiDmini R1170I readers.</td>
</tr>
<tr>
<td>Bit 6</td>
<td>XPC: a 1 value allows the reader to get the XPC word if backscattered by a tag. Tags that do not backscatter the XPC words will return an XPC array with all the 4 bytes set to 0</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Match tag: a 1 value enables the matching of read tags with a tag present in the memory (A828BT reader only).</td>
</tr>
<tr>
<td>Bit 8</td>
<td>PC: a 1 value allows the reader to return the PC of a Gen2 tag in addition to the ID (A828BT reader only).</td>
</tr>
</tbody>
</table>

### InventoryAbort Method

**Description:**
This method stops the EventInventoryTag execution.

**Syntax:**

**C# representation:**
```csharp
public void InventoryAbort()
```

**Java and Android representation:**
```java
public void InventoryAbort() throws CAENRFIDException
```

**C representation:**
```c
CAENRFIDErrorCodes CAENRFID_InventoryAbort( CAENRFIDHandle handle);
```
ForceAbort Method

Description:
This method tries to stop a pending continuous inventory (see EventInventoryTag Method) that has not been stopped correctly by an InventoryAbort or Disconnect call. Choose the timeout value based on the expected reader’s load (large value if in presence of a large population of tags, small value if only few tags/seconds must be read).

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeout</td>
<td>The time (in ms) to wait for the end of the continuous inventory data.</td>
</tr>
</tbody>
</table>

Return value:
True if a continuous inventory end has been detected, false otherwise.

Syntax:

C# representation:
```csharp
public bool ForceAbort(
                    long timeout );
```

Java and Android representation:
```java
public boolean ForceAbort(
                    long timeout) throws CAENRFIDException
```

C representation:
```c
CAENRFIDErrorCodes CAENRFID_ForceAbort(
                    CAENRFIDHandle handle,
                    int timeout,
                    BOOL *endOfStreamMatched);
```

Remarks:
If continuous data stream is detected, the ForceAbort waits for its end even if the necessary amount of time exceeds the timeout parameter value.
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:
```csharp
public CAENRFIDNotify[] getData()
```

CAENRFIDEventHandler Delegate

CAENRFIDEventHandler delegate declaration.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>the Data Event.</td>
</tr>
</tbody>
</table>

Syntax:
C# representation:
```csharp
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:
```csharp
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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Represents the event object value.</td>
</tr>
</tbody>
</table>
Java and Android Event Handling

CAENRFIDEEvent 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 and Android representation:
```java
public java.util.ArrayList getData()
```

CAENRFIDEEventListener 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 and Android representation:
```java
void CAENRFIDTagNotify(CAENRFIDEvent evt)
```

**addCAENRFIDEEventListener**

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 and Android representation:
```java
public void addCAENRFIDEEventListener(CAENRFIDEEventListener listener)
```

**removeCAENRFIDEEventListener**

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 and Android representation:
```java
public void removeCAENRFIDEEventListener(CAENRFIDEEventListener 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:
```c
typedef CAENRFIDErrorCodes (calls *CAENRFID_INVENTORY_CALLBACK)
  (const CAENRFIDNotify* Tags, const int Size);
```
CAENRFIDBitRate Enumeration

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

**Syntax:**

**C# representation:**

```csharp
public enum CAENRFIDBitRate
```

**Java and Android representation:**

```java
public final class CAENRFIDBitRate
```

**C representation:**

```c
typedef enum CAENRFID_Bitrate;
```

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

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSB_ASK_FM0_TX10RX40</td>
<td>DSB-ASK transmission modulation, FM0 return link encoding, 10 Kbit in transmission, 40 Kbit in reception.</td>
</tr>
<tr>
<td>DSB_ASK_FM0_TX40RX40</td>
<td>DSB-ASK transmission modulation, FM0 return link encoding, 40 Kbit in transmission, 40 Kbit in reception.</td>
</tr>
<tr>
<td>DSB_ASK_FM0_TX40RX160</td>
<td>DSB-ASK transmission modulation, FM0 return link encoding, 40 Kbit in transmission, 160 Kbit in reception.</td>
</tr>
<tr>
<td>DSB_ASK_FM0_TX160RX400</td>
<td>DSB-ASK transmission modulation, FM0 return link encoding, 160 Kbit in transmission, 400 Kbit in reception.</td>
</tr>
<tr>
<td>DSB_ASK_M2_TX40RX160</td>
<td>DSB-ASK transmission modulation, Miller (M=2) return link encoding, 40 Kbit in transmission, 160 Kbit in reception.</td>
</tr>
<tr>
<td>DSB_ASK_M4_TX40RX256</td>
<td>DSB-ASK transmission modulation, Miller (M=4) return link encoding, 40 Kbit in transmission, 256 Kbit in reception.</td>
</tr>
<tr>
<td>PR_ASK_FM0_TX40RX640</td>
<td>PR-ASK transmission modulation, FM0 return link encoding, 40 Kbit in transmission, 640 Kbit in reception.</td>
</tr>
<tr>
<td>PR_ASK_M2_TX40RX250</td>
<td>PR-ASK transmission modulation, Miller (M=2) return link encoding, 40 Kbit in transmission, 256 Kbit in reception.</td>
</tr>
<tr>
<td>PR_ASK_M4_TX40RX250</td>
<td>PR-ASK transmission modulation, Miller (M=4) return link encoding, 40 Kbit in transmission, 256 Kbit in recepction.</td>
</tr>
<tr>
<td>PR_ASK_M4_TX40RX256</td>
<td>PR-ASK transmission modulation, Miller (M=4) return link encoding, 40 Kbit in transmission, 256 Kbit in reception.</td>
</tr>
<tr>
<td>PR_ASK_M4_TX40RX300</td>
<td>PR-ASK transmission modulation, Miller (M=4) return link encoding, 40 Kbit in transmission, 300 Kbit in reception.</td>
</tr>
<tr>
<td>PR_ASK_M4_TX40RX320</td>
<td>PR-ASK transmission modulation, Miller (M=4) return link encoding, 40 Kbit in transmission, 320 Kbit in reception.</td>
</tr>
<tr>
<td>PR_ASK_M4_TX80RX320</td>
<td>PR-ASK transmission modulation, Miller (M=4) return link encoding, 80 Kbit in transmission, 320 Kbit in reception.</td>
</tr>
<tr>
<td>DSB_ASK_M8_TX40RX256</td>
<td>DSB-ASK transmission modulation, Miller (M=4) return link encoding, 80 Kbit in transmission, 320 Kbit in reception.</td>
</tr>
</tbody>
</table>
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 and Android representation:
public final class CAENRFIDLogicalSourceConstants

C representation:
typedef enum CAENRFIDLogicalSourceConstants;

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

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPC_C1G2_SESSION_S0</td>
<td>Session 0 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).</td>
</tr>
<tr>
<td>EPC_C1G2_SESSION_S1</td>
<td>Session 1 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).</td>
</tr>
<tr>
<td>EPC_C1G2_SESSION_S2</td>
<td>Session 2 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).</td>
</tr>
<tr>
<td>EPC_C1G2_SESSION_S3</td>
<td>Session 3 is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).</td>
</tr>
<tr>
<td>EPC_C1G2_TARGET_A</td>
<td>Target A is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).</td>
</tr>
<tr>
<td>EPC_C1G2_TARGET_B</td>
<td>Target B is selected for the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).</td>
</tr>
<tr>
<td>EPC_C1G2_SELECTED_YES</td>
<td>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).</td>
</tr>
<tr>
<td>EPC_C1G2_SELECTED_NO</td>
<td>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).</td>
</tr>
<tr>
<td>EPC_C1G2_ALL_SELECTED</td>
<td>All the tags are considered in the anticollision algorithm execution on the logical source (valid only for the EPC Class1 Gen2 air protocol).</td>
</tr>
<tr>
<td>ISO18006B_DESB_ON</td>
<td>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).</td>
</tr>
<tr>
<td>ISO18006B_DESB_OFF</td>
<td>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).</td>
</tr>
</tbody>
</table>
CAENRFIDLogicalSource.InventoryFlag Enumeration

The CAENRFIDLogicalSource.InventoryFlag Enumeration gives a list of constants used for the configuration of the inventory function that comes with Flag parameter.

Syntax:

C# representation:
```csharp
public enum CAENRFIDLogicalSource.InventoryFlag
```

Java and Android representation:
```java
public final class CAENRFIDLogicalSource.InventoryFlag
```

C representation:
```c
typedef enum CAENRFIDLogicalSource.InventoryFlag;
```

In the following table, the CAENRFIDLogicalSource.InventoryFlag Enumeration members are listed:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSSI</td>
<td>When enabled, the RSSI value representing the backscattered RF field strength is returned by the reader for each tag read. Some reader cannot have this feature.</td>
</tr>
<tr>
<td>FRAMED</td>
<td>Tags found in an inventory cycle are not buffered in reader and sent all together, but sent one by one as soon as a tag is detected. It is used in conjunction with the continuous flag.</td>
</tr>
<tr>
<td>CONTINUOUS</td>
<td>Enables the continuous mode acquisition. Logical source must have ReadCycle parameter set to 0.</td>
</tr>
<tr>
<td>COMPACT</td>
<td>Instruct the reader to not return any other information than the ID. Other values are fake and filled by the library.</td>
</tr>
<tr>
<td>TID_READING</td>
<td>Instruct the reader to return the TID memory. On some reader it must be used in conjunction with SetTIDLength to work more efficiently.</td>
</tr>
<tr>
<td>EVENT_TRIGGER</td>
<td>Work only in combination with continuous mode. In reader provided with identification button, it instructs the reader to do an inventory cycle only when the button is pressed.</td>
</tr>
<tr>
<td>XPC</td>
<td>It instructs the reader to return XPC. If no XPC is present on the tag, the XPC field of a tag is filled up with zero values.</td>
</tr>
<tr>
<td>PC</td>
<td>Instruct the reader to return the PC of the EPC bank for each inventoried tag.</td>
</tr>
</tbody>
</table>
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 and Android 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:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAENRFID_RS232</td>
<td>Serial port communication link.</td>
</tr>
<tr>
<td>CAENRFID_TCP</td>
<td>TCP/IP network communication link.</td>
</tr>
<tr>
<td>CAENRFID_USB</td>
<td>USB communication link.</td>
</tr>
</tbody>
</table>

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 and Android 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:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAENRFID_ISO18000_6b</td>
<td>ISO18000-6B air protocol.</td>
</tr>
<tr>
<td>CAENRFID_EPC119</td>
<td>EPC 1.19 air protocol.</td>
</tr>
<tr>
<td>CAENRFID_EPC_C1G1</td>
<td>EPCGlobal Class1 Gen1 air protocol.</td>
</tr>
<tr>
<td>CAENRFID_ISO18000_6a</td>
<td>ISO18000-6A air protocol.</td>
</tr>
<tr>
<td>CAENRFID_EPC_C1G2</td>
<td>EPCGlobal Class1 Gen2 (aka ISO18000-6C) air protocol.</td>
</tr>
<tr>
<td>CAENRFID_MULTIPROTOCOL</td>
<td>This value permits to use all the supported air protocol at the same time. Suggested setting only for demo purposes.</td>
</tr>
</tbody>
</table>
CAENRFIDReadPointStatus Enumeration

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

Syntax:

C# representation:
```csharp
public enum CAENRFIDReadPointStatus
```

Java and Android representation:
```java
public final class CAENRFIDReadPointStatus
```

C representation:
```c
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:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS_BAD</td>
<td>Bad antenna connection.</td>
</tr>
<tr>
<td>STATUS_GOOD</td>
<td>Good antenna connection.</td>
</tr>
<tr>
<td>STATUS_POOR</td>
<td>Poor antenna connection.</td>
</tr>
</tbody>
</table>

CAENRFIDRFRegulations Enumeration

The CAENRFIDRFRegulations gives a list of country radiofrequency regulations.

Syntax:
C# representation:
public enum CAENRFIDRFRegulations

Java and Android 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:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETSI_302208</td>
<td>ETSI_302208 radiofrequency regulation.</td>
</tr>
<tr>
<td>ETSI_300220</td>
<td>ETSI_300220 radiofrequency regulation.</td>
</tr>
<tr>
<td>MALAYSIA</td>
<td>MALAYSIA radiofrequency regulation.</td>
</tr>
<tr>
<td>JAPAN</td>
<td>JAPAN radiofrequency regulation.</td>
</tr>
<tr>
<td>KOREA</td>
<td>KOREA radiofrequency regulation.</td>
</tr>
<tr>
<td>AUSTRALIA</td>
<td>AUSTRALIA radiofrequency regulation.</td>
</tr>
<tr>
<td>CHINA</td>
<td>CHINA radiofrequency regulation.</td>
</tr>
<tr>
<td>TAIWAN</td>
<td>TAIWAN radiofrequency regulation.</td>
</tr>
<tr>
<td>SINGAPORE</td>
<td>SINGAPORE radiofrequency regulation.</td>
</tr>
<tr>
<td>BRAZIL</td>
<td>BRAZIL radiofrequency regulation.</td>
</tr>
<tr>
<td>JAPAN_STD_T106 11</td>
<td>JAPAN radiofrequency regulation (ARIB STD-T106 Premises radio station (1W) - LBT free)</td>
</tr>
<tr>
<td>JAPAN_STD_T107 12</td>
<td>JAPAN radiofrequency regulation (ARIB STD-T107 Specified low power radio station (250mW) - with LBT)</td>
</tr>
<tr>
<td>PERU</td>
<td>PERU radiofrequency regulation.</td>
</tr>
<tr>
<td>SOUTH_AFRICA</td>
<td>SOUTH_AFRICA radiofrequency regulation.</td>
</tr>
<tr>
<td>CHILE</td>
<td>CHILE radiofrequency regulation.</td>
</tr>
</tbody>
</table>
CAENRFIDRS232Constants Enumeration

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

Syntax:
C# representation:
public enum CAENRFIDRS232Constants

Java and Android 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:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAENRS232_Parity_None</td>
<td>No parity bit is sent at all.</td>
</tr>
<tr>
<td>CAENRS232_Parity_Odd</td>
<td>Odd parity.</td>
</tr>
<tr>
<td>CAENRS232_Parity_Even</td>
<td>Even parity.</td>
</tr>
<tr>
<td>CAENRFID_RS232_FlowControl_XonXoff</td>
<td>Software flow control.</td>
</tr>
<tr>
<td>CAENRFID_RS232_FlowControl_Hardware</td>
<td>Hardware flow control.</td>
</tr>
<tr>
<td>CAENRFID_RS232_FlowControl_None</td>
<td>No flow control.</td>
</tr>
</tbody>
</table>
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 and Android representation:**

```
public final class CAENRFIDSelUnselOptions
```

**C representation:**

```
typedef enum CAENRFID_SelUnsel_Op;
```

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

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEL_EQUAL</td>
<td>select equal to.</td>
</tr>
<tr>
<td>SEL_NOT_EQUAL</td>
<td>select not equal to.</td>
</tr>
<tr>
<td>SEL_GREATER_THAN</td>
<td>select greater than.</td>
</tr>
<tr>
<td>SEL_LOWER_THAN</td>
<td>select lower than.</td>
</tr>
<tr>
<td>UNS_EQUAL</td>
<td>unselect equal to.</td>
</tr>
<tr>
<td>UNS_NOT_EQUAL</td>
<td>unselect not equal to.</td>
</tr>
<tr>
<td>UNS_GREATER_THAN</td>
<td>unselect greater than.</td>
</tr>
<tr>
<td>UNS_LOWER_THAN</td>
<td>unselect lower than.</td>
</tr>
</tbody>
</table>
CAENRFIDTag.MemBanks Enumeration

The CAENRFIDTag.MemBanks enumerates the bank name of a generic ISO18000-6C tag.

Syntax:
C# representation:
```csharp
public enum MemBanks {
    RESERVED = 0,
    EPC = 1,
    TID = 2,
    USER = 3
}
```

Java and Android representation:
```java
public enum MemBanks {
    RESERVED(0), EPC(1), TID(2), USER(3);
    private int code;
    private MemBanks(int c) { code = c; }
    public int getBankNum() { return code; }
}
```

C representation:
```c
typedef enum {
    RESERVED = 0,
    EPC = 1,
    TID = 2,
    USER = 3
} CAENRFIDMemBanks;
```

In the following table, the CAENRFIDTag.MemBanks Enumeration members are listed:

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVED</td>
<td>Indicates the reserved bank</td>
</tr>
<tr>
<td>EPC</td>
<td>Indicates the EPC bank</td>
</tr>
<tr>
<td>TID</td>
<td>Indicates the TID bank</td>
</tr>
<tr>
<td>USER</td>
<td>Indicates the USER bank</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Channel Class</strong></td>
<td></td>
</tr>
<tr>
<td>AddSource</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>AddTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetChannelStatus</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetChannelType</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetName</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>IsSourcePresent</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>IsTriggerPresent</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveSource</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td><strong>LogicalSource Class</strong></td>
<td></td>
</tr>
<tr>
<td>AddTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_BurstErase</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_BurstWrite</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgBlockGroupPassword</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgBlockLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgWordLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ReadBlockLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_Refresh</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetLostThreshold</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetObservedThreshold</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_BlockLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_BlockReadLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_GetSystemInformation</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_ReadLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_SetAttenuate</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_WriteMultipleWords</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Inventory</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>KillTag</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>LockTag</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>NXP_Calibrate</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>ProgramID</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>SetLostThreshold</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>SetObservedThreshold</td>
<td>This method is now obsolete.</td>
</tr>
</tbody>
</table>

**Reader Class**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectRS232</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>CreateChannel</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>CreateTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>FWUpgradeTFTP</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetAllocatedChannels</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetAllocatedTriggers</td>
<td>This method is now obsolete.</td>
</tr>
</tbody>
</table>
### C# Obsolete Members

<table>
<thead>
<tr>
<th>Member</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BitRate Enumeration</strong></td>
<td></td>
</tr>
<tr>
<td>TX10RX40</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td>TX40RX40</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td>TX40RX160</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td><strong>EventMode Enumeration</strong></td>
<td></td>
</tr>
<tr>
<td>READCYCLE_MODE</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td>TIME_MODE</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td>NOEVENT_MODE</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td><strong>TagEventType Enumeration</strong></td>
<td></td>
</tr>
<tr>
<td>TAG_GLIMPSED</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td>TAG_LOST</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td>TAG_OBSERVED</td>
<td>This member is now obsolete.</td>
</tr>
<tr>
<td>TAG_UNKNOWN</td>
<td>This member is now obsolete.</td>
</tr>
</tbody>
</table>

### Java and Android Obsolete Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BitRate Class</strong></td>
<td></td>
</tr>
<tr>
<td>TX10RX40</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>TX40RX40</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>TX40RX160</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td><strong>Channel Class</strong></td>
<td></td>
</tr>
<tr>
<td>AddSource</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>AddTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetChannelStatus</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetChannelType</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetName</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>IsSourcePresent</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>IsTriggerPresent</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveSource</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td><strong>Event Class</strong></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>This method is now obsolete. Use getData instead.</td>
</tr>
<tr>
<td><strong>EventMode Class</strong></td>
<td></td>
</tr>
<tr>
<td>READCYCLE_MODE</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>TIME_MODE</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>NOEVENT_MODE</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>LogicalSource Class</td>
<td></td>
</tr>
<tr>
<td>AddTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_BurstErase</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_BurstWrite</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgBlockGroupPassword</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgBlockLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgWordLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ReadBlockLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_Refresh</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetLostThreshold</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetObservedThreshold</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_GetSystemInfo</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_BlockLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_BlockReadLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_GetSystemInformation</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_ReadLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_SetAttenuate</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_WriteMultipleWords</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Inventory</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>NXP_Calibrate</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>NXP_ChangeEAS (only non secure version)</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>NXP_EAS_Alarm (only secure version)</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>NXP_ResetReadProtect (only secure version)</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>SetLostThreshold</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>SetObservedThreshold</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Notify Class</td>
<td></td>
</tr>
<tr>
<td>getAntenna</td>
<td>This method is now obsolete. Use getReadPoint instead.</td>
</tr>
<tr>
<td>Reader Class</td>
<td></td>
</tr>
<tr>
<td>CreateChannel</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>CreateTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>FWUpgradeTFTP</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetAllocatedChannels</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetAllocatedTriggers</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetChannelData</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetEventMode</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetLBMMode</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveChannel</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>RemoveTrigger</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>SetEventMode</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>SetReaderOptions</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Receiver Class</td>
<td></td>
</tr>
<tr>
<td>KillServer</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>TagEventType Class</td>
<td></td>
</tr>
<tr>
<td>TAG_GLIMPSED</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>TAG_LOST</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>TAG_OBSERVED</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>TAG_UNKNOWN</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Trigger Class</td>
<td></td>
</tr>
<tr>
<td>GetIOLineValue</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetName</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetTimerValue</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>IsLinkedToChannel</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>IsLinkedToSource</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>HitachiSysInfo</td>
<td></td>
</tr>
<tr>
<td>GetBankLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetBlockReadLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetBlockReadWriteLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetBlockWriteLock</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>GetInfoFlag</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>getReserved</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>getSetAttenuateLevel</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>getTID</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>getUI</td>
<td>This method is now obsolete.</td>
</tr>
<tr>
<td>getUser</td>
<td>This method is now obsolete.</td>
</tr>
</tbody>
</table>

Tab. 6.3: Java and Android Obsolete Methods
# C Obsolete Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddNotifyTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>AddReadTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>AddSourceToChannel</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>AllocateChannel</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>AllocateTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Custom_Cmd_C1G2</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>DeallocateChannel</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>DeallocateTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>ExtendedInventoryTag</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>FirmwareUpgrade</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>FreeNotifyMemory</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_BurstErase</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_BurstWrite</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgBlockGroupPassword</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgBlockLock</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ChgWordLock</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_ReadBlockLock</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Fujitsu_Refresh</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetAllocatedChannels</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetAllocatedTriggers</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetChannelData</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetChannelInTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetChannelStatus</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetDE_S8</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetEventMode</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetFirmwareRelease</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetLBTMode</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetNotification</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetQ_C1G2</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetQ_EPC_C1G2</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetReadPointInSource</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetSourceConfiguration</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetSourceInChannel</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetSourceInTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetSWRelease</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>GetTriggerInChannel</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_BlockLock</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_BlockReadLock</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_GetSystemInformation</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_ReadLock</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_SetAttenuate</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Hitachi_WriteMultipleWords</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Inventory</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>KillTag</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>KillTag_C1G2</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Lock</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>NXP_Calibrate</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>NXP_ChangeEAS</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>NXP_SecureCalibrate</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>NXP_SecureEAS_Alarm</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>NXP_SecureResetReadProtect</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ProgramID</td>
<td>This function is now obsolete. Use ProgramID_EPC_C1G1 instead.</td>
</tr>
<tr>
<td>ProgramID_C1G2</td>
<td>This function is now obsolete. Use ProgramID_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>QueryAck_C1G2</td>
<td>This function is now obsolete. Use QueryAck_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>QueryTag_C1G2</td>
<td>This function is now obsolete. Use Query_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>Read_C1G2</td>
<td>This function is now obsolete. Use ReadTagData instead.</td>
</tr>
<tr>
<td>Read_C1G2</td>
<td>This function is now obsolete. Use ReadTagData_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>RemoveNotifyTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>RemoveReadTrigger</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>RemoveSourceFromChannel</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>SecureCustomCmd_C1G2</td>
<td>This function is now obsolete. Use SecureCustomCommand_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>SecureLock_C1G2</td>
<td>This function is now obsolete. Use SecureLockTag_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>SecureProgramID_C1G2</td>
<td>This function is now obsolete. Use SecureProgramID_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>SecureRead_C1G2</td>
<td>This function is now obsolete. Use SecureReadTagData_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>SecureWrite_C1G2</td>
<td>This function is now obsolete. Use SecureWriteTagData_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>SetBitrate</td>
<td>This function is now obsolete. Use CAENRFID_SetBitRate instead.</td>
</tr>
<tr>
<td>SetDE_SB</td>
<td>This function is now obsolete. Use SetDESB_ISO1800006B instead.</td>
</tr>
<tr>
<td>SetEventMode</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>SetModulation</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>SetQ_C1G2</td>
<td>This function is now obsolete. Use SetQValue_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>SetQ_EPC_C1G2</td>
<td>This function is now obsolete. Use SetQValue_EPC_C1G2 instead.</td>
</tr>
<tr>
<td>SetSourceConfiguration</td>
<td>This function is now obsolete.</td>
</tr>
<tr>
<td>Write</td>
<td>This function is now obsolete. Use WriteTagData instead.</td>
</tr>
<tr>
<td>Write_C1G2</td>
<td>This function is now obsolete. Use WriteTagData_EPC_C1G2 instead.</td>
</tr>
</tbody>
</table>

Tab. 6.4: C Obsolete Functions
# C Obsolete Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CAENRFID_SOURCE_Parameter</strong></td>
<td></td>
</tr>
<tr>
<td>CONFIG_READCYCLE</td>
<td>This data type is now obsolete. Use Get/SetReadCycle Method instead.</td>
</tr>
<tr>
<td>CONFIG_OBSERVEDTHRESHOLD</td>
<td>This data type is now obsolete.</td>
</tr>
<tr>
<td>CONFIG_LOSTTHRESHOLD</td>
<td>This data type is now obsolete.</td>
</tr>
<tr>
<td>CONFIG_G2_Q_VALUE</td>
<td>This data type is now obsolete. Use Get/SetQ_EPC_C1G2 Method instead.</td>
</tr>
<tr>
<td>CONFIG_G2_SESSION</td>
<td>This data type is now obsolete. Use Get/Session_EPC_C1G2 Method instead.</td>
</tr>
<tr>
<td>CONFIG_G2_SESSION</td>
<td>This data type is now obsolete. Use Get/Session_EPC_C1G2 Method instead.</td>
</tr>
<tr>
<td>CONFIG_G2_SELECTED</td>
<td>This data type is now obsolete. Use Get/Selected_EPC_C1G2 Method instead.</td>
</tr>
<tr>
<td>CONFIG_ISO18006B_DESB</td>
<td>This data type is now obsolete. Use Get/Selected_ISO180006B Method instead.</td>
</tr>
<tr>
<td><strong>CAENRFID_EventMode</strong></td>
<td></td>
</tr>
<tr>
<td>READCYCLE_MODE</td>
<td>This data type is now obsolete.</td>
</tr>
<tr>
<td>TIME_MODE</td>
<td>This data type is now obsolete.</td>
</tr>
<tr>
<td>NOEVENT_MODE</td>
<td>This data type is now obsolete.</td>
</tr>
<tr>
<td><strong>CAENRFID_FWUpgradeType</strong></td>
<td></td>
</tr>
<tr>
<td>RFID_TFTP</td>
<td>This data type is now obsolete.</td>
</tr>
<tr>
<td><strong>CAENRFID_ExtendedInventoryParams</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This data type is now obsolete.</td>
</tr>
</tbody>
</table>

Tab. 6.5: C Obsolete Data Types