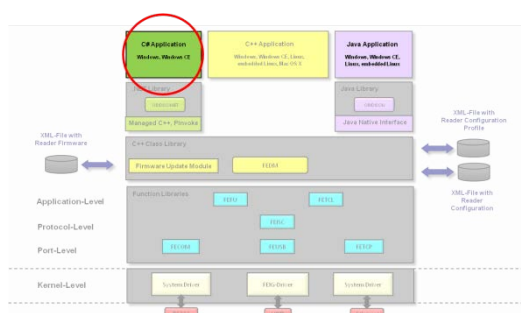


.NET Klassenbibliothek ID ISC.SDK.NET

Version 4.05.00

Software-Support für OBID i-scan® und OBID® classic-pro



.NET Framework	Ausführung		Betriebssysteme
	32-Bit (x86)	64-Bit (x64)	
V2.0 – 3.5	X		Windows XP / Vista / 7 / 8 (32- oder 64-Bit)
V4.0 und 4.5	X	X	Windows XP / Vista / 7 / 8 (32- oder 64-Bit)
Compact V2.0 und 3.5	X	-	Windows CE 5 und CE 6

Hinweis

© Copyright 2003-2013 by FEIG ELECTRONIC GmbH
Lange Straße 4
D-35781 Weilburg-Waldhausen
eMail: obid-support@feig.de

Alle früheren Ausgaben verlieren mit diesem Handbuch ihre Gültigkeit.
Die Angaben in diesem Handbuch können ohne vorherige Ankündigung geändert werden.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung ihres Inhalts sind nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlung verpflichtet zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder Gebrauchsmuster-Eintragung vorbehalten.

Die Zusammenstellung der Informationen in diesem Handbuch erfolgt nach bestem Wissen und Gewissen. FEIG ELECTRONIC GmbH übernimmt keine Gewährleistung für die Richtigkeit und Vollständigkeit der Angaben in diesem Handbuch. Insbesondere kann FEIG ELECTRONIC GmbH nicht für Folgeschäden aufgrund fehlerhafter oder unvollständiger Angaben haftbar gemacht werden. Da sich Fehler, trotz aller Bemühungen nie vollständig vermeiden lassen, sind wir für Hinweise jederzeit dankbar.

FEIG ELECTRONIC GmbH übernimmt keine Gewährleistung dafür, dass die in diesem Dokument enthaltenen Informationen frei von fremden Schutzrechten sind. FEIG ELECTRONIC GmbH erteilt mit diesem Dokument keine Lizenzen auf eigene oder fremde Patente oder andere Schutzrechte.

Die in diesem Handbuch gemachten Installationsempfehlungen gehen von günstigsten Rahmenbedingungen aus. FEIG ELECTRONIC GmbH übernimmt keine Gewähr für die einwandfreie Funktion einer OBID®-Anlage in systemfremden Umgebungen.

OBID® and OBID i-scan® are registered trademarks of FEIG ELECTRONIC GmbH.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Windows Vista is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries

Electronic Product Code (TM) is a Trademark of EPCglobal Inc.

I-CODE® and Mifare® are registered Trademarks of Philips Electronics N.V.

Tag-it (TM) is a registered Trademark of Texas Instruments Inc.

Jewel (TM) is a trademark of Innovision Research & Technology plc.

Lizenzvertrag über die Nutzung der Software

Dies ist ein Vertrag zwischen Ihnen und der FEIG ELECTRONIC GmbH (nachfolgend "FEIG") über die Nutzung der überlassenen Software **ID ISC.SDK.NET** (Anwendungsprogramme, Programmbibliotheken, Source-Code Beispiele und Dokumente), nachfolgend Lizenzmaterial genannt. Mit der Installation und Benutzung der Software erklären Sie sich mit allen Bestimmungen dieses Vertrages ausnahmslos und ohne Einschränkung einverstanden. Wenn Sie mit den Bestimmungen dieses Vertrages nicht oder nicht vollständig einverstanden sind, dürfen Sie das Lizenzmaterial nicht installieren oder anderweitig benutzen. Das überlassene Lizenzmaterial ist Eigentum von FEIG und ist international urheberrechtlich geschützt.

§1 Vertragsgegenstand und Vertragsumfang

1. FEIG gewährt Ihnen das Recht, das überlassene Lizenzmaterial zu installieren und zu den nachstehenden Bedingungen zu nutzen.
2. Das Lizenzmaterial ist für den Gebrauch eines einzelnen Entwicklers bestimmt (Einzelplatzlizenz). Sie dürfen sämtliche Bestandteile des Lizenzmaterials auf einer Festplatte eines einzelnen, zu Ihrem Gebrauch bestimmten Computers installieren.
3. Die Installation und Nutzung darf auch auf einem Netzwerk-Fileserver erfolgen, wenn die Nutzung ausschließlich vom Lizenznehmer erfolgt. Für jeden zusätzlichen Nutzer ist eine separate Lizenz erforderlich.
4. Sie dürfen eine Sicherheitskopie des Lizenzmaterials anfertigen.
5. FEIG gewährt Ihnen das Recht, die dokumentierte .NET-Bibliothek OBIDISC4NET.dll, sowie die notwendigen nativen Bibliotheken, für die Entwicklung eigener Anwendungsprogramme zu verwenden und die .NET-Bibliothek OBIDISC4NET.dll, sowie die notwendigen nativen Bibliotheken, ausschließlich zusammen mit Ihren Anwendungsprogrammen ohne Abgabe von Lizenzgebühren zu vertreiben, unter der Voraussetzung, dass diese Anwendungsprogramme dazu dienen Geräte und/oder Anlagen anzusteuern oder zu betreiben, die von FEIG entwickelt und/oder vertrieben werden.
6. FEIG gewährt Ihnen das Recht den Quellcode mitgelieferter Programmbeispiele zu modifizieren und für die Entwicklung eigener Anwendungsprogramme zu verwenden und diese Anwendungsprogramme zusammen mit der .NET-Bibliothek OBIDISC4NET.dll, sowie der notwendigen nativen Bibliotheken, ohne Abgabe von Lizenzgebühren zu vertreiben, unter der Voraussetzung, dass diese Anwendungsprogramme dazu dienen Geräte und/oder Anlagen anzusteuern oder zu betreiben, die von FEIG entwickelt und/oder vertrieben werden.
7. Dieses Lizenzmaterial kann Software Dritter enthalten. Bei Verwendung dieser Drittanbieter-Software gelten die im Abschnitt [Lizenzbestimmungen Dritter](#) genannten Lizenzbestimmungen.

§2. Schutz des Lizenzmaterials

1. Das Lizenzmaterial ist geistiges Eigentum von FEIG und seinen Lieferanten. Es ist gemäß Urheberrecht, internationalen Verträgen und einschlägigen Gesetzen des Landes geschützt, in dem sie genutzt wird. Struktur, Organisation und Code der Software sind wertvolles Geschäftsgeheimnis und vertrauliche Information von FEIG und seinen Lieferanten.
2. Sie verpflichten sich, ausführbare Anwendungsprogramme, Programmbibliotheken und Dokumente nicht zu ändern, anzupassen, zu übersetzen, rückzuentwickeln, zu dekompileieren, zu disassemblieren oder auf andere Weise zu versuchen, den Quellcode dieser Software herauszufinden.
3. Soweit FEIG im Lizenzmaterial Schutzvermerke, wie Copyright-Vermerke und andere Rechtsvorbehalte angebracht hat, sind Sie verpflichtet, diese unverändert beizubehalten sowie in alle von Ihnen hergestellten vollständigen oder teilweisen Kopien in unveränderter Form zu übernehmen.
4. Die Veröffentlichung und Weitergabe an Dritte von Lizenzmaterial ist weder vollständig noch auszugsweise gestattet, solange dazu keine explizite anderslautende Vereinbarung zwischen Ihnen und FEIG getroffen wurde. Nicht betroffen

von dieser Regelung sind solche Anwendungsprogramme, die gem. §1 Absatz 5 und Absatz 6. dieser Vereinbarung erstellt und vertrieben werden.

§ 3 Gewährleistung und Haftungsbeschränkungen

1. Sie stimmen mit FEIG darüber überein, dass es nicht möglich ist, EDV-Programme so zu entwickeln, dass sie für alle Anwendungsbedingungen fehlerfrei sind. FEIG weist Sie ausdrücklich darauf hin, dass die Installation eines neuen Programms bereits vorhandene Software beeinflussen kann, und zwar auch solche Software, die nicht gleichzeitig mit der neuen Software ausgeführt wird. FEIG haftet in keinem Fall für direkte oder indirekte Schäden, für Folgeschäden oder Sonderschäden, einschließlich entgangenen Geschäftsgewinn oder entgangener Einsparungen. Wenn Sie sicherstellen wollen, dass es zu keinerlei Beeinflussung eines bereits installierten Programms kommt, dürfen Sie die vorliegende Software nicht installieren.

2. FEIG weist ausdrücklich darauf hin, dass mit der Software irreversible Einstellungen und Anpassungen an Geräten vorgenommen werden können, wodurch diese Geräte zerstört oder unbrauchbar gemacht werden können. FEIG übernimmt für derartiges Handeln unabhängig davon ob dies bewusst oder unbewusst erfolgte, keinerlei Gewährleistung.

3. FEIG liefert Ihnen die Software "wie besehen" ohne jegliche Gewährleistung. FEIG kann für die Leistung oder die Ergebnisse, die Sie durch die Nutzung der Software erzielen, nicht garantieren. FEIG übernimmt keine Gewährleistung oder Garantie dafür, dass keine Schutzrechte Dritter verletzt werden, auch nicht dafür, dass die Software für irgendeinen bestimmten Zweck geeignet ist.

4. FEIG weist ausdrücklich darauf hin, dass das Lizenzmaterial nicht für den Einsatz mit oder in medizinischen Geräten oder für Geräte für lebenserhaltende Maßnahmen konzipiert ist, bei denen ein Fehler eine Gefahr für menschliches Leben oder für die gesundheitliche Unversehrtheit zur Folge haben kann.

Der Anwender des Lizenzmaterials ist dafür verantwortlich, geeignete Maßnahmen zu ergreifen um Gefahren, Schäden oder Verletzungen zu vermeiden.

§ 4 Schlussbestimmungen

1. Dieser Vertrag enthält die vollständigen Lizenzbestimmungen und ersetzt alle eventuell vorangegangenen Regelungen und Absprachen. Änderungen und Ergänzungen bedürfen der Schriftform.

2. Sollte eine der in diesem Vertrag enthaltenen Bestimmungen unwirksam sein oder werden, so wird die Gültigkeit der übrigen Bestimmungen hierdurch nicht berührt. Beide Vertragsparteien verpflichten sich, die unwirksame Bestimmung durch eine solche wirksame Bestimmung zu ersetzen, die dem wirtschaftlichen Zweck der zu ersetzenden Bestimmung am nächsten kommt.

3. Dieser Vertrag unterliegt dem Recht der Bundesrepublik Deutschland. Gerichtsstand ist Frankfurt a. M.

Haben Sie Fragen zu diesen Verträgen, wenden Sie sich bitte an

FEIG ELECTRONIC GmbH
Lange Straße 4
35781 Weilburg-Waldhausen
Tel.: 06471 / 31090
Fax: 06471 / 310999
E-Mail: info@feig.de
Internet: <http://www.feig.de>

Lizenzbestimmungen Dritter

Lizenzbestimmung der openSSL Organisation

Die nachfolgende Lizenzbestimmung ist relevant für den Fall, dass verschlüsselte Datenübertragung zur Anwendung kommt.

LICENSE ISSUES
=====

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License

=====
Copyright (c) 1998-2008 The OpenSSL Project. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment:
"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)"
4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment:
"This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)"

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
=====

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License

Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used.

This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

"This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)"

The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement:

"This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

Inhalt:

Lizenzvertrag über die Nutzung der Software	3
Lizenzbestimmungen Dritter.....	5
Lizenzbestimmung der openssl Organisation.....	5
Inhalt:	7
Anmerkungen zur Dokumentation dieser Bibliothek.....	13
1. Einführung.....	14
1.1. Übersicht über alle Software-Bausteine	15
1.2. Unterstützte Betriebssysteme	16
2. Änderungen gegenüber der Vorversion	17
3. Installation.....	18
3.1. Installation auf dem Entwicklungsrechner	18
3.2. Installation auf dem Zielrechner	20
3.2.1. Abhängigkeiten bei Verwendung mit 32-Bit Framework V2.0 - V3.5.....	20
3.2.2. Abhängigkeiten bei Verwendung mit 32-Bit Framework V4.0 bzw. 4.5	21
3.2.3. Abhängigkeiten bei Verwendung mit 64-Bit Framework V4.0 bzw. 4.5	22
3.3. Kontrolle der installierten MFC-Version 8.0.....	23
3.4. Kontrolle der installierten MFC-Version 10.0.....	23
3.5. Applikations-Entwicklung mit 32-Bit Framework für 64-Bit Windows	24
3.6. Unterstützte Entwicklungsumgebungen	25
4. Übersicht über die Klassen.....	26
4.1. Leserkasse FedmlscReader	26
4.2. Tabellenklassen FedmlsoTableItem und FedmBrmTableItem	27
4.3. Klasse FedmlscFunctionUnit	29
4.4. Klasse FedmlscPeopleCounter	30
4.5. TagHandler-Klassen.....	30
4.6. Hilfsklassen, Strukturen und Schnittstellen	31
4.6.1. Die Klasse FedmlscReaderInfo.....	31
4.6.2. Die Klasse FedmlscReaderTime.....	31
4.6.3. Die Klasse FedmCprApdu.....	31
4.6.4. Die Klasse FedmCprCommandQueue	31
4.6.5. Die Klassen FeUsb und FeUsbScanSearch	32
4.6.6. Die Klasse FeHexConvert	32
4.6.7. Das Interface FelscListener	32

4.6.8. Die Struktur FelscListenerConst.....	32
4.6.9. Das Interface FeUsbListener.....	33
4.6.10. Die Struktur FeUsbListenerConst.....	33
4.6.11. Das Interface FedmTaskListener	33
4.6.12. Die Property FedmTaskOption.....	33
4.6.13. Die Struktur Fedm.....	33
4.6.14. Die Struktur FedmlscReaderConst.....	33
4.6.15. Die Struktur FedmlscReaderID	33
4.6.16. Die Struktur FedmlscFunctionUnitID	33
4.7. Ausnahmen.....	34
4.7.1. Die Klasse FedmException	34
4.7.2. Die Klasse FePortDriverException	34
4.7.3. Die Klasse FeReaderDriverException	34
5. Grundlegende Eigenschaften der Kommunikationsklassen	35
5.1. Initialisierung und Finalisierung.....	35
5.1.1. Initialisierung.....	35
5.1.2. Finalisierung	36
5.2. Verwaltung der Kommunikationskanäle.....	36
5.3. Kommunikation mit dem Leser	37
5.3.1. Synchrone Kommunikation	37
5.3.2. Asynchrone Kommunikation.....	39
5.3.3. Sicherheit in der Datenübertragung.....	41
5.4. Kommunikation mit einer Funktionseinheit	43
5.5. Kommunikation mit einem People Counter.....	44
5.6. Datencontainer	45
5.6.1. Datenaustausch	45
5.6.2. Zugriffskonstanten für temporäre Protokolldaten.....	47
5.6.3. Leser-Konfigurationsparameter im Namespace OBID.ReaderConfig	49
5.6.4. Verwaltung der Leserkonfiguration.....	50
5.6.5. Serialisierung	53
5.7. Tabellen.....	55
5.8. Kommunikation mit Transpondern im Host-Mode.....	56
6. Fehlerbehandlung.....	58
6.1. Rückgabewert.....	58
6.2. Ausnahmen (Exceptions).....	58
7. Bibliotheksreferenz.....	59
7.1. FedmlscReader.....	59
7.1.1. GetDependentLibVersions	59
7.1.2. FedmlscReader	60

7.1.3. ConnectCOMM	61
7.1.4. ConnectTCP	62
7.1.5. ConnectUSB	63
7.1.6. DisConnect	64
7.1.7. GetTcpConnectionState	65
7.1.8. SetPortPara	66
7.1.9. GetPortPara	68
7.1.10. SetBusAddress	68
7.1.11. GetBusAddress	69
7.1.12. GetFamilyCode	69
7.1.13. GetReaderName	70
7.1.14. GetTransponderName	70
7.1.15. GetReaderType	71
7.1.16. SetReaderType	71
7.1.17. SendProtocol	72
7.1.18. SendTransparent	72
7.1.19. TagInventory	73
7.1.20. TagSelect	74
7.1.21. SendTclApu. SendTclPing, SendTclDeselect	75
7.1.22. SendCommandQueue	76
7.1.23. SendSAMCommand	77
7.1.24. FindBaudrate	78
7.1.25. Serialize	79
7.1.26. TransferReaderCfgToXmlFile, TransferXmlFileToReaderCfg	79
7.1.27. ReaderAuthentication	80
7.1.28. ReadReaderInfo	81
7.1.29. ReadCompleteConfiguration	81
7.1.30. WriteCompleteConfiguration	81
7.1.31. ResetCompleteConfiguration	82
7.1.32. ApplyConfiguration	82
7.1.33. GetLastError	83
7.1.34. GetLastStatus	83
7.1.35. GetErrorText	83
7.1.36. GetStatusText	83
7.1.37. GetData	84
7.1.38. SetData	84
7.1.39. GetConfigPara	85
7.1.40. SetConfigPara	85
7.1.41. TestConfigPara	86
7.1.42. GetByteContainer	87
7.1.43. SetByteContainer	87
7.1.44. GetTagList	88
7.1.45. GetTagHandler, GetSelectedTagHandler	88
7.1.46. CreateNonAddressedTagHandler	88
7.1.47. GetTableItem	89
7.1.48. SetTableItem	89
7.1.49. GetTable	90

7.1.50. SetTable	90
7.1.51. GetTableSize	91
7.1.52. SetTableSize.....	91
7.1.53. GetTableLength	92
7.1.54. SetTableLength.....	92
7.1.55. ResetTable	93
7.1.56. GetTableData.....	94
7.1.57. SetTableData	95
7.1.58. VerifyTableDataBlocks	96
7.1.59. FindTableIndex	97
7.1.60. AddEventListener.....	98
7.1.61. RemoveEventListener	101
7.1.62. StartAsyncTask.....	102
7.1.63. CancelAsyncTask	104
7.1.64. TriggerAsyncTask	104
7.2. FedmIsoTableItem, FedmBrmTableItem	105
7.2.1. Datenfelder in FedmISOTableItem.....	105
7.2.2. Datenfelder in FedmBRMTableItem	107
7.2.3. GetData	109
7.2.4. SetData.....	110
7.2.5. GetRSSI.....	111
7.2.6. VerifyDataBlocks.....	112
7.2.7. IsDataValid	113
7.2.8. GetIdentifier	113
7.3. FedmIscFunctionUnit.....	114
7.3.1. FedmIscFunctionUnit	114
7.3.2. GetFUType	114
7.3.3. GetLastError	115
7.3.4. GetErrorText	115
7.3.5. SendProtocol	116
7.3.6. GetData	117
7.3.7. SetData.....	117
7.3.8. AddChild	118
7.3.9. DeleteChild	118
7.3.10. GetChild.....	118
7.4. FedmIscPeopleCounter	119
7.4.1. GetCounterValues	119
7.4.2. SetCounterValues.....	120
7.4.3. SetOutputsOn	121
7.4.4. SetOutputsOff	122
7.4.5. SetOutputsFlashing.....	123
7.5. FeHexConvert	124
7.5.1. ByteArrayToHexStringWithSpaces.....	124
7.5.2. ByteArrayToHexString	124
7.5.3. ByteToHexString.....	125

7.5.4. IntegerToHexString	125
7.5.5. LongToHexString	126
7.5.6. HexStringToByte	126
7.5.7. HexStringToByteArray	127
7.5.8. HexStringToInteger	127
7.5.9. HexStringToLong	128
7.5.10. isHexString	128
7.5.11. GetMemIDOfID	128
7.5.12. GetByteCntOfID	129
7.5.13. GetAdrOfID	129
7.6. FelscListener	130
7.6.1. OnSendProtocol	130
7.6.2. OnReceiveProtocol	130
7.7. FeUsbListener	131
7.7.1. OnConnectReader	131
7.7.2. OnDisConnectReader	131
7.8. FedmTaskListener	132
7.8.1. OnNewTag	132
7.8.2. OnNewNotification	132
7.8.3. OnNewReaderDiagnostic	133
7.8.4. OnNewPeopleCounterEvent	133
7.8.5. OnNewSAMResponse	134
7.8.6. OnNewApduResponse	134
7.8.7. OnNewQueueResponse	134
7.9. FedmException	135
7.10. FedmPortDriverException	135
7.11. FedmReaderDriverException	135
8. Beispiele für die Verwendung der Methode SendProtocol	136
8.1. Grundkommandos für den Leser	137
8.2. Tabellen orientierte Kommandos für den Leser	148
8.2.1. Besonderheiten des addressed mode	148
8.2.2. Beispiele für die Verwendung der ISO Tabelle mit [0xB0] Protokollen	148
8.2.3. Beispiele für die Verwendung der ISO Tabelle mit [0xB3] Protokollen	159
8.2.4. Kommandos für Buffered Read Mode	161
8.3. Kommandos für eine Funktionseinheit	163
9. Beispiel für die Verwendung von TagHandler-Klassen	165
10. Anhang	167
10.1. Liste der Fehlercodes	167
10.2. Unterstützte OBID® Leser	170

10.3. Unterstützte Transponder	171
10.4. TCP-Status	172
10.5. Liste der Konstanten.....	173
10.5.1. Allgemeine Konstanten	173
10.5.2. Konstanten für tableID	173
10.5.3. Konstanten für dataID	173
10.6. Änderungshistorie.....	180

Anmerkungen zur Dokumentation dieser Bibliothek

Zum Verständnis der Klassen und Methoden müssen immer auch die Systemhandbücher der eingesetzten OBID®-Leser herangezogen werden.

FEIG ELECTRONIC GmbH verzichtet darauf, Informationen zu OBID®-Lesern in verschiedenen Handbüchern mehrfach darzustellen oder Querverweise auf bestimmte Seitenzahlen eines anderen Dokumentes einzubauen. Dies ist durch die ständige Aktualisierung der Handbücher notwendig und vermeidet Irrtümer durch Informationen in veralteten Dokumenten. Dem Anwender dieser Bibliothek sei deshalb empfohlen, sich ständig zu vergewissern, dass er die aktuellen Handbücher vorliegen hat. Diese kann er selbstverständlich jederzeit bei FEIG ELECTRONIC GmbH anfordern.

Wichtige Hinweise:

Sie dürfen diese Bibliothek nur verwenden, wenn Sie zuvor den umseitig abgedruckten Lizenzbestimmungen zugestimmt haben.

1. Einführung

Mit der .NET Klassenbibliothek ID OBIDISC4NET stellt Ihnen FEIG ELECTRONIC GmbH eine weitere Komponente zur Verfügung, mit der Ihnen die Entwicklung von Anwendungsprogrammen für das Microsoft®.NET Framework für OBID *i-scan*® und OBID®*classic-pro* Leser vereinfacht werden soll.

Dieses Handbuch ist als Einstiegshilfe und Referenz in die Bibliothek gedacht.

Die .NET Klassenbibliothek ID OBIDISC4NET unterstützt z. Zt. Windows und Windows CE.

Die .NET Klassenbibliothek ID OBIDISC4NET basiert auf der C++ Klassenbibliothek ID FEDM, sowie den nativen Funktionsbibliotheken ID FECOM, ID FEUSB, ID FETCP, ID FEISC, ID FETCL und ID FEFU, die auch separat erhältlich sind. Die .NET-Klassenbibliothek besteht somit nur aus einer Wrapperschicht. Trotzdem ist der volle Funktionsumfang der C++ Klassenbibliothek ID FEDM für .NET zugänglich:

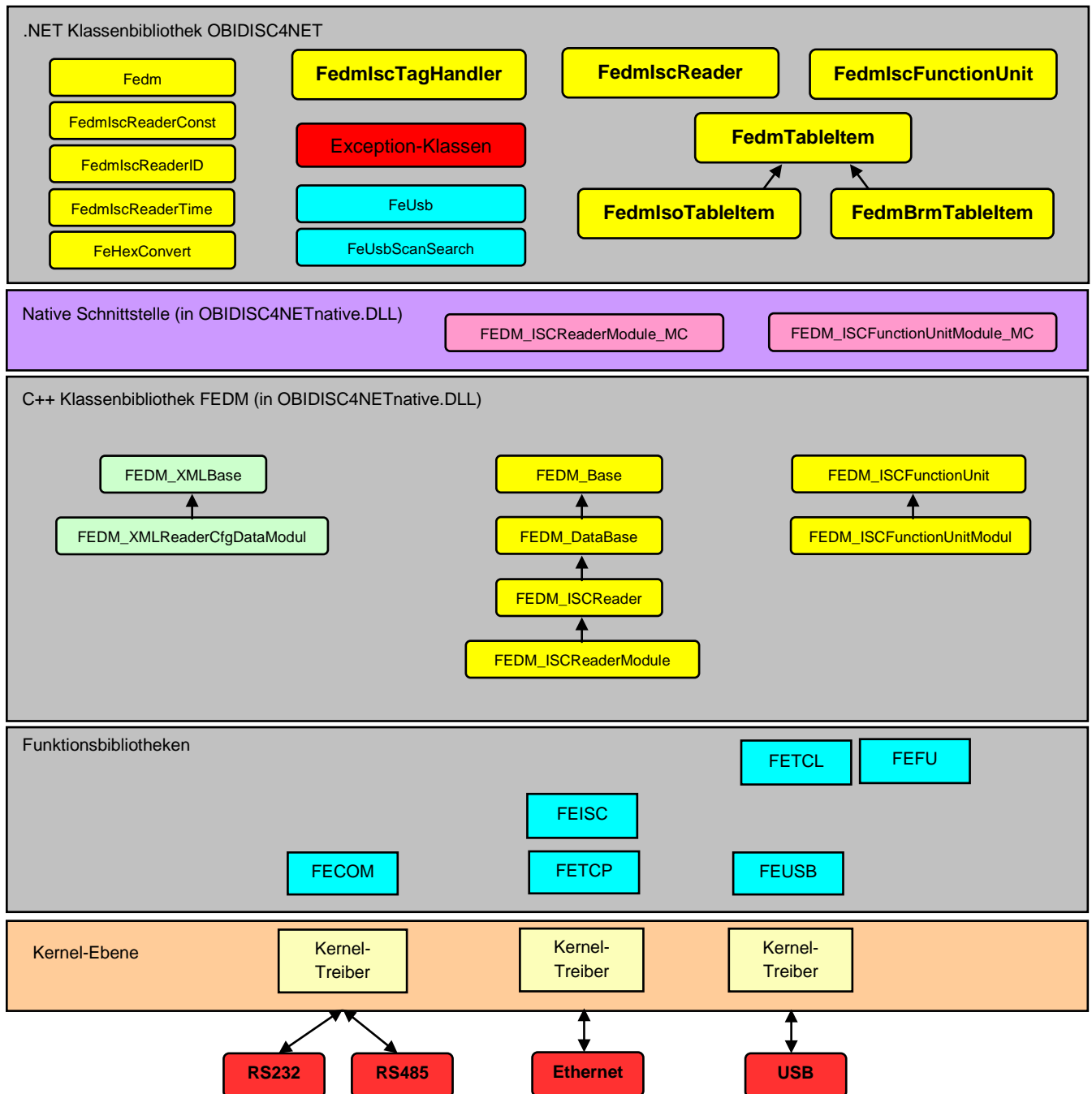
- Ein einheitliches Organisationsprinzip für die speicherbaren Daten von Leser und Transponder in Datencontainern und Tabellen.
- Überladene Methoden für den Zugriff auf die Datencontainer und Tabellen.
- Eine einzige, einfach zu verwendende Kommunikationsmethode.
- Synchrone und asynchrone Kommunikation.
- Transponder-Klassen
- Vollständige Fehlerbehandlung durch Ausnahmen oder Rückgabewerte der Methoden.
- Eine einfache Art der Serialisierung von Daten der Leserkonfiguration in einer XML-Datei.

Wichtiger Hinweis:

Die Klassenbibliothek ID OBIDISC4NET ist einem ständigen Anpassungsprozeß ausgesetzt. Wir werden uns bemühen, den dokumentierten Stand beizubehalten. Änderungen sind trotzdem nicht auszuschließen.

1.1. Übersicht über alle Software-Bausteine

Die folgende Abbildung zeigt die einzelnen Software-Bausteine, auf denen die .NET Klassenbibliothek ID OBIDISC4NET aufbaut. Die Klasse **FedmlscReader** ist die Hauptklasse. Über sie wird der Kommunikationskanal geöffnet und wird die gesamte Kommunikation mit dem Leser an diesem Kanal durchgeführt. FedmlscReader baut indirekt auf der C++ Klasse FEDM_ISCReaderModule auf, dazwischen nimmt die managed C++ Klasse FEDM_ISCReaderModule_MC die notwendigen Anpassungen vor.



1.2. Unterstützte Betriebssysteme

Die nachfolgende Matrix gibt Auskunft darüber, welche .NET-Framework-Versionen unterstützt werden.

.NET Framework	Ausführung		Betriebssysteme
	32-Bit (x86)	64-Bit (x64)	
V2.0 – 3.5	X		Windows XP / Vista / 7 / 8 (32- oder 64-Bit) mit installiertem 32-Bit Framework
V4.0 und 4.5	X	X	Windows XP / Vista / 7 / 8 (32- oder 64-Bit) mit installiertem 32- oder 64-Bit Framework
Compact V2.0 und 3.5	X	-	Windows CE 5 und CE 6

Eine Version für Windows CE 5 und CE 6 ist auf Anfrage erhältlich.

Das .NET Micro Framework wird nicht unterstützt.

2. Änderungen gegenüber der Vorversion

- Fehlerkorrektur in Leser-Klasse **FedmlscReader**: **AbandonedMutexException** konnte in den Methoden TagInventory und TagSelect auftreten.
- Neue Methode GetDependentLibVersions in der Leser-Klasse **FedmlscReader**
- Modifikationen für die Methode **StartAsyncTask** der Leser-Klasse **FedmlscReader**:
 - a) Der Listener-Port muss bei der Initialisierung des asynchronen Tasks systemweit frei sein. Andernfalls wird der Fehlercode -4086 zurückgegeben.
 - b) Listener-Port für Notification-Mode nimmt nur noch eine Verbindung zur gleichen Zeit an. Alle weiteren Verbindungsversuche werden abgelehnt.
- **FedmlscTagHandler_ISO15693_NXP_ICODE_SLI_L**: neue Methode PasswordProtectAFI
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

Bitte beachten Sie auch die Änderungshistorie im Anhang.

3. Installation

Das Supportpaket wird in der Regel mit einem Software Development Kit (SDK) ausgeliefert. Kopieren Sie das SDK unter Beibehaltung der Verzeichnisstruktur in ein Verzeichnis Ihrer Wahl.

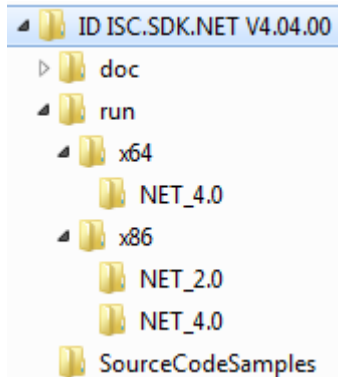


Abb: Verzeichnisstruktur des SDKs

3.1. Installation auf dem Entwicklungsrechner

Zum Lieferumfang gehören zwei 32-Bit Pakete von Bibliotheksdateien, die entweder kompatibel zum Framework V2.0 bis einschließlich V3.5 (im Verzeichnis run\x86\NET_2.0) oder kompatibel zum Framework 4.0 bzw 4.5 (im Verzeichnis run\x86\NET_4.0) sind. Zusätzlich ist für die 64-Bit Framework-Version 4.0 bzw. 4.5 ein Bibliotheks-Paket im Verzeichnis run\x64\NET_4.0 abgelegt.

Wichtig: Die DLLs der Pakete dürfen nicht gemischt werden!

Dateien im Verzeichnis	Beschreibung
run\x86\NET_2.0	
FECOM.DLL	32-Bit native Bibliothek für serielle Schnittstelle
FETCP.DLL	32-Bit native Bibliothek für TCP/IP
FEUSB.DLL	32-Bit native Bibliothek für USB
FEISC.DLL	32-Bit native Bibliothek für OBID <i>i-scan</i> ® und OBID <i>classic-pro</i> Leser
FETCL.DLL	32-Bit native Bibliothek für OBID <i>classic-pro</i> Leser
FEFU.DLL	32-Bit native Bibliothek für OBID <i>i-scan</i> ® Funktionseinheiten
FedmlscCoreVC80.DLL	32-Bit native Bibliothek für OBID <i>i-scan</i> ® und OBID <i>classic-pro</i> Leser
FedmlscMyAxxessVC80.DLL	32-Bit native Bibliothek für OBID myAXXESS® Leser
OBIDISC4NETnative.DLL	32-Bit native Bibliothek mit Wrapperschicht für Framework von V2.0 bis einschließlich V3.5
OBIDISC4NET.DLL	32-Bit .NET Bibliothek für Framework von V2.0 bis einschließlich V3.5

Dateien im Verzeichnis run\x86\NET_4.0	Beschreibung
FECOM.DLL	32-Bit native Bibliothek für serielle Schnittstelle
FETCP.DLL	32-Bit native Bibliothek für TCP/IP
FEUSB.DLL	32-Bit native Bibliothek für USB
FEISC.DLL	32-Bit native Bibliothek für OBID i-scan® und OBID® classic-pro Leser
FETCL.DLL	32-Bit native Bibliothek für OBID® classic-pro Leser
FEFU.DLL	32-Bit native Bibliothek für OBID i-scan® Funktionseinheiten
FedmlscCoreVC100.DLL	32-Bit native Bibliothek für OBID i-scan® und OBID® classic-pro Leser
FedmlscMyAxxessVC100.DLL	32-Bit native Bibliothek für OBID myAXXESS® Leser
OBIDISC4NETnative.DLL	32-Bit native Bibliothek mit Wrapperschicht für Framework V4.0
OBIDISC4NET.DLL	32-Bit .NET Bibliothek für Framework V4.0 bzw. V4.5

Dateien im Verzeichnis run\x64\NET_4.0	Beschreibung
FECOM.DLL	64-Bit native Bibliothek für serielle Schnittstelle
FETCP.DLL	64-Bit native Bibliothek für TCP/IP
FEUSB.DLL	64-Bit native Bibliothek für USB
FEISC.DLL	64-Bit native Bibliothek für OBID i-scan® und OBID® classic-pro Leser
FETCL.DLL	64-Bit native Bibliothek für OBID® classic-pro Leser
FEFU.DLL	64-Bit native Bibliothek für OBID i-scan® Funktionseinheiten
FedmlscCoreVC100.DLL	64-Bit native Bibliothek für OBID i-scan® und OBID® classic-pro Leser
OBIDISC4NETnative.DLL	64-Bit native Bibliothek mit Wrapperschicht für Framework V4.0
OBIDISC4NET.DLL	64-Bit .NET Bibliothek für Framework V4.0 bzw. V4.5

Die Installation gestaltet sich sehr einfach:

Kopieren Sie alle DLL-Dateien aus einem run-Verzeichnis in Ihr Arbeitsverzeichnis. Es wird nicht empfohlen, die DLL-Dateien in das Systemverzeichnis von Windows zu kopieren um möglichen Versionskonflikten mit gleichen DLL-Dateien in anderen Versionsständen vorzubeugen.

Erstellen Sie einen Verweis auf die Datei OBIDISC4NET.DLL in Ihrem Projekt.

Hinweis: Die Assembly-Datei OBIDISC4NET.DLL ist mit einem starken Namen signiert. Dies ermöglicht die Signatur der davon abhängigen Applikationen und erhöht die Sicherheit des Systems.

3.2. Installation auf dem Zielrechner

Zusammen mit den Dateien der Applikation sind die Laufzeitdateien OBIDISC4NET.dll, OBIDISC4NETnative.dll, FedmlscCoreVC80.dll und FedmlscMyAxxessVC80.dll bzw. FedmlscCoreVC100.dll und FedmlscMyAxxessVC100.dll und die Laufzeitdateien der Funktionsbibliotheken FECOM, FEUSB, FETCP, FEISC, FETCL und FEFU auf dem Zielrechner zu installieren.

Es wird empfohlen, die Bibliotheksdateien im Verzeichnis der Applikation zu halten. Damit vermeidet man Versionskonflikte mit späteren Installationen, die ebenfalls diese Bibliotheksdateien, aber möglicherweise in anderen Versionsständen, installieren.

Die Bibliotheksdateien hängen von neueren MFC-Bibliotheken ab, die in der Regel auf dem Zielrechner nicht vorhanden sind. Sie müssen somit installiert werden. Mit dem Visual Studio werden sogenannte Merge-Module mitgeliefert, die man in ein Setup-Projekt aufnehmen kann und die die MFC-Bibliotheken installieren.

3.2.1. Abhängigkeiten bei Verwendung mit 32-Bit Framework V2.0 - V3.5

Folgende Merge-Module sind notwendig:

MFC-Version	Merge-Module
Version 8.0 (8.0.50727.6195 s. MS11-025 ¹)	Microsoft_VC80_MFC_x86.msm Microsoft_VC80_CRT_x86.msm policy_8_0_Microsoft_VC80_MFC_x86.msm policy_8_0_Microsoft_VC80_CRT_x86.msm Hinweis: Alle vier Merge-Module gibt es auch in 64-Bit (x64) Versionen. Diese sind nicht für die 32-Bit Bibliotheksdateien des SDKs geeignet. Allerdings können 32- und 64-Bit Merge-Module zusammen auf einem Betriebssystem installiert sein.

Diese Merge-Module können mit Windows-Update auf dem Entwicklungsrechner aktualisiert werden (empfohlen) und anschließend in ein Setup-Projekt eingebunden werden.

Eine Alternative ist die Installation der Visual C++ Laufzeitbibliotheken über das Internet-Portal von Microsoft. Dort findet sich für jede MFC-Version eine Datei vcredist_x86.exe zum herunterladen.

In beiden Fällen ist es wichtig, dass mindestens die oben angegebene Versionsnummer auf dem Zielrechner installiert wird. Zur Kontrolle der installierten Version, siehe [3.3. Kontrolle der installierten MFC-Version 8.0](#)

Link zur Webpage von Microsoft Visual C++ 2005 SP1 Redistributable Packages:

<http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=26347>

¹ Microsoft Sicherheitsbulletin Artikel-ID: 2538218 vom 14. Juni 2011

3.2.2. Abhängigkeiten bei Verwendung mit 32-Bit Framework V4.0 bzw. 4.5

Folgende Merge-Module sind notwendig:

MFC-Version	Merge-Module
Version 10.0 (10.0.30319.460 s. MS11-025 ²)	Microsoft_VC100_MFC_x86.msm Microsoft_VC100_CRT_x86.msm Hinweis: Beide Merge-Module gibt es auch in 64-Bit (x64) Versionen. Diese sind nicht für die 32-Bit Bibliotheksdateien des SDKs geeignet. Allerdings können 32- und 64-Bit Merge-Module zusammen auf einem Betriebssystem installiert sein.

Diese Merge-Module können mit Windows-Update auf dem Entwicklungsrechner aktualisiert werden (empfohlen) und anschließend in ein Setup-Projekt eingebunden werden.

Eine Alternative ist die Installation der Visual C++ Laufzeitbibliotheken über das Internet-Portal von Microsoft. Dort findet sich für jede MFC-Version eine Datei vcredist_x86.exe zum herunterladen.

In beiden Fällen ist es wichtig, dass mindestens die oben angegebene Versionsnummer auf dem Zielrechner installiert wird. Zur Kontrolle der installierten Version, siehe [3.4. Kontrolle der installierten MFC-Version 10.0](#)

Link zur Webpage von Microsoft Visual C++ 2010 Redistributable Package (x86):

<http://www.microsoft.com/en-us/download/details.aspx?id=5555>

² Microsoft Sicherheitsbulletin Artikel-ID: 2538218 vom 14. Juni 2011

3.2.3. Abhängigkeiten bei Verwendung mit 64-Bit Framework V4.0 bzw. 4.5

Folgende Merge-Module sind notwendig:

MFC-Version	Merge-Module
Version 10.0 (10.0.40219.1)	Microsoft_VC100_MFC_x64.msm Microsoft_VC100_CRT_x64.msm Hinweis: Beide Merge-Module gibt es auch in 32-Bit (x86) Versionen. Diese sind nicht für die 64-Bit Bibliotheksdateien des SDKs geeignet. Allerdings können 32- und 64-Bit Merge-Module zusammen auf einem Betriebssystem installiert sein.

Diese Merge-Module können mit Windows-Update auf dem Entwicklungsrechner aktualisiert werden (empfohlen) und anschließend in ein Setup-Projekt eingebunden werden.

Eine Alternative ist die Installation der Visual C++ Laufzeitbibliotheken über das Internet-Portal von Microsoft. Dort findet sich für jede MFC-Version eine Datei vcredist_x64.exe zum herunterladen.

In beiden Fällen ist es wichtig, dass mindestens die oben angegebene Versionsnummer auf dem Zielrechner installiert wird. Zur Kontrolle der installierten Version, siehe [3.4. Kontrolle der installierten MFC-Version 10.0](#)

Link zur Webpage von Microsoft Visual C++ 2010 Redistributable Package (x64):

<http://www.microsoft.com/en-us/download/details.aspx?id=14632>

3.3. Kontrolle der installierten MFC-Version 8.0

Im Verzeichnis C:\Windows\WinSxS befinden sich Unterverzeichnisse mit den installierten Microsoft Laufzeitbibliotheken. Die Verzeichnisse, beginnend mit x86_microsoft.vc80.mfc_ und x86_microsoft.vc80.crt_ geben Auskunft über die installierten Versionen.



Im Verzeichnisnamen steht u.a. auch die Versionsnummer, z.B. **8.0.50727.6195**.

3.4. Kontrolle der installierten MFC-Version 10.0

Im Systemverzeichnis C:\Windows\System32³ bzw. C:\Windows\SysWOW64⁴ befinden sich die installierten Microsoft Laufzeitbibliotheken MFC100.DLL, MSVCP100.DLL und MSVCR100.DLL. In den Dateieigenschaften steht die Versionsnummer.

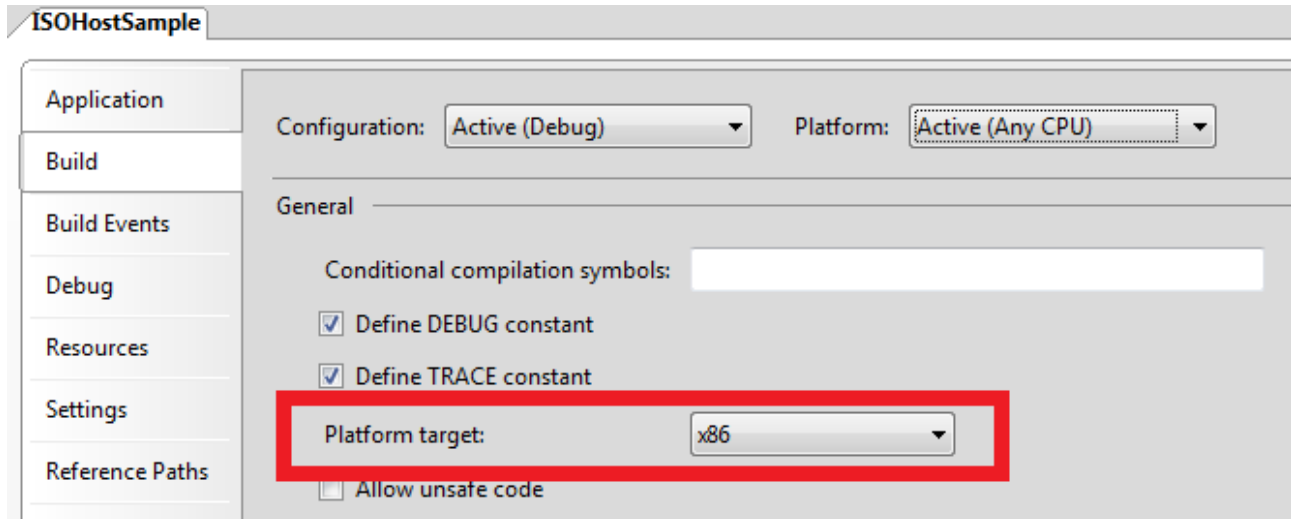
³ 32-Bit DLLs von 32-Bit Windows bzw. 64-Bit DLLs von 64-Bit Windows

⁴ 32-Bit DLLs von 64-Bit Windows

3.5. Applikations-Entwicklung mit 32-Bit Framework für 64-Bit Windows

Die 32-Bit .NET-Bibliothek ist ausschließlich für 32-Bit Applikationen geeignet. Wenn eine Applikation für ein 64-Bit Windows entwickelt wird, muss Platform Target in den Projekteinstellungen auf x86 eingestellt werden.

Auf dem Zielrechner muss sichergestellt sein, dass die 32-Bit (x86) Version des .NET-Framework 2.0, 3.0, 3.5, 4.0 oder 4.5 installiert ist.



3.6. Unterstützte Entwicklungsumgebungen

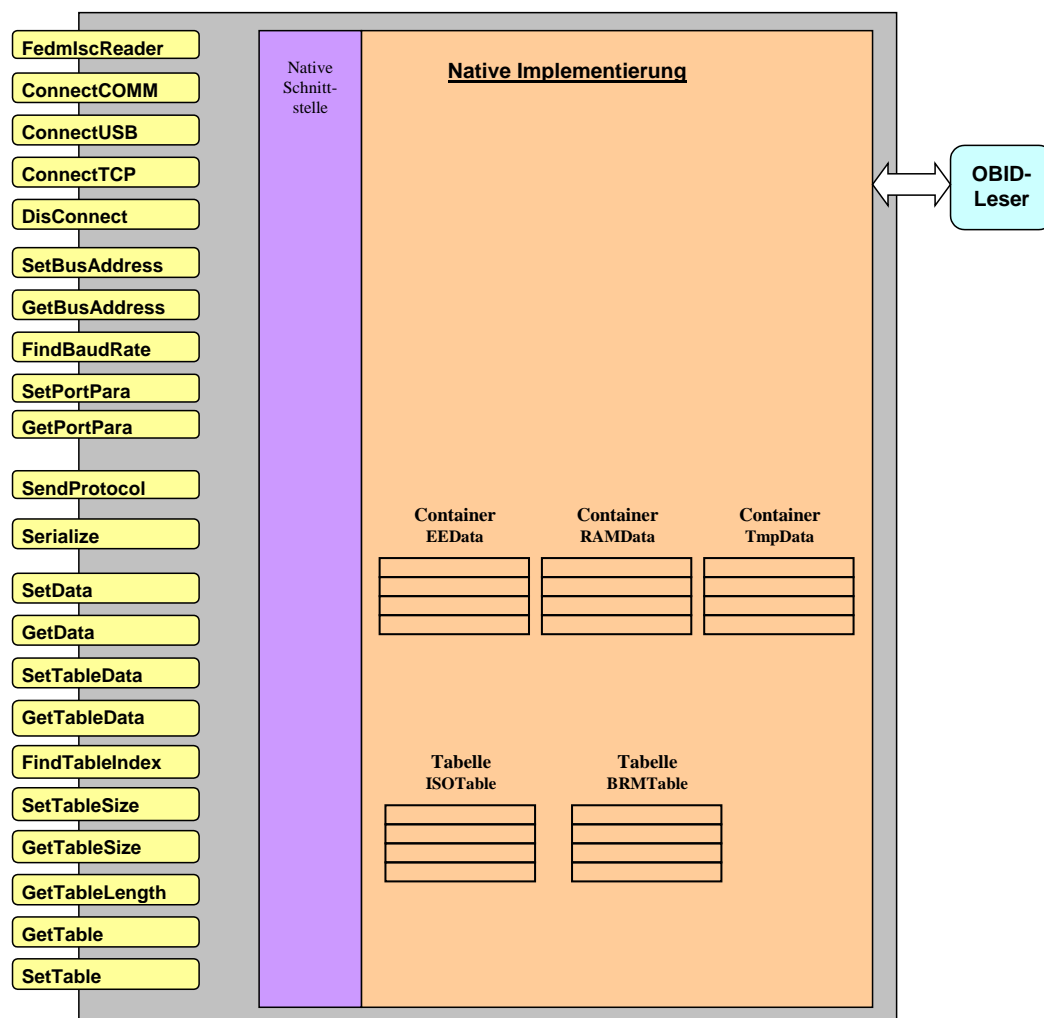
Betriebssystem	Entwicklungsumgebung	Unterstützung
Windows XP / Vista / 7	Visual Studio 6	nein
	Visual Studio 2005 / 2008 / 2010 / 2012	ja, ab Professional Version
Windows CE	eMbedded Visual C++ 4	nein
	Visual Studio 2005 / 2008	ja, ab Professional Version

4. Übersicht über die Klassen

4.1. Leserklasse FedmlscReader

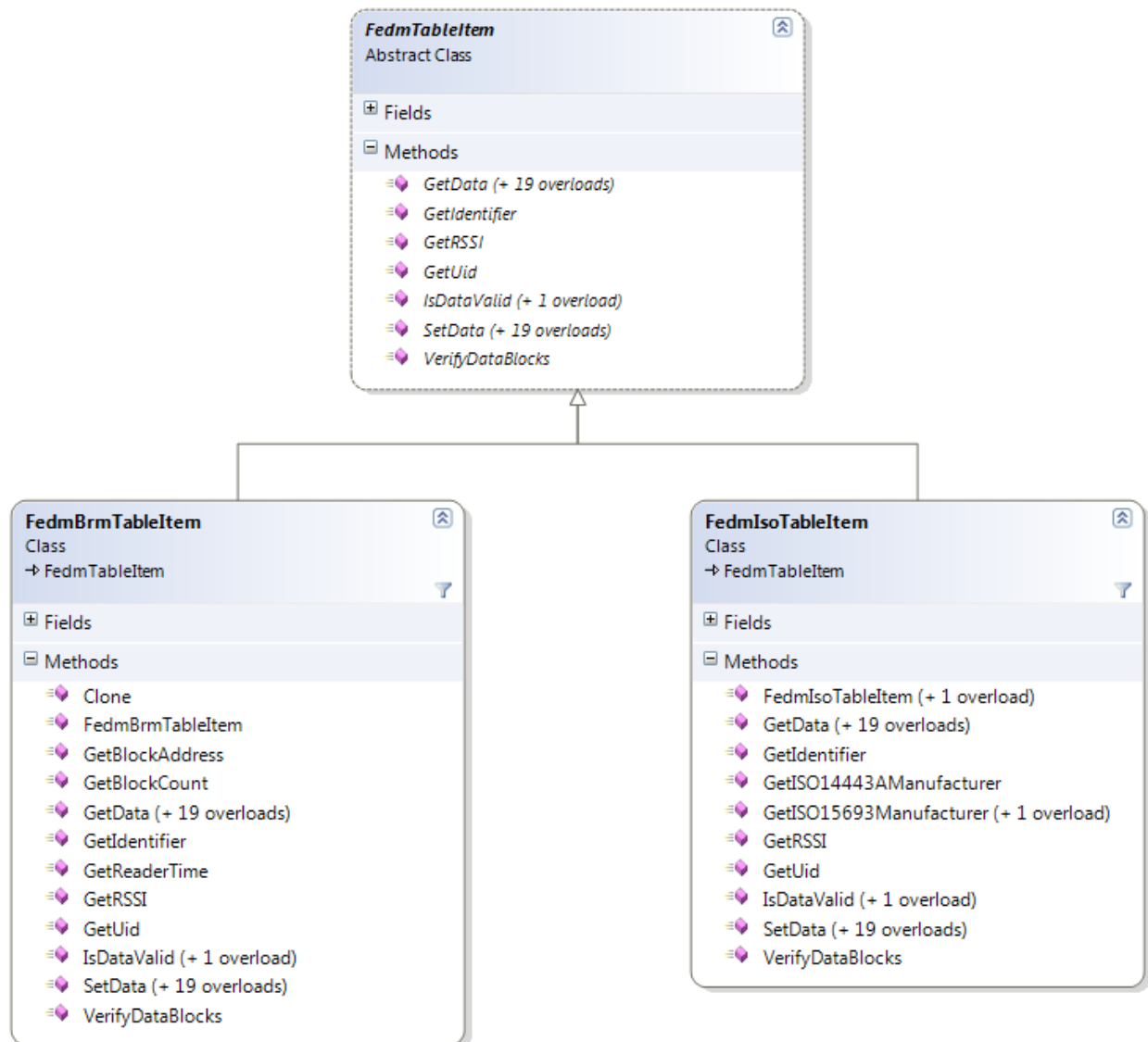
Die Leserklasse **FedmlscReader** ist die Hauptklasse der .NET-Bibliothek. Sie repräsentiert einen Leser und enthält Datencontainer und Tabellen. Das Komponenten-Diagramm zeigt eine Übersicht auf die Leserklasse.

Es sind nur die wichtigsten Methoden dargestellt. Attribute sind in der .NET-Klasse nicht enthalten.



4.2. Tabellenklassen **FedmIsoTableItem** und **FedmBrmTableItem**

Die Tabellenklassen **FedmIsoTableItem** und **FedmBrmTableItem** sind von der abstrakten Klasse **FedmTableItem** abgeleitet und enthalten Daten eines Transponders. Ein Array aus diesen Klassen bildet eine Tabelle, wobei keine gemischte Tabelle erlaubt ist. Die Klassen sind reine .NET-Implementierungen.



Beide Klassen sind zu den Methoden *GetTableData* und *SetTableData* der Leserklasse **FedmIsCReader** alternative Schnittstellen zum Transponder.

FedmIsoTableItem enthält Transponderdaten, die mit den Leserkommandos des ISO-Host-Modus gelesen wurden, bzw. vor dem Schreiben auf dem Transponder dort abgelegt werden.

FedmBrmTableItem enthält Transponderdaten, die vom Leser im Buffered Read Mode oder Notification Mode gelesen wurden. Da beide Modi reine Lesemodi sind, können in **FedmBrmTableItem** keine Daten mit *SetTableData* geschrieben werden.

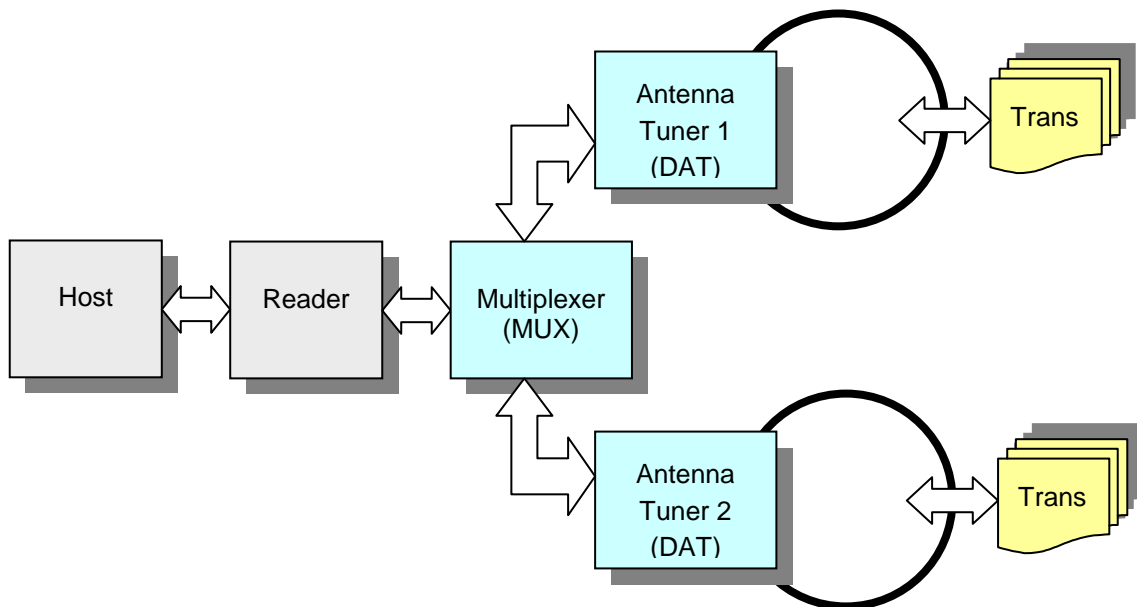
Der Datenaustausch mit den Transpondern erfolgt immer über die Methode *SendProtocol* der Leserklassse **FedmlscReader**.

Einen einfachen Zugang zu den Tabellendaten erlaubt die Methode *GetTableItem* der Leserklassse **FedmlscReader** und der anschließende direkte Zugriff (ab SDK-Version 4.03.00) auf die public Transponderdaten. Lediglich Datenarrays, die Datenblöcke eines Transponders enthalten, müssen mit *GetData* Methoden abgefragt werden.

4.3. Klasse FedmlscFunctionUnit

Die Klasse **FedmlscFunctionUnit** repräsentiert eine externe Funktionseinheit (engl. Function Unit, Abk. FU) in der Antennen-Leitung des Lesers. Zum Verständnis der Funktionseinheiten ist das Systemhandbuch H30701-xe-ID-B (HF) bzw. H80302-xe-ID-B (UHF) zwingend erforderlich. Nützliche Informationen finden sich auch in den Montageanleitungen zu den jeweiligen Funktionseinheiten.

In Anbetracht der Tatsache, dass eine Funktionseinheit immer einen Leser als Kommunikationsbrücke voraussetzt, kann auch die Klasse **FedmlscFunctionUnit** nur instanziiert werden, wenn ein Leserobjekt vom Typ **FedmlscReader** vorhanden ist.



Die Abbildung verdeutlicht auch, dass externe Funktionseinheiten baumartig mit dem Leser verbunden sind. Diese Topologie wird in der Leserklassse durch eine Liste für nachfolgende Funktionseinheiten nachgebildet. Mit dieser Liste ist es möglich, durch den Baum, beginnend mit der ersten Funktionseinheit, zu traversieren.

4.4. Klasse FedmlscPeopleCounter

Die Klasse **FedmlscPeopleCounter** repräsentiert eine externe Einheit am RS485-Bus des Lesers zum Zählen von Personen. Zum Verständnis des People Counters ist das Systemhandbuch H01011-xe-ID-B zwingend erforderlich. Nützliche Informationen finden sich auch in den Montageanleitungen zu den Gate-Antennen.

Zur Verwendung der Klasse siehe [5.5. Kommunikation mit einem People Counter](#) und die Referenz in [7.4. FedmlscPeopleCounter](#).

4.5. TagHandler-Klassen

TagHandler-Klassen bieten ein auf den jeweiligen Transponder-Standard (ISO 14443, ISO 15693, EPC Class 1 Gen 2) bzw. einen speziellen Chip-Typ zugeschnittenes API an. Jeder Standard bzw. Chip-Typ wird als Klasse implementiert, wobei die Klassen ein hierarchisches System aus abgeleiteten Klassen bilden. Grundklasse ist die Klasse FedmlscTagHandler.

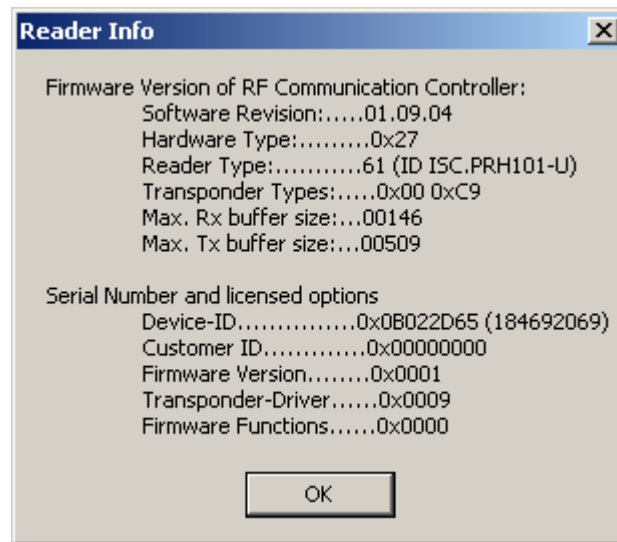
Zur Verwendung der Klassen siehe [5.8. Kommunikation mit Transpondern im Host-Mode](#)

4.6. Hilfsklassen, Strukturen und Schnittstellen

4.6.1. Die Klasse FedmlscReaderInfo

FedmlscReaderInfo ist eine Klasse, die alle wichtigen Informationen eines Lesers nach dem Aufruf der Methode `ReadReaderInfo` speichert.

Mit der Methode `GetReport()` erhält man einen formatierten String mit allen Informationen zum angeschlossenen Leser.



4.6.2. Die Klasse FedmlscReaderTime

FedmlscReaderTime ist eine Klasse, die die Zeit des Lesers im Buffered Read Mode repräsentiert.

Man erhält das Objekt nur mit der Methode `GetReaderTime` der Klasse **FedmlscBrmTableItem**.

Das Datumsformat ist kompatibel mit ISO 8601.

Beispiel für Datum: "2012-08-15"

Beispiel für Zeit: "13:01:25.123" (Stunde:Minuten: Sekunden.Millisekunden)

4.6.3. Die Klasse FedmCprApu

FedmCprApu ist eine Klasse, die die Leserkasse in der asynchronen Ausführung von ISO14443-4 T=CL Protokollen unterstützt.

4.6.4. Die Klasse FedmCprCommandQueue

FedmCprCommandQueue ist eine Klasse, die die Leserkasse in der asynchronen Ausführung von [0xBC] Command Queue Protokollen unterstützt.

4.6.5. Die Klassen FeUsb und FeUsbScanSearch

Die Klasse **FeUsb** ist eine Hilfsklasse zur Erkennung von mehr als einem USB-Leser, die gleichzeitig am USB angeschlossen sind. **FeUsbScanSearch** ist eine Klasse mit Suchoptionen für einen Scanvorgang am USB.

Verwendet man in Applikationen nie mehr als einen USB-Leser gleichzeitig, kann man auf diese Klassen verzichten.

FeUsb baut direkt auf der nativen Bibliothek ID FEUSB auf. Deshalb wird in diesem Dokument FeUsb nicht beschrieben. Mehr Informationen finden sich in H00501-x-ID-B.

4.6.6. Die Klasse FeHexConvert

Die Klasse **FeHexConvert** enthält nützliche statische Methoden zum Konvertieren von Daten.

4.6.7. Das Interface FelscListener

Das Interface **FelscListener** ermöglicht eine Ereignisbehandlung aus der nativen Bibliothek. Mit diesem Interface kann z. B. sehr einfach ein Protokollfenster für Leserprotokolle realisiert werden.

4.6.8. Die Struktur FelscListenerConst

In der Struktur **FelscListenerConst** sind Konstanten für das Interface FelscListener definiert, die ein bestimmtes Ereignis repräsentieren.

4.6.9. Das Interface FeUsbListener

Das Interface **FeUsbListener** ermöglicht eine Ereignisbehandlung aus der nativen Bibliothek. Mit diesem Interface kann eine Applikation über die Ereignisse USB-Leser eingesteckt bzw. entfernt informiert werden.

4.6.10. Die Struktur FeUsbListenerConst

In der Struktur **FeUsbListenerConst** sind Konstanten für das Interface FeUsbListener definiert, die ein bestimmtes Ereignis repräsentieren.

4.6.11. Das Interface FedmTaskListener

Das Interface **FedmTaskListener** ermöglicht eine Ereignisbehandlung aus der nativen Bibliothek. Über dieses Interface werden die Transponder- oder Leser-Informationen aus asynchronen Tasks in die Applikation übertragen.

4.6.12. Die Property FedmTaskOption

Die Klasse **FedmTaskOption** sammelt die Einstellungen für asynchrone Tasks.

4.6.13. Die Struktur Fedm

In der Struktur **Fedm** sind allgemeine Konstanten für die Klassenbibliothek definiert.

4.6.14. Die Struktur FedmlscReaderConst

In der Struktur **FedmlscReaderConst** sind allgemeine Konstanten für die Leserkasse **FedmlscReader** enthalten.

4.6.15. Die Struktur FedmlscReaderID

In der Struktur **FedmlscReaderID** sind alle Zugriffskonstanten für temporäre Protokolldaten für die OBID *i-scan*® und OBID® *classic-pro* Leser enthalten.

4.6.16. Die Struktur FedmlscFunctionUnitID

In der Struktur **FedmlscFunctionUnitID** sind alle Zugriffskonstanten für die OBID *i-scan*® Funktionseinheiten enthalten.

4.7. Ausnahmen

4.7.1. Die Klasse FedmException

FedmException ist eine Klasse, die in Ausnahmesituationen im Bereich der nativen C++ Klassenbibliothek FEDM ausgelöst wird.

4.7.2. Die Klasse FePortDriverException

FePortDriverException ist eine Klasse, die in Ausnahmesituationen im Bereich der nativen Funktionsbibliotheken FECOM, FEUSB und FETCP ausgelöst wird.

4.7.3. Die Klasse FeReaderDriverException

FeReaderDriverException ist eine Klasse, die in Ausnahmesituationen im Bereich der nativen Funktionsbibliothek FEISC ausgelöst wird.

5. Grundlegende Eigenschaften der Kommunikationsklassen

Die Methoden der Leserklasse und FU-Klasse kann man grob in fünf Kategorien aufteilen:

- a) Methoden für die Initialisierung und Finalisierung
- b) Methoden für die Kommunikationskanäle
- c) Methoden für die Kommunikation
- d) Methoden für Datencontainer und Serialisierung
- e) Methoden für Tabellen

5.1. Initialisierung und Finalisierung

5.1.1. Initialisierung

Vor der ersten Verwendung der Leserklasse müssen einige Initialisierungen durchgeführt werden:

1. Busadresse

Die Busadresse des Lesers ist in der Klasse mit 255 voreingestellt. Eine andere Adresse stellt man mit der Methode *SetBusAddress* ein. Die Einstellung ist allerdings nur für Leser am seriellen Port sinnvoll.

Die Adresse für eine Funktionseinheit setzt man in der Klasse mit *SetData(FedmIscFunctionUnitID.FEDM_ISC_FU_TMP_DAT_ADR, (byte)1)*
2. Größe einer Tabelle

Die in der Leserklasse **FedmIscReader** enthaltenen Tabellen *ISOTable* und *BRMTable* haben voreingestellt keine Größe. Deshalb **muss** (!) man mit der Methode *SetTableSize* die benötigte Tabelle vor der ersten Kommunikation mit einem Transponder dimensionieren.

Anhaltspunkt für die Größe einer Tabelle ist die maximal gleichzeitig im Antennenfeld des Lesers befindliche Transponderzahl.

Üblicherweise dimensioniert man nur eine Tabelle, denn der Leser kann nicht gleichzeitig im Buffered Read Mode und im ISO-Host Mode arbeiten.

Für die Tabellen wird je Tabelleneintrag die folgende Speichermenge reserviert:

 - *BRMTable*: 1104 Bytes
 - *ISOTable*: 17496 Bytes

3. Lesertyp Der Lesertyp muss in der Leserklasse gesetzt werden. Dies kann auf drei Arten erfolgen:
1. Nach dem erfolgreichen Verbindungsaufbau wird automatisch mit den Methoden *ConnectCOMM*(..., *true*), *ConnectUSB* und *ConnectTCP* intern die Methode *ReadReaderInfo* ausgeführt und der Lesertyp gesetzt.
 2. Nach dem Öffnen der seriellen Schnittstelle mit *ConnectCOMM*(..., *false*) durch den externen Aufruf von *ReadReaderInfo*.
 3. Mit der Methode *SetReaderType*. Die Konstanten für die Lesertypen sind im Interface *FedmlscReaderConst* enthalten.

5.1.2. Finalisierung

In .NET überlässt man dem Garbage Collector die Beseitigung nicht mehr benötigter Objekte. Das funktioniert bei reinen .NET-Anwendungen vorzüglich. Objekte, die im 'nativen Adressraum' erzeugt werden, unterliegen allerdings nicht der Aufsicht des Garbage Collectors. Deshalb ist es notwendig, dass der Programmierer diese Arbeit übernimmt. Nach dem erfolgreichen Verbinden der Applikation mit einem Leser sollte die Verbindung mit einem Aufruf der Methode *Disconnect* geschlossen werden.

5.2. Verwaltung der Kommunikationskanäle

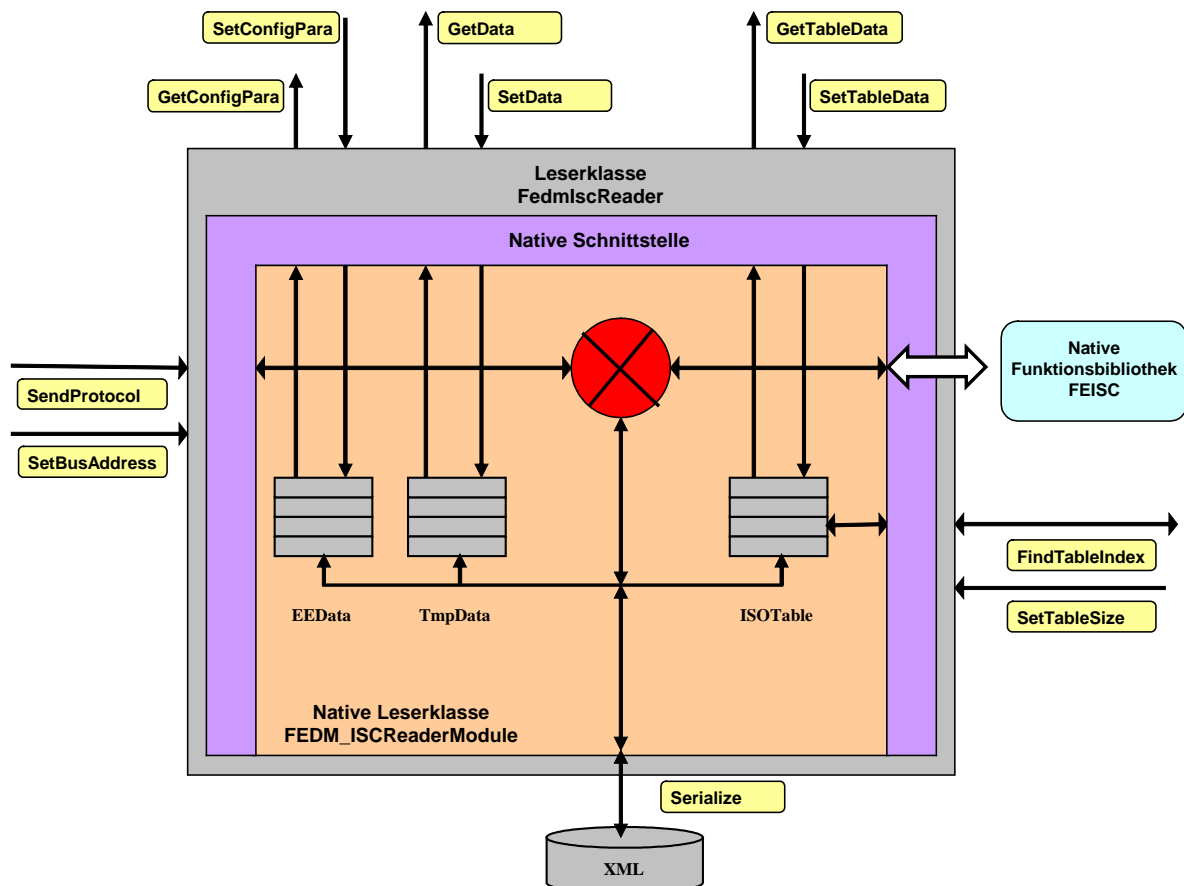
Es gibt innerhalb der Klassenbibliothek, mit einer Ausnahme, keine Klassen für die Kommunikationskanäle. Stattdessen sind Methoden in der Leserklasse *FedmlscReader* integriert: *ConnectCOMM*, *ConnectUSB*, *ConnectTCP* öffnen jeweils einen Kanal zum Leser. Mit *Disconnect* schließt man diesen Kanal wieder. Für die serielle Schnittstelle gibt es zusätzlich die Methode *FindBaudrate*, die einen Leser detektiert und die Schnittstelle entsprechend der Kommunikationsparameter (Baudrate, Frame) richtig konfiguriert.

Für den Ausnahmefall, dass mehrere USB-Leser in einer Applikation gleichzeitig unterstützt werden müssen, gibt es die Klasse *FeUsb*, die spezielle Methoden für diesen Fall bereitstellt.

5.3. Kommunikation mit dem Leser

5.3.1. Synchrone Kommunikation

Der Ablauf der synchronen, also von einem Host initiierten Kommunikation in der Leserklasse **FedmlscReader** lässt sich sehr schön am nachfolgenden Schaubild verdeutlichen: In der Vertikalen aufgetragen sind die Datenströme, die mit den (überladenen) Methoden *GetData* bzw. *GetConfigPara* und *SetData* bzw. *SetConfigPara*, sowie *GetTableData* und *SetTableData* bewegt werden. Zusätzlich ist mit der Methode *Serialize* der Datenfluss zwischen einem Leserobjekt und einer Datei möglich.



In der Horizontalen sieht man den Steuerfluss, der mit der Methode *SendProtocol*, der einzigen Kommunikationsmethode, ausgelöst wird. Sie holt sich intern selbstständig vor der Ausgabe des Sendeprotokolls alle notwendigen Daten aus den integrierten Datencontainern und legt auch dort die empfangenen Protokolldaten wieder ab. Somit muss das Anwendungsprogramm **vor** dem Aufruf von *SendProtocol* **alle** für dieses Protokoll notwendigen Daten in die entsprechenden Datencontainern an die richtigen Stellen schreiben. Ebenso sind die Empfangsdaten an bestimmten Stellen in entsprechenden Datencontainern hinterlegt.

Der Schlüssel zu den Protokolldaten sind sogenannte Zugriffskonstanten für temporäre Protokolldaten im Namespace **OBID.ReaderCommand** (z.B. `OBID.ReaderCommand._0x6A.Req.RF_OUTPUT`) und der Namespace **OBID.ReaderConfig** mit

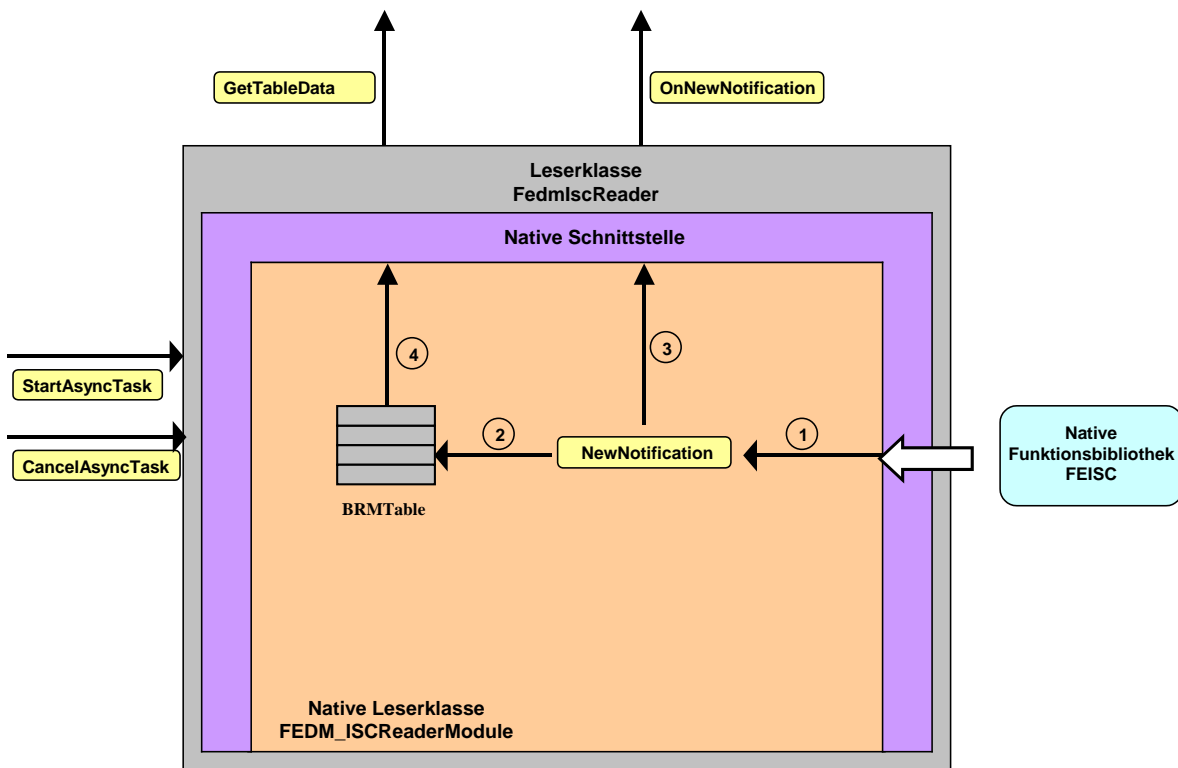
allen Parameternamen der Leserkonfiguration (z.B. `OBID.ReaderConfig.OperatingMode.Mode`). Für jede Leserklasse können einige Dutzend bis hundert Konstanten sowie Namen im Namespace **OBID.ReaderConfig** definiert sein. Ihr Aufbau ist für alle Lesertypen gleich und von besonderer Bedeutung. Er wird genau in [5.6.2. Zugriffskonstanten für temporäre Protokolldaten](#) und [5.6.3. Leser-Konfigurationsparameter im Namespace OBID.ReaderConfig](#) erläutert. Da die Zugriffskonstanten von elementarer Bedeutung für die gesamte Funktion der Leserklasse sind, werden sie im Zusammenhang mit ihrer Verwendung in dem [Kapitel 8. Beispiele für die Verwendung der Methode SendProtocol](#) genau in ihrem Zusammenhang beschrieben. Die Abbildung eines jeden Konfigurationsparameters in einen Parameternamen innerhalb des Namespaces **OBID.ReaderConfig** ist in den Systemhandbüchern der Leserfamilien dokumentiert.

Die OBID®-Leser an der seriellen Schnittstelle sind busfähig und benötigen im Protokoll die Busadresse. Diese sollte mit der Methode `SetBusAddress` gesetzt werden.

5.3.2. Asynchrone Kommunikation

Die asynchrone Kommunikation über die Leserklasse **FedmlscReader** wird durch die Methode **StartAsyncTask** initiiert und mit Ereignissen im Leser angestoßen. Asynchrone Tasks können nur gestartet werden, wenn ein Leser den Notification Mode unterstützt oder im Host Mode den Inventory-Befehl asynchron beantworten kann⁵. Pro Instanz von **FedmlscReader** kann nur ein asynchroner Task gestartet werden.

Der Informationsfluß läßt sich sehr schön am nachfolgenden Schaubild verdeutlichen:



Die Notification des Lesers wird in den nativen Teil der Bibliothek gesendet (1. Schritt). Im zweiten Schritt werden die Transponderinformationen in die Tabelle geschrieben und mit dem dritten Schritt die Eventmethode der Applikation aufgerufen. In dieser Eventmethode kann die Applikation die Tabelleninformation mit den überladenen **GetTableData** Methoden auslesen und weiter verarbeiten.

Sendet der Leser im Notification Mode Transponderdaten, werden diese in der BRM-Tabelle abgelegt. Die Transponderdaten eines asynchronen Inventory im Host Mode des Lesers werden in die ISO-Tabelle geschrieben.

⁵ Letzteres ist nur in der OBID® *classic-pro* Leserfamilie realisiert.

Die nachfolgende Liste zeigt die Zuordnung der Tasks zu den Listener-Methoden:

Task	Task-ID (FedmTaskOption)	Start-Methode	Listener-Methode (FedmTaskListener)
Einzelner Inventory	ID_FIRST_NEW_TAG	StartAsyncTask	OnNewTag
Repetierender Inventory	ID_EVERY_NEW_TAG	StartAsyncTask	OnNewTag
Notification	ID_NOTIFICATION	StartAsyncTask	OnNewNotification oder OnNewReaderDiagnostic oder OnNewPeopleCounterEvent
SAM-Kommunikation	-	SendSAMCommand	OnNewSAMResponse
Queue Kommando	-	SendQueueCommand	OnNewQueueResponse
T=CL APDU	-	SendTclApdu	OnNewApduResponse
Zutrittsereignis (FedmIscMyAxxessReader)	-	StartEventHandler	OnNewMaxAccessEvent oder OnNewMaxKeepAliveEvent

5.3.3. Sicherheit in der Datenübertragung

5.3.3.1. Übersicht

Optional können OBID *i-scan*®- oder OBID®*classic-pro* Leser die Protokolle verschlüsselt übertragen. Zur Anwendung kommt ein AES-Algorithmus mit einer Schlüssellänge von 256 Bit. Der Authentifizierungsschlüssel (Passwort) ist im Leser gespeichert und kann nicht ausgelesen werden. Der Kryptomode im Leser ist ab Werk abgeschaltet.

Die Datenverschlüsselung basiert auf Funktionen der Open-Source Organisation openssl (<http://www.openssl.org>), die in der Bibliotheksdatei libeay32.dll enthalten sind. Die Bindung an die openssl-Bibliothek erfolgt erst zur Laufzeit beim ersten Aufruf einer openssl-Funktion. Dies bedeutet, dass alle Applikationen, die keine Datenverschlüsselung nutzen, die genannte openssl-Bibliothek nicht installieren müssen. Für den Fall, dass verschlüsselte Datenübertragung zur Anwendung kommt, müssen Sie die Lizenzbedingungen zu openssl beachten.

Die Datenverschlüsselung wird durch das Aktivieren des Kryptomodes in der Leserkonfiguration mit einem anschließenden CPU-Reset eingeschaltet. Danach akzeptiert ein Leser im Kryptomode ausschließlich verschlüsselte Protokolle. Vor dem ersten Leserbefehl muss eine neue Verbindung mit FedIscReader.ConnectTCP (ConnectCOMM oder ConnectUSB) aufgebaut werden, mit dem das Passwort (Werkseinstellung: Passwort besteht aus Nullen) verschlüsselt übertragen und eine neue Session gestartet wird. Jedes nachfolgende Protokoll wird dann automatisch verschlüsselt übertragen.

Hinweis: Nach der ersten Authentifizierung sollte ein neues Passwort vergeben und eine erneute Authentifizierung mit dem neuen Passwort durchgeführt werden. Diese Vorgehensweise – erst in den Kryptomode wechseln und dann ein Passwort vergeben – stellt sicher, dass das neue Passwort verschlüsselt übertragen wird! Andernfalls wird das neue Passwort im Klartext übertragen.

5.3.3.2. Rückmeldung von Fehlerzuständen

Ein Leser im Kryptomode lehnt alle nicht verschlüsselten Protokolle mit dem Status 0x19 (Crypto Processing Error) ab.

Ein Leser im Klartextmode lehnt alle verschlüsselten Protokolle mit dem Status 0x82 (Command not available) ab.

Ein Reader-Authent mit einem falschen Passwort wird mit dem Status 0x12 (Authent Error) signalisiert.

Ein Leser im Kryptomode signalisiert mit dem Status 0x19 (Crypto Processing Error) einen fehlerhaften Zustand in der Datenverschlüsselung. Der Host muss sich daraufhin erneut authentifizieren.

Die Fehlercodes -4093 und -4094 aus FedmlscReader.SendProtocol signalisieren einen Host-seitigen fehlerhaften Zustand in der Datenverschlüsselung. Der Host muss sich daraufhin erneut authentifizieren.

Der Fehlercode -4090 signalisiert einen Fehler beim Laden der openssl-Bibliotheksdatei. Möglicherweise ist diese Bibliotheksdatei nicht installiert oder eine nicht kompatible Version ist installiert.

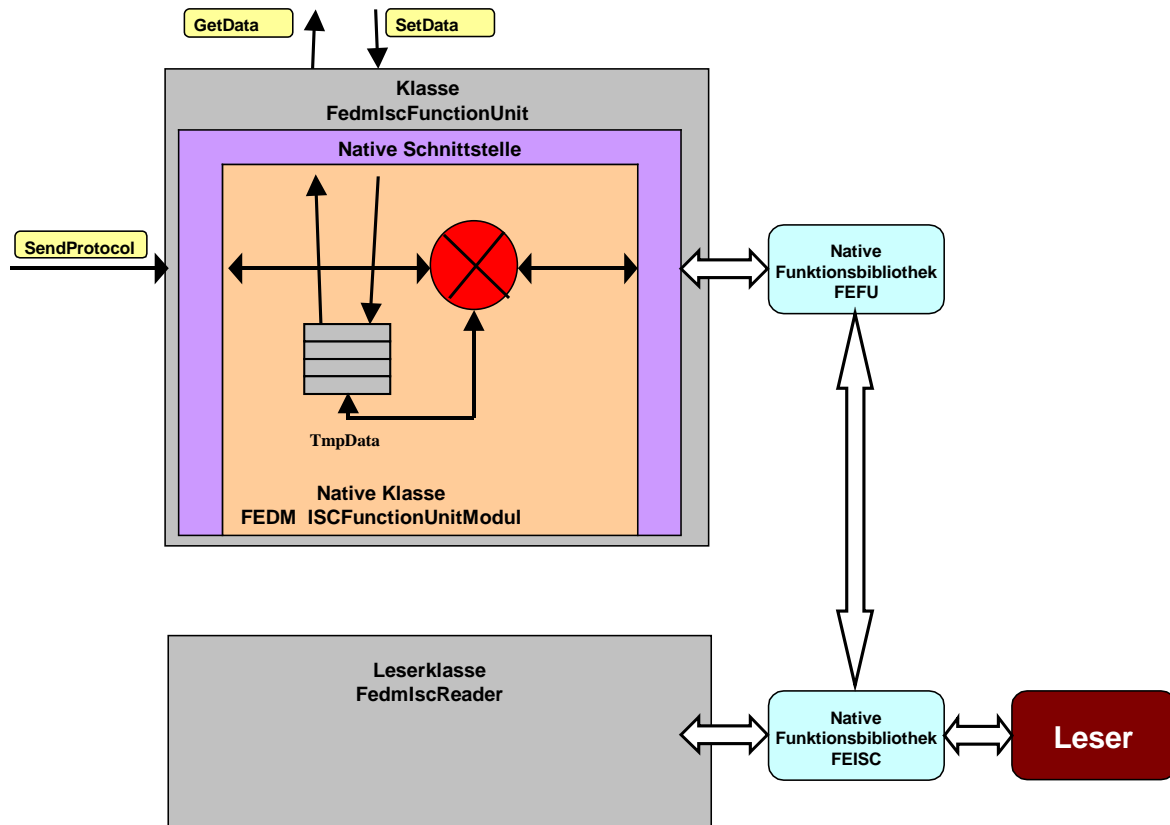
5.3.3.3. Hinweise für den Programmierer

Der Programmierer, der die Datenverschlüsselung in sein Projekt – auch nachträglich - integriert, muss nur wenige Aspekte beachten:

1. Alle Kommunikationsmethoden beginnend mit Send... der Klasse FedmlscReader sind sowohl für die verschlüsselte als auch unverschlüsselte Datenübertragung geeignet.
2. Es muss sichergestellt sein, dass jeder OBID i-scan®- oder OBID®classsic-pro Leser über ein eigenes Leser-Objekt programmiert wird, denn dieses verwaltet die individuell für jeden Leser kalkulierten Sessiondaten.
3. Bei einem Verbindungsaufbau mit FedmlscReader.ConnectTCP muss dasPasswort übergeben werden.
4. Erhält der Host nach der Übertragung eines verschlüsselten oder unverschlüsselten Protokolls den Status 0x19 muss er einen Reader-Authent ausführen.
5. Erhält die Applikation die Fehlercodes -4093 oder -4094 muss ein Reader-Authent ausgeführt werden.
6. Die Datenübertragung im Notification- bzw. Access-Mode erfolgt bei aktiviertem Kryptomode verschlüsselt. Deshalb muss das Passwort in der Klasse FedmTaskOption hinterlegt werden.
7. Wird der Kryptomode in der Leserkonfiguration abgeschaltet, wechselt das Leser-Objekt mit dem nächsten unverschlüsselten Protokoll selbständig wieder in den Mode der unverschlüsselten Datenübertragung. Das bestehende Leser-Objekt kann also weiter benutzt werden. Ebenso ist ein Verbindungsabbau und erneuter Verbindungsaufbau nicht notwendig.

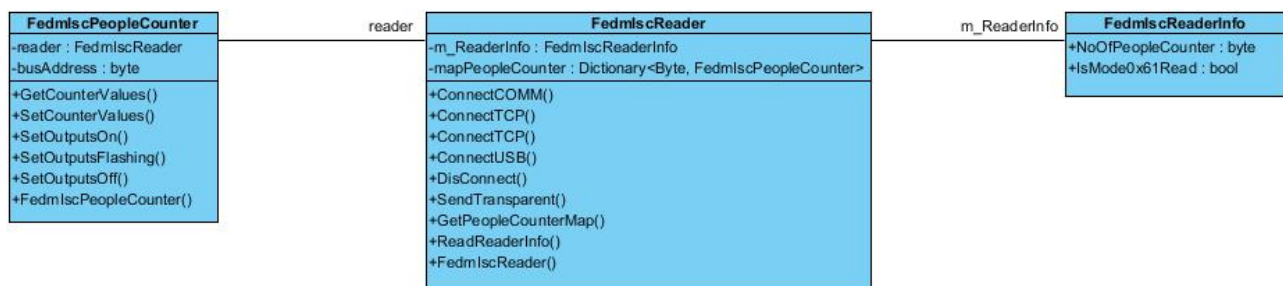
5.4. Kommunikation mit einer Funktionseinheit

Der Ablauf der Kommunikation mit einer Funktionseinheit funktioniert analog zur Kommunikation mit einem Leser. Somit muss das Anwendungsprogramm **vor** dem Aufruf von *SendProtocol* alle für dieses Protokoll notwendigen Daten im Datencontainer TmpData an die richtigen Stellen schreiben. Ebenso sind die Empfangsdaten an bestimmten Stellen im Datencontainer TmpData hinterlegt. Den Schlüssel zu den Daten im Datencontainer TmpData sind auch hier die Zugriffskonstanten.



5.5. Kommunikation mit einem People Counter

People Counter werden vom Leser eigenständig nach dem Einschalten erkannt. Mit dem Abfrageprotokoll [0x66] Get Reader Info mit Mode-Byte 0x61 bzw. mit der Methode ReadReaderInfo() wird der Leserklasse FedmlscReader die Anzahl der detektierten People Counter mitgeteilt. Mit der Methode GetPeopleCounterMap() kann der Applikationsentwickler zur Laufzeit ein Dictionary aller People Counter Objekte vom Typ FedmlscPeopleCounter abrufen und anschließend mit jedem People Counter kommunizieren. Der Schlüssel des Dictionary ist die Busadresse des People Counters im Bereich 1...3.



Die Methoden ConnectUSB, ConnectTCP und (optional) ConnectCOM führen nach erfolgreichem Verbindungsaufbau intern ein ReadReaderInfo() aus. Damit sind alle Informationen über den angeschlossenen Leser und People Counter in der Leserklasse FedmlscReader gespeichert und die Liste der angeschlossenen People Counter kann mit der Methode GetPeopleCounterMap() sofort abgefragt werden. Die Verwendung der Klasse FedmlscPeopleCounter wird in der Bibliotheksreferenz [7.4. FedmlscPeopleCounter](#) mit Beispielen erklärt.

5.6. Datencontainer

Datencontainer haben die Aufgabe, alle Leserparameter bzw. temporäre Protokolldaten strukturiert zu verwalten. Intern sind alle Datencontainer als Byte-Arrays im Motorola-Format (Big Endian) organisiert. Dieses Format entspricht jedem OBID®-Leser. Die Umwandlung in das für Intel-basierte PCs notwendige Intel-Format (Little Endian) übernehmen die überladenen Zugriffsmethoden.

Die Byte-Arrays sind organisiert in Blöcke zu je 16 bzw. 32 Byte. Auch diese Organisation entspricht dem der Leser.

Insgesamt 3 Datencontainer sind integriert.

Datencontainer	Beschreibung
EEData	für Konfigurationsparameter des Lesers
RAMData	für temp. Konfigurationsparameter des Lesers
TmpData	für allgemeine temp. Protokolldaten

5.6.1. Datenaustausch

Der Zugriff auf die temporären Protokolldaten ist vorrangig mit den überladenen Methoden *SetData* und *GetData* möglich. Mit jedem Methodenaufruf kann genau ein Parameter gelesen bzw. geschrieben werden, der durch eine Zugriffskonstante (s. [5.6.2. Zugriffskonstanten für temporäre Protokolldaten](#)) identifiziert wird.

Der Datenaustausch mit Konfigurationsparametern wird mit den überladenen Methoden *SetConfigPara* und *GetConfigPara* realisiert.

Die nachfolgenden Kapitel zeigen die Verwendung von *GetData* und *SetData* für verschiedene Datentypen. *GetConfigPara* und *SetConfigPara* sind analog zu verwenden, mit dem Unterschied, dass für die Zugriffskonstante der String des Parameternamens aus dem Namespace OBID.ReaderConfig übergeben wird.

5.6.1.1. Konstante Daten

```
int iErr = SetData(OBID.ReaderCommand._0x80.Req.CfgAdr.LOCATION, false);           // bool
int iErr = SetData(OBID.ReaderCommand._0x66.Req.MODE, (byte)1);                  // byte
int iErr = SetData(OBID.ReaderCommand._0x66.Req.MODE, (long)134);                // long
int iErr = SetData(OBID.ReaderCommand._0x66.Req.MODE, "0134");                  // String
```

5.6.1.2. Datentyp bool

```
bool data = false;
int iErr = GetData(OBID.ReaderCommand._0x74.Rsp.Inputs.IN1, out data);
iErr = SetData(OBID.ReaderCommand._0xB0.SubCmd._0x01.Req.Mode.MORE, data);
```

5.6.1.3. Datentyp byte

```
byte data = 1;
int iErr = GetData(OBID.ReaderCommand._0x74.Rsp.INPUTS, out data);
iErr = SetData(OBID.ReaderCommand._0x66.Req.MODE, data);
```

5.6.1.4. Datentyp byte[]

```
byte[] data = new byte[31];
int iErr = GetData(OBID.ReaderCommand._0x66.Rsp.READER_INFO, out data);
iErr = SetData(OBID.ReaderCommand._0xA0.Req.PASSWORD, data);
```

5.6.1.5. Datentyp uint

```
uint data = 0;
int iErr = 0;
iErr = GetData(OBID.ReaderCommand._0xB0.SubCmd._0x24.Rsp.NormAddrMode.DB_ADDRESS_ERROR, out data);
iErr = SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24.Req.ExtAddrMode.DB_ADR, data);
```

5.6.1.6. Datentyp long

```
long data = 0;
int iErr = GetData(OBID.ReaderCommand._0x74.Rsp.INPUTS, out data);
iErr = SetData(OBID.ReaderCommand._0x66.Req.MODE, data);
```

5.6.1.7. Datentyp string

ALLE Daten, die mit einer *GetData(string id, out string data)* Methode gelesen werden, sind "Hex"-Strings. Das bedeutet z. B., dass der numerische Wert 159 nicht als "159" sondern als "9F" übergeben wird. String-Werte bestehen somit immer aus einer geraden Anzahl Zeichen. Zur Konvertierung von String-Werten in andere Datentypen bzw. umgekehrt steht die Methodensammlung in der Klasse **FeHexConvert** (s. [4.6.6. Die Klasse FeHexConvert](#)) zur Verfügung.

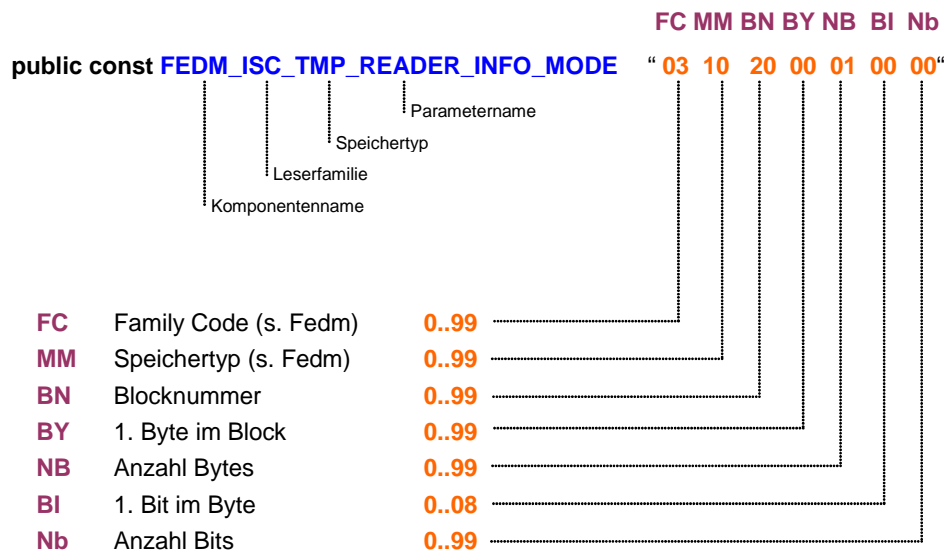
Zur Umwandlung von numerischen Werten in Strings, die aus dem o. g. Beispiel ein "159" machen, bedient man sich am Besten bei den Methoden des .Net-Framework.

```
String data;
int iErr = GetData(OBID.ReaderCommand._0x74.Rsp.INPUTS, out data);
iErr = SetData(OBID.ReaderCommand._0x66.Req.MODE, data);
```

5.6.2. Zugriffskonstanten für temporäre Protokolldaten

Eine zentrale Rolle im Datenverkehr zwischen Anwendungsprogramm und Datencontainer der Leserkasse bzw. FU-Klasse, sowie innerhalb der Klassen zwischen Protokollmethode und Datencontainer spielen die Zugriffskonstanten. Sie identifizieren einerseits den Parameter und enthalten gleichzeitig kodiert den Ablageort in einem der Datencontainer.

Eine Zugriffskonstante ist ein String und hat generell folgenden Aufbau:



Diese Zugriffskonstanten werden ausschließlich mit den Methoden *SetData* und *GetData* der Leserkasse **FedmlscReader** bzw. der Klasse **FedmlscFunctionUnit** verwendet. Die Zugriffskonstante sagt nichts über den Datentyp eines Parameters aus. Dieser wird nur durch den Datentyp der Zugriffsmethode bestimmt. Man kann also die im obigen Beispiel dargestellte Busadresse entweder z. B. als Integer oder als String oder einen anderen plausiblen Datentyp auslesen (s. [5.6.1. Datenaustausch](#)).

Alle Zugriffskonstanten sind in den Strukturen **FedmlscReaderID** und **FedmlscFunctionUnitID** enthalten.

Im Zuge der Entwicklung hat sich die Sammlung von Zugriffskonstanten extrem erweitert und wurde unübersichtlich. Als Alternative gibt es ab der V4.04.00 den Namespace **OBID.ReaderCommand** mit einer strukturierten Sammlung aller Zugriffskonstanten. Beispiele finden sich bereits in den Kapiteln [5.6.1.1](#) bis [5.6.1.7](#).

Die Strukturierung erfolgt nach folgendem Prinzip:

<Namespace>.<CommandByte>.Req.<Parameter> für einen Parameter im Sendeprotokoll

oder

< Namespace>.<CommandByte>.Rsp.<Parameter> für einen Parameter im Empfangsprotokoll

Parameter sind immer in Grossbuchstaben, Gruppen in Gross-Klein-Buchstaben anzugeben. Mit Intellisense der Entwicklungsumgebungen wird aber auch der aktuell verfügbare Bereich zur Auswahl angezeigt.

Beispiel:

```
// command byte for inventory
```

```
SetData(ReaderCommand.|
```

▲ 1 of 6 ▼ int FedmIscReader.SetData(string id, bool data) ✓

```
if(bAll)
```

```
{
```

```
    SetData(FedmIscRea
```

```
    SetData(FedmIscRea
```

```
    ResetTable(FedmIsc
```

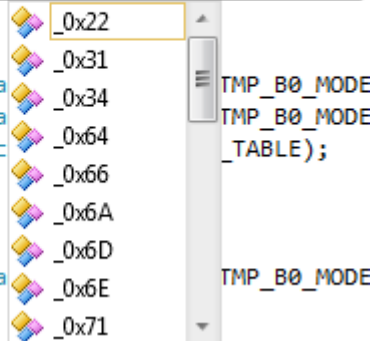
```
}
```

```
else
```

```
{
```

```
    SetData(FedmIscRea
```

```
}
```



```
// command byte for inventory
```

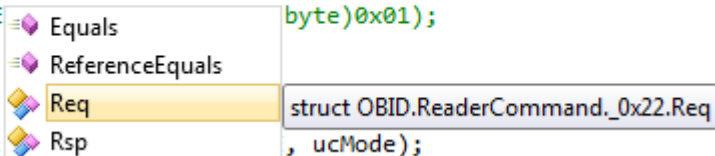
```
SetData(ReaderCommand._0x22.|
```

```
//SetData(FedmIscReaderID.FEDM_I
```

```
if(bAll)
```

```
{
```

```
    SetData(FedmIscReaderID.
```



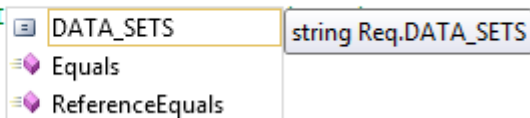
```
// command byte for inventory
```

```
SetData(ReaderCommand._0x22.Req.
```

```
//SetData(FedmIscReaderID.FEDM_I
```

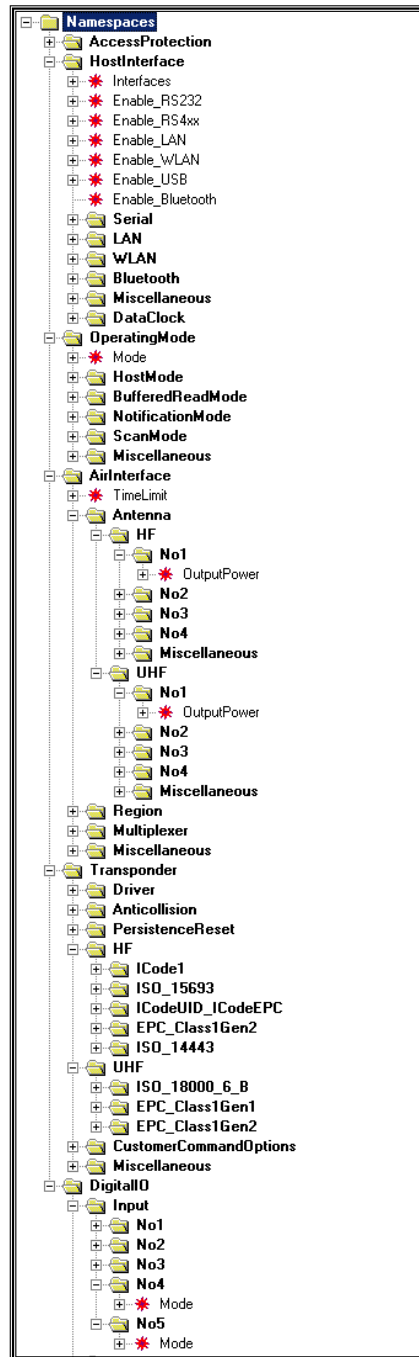
```
if(bAll)
```

```
{
```



5.6.3. Leser-Konfigurationsparameter im Namespace OBID.ReaderConfig

Der Datenverkehr zwischen dem Anwendungsprogramm und den Datencontainern für Konfigurationsparameter in der Klassenbibliothek wird mit überladenen Methoden realisiert, die als Identifikator einen Parameternamen als String aus dem Namespace OBID.ReaderConfig übergeben bekommen. Die Namen aller Konfigurationsparameter sind für alle Lesertypen vereinheitlicht worden, was den Vorteil der Vereinfachung der Programmstruktur zur Folge hat. Der Namensaufbau ist hierarchisch gegliedert in Gruppen und Untergruppen, die durch Punkte geteilt sind. Alle Namen zusammen ergeben eine Baumstruktur.



Ausschnitt aus der Baumstruktur des Namespace OBID.ReaderConfig

5.6.4. Verwaltung der Leserkonfiguration

Jeder OBID i-scan® und OBID®*classic-pro* Leser wird durch Parameter gesteuert, die Blockweise gruppiert in einem EEPROM gespeichert und im Systemhandbuch zum jeweiligen Leser ausführlich beschreiben sind. Nach dem Einschalten bzw. Reset des Lesers werden alle Parameter in das RAM geladen, ausgewertet und in die Steuerung einbezogen.

Alle Parameter können mit einem Protokoll verändert werden, damit das Verhalten des Lesers an die Applikation angepasst werden kann. Idealerweise verwendet man das Programm ISOStart für diese Anpassung und normalerweise müssen in der Applikation keine Parameter mehr geändert werden. Trotzdem kann es vorkommen, dass aus einem Programm heraus ein oder mehrere Parameter umgestellt werden müssen. Dieses Kapitel soll Sie mit der Vorgehensweise unter Verwendung der Leserklasse an einem Beispiel vertraut machen.

Eine gemeinsame Eigenschaft aller Leser ist die Blockweise Gruppierung von thematisch verwandten Parametern zu je 14 Byte je Konfigurationsblock. Jeder Parameter kann nicht einzeln adressiert werden, sondern muss immer zusammen mit einem Konfigurationblock mit dem Protokoll [0x80] Read Configuration ausgelesen, dann modifiziert und schließlich wieder mit dem Protokoll [0x81] Write Configuration in den Leser geschrieben werden. Dieser Zyklus ist immer einzuhalten und wird von der Leserklasse FedmlscReader auch kontrolliert. Das bedeutet, dass das Schreiben eines Konfigurationsblockes ohne vorheriges Lesen desselben Blocks nicht möglich ist, mit der Ausnahme, dass nach der Übernahme von Konfigurationsparametern aus einer XML-Datei das Schreiben in den Leser immer möglich ist.

Die Leserklasse verwaltet die Konfigurationsdaten in einem (public) Byte-Array EEData für Daten aus dem EEPROM bzw. RAMData für Daten aus dem RAM des Lesers. Die Unterscheidung ist von Bedeutung, da Änderungen im RAM sofort zur Anwendung kommen, während Änderungen im EEPROM des Lesers erst nach einem Reset wirksam werden. Deshalb hat die Leserklasse für beide Konfigurationssätze eigene Byte-Arrays.

Am Beispiel des Konfigurationsblockes CFG2 des Lesers ID ISC.LR2000, der Parameter zur Einstellung der digitalen Ein- und Ausgänge enthält, soll erläutert werden, wie man gezielt einen Parameter mit Hilfe der Leserklasse modifiziert.

Byte	0	1	2	3	4	5	6
Contents	IDLE-MODE		FLASH-IDLE		IN-ACTIVE	0x00	REL1-TIME
Default	0x88A8		0xCC00		0x00		0x00

Byte	7	8	9	10	11	12	13
Contents	REL1-TIME	OUT1-TIME		REL2-TIME		REL3-TIME	REL4-TIME
Default	0x00	0x0000		0x0000			0x0000

IDLE-MODE:

Defines the status of the signal emitters (OUT1 and RELx) during the idle mode.

Bit:	15	14	13	12	11	10	9	8
Function:	REL1 mode		0	0	OUT1 mode		0	0

	7	6	5	4	3	2	1	0
	REL2 mode		REL3 mode		REL4 mode		0	0

Mode	Function	
b 0 0	UNCHANGED	no effect on the status of the signal emitter
b 0 1	ON	signal emitter on
b 1 0	OFF	signal emitter off
b 1 1	FLASH	signal emitter alternating on

Dargestellt ist die Belegung des Konfigurationsblockes CFG2. Der Parameter IDLE-MODE belegt zwei Bytes und enthält Sub-Parameter für vier Relais und einen digitalen Ausgang. Jeder Ausgang kann für einen von vier Zuständen entsprechend der Tabelle konfiguriert werden. Da das Feld IDLE-MODE nicht grau unterlegt ist, kann die Modifikation im RAM des Lesers erfolgen.

Für die Änderung von REL1 Mode innerhalb von IDLE-MODE sind nun folgende Schritte notwendig:

```
// das Beispiel zeigt das Lesen, Modifizieren und Zurückschreiben eines Blocks der Leserkonfiguration
// Modifiziert wird der Idle-Mode des Relais 1
// reader ist ein Objekt der Leserklasse FedmlscReader

byte CfgAdr = 2;           // Adresse des Konfigurationsblocks
bool EEProm = false;       // Konfigurationsdaten aus/in RAM des Lesers
uint IdleModeRel1          // Parameter IDLE-MODE für Relais 1

// Voreinstellungen für das nächste SendProtocol
reader.SetData(OBID.ReaderCommand._0x80.Req.CFG_ADDRESS, (byte)0); // alles zurücksetzen
reader.SetData(OBID.ReaderCommand._0x80.Req.CfgAddr.ADDRESS, CfgAdr); // Adresse setzen
reader.SetData(OBID.ReaderCommand._0x80.Req.CfgAddr.LOCATION, EEProm); // Speicherort auf RAM setzen

// Konfigurationsdaten lesen
reader.SendProtocol(0x80);

IdleModeRel1 = 3;          // REL1 soll alternierend anziehen (Anm: Frequenz wäre im Parameter IDLE-FLASH einzustellen)
```

```
// neuen Wert in Leserklasse speichern (Ziel: RAM des Lesers)
```

```
reader.SetConfigPara(OBID.ReaderConfig.DigitalIO.Relay.No1.IdleMode, IdleModeRel1, false);
```

```
// Voreinstellungen für das nächste SendProtocol
```

```
reader.SetData(OBID.ReaderCommand._0x81.Req.CFG_ADDRESS, (byte)0); // alles zurücksetzen
```

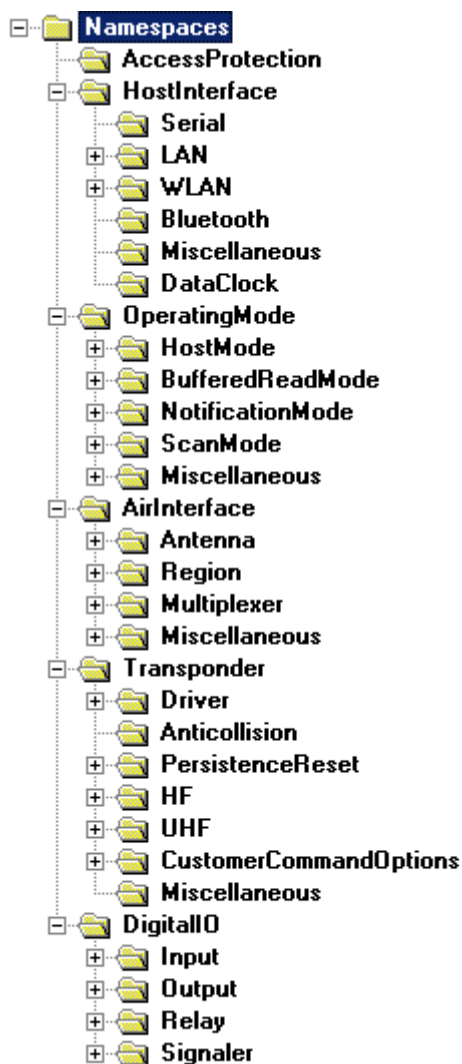
```
reader.SetData(OBID.ReaderCommand._0x81.Req.CfgAddr.ADDRESS, CfgAdr); // Adresse setzen
```

```
reader.SetData(OBID.ReaderCommand._0x81.Req.CfgAddr.LOCATION, EEPROM); // Speicherort auf EEPROM setzen
```

```
// Konfigurationsdaten zurückschreiben
```

```
reader.SendProtocol(0x81);
```

Die Methoden *GetConfigPara* und *SetConfigPara* bekommen im ersten Übergabeparameter einen Parameternamen als String aus dem Namespace **OBID.ReaderConfig** übergeben. Dieser Namespace kapselt in weiteren internen Strukturen einheitlich alle Parameternamen aller OBID i-scan® und OBID®classic-pro Leser. Nachfolgend sind die Haupt- und die ersten Untergruppen dargestellt.



Dieses Schema hat den Vorteil, dass mit den heute verfügbaren Intellisense-Funktionen der IDEs die Auswahl des gewünschten Namens für einen Konfigurationsparameter schnell gefunden werden kann.

5.6.5. Serialisierung

Die integrierte Serialisierungsmethode *Serialize* erlaubt das Speichern der Leserkonfiguration aus den Datencontainern in eine Datei bzw. laden der Leserkonfiguration aus einer Datei in Datencontainer.

Mit der Standardisierung von XML (Extensible Markup Language) hat sich eine Beschreibungssprache für Dokumente durchgesetzt, die unabhängig der eingesetzten Computersprache und Betriebssysteme verwendet werden kann. Es ist folglich naheliegend, mit dieser Sprache die Struktur einer Leserkonfigurationsdatei zu definieren. Nachfolgend ist der Inhalt einer XML-Datei dargestellt, die mit dem Programm ISOStart erstellt wurde:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<OBID>
  <file-header>
    <document-type>Reader Configuration File</document-type>
    <document-version>1.0</document-version>
    <reader-family>ISC</reader-family>
    <reader-name>ID ISC.MR100</reader-name>
    <reader-type>74</reader-type>
    <host-address>192.168.3.3</host-address>
    <port-number>10001</port-number>
    <communication-mode>TCP</communication-mode>
    <program-name>ID ISOStart</program-name>
    <program-version>05.03.03</program-version>
    <fedm-version>01.08</fedm-version>
    <date>07/18/03</date>
    <time>11:13:28</time>
  </file-header>
  <data-array name="Reader EEPROM-Parameter" blocks="16" size="16">
    <CFG0 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG1 b0="00" b1="00" b2="08" b3="01" b4="00" b5="00" b6="00" b7="0A" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG2 b0="00" b1="20" b2="00" b3="25" b4="00" b5="04" b6="00" b7="2F" b8="0A" b9="64" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG3 b0="00" b1="39" b2="00" b3="07" b4="00" b5="00" b6="06" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG4 b0="00" b1="00" b2="00" b3="00" b4="09" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG5 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="04" b12="00" b13="00" b14="00" b15="00"/>
    <CFG6 b0="00" b1="00" b2="00" b3="01" b4="00" b5="00" b6="00" b7="0A" b8="00" b9="00" b10="00"
      b11="05" b12="04" b13="00" b14="00" b15="00"/>
    <CFG7 b0="02" b1="20" b2="2C" b3="01" b4="0D" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG8 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG9 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG10 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG11 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG12 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG13 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG14 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG15 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
      b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
  </data-array>
  <data-array name="Reader RAM-Parameter" blocks="16" size="16">
    <CFG0 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG1 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG2 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
      b11="00" b12="00" b13="00" b14="00" b15="00"/>
    <CFG3 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
```

```
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG4 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG5 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG6 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG7 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG8 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG9 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00" b10="00"
b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG10 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG11 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG12 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG13 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG14 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
<CFG15 b0="00" b1="00" b2="00" b3="00" b4="00" b5="00" b6="00" b7="00" b8="00" b9="00"
b10="00" b11="00" b12="00" b13="00" b14="00" b15="00"/>
</data-array>
</OBID>
```

Neben einigen Headerdaten finden sich in den Tags `<data-array name="Reader EEPROM-Parameter" blocks="16" size="16">` und `<data-array name="Reader RAM-Parameter" blocks="16" size="16">` die Leserparameter als Hex-Werte.

Mit der Methode *Serialize* kann nun diese Datei erstellt, bzw. die Leserkonfiguration einer solchen Datei ausgelesen und in den internen Speicher *EEData* bzw. *RAMData* übernommen werden. Voraussetzung für die Generierung der Konfigurationsdatei ist, dass zuvor die gesamte Leserkonfiguration mit *SendProtocol* ausgelesen wurde.

Zum Erzeugen einer Leserkonfigurationsdatei verwendet man den Aufruf:

```
Serialize(false, "c:\\tmp\\myreader.xml")
```

und zum Einlesen der Daten aus einer Leserkonfigurationsdatei den Aufruf:

```
Serialize(true, "c:\\tmp\\myreader.xml")
```

5.7. Tabellen

OBID *i-scan*® und OBID® *classic-pro* Leser unterstützen Protokolle, die für mehrere Transponder Daten transportieren können (ISO-Host Mode, Buffered Read Mode, Notification Mode) und eine Speicherung in den Containern unmöglich machen. Idealerweise speichert man diese Daten strukturiert in einer Tabelle. Die Leserklasse **FedmlscReader** enthält für diese Transponderdaten die Tabellen ISOTable und BRMTable. Mit den Methoden *GetTableData*, *SetTableData* und *FindTableIndex* ist ein Zugriff auf die Tabellendaten möglich. Die Methoden *GetTableSize*, *SetTableSize*, *GetTableLength* und *ResetTable* sind für die Tabellenverwaltung. Zusätzlich kann mit der Methode *VerifyTableDataBlocks* ein Vergleich der gesendeten zu den empfangenen Transponderdaten (nur für ISO-Host Mode) vorgenommen werden.

Der Zugriff auf Tabellendaten mittels der Methoden *SetTableData* und *GetTableData* erfolgt zwar auch mit eindeutigen Konstanten wie in *GetData* und *GetData* für Datencontainer. Sie stellen aber keinen String dar und enthalten somit auch keine Ortskodierung. Stattdessen ist mit dem Tabellenindex (idx) und den Konstanten für den Tabellentyp (tableID) und die Tabellenvariable (dataID) eine eindeutige Identifizierung eines Tabellenwertes möglich.

Beispiel:

```
int GetTableData( int dataID, out uint data )
```

Alle Konstanten für den Tabellentyp und für die Tabellenvariablen sind im Interface **FedmlscReaderConst** enthalten.

Tabellen können alternativ auch mit der Methode *GetTable* als Tabellenobjekte vom Typ **FedmlsoTableItem[]** oder **FedmBrmTableItem[]** ausgegeben werden. Die Verwendung des Methodeninterfaces der Tabellenklassen ist analog. Mit der Methode *SetTable* kann man auch eine in der Applikation erstellte und gefüllte Tabelle in die Leserklasse übergeben und anschließend diese Daten auf den Transponder schreiben.

Die Methoden *GetTableItem* und *SetTableItem* erlauben den Austausch von einzelnen Tabellenelementen.

Wichtiger Hinweis: ein neues Leserobjekt hat undimensionierte Tabellen. Sie müssen nach dem Erstellen des Leserobjektes daher unbedingt sofort die Größe der verwendeten Tabelle mit der Methode *SetTableSize* einstellen (s. [5.1.1. Initialisierung](#)).

5.8. Kommunikation mit Transpondern im Host-Mode

Für die Programmierung der Kommunikation mit Transpondern stehen zwei Möglichkeiten in der Leserklasse FedmlscReader bereit:

1. Tabellen-orientiertes API (s. [8.2. Tabellen orientierte Kommandos für den Leser](#))
2. TagHandler API, basierend auf spezialisierten Transponder-Klassen (s. 9. Beispiel für die Verwendung von TagHandler-Klassen)

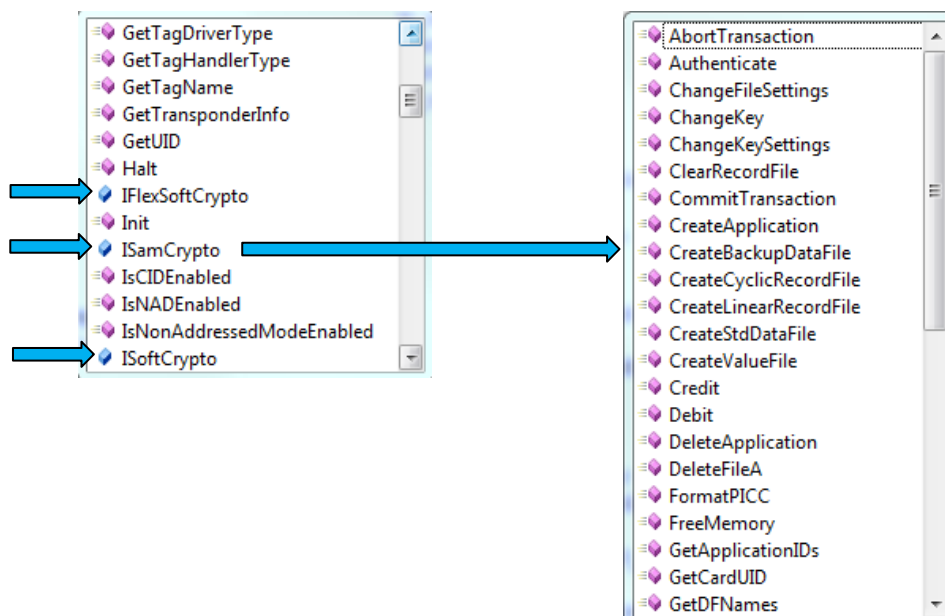
Für neue Projekte wird letzteres empfohlen. TagHandler-Klassen bieten ein auf den jeweiligen Transponder-Standard (ISO 14443, ISO 15693, EPC Class 1 Gen 2) bzw. einen speziellen Chip-Typ zugeschnittenes API an. Jeder Standard bzw. Chip-Typ wird als Klasse implementiert, wobei die Klassen ein hierarchisches System aus abgeleiteten Klassen bilden. Grundklasse ist die Klasse FedmlscTagHandler.

Voraussetzung zur Nutzung der TagHandler-Klassen ist die Verwendung der Methoden TagInventory und TagSelect in der Leserklasse FedmlscReader und die Identifizierbarkeit des Transponder-Standards bzw. Chip-Typs zur Auswahl der zugehörigen TagHandler-Klasse. Nicht unterstützte Chip-Typen werden automatisch der Grundklasse zugeordnet.

TagHandler-Objekte werden von der Tabelle FedmlscTableItem verwaltet. Sie nutzen folglich intern das Tabellen-orientierte API.

Das Methoden-Interface von TagHandler-Klassen setzt sich aus der Befehlsliste von Transponder-Standards bzw. Chip-Typen zusammen. Folglich arbeitet man mit der Dokumentation von Standards bzw. der Dokumentation des Transponder-Herstellers mit denen die Bedeutung des Methoden-Interfaces erst verständlich wird.

In der folgenden Abbildung wird am Beispiel des ISO 14443-A Transponders MIFARE DESFire das Methoden-Interface (links) der TagHandler Klasse FedmlscTagHandler_ISO14443_4_MIFARE_DESFire und, nach Auswahl eines internen Interfaces – hier: ISamCrypto – das eigentliche Methoden-Interface des Transponders dargestellt.



H40301-23d-ID-B.docx



6. Fehlerbehandlung

6.1. Rückgabewert

Viele Methoden der Klassenbibliothek führen intern Fehleruntersuchungen durch und geben im Fehlerfall einen negativen Wert zurück. Die Fehlercodes der .NET-Klassenbibliothek ID OBIDISC4NET sind direkt von den nativen Implementierungen übernommen worden. Sie sind so in Bereiche organisiert, dass sie sich nicht überlappen können. Folgende Bereiche sind für die C++ Klassenbibliothek ID FEDM und die nativen OBID®-Funktionsbibliotheken reserviert:

Bibliothek	Wertebereich für Fehlercodes	Referenz
ID FEDM	-101 ... -999	10.1.Liste der Fehlercodes
ID FECOM	-1000...-1099	H80592-xx-ID-B
ID FEUSB	-1100...-1199	H00501-xx-ID-B
ID FETCP	-1200...-1299	H30802-xx-ID-B
ID FEISC	-4000...-4099	H9391-xx-ID-B
ID FEFU	-4100...-4199	H30801-xx-ID-B
IF FETCL	-4200...-4299	H50401-xx-ID-B

Mit der Methode *GetErrorText* der Leserkasse kann man zum Fehlercode einen Fehlertext abrufen. Der übergebene Fehlercode kann auch aus dem Bereich einer nativen OBID®-Funktionsbibliothek kommen.

Der letzte Fehlercode wird intern gespeichert und kann mit der Methode *GetLastError* abgerufen werden.

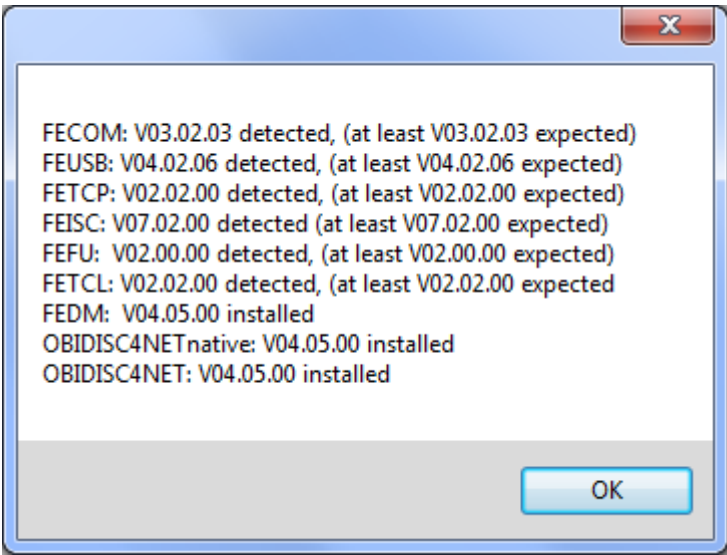
6.2. Ausnahmen (Exceptions)

In Ausnahmesituationen während der Kommunikation werden Ausnahmen ausgelöst. In der Bibliotheksreferenz der Klasse *FedmlscReader* ist angegeben, ob und welche Exception geworfen wird.

7. Bibliotheksreferenz

7.1. FedmlscReader

7.1.1. GetDependentLibVersions

Funktion	Methode zum Abfragen der Bibliotheks-Versionen.
Syntax	string GetDependentLibVersions()
Beschreibung	Diese Methode erzeugt einen String mit allen Versionsnummern abhängiger Bibliotheken. Dies kann nützlich sein für Logdateien oder zur Info-Ausgabe.
Exceptions	-
Beispiel	<pre>using OBID; ... string Info = reader.GetDependentLibVersions(); Console.WriteLine(Info);</pre>  <p>The screenshot shows a standard Windows dialog box with a blue title bar and a red 'X' button in the top right corner. The main area contains the following text:</p> <pre>FECOM: V03.02.03 detected, (at least V03.02.03 expected) FEUSB: V04.02.06 detected, (at least V04.02.06 expected) FETCP: V02.02.00 detected, (at least V02.02.00 expected) FEISC: V07.02.00 detected, (at least V07.02.00 expected) FEFU: V02.00.00 detected, (at least V02.00.00 expected) FETCL: V02.02.00 detected, (at least V02.02.00 expected) FEDM: V04.05.00 installed OBIDISC4NETnative: V04.05.00 installed OBIDISC4NET: V04.05.00 installed</pre> <p>At the bottom right of the dialog box is an 'OK' button.</p>

7.1.2. FedmIscReader

Funktion	Erzeugt eine neue Instanz der Klasse FedmIscReader (Konstruktor).
Syntax	FedmIscReader()
Beschreibung	Es wird ein Leser-Objekt erzeugt. Nur mit einem Leser-Objekt können die Protokollfunktionen ausgeführt werden.
Rückgabewert	Wenn ein Leser-Objekt fehlerfrei erstellt werden konnte, wird die neue Instanz der Klasse FedmIscReader zurückgeliefert.
Exceptions	Im Fehlerfall wirft die Methode die Ausnahme FedmException .
Beispiel	<pre>using OBID; ... try { reader = new FedmIscReader(); // neue Instanz erzeugen } catch (FedmException e) { Console.WriteLine("Ausnahme beim Erzeugen eines neuen Reader-Objektes: " + e.Message); } private FedmIscReader reader;</pre>

7.1.3. ConnectCOMM

Funktion	Öffnet eine serielle Schnittstelle und stellt optional die Verbindung zu einem Leser her
Syntax	void ConnectCOMM(int portNumber, bool withDetect) void ConnectCOMM(int portNumber, bool withDetect, uint authenticType, string authenticKey)
Beschreibung	<p>Es wird eine serielle Schnittstelle geöffnet.</p> <p>Optional, wenn <i>withDetect</i> = true ist, kann ein angeschlossener Leser detektiert, wichtige Informationen vom Leser ausgelesen und der Lesertyp gesetzt werden. Im Fehlerfall, z.B. wenn kein Leser angeschlossen ist, wird die serielle Schnittstelle wieder geschlossen.</p> <p>Der Parameter <i>portNumber</i> muss im Bereich 1..256 liegen.</p> <p>Diese Methode kann mit jedem Leser-Objekt verwendet werden. Falls der Port gewechselt werden soll, muss der geöffnete Port zuerst mit der Methode <i>Disconnect</i> geschlossen werden.</p> <p>Die zweite Methode ruft intern zusätzlich eine Authentifikationsfunktion zur Einleitung einer verschlüsselten Datenübertragung auf.</p> <p><i>authenticType</i>: 0 – AES-128 1 – AES-192 2 – AES-256</p> <p><i>authenticKey</i> enthält das Passwort und ist eine Zeichenkette mit Hexadezimalen Zeichen (0..9, A..F). Bei AES-128 sind 32 Zeichen, bei AES-192 sind 48 Zeichen und bei AES-256 sind 64 Zeichen zu übergeben.</p>
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1, false); // nur COM1 wird geöffnet reader.FindBaudrate(); // Baudrate und Frame werden detektiert reader.ReadReaderInfo(); // Lesertyp wird gesetzt // oder alternativ mit Detekt des Lesers reader.ConnectCOMM(1, true); ... private FedmIscReader reader;</pre>

7.1.4. ConnectTCP

Funktion	Stellt die Verbindung zu einem Server via TCP/IP her.
Syntax	void ConnectTCP(string host, int portNumber) void ConnectTCP(string host, int portNumber, uint authenticType, string authenticKey)
Beschreibung	<p>Beide Methoden stellen eine Verbindung zu einem TCP/IP-Server im Netzwerk her, lesen wichtige Informationen vom Leser aus und setzen intern den Lesertyp. Die Portnummer sollte im Bereich zwischen 1024 und 65535 liegen, um Konflikte mit den „well known ports“ zu vermeiden.</p> <p>Diese Methode kann nur einmal für jedes Leser-Objekt benutzt werden. Falls die Verbindung gewechselt werden soll, muss die geöffnete Verbindung zuerst mit der Methode Disconnect geschlossen werden.</p> <p>Die zweite Methode ruft intern zusätzlich eine Authentifikationsfunktion zur Einleitung einer verschlüsselten Datenübertragung auf.</p> <p><i>authenticType:</i> 0 – AES-128 1 – AES-192 2 – AES-256</p> <p><i>authenticKey</i> enthält das Passwort und ist eine Zeichenkette mit Hexadezimalen Zeichen (0..9, A..F). Bei AES-128 sind 32 Zeichen, bei AES-192 sind 48 Zeichen und bei AES-256 sind 64 Zeichen zu übergeben.</p>
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>using OBID; ... reader.ConnectTCP(„192.168.1.127“, 10001); // Verbindung zum Server an der // Adresse „192.168.1.127“ private FedmIscReader reader;</pre>

7.1.5. ConnectUSB

Funktion	Stellt die Verbindung zu einem Leser via USB her.
Syntax	void ConnectUSB(long deviceId) void ConnectUSB(long deviceId, uint authenticType, string authenticKey)
Beschreibung	<p>Die Methode stellt die Verbindung zu einem Leser über USB her, liest wichtige Informationen vom Leser aus und setzt den Lesertyp.</p> <p>Wenn für den Parameter <i>deviceId</i> eine 0 übergeben wird, dann wird der zuerst gescannte USB-Leser benutzt. In diesem Fall sollte nur ein USB-Leser mit dem PC verbunden sein.</p> <p>Die zweite Methode ruft intern zusätzlich eine Authentifikationsfunktion zur Einleitung einer verschlüsselten Datenübertragung auf.</p> <p><i>authenticType</i>: 0 – AES-128 1 – AES-192 2 – AES-256</p> <p><i>authenticKey</i> enthält das Passwort und ist eine Zeichenkette mit Hexadezimalen Zeichen (0..9, A..F). Bei AES-128 sind 32 Zeichen, bei AES-192 sind 48 Zeichen und bei AES-256 sind 64 Zeichen zu übergeben.</p> <p>Sind mehrere USB-Leser angeschlossen und müssen zur gleichen Zeit behandelt werden, kann die Hilfsklasse FeUsb herangezogen werden. Mit ihr kann man den USB-Bus nach Lesern scannen.</p> <p>Falls die Verbindung gewechselt werden soll, muss die geöffnete Verbindung zuerst mit der Methode Disconnect geschlossen werden.</p>
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>using OBID; reader.ConnectUSB(0); // Verbindung zum USB-Leser herstellen ... private FedmIscReader reader;</pre>

7.1.6. Disconnect

Funktion	Diese Methode trennt die Verbindung zwischen Leser und PC und gibt reservierten Speicher wieder frei.
Syntax	int Disconnect()
Beschreibung	<p>Diese Methode trennt die Verbindung zwischen Leser und PC und gibt reservierten Speicher wieder frei.</p> <p>Wenn eine TCP/IP-Verbindung geschlossen wurde, entspricht der Rückgabewert dem letzten Status der Verbindung. Wenn der Zustand TIME_WAIT ermittelt wird, dann gibt diese Funktion 0 zurück um anzuzeigen, dass die Verbindung korrekt geschlossen wurde.</p> <p>Siehe auch: 10.4. TCP-Status</p>
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException .
Beispiel	<pre>using OBID; ... reader.ConnectUSB(0); // Verbindung zum USB-Leser herstellen if (reader.Connected) { back = reader.Disconnect(); // Verbindung zum Leser trennen } ... private FedmIscReader reader;</pre>

7.1.7. GetTcpConnectionState

Funktion	Diese Methode gibt den aktuellen Status einer TCP/IP-Verbindung zurück.
Syntax	int GetTcpConnectionState() int GetTcpConnectionState(string HostAdr, uint PortNumber)
Beschreibung	<p>Ermittelt mit einer Kernelfunktion den Status einer TCP/IP-Verbindung.</p> <p>Die 1. Methode kann für eine bestehende Verbindung, also nach einem Aufruf von ConnectTCP() und vor Disconnect(), benutzt werden.</p> <p>Die 2. Methode kann nach einem Aufruf von Disconnect() benutzt werden, wenn die Verbindung noch nicht komplett abgebaut wurde (TCP-Status ist nicht TIME_WAIT), um den Abbauvorgang solange beobachten zu können, bis der Status TIME_WAIT erreicht wurde.</p> <p>Es ist nicht möglich, durch permanentes Polling eine durch Strom- oder Netzwerkausfall unterbrochene Verbindung zu erkennen. Diese Methode ist hilfreich <u>nach</u> einem Kommunikationsfehler um die Ursache eingrenzen zu können.</p> <p>Siehe auch: 10.4. TCP-Status</p>
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException
Beispiel	

7.1.8. SetPortPara

Funktion	Setzt neue Parameter für die serielle Schnittstelle.					
Syntax	void SetPortPara (string parameter, string val)					
Beschreibung	Setzt neue Parameter für die Schnittstelle. Folgende Parameter sind erlaubt:					
	Parameter-kennung	Wertebereich	Default	Einheit	Port	Beschreibung
	Baud	300...115200	9600	bit/s	<input checked="" type="checkbox"/> Seriell <input type="checkbox"/> USB <input type="checkbox"/> TCP/IP	Baudrate der Schnittstelle
	Frame	7N1, 7E1, 7O1, 7N2, 7E2, 7O2, 8N1, 8E1, 8O1	8E1		<input checked="" type="checkbox"/> Seriell <input type="checkbox"/> USB <input type="checkbox"/> TCP/IP	Zeichenrahmen (Datenbits, Parität, Stoppbits)
	Timeout	0...99999	1000 3000 3000	ms	<input checked="" type="checkbox"/> Seriell <input checked="" type="checkbox"/> USB <input checked="" type="checkbox"/> TCP/IP	Maximale Wartezeit auf Empfangs-Protokoll
	TxTimeControl	0, 1	1	-	<input checked="" type="checkbox"/> Seriell <input type="checkbox"/> USB <input type="checkbox"/> TCP/IP	Wenn gesetzt (1), dann wird intern die Ausgabe des nächsten Sendeprotokolls solange verzögert, bis mindestens TxDelayTime (ms) nach dem letzten Empfangsprotokoll vergangen sind. Wenn ungesetzt (0), dann wird das Sendeprotokoll immer schnellstmöglich ausgegeben.
	TxDelayTime	0...999	5	ms	<input checked="" type="checkbox"/> Seriell <input type="checkbox"/> USB <input type="checkbox"/> TCP/IP	Minimale Zeitspanne zwischen letztem Empfangs- und nächstem Sendeprotokoll. Wird nur berücksichtigt, wenn TxTimeControl=1
	CharTimeoutMpy	1...99	1	-	<input checked="" type="checkbox"/> Seriell <input type="checkbox"/> USB <input type="checkbox"/> TCP/IP	Die Zeichen-Timeout für die serielle Schnittstelle wird intern berechnet. Die Zeichen-Timeout bestimmt, nach welcher Zeit nach dem Empfang des letzten Zeichens der Empfangsprozess beendet wird. Mit einigen PCs kann es vermehrt zu Protokollängenfehlern kommen, weil die Wartezeit zu gering ist. In diesem Fall kann mit diesem Parameter die Wartezeit multipliziert werden.
	SetDTR	0, 1	0	-	<input checked="" type="checkbox"/> Seriell <input type="checkbox"/> USB <input type="checkbox"/> TCP/IP	Setzt die DTR Steuerleitung

	SetRTS	0, 1	0	-	<input checked="" type="checkbox"/> Seriell <input type="checkbox"/> USB <input type="checkbox"/> TCP/IP	Setzt die RTS Steuerleitung
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .					
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen reader.setPortPara(„Baud“, „9600“); // Ändern der Baudrate ... private FedmIscReader reader;</pre>					

7.1.9. GetPortPara

Funktion	Fragt die Parameter der Schnittstelle ab.
Syntax	string GetPortPara(string parameter)
Beschreibung	Fragt die Parameter für die serielle Schnittstelle ab. Weitere Informationen befinden sich im Abschnitt SetPortPara
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen string baud = reader.GetPortPara(„Baud“); // Abfragen der Baudrate ... private FedmIscReader reader;</pre>

7.1.10. SetBusAddress

Funktion	Setzt die Busadresse im Leserobjekt.
Syntax	void SetBusAddress(byte busAddress)
Beschreibung	Die Methode setzt die Busadresse im Leserobjekt für die Kommunikation. Die voreingestellte Adresse ist 255 und kann benutzt werden, falls nur ein Leser an der seriellen Schnittstelle angeschlossen ist.
Exceptions	Keine
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen reader.SetBusAddress(1); // Setzen der Busadresse auf 1 ... private FedmIscReader reader;</pre>

7.1.11. GetBusAddress

Funktion	Holt die Busadresse aus einem Leserobjekt.
Syntax	byte GetBusAddress()
Beschreibung	Die Methode holt die eingestellte Busadresse aus einem Leserobjekt.
Rückgabewert	Die Methode gibt die Busadresse zurück.
Exceptions	Keine
Beispiel	<pre>Using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen byte busAddress = reader.GetBusAddress(); // Abfragen der Busadresse Console.Write(„Die eingestellte Busadresse ist: „); Console.WriteLine(busAddress); ... private FedmIscReader reader;</pre>

7.1.12. GetFamilyCode

Funktion	Gibt einen kurzen String mit dem Familiencode zurück.
Syntax	string GetFamilyCode()
Beschreibung	Gibt einen kurzen String mit dem Familiencode zurück.
Rückgabewert	„ISC“
Exceptions	Keine
Beispiel	<pre>Using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen reader.sendProtocol(0x65); // GetSoftVersion string familyCode = reader.GetFamilyCode(); // Abfragen der Leserfamilie Console.WriteLine(„Der Leser ist ein „ + familyCode + „ – Leser“); ... private FedmIscReader reader;</pre>

7.1.13. GetReaderName

Funktion	Gibt einen kurzen String mit dem Lesernamen zurück.
Syntax	String GetReaderName()
Beschreibung	Gibt einen kurzen String mit dem Lesernamen zurück. Dazu muss vorher die Softwareversion gelesen worden sein.
Rückgabewert	Einen kurzen String mit dem Lesernamen.
Exceptions	Keine
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen reader.sendProtocol(0x65); // GetSoftVersion string readerName = reader.GetReaderName(); // Abfragen des Lesernamens Console.WriteLine(„Der Leser hat die Verkaufsbezeichnung: „ + readerName); ... private FedmIscReader reader;</pre>

7.1.14. GetTransponderName

Funktion	Gibt einen kurzen String mit der Transponderbezeichnung zurück.
Syntax	String GetTransponderName(byte type)
Beschreibung	Gibt einen kurzen String mit der Transponderbezeichnung zurück. Der Parameter type enthält den Transpondertyp gemäß der Auflistung im Systemhandbuch des Lesers.
Rückgabewert	Einen kurzen String mit dem Transpondernamen.
Exceptions	Keine
Beispiel	<pre>using OBID; ... string trpName = reader.GetTransponderName(FedmIscReaderConst.TR_TYPE_EPC); Console.WriteLine(„Der Transponder hat die Bezeichnung: „ + trpName); ... private FedmIscReader reader;</pre>

7.1.15. GetReaderType

Funktion	Gibt Lesertypnummer zurück.
Syntax	byte GetReaderType()
Beschreibung	Gibt die Lesertypnummer zurück. Dazu muss vorher die Softwareversion gelesen worden sein.
Rückgabewert	Lesertyp.
Exceptions	Keine
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen reader.sendProtocol(0x65); // GetSoftVersion byte readerType = reader.GetReaderType(); // Abfragen des Lesernamens Console.Write(„Der Leser ist vom Typ: „); Console.WriteLine(readerType); ... private FedmIscReader reader;</pre>

7.1.16. SetReaderType

Funktion	Setzt eine neue Lesertypnummer.
Syntax	void SetReaderType(byte readerType)
Beschreibung	Diese Methode setzt den Lesertyp und die Verkaufsbezeichnung des Lesers. Der Lesertyp muss im Systemhandbuch definiert worden sein. Diese Methode wird automatisch nach einem Aufruf von <i>SendProtocol(0x65)</i> , mit dem die Software- und Leserversion gelesen wird, aufgerufen. Alle Lesertypkonstanten sind in der Struktur FedmIscReaderConst aufgelistet.
Exceptions	FedmException , wenn ein <i>readerType</i> unbekannt ist
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen int readerType = reader.SetReaderType(41); // Der Leser ist ein ID ISC.LR200 ... private FedmIscReader reader;</pre>

7.1.17. SendProtocol

Funktion	Die zentrale Kommunikationsmethode.
Syntax	(1) int SendProtocol(byte cmd) (2) int SendProtocol(byte cmd, string RequestData, out string ResponseData)
Beschreibung	<p>Der Protokollverkehr wird üblicherweise mit der Methode <i>SendProtocol</i> (1) durchgeführt. Dieser wird nur das Steuerbyte für das gewählte Protokoll übergeben. Alle für den Protokolltransfer benötigten Daten werden aus den Datencontainern bzw. den Datentabellen entnommen. Deshalb muss sichergestellt sein, dass alle benötigten Parameter vorher eingestellt sind.</p> <p>Die überladene Methode (2) kann in den Fällen verwendet werden, wenn für ein Protokoll kein direkter Support in der Bibliothek vorhanden ist (z.B. einige [0xB1] ISO15693 Customer and Proprietary Commds). Der Parameter cmd enthält das Steuerbyte und der Parameter RequestData einen String mit allen gemäß dem Systemhandbuch dokumentierten Datenbytes, die nach dem Steuerbyte folgen. Im Parameter ResponseData sind die Antwortdaten enthalten.</p> <p>Beide Methoden können mit allen Porttreibern verwendet werden.</p>
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder Statusbyte der Antwort (≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	Eine ausführliche Beschreibung zur Verwendung dieser Methode befindet sich in 8. Beispiele für die Verwendung der Methode SendProtocol

7.1.18. SendTransparent

Funktion	Kommunikationsmethode für seltene Fälle
Syntax	int SendTransparent(string SndProtocol, bool CalcChecksum, out string RecProtocol)
Beschreibung	<p>Der Methode <i>SendTransparent</i> erlaubt die Konstruktion von Protokollen mit Protokollrahmen auf der Applikationsebene. Eingehende Kenntnisse zu den Protokollen ist daher zwingend erforderlich. Mit dem Parameter CalcChecksum kann man optional die Checksumme berechnen und dem String SndProtocol anhängen lassen.</p> <p>Die Methode kann mit allen Porttreibern verwendet werden.</p>
Rückgabewert	Ein Integer mit dem Fehlercode(< 0) oder der Länge des Empfangsprotokolls(≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	

7.1.19. TagInventory

Funktion	Kommunikationsmethode für Inventory
Syntax	Dictionary<string, FedmIscTagHandler> TagInventory(bool bAll, byte ucMode, byte ucAntennas)
Beschreibung	<p>Ein Inventory auf der RF-Schnittstelle wird ausgeführt. Für jeden detektierten Transponder wird eine TagHandler-Klasse erzeugt und im Dictionary zurückgegeben. Der Key eines jeden Dictionary-Eintrags ist die UID des Transponders.</p> <p>Die Übergabeparameter steuern das Inventory und entsprechen den Inventory-Parametern im Systemhandbuch zum jeweiligen Leser. Im Standardfall setzt man <i>bAll</i> auf true, <i>ucMode</i> auf 0x00 und <i>ucAntennas</i> auf 1.</p> <p>Die Methode kann mit allen Porttreibern verwendet werden.</p>
Rückgabewert	Dictionary mit erkannten Transpondern bzw. leeres Dictionary, wenn keine Transponder im Feld sind.
Querverweis	
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre> using OBID; using OBID.TagHandler; Dictionary<string, FedmIscTagHandler> TagList; Dictionary<string, FedmIscTagHandler>.ValueCollection listTagHandler; FedmIscTagHandler TagHandler = null; FedmIscTagHandler_ISO14443_4_MIFARE_DESFire DESFire = null; // Inventory command TagList = Reader.TagInventory(true, 0x00, 1); if (TagList.Count > 0) { listTagHandler = TagList.Values; foreach (FedmIscTagHandler tagHandler in listTagHandler) { if (tagHandler != null) { TagHandler = tagHandler; break; // we break as we want to have the first item } } // Try to select tranponder as Mifare DESFire TagHandler = Reader.TagSelect(TagHandler, 9); if (TagHandler is FedmIscTagHandler_ISO14443_4_MIFARE_DESFire) { DESFire = (FedmIscTagHandler_ISO14443_4_MIFARE_DESFire)TagHandler; // do anything with this DESFire-TagHandler } } </pre>

7.1.20. TagSelect

Funktion	Kommunikationsmethode für Select-Command mit einem Transponder
Syntax	FedmlscTagHandler TagSelect(FedmTagHandler tagHandler, uint tagDriver)
Beschreibung	<p>Ein Select-Command auf der RF-Schnittstelle wird ausgeführt. Nach einem erfolgreichen Select wird der aktualisierte TagHandler gleichen Typs oder ein neuer TagHandler eines anderen TagTyps zurückgegeben.</p> <p>Der optionale 2. Parameter <i>tagDriver</i> stellt den zu verwendenden Transpondertreiber im Leser ein. Näher Informationen dazu finden sich im Systemhandbuch zum Leser. Wenn <i>tagDriver</i> nicht zum Einsatz kommt, übergibt man eine 0.</p> <p>Die Methode kann mit allen Porttreibern verwendet werden.</p>
Rückgabewert	TagHandler des selektierten Transponders oder null im Fehlerfall.
Querverweis	
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	s. TagInventory

7.1.21. SendTclApu. SendTclPing, SendTclDeselect

Funktion	Kommunikationsmethode für APDUs
Syntax	int SendTclApu(FedmCprApu apdu) int SendTclPing(FedmCprApu apdu) int SendTclDeselect(FedmCprApu apdu)
Beschreibung	<p>Eine APDU wird zur Ausführung an den Leser gesendet. Die Ausführung erfolgt asynchron. Die Applikation wird per OnNewApuResponse benachrichtigt.</p> <p>Die Methoden SendTclPing und SendTclDeselect werden synchron ausgeführt.</p> <p>Die Methode kann mit allen Porttreibern verwendet werden.</p>
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder OK(0). Für SendTclPing und SendTclDeselect wird mit dem Rückgabewert > 0 das Statusbyte der Leserantwort reflektiert.
Querverweis	
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre> using OBID; // one APDU object as a member in your application FedmCprApu apdu = new FedmCprApu(this); SendApu() { byte[] apduData = new byte[1]; apduData[0] = 0x60; apdu.SetApu(apduData); fedm.SendTclApu(apdu); } ... OnNewApuResponse(int iError) { byte[] rspData; if(iError == 0) rspData = apdu.GetLastResponseData(); } </pre>

7.1.22. SendCommandQueue

Funktion	Kommunikationsmethode für [0xBC] Command Queue
Syntax	int SendCommandQueue(FedmCprCommandQueue queue)
Beschreibung	<p>Eine gefüllte Command Queue wird zur Ausführung an den Leser gesendet. Die Ausführung erfolgt asynchron. Die Applikation wird per OnNewQueueResponse benachrichtigt.</p> <p>Die Methode kann mit allen Porttreibern verwendet werden.</p>
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder OK(0).
Querverweis	
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre> using OBID; // one command queue object as a member in your application FedmCprCommandQueue queue = new FedmCprCommandQueue(this); SendQueue() { string serialNumber; // get serialNumber after Inventory queue.Clear(); // queue must be first cleared // prepare [0xB0][0x25] Select fedm.SetData(FedmIscReaderID.FEDM_ISC_TMP_B0_MODE, (byte)0x01); // addressed fedm.SetData(FedmIscReaderID.FEDM_ISC_TMP_B0_REQ_UID, serialNumber); fedm.SetData(FedmIscReaderID.FEDM_ISC_TMP_B0_CMD, 0x25); fedm.AddCommand(queue, 0xB0); // prepare [0xB0][0x23] Read Multiple Blocks fedm.SetData(FedmIscReaderID.FEDM_ISC_TMP_B0_MODE, (byte)0x02); // selected fedm.SetData(FedmIscReaderID.FEDM_ISC_TMP_B0_CMD, 0x23); fedm.SetData(FedmIscReaderID.FEDM_ISC_TMP_B0_REQ_DBN, (byte)1); fedm.SetData(FedmIscReaderID.FEDM_ISC_TMP_B0_REQ_DB_ADR, (byte)0); fedm.AddCommandToQueue(queue, 0xB0); fedm.SendCommandQueue(queue); } ... OnNewQueueResponse(int iError) { byte[] rspData; if(iError == 0) rspData = queue.GetLastResponseData(); } </pre>

7.1.23. SendSAMCommand

Funktion	Kommunikationsmethode für [0xC0] SAM Command
Syntax	(1) int SendSAMCommand(FedmTaskListener l, int iSlot, byte[] SndProtocol, uint uiTimeout) (2) int SendSAMCommand(int iSlot, uint uiTimeout, byte[] SndProtocol, ref byte[] RecProtocol, ref int RecProtocolLen)
Beschreibung	Ein SAM-Command wird asynchron (1) oder synchron (2) zur Ausführung im Reader gebracht. Die Applikation wird für (1) per OnNewSAMResponse benachrichtigt. Die Methode kann mit allen Porttreibern verwendet werden.
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder OK(0). Für (2) wird mit dem Rückgabewert > 0 das Statusbyte der Leserantwort reflektiert.
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	

7.1.24. FindBaudrate

Funktion	Methode, um die Baudrate und den Protokollrahmen festzustellen.
Syntax	int FindBaudrate()
Beschreibung	Detektiert die Baudrate und den Protokollrahmen eines Lesers und stellt den Port auf die erkannten Parameter ein. Die Methode kann nur in Verbindung mit seriellen Schnittstellen genutzt werden.
Rückgabewert	Fedm.OK, wenn ein Leser erkannt wurde oder ein Fehlercode (< 0)
Exceptions	Keine
Beispiel	<pre>using OBID; ... reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen int state = reader.FindBaudrate(); // Baudrate und Protokollrahmen suchen if (state == 0) { Console.WriteLine(„Der Leser wurde mit eine Baudrate von: „); Console.WriteLine(reader.GetPortPara(„Baud“) + “ erkannt.”); Console.WriteLine(„Der Leser wurde mit einem Protokollrahmen von: „); Console.WriteLine(reader.GetPortPara(„Frame“) + “ erkannt.”); } else { Console.WriteLine(„Es wurde kein Leser erkannt“); } ... private FedmIscReader reader;</pre>

7.1.25. Serialize

Funktion	Methode um die Leserkonfiguration in eine oder aus einer XML-Datei zu serialisieren.
Syntax	void Serialize(bool read, string fileName)
Beschreibung	Diese Methode erlaubt das Laden (falls read == true) einer Leserkonfiguration aus einer XML-Datei in die Datencontainer EEPROM und RAM, oder Schreiben (wenn read == false) der Inhalte der Datencontainer in eine XML-Datei. Die Konfiguration des Lesers bleibt unverändert.
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	

7.1.26. TransferReaderCfgToXmlFile, TransferXmlFileToReaderCfg

Funktion	Methoden für das Schreiben bzw. Lesen der gesamten Leserkonfiguration in eine oder aus einer XML-Datei.
Syntax	int TransferReaderCfgToXmlFile(string fileName) int TransferXmlFileToReaderCfg(string fileName)
Beschreibung	Die erste Methode erlaubt das Auslesen und Schreiben einer Leserkonfiguration in eine XML-Datei. Die Konfiguration des Lesers bleibt unverändert. Die zweite Methode realisiert das Lesen einer Leserkonfiguration aus einer XML-Datei mit anschließendem Schreiben in den Leser. Die Konfiguration des Lesers wird dadurch geändert.
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder Statusbyte der Antwort (≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	

7.1.27. ReaderAuthentication

Funktion	Authentifikationsfunktion.
Syntax	int ReaderAuthentication(uint uiAuthentType, string sAuthentKey)
Anmerkung	Bei dieser Funktion bitte Sorgfalt walten lassen! Der Authentifizierungsschlüssel kann nicht ausgelesen werden.
Beschreibung	Authentifikationsfunktion zur Einleitung einer verschlüsselten Datenübertragung. Zur erstmaligen Anmeldung muss ConnectTCP verwendet werden. Anschließend kann mit dieser Methode immer dann eine neue Session gestartet werden, wenn ein Authentifizierungsfehler vom Leser gemeldet wurde und dieser die aktuelle Session beendet hat. <i>authentType:</i> 0 – AES-128 1 – AES-192 2 – AES-256 <i>authentKey</i> enthält das Passwort und ist eine Zeichenkette mit Hexadezimalen Zeichen (0..9, A..F). Bei AES-128 sind 32 Zeichen, bei AES-192 sind 48 Zeichen und bei AES-256 sind 64 Zeichen zu übergeben.
Rückgabewert	Fedm.OK (0) oder Statusbyte der Antwort (>0)
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .

7.1.28. ReadReaderInfo

Funktion	Methode liest alle wichtigen Informationen vom Leser aus.
Syntax	FedmlscReaderInfo ReadReaderInfo()
Beschreibung	Methode liest alle wichtigen Informationen mit Protokollen [0x66] Get Reader Info vom Leser aus und speichert die Werte in der Hilfsklasse FedmlscReaderInfo. Es wird dringend empfohlen, diese Methode immer nach dem ersten Verbindungsaufbau aufzurufen.
Rückgabewert	Instanz von FedmlscReaderInfo
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .

7.1.29. ReadCompleteConfiguration

Funktion	Liest die gesamte Leserkonfiguration aus.
Syntax	int ReadCompleteConfiguration(bool EEPROM)
Beschreibung	Die Methode liest vom EEPROM (EEPROM=true) bzw. RAM (EEPROM=false) des Lesers alle Konfigurationsblöcke aus und speichert die Werte im internen Datencontainer.
Rückgabewert	Fedm.OK (0) oder Statusbyte der Antwort (>0)
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .

7.1.30. WriteCompleteConfiguration

Funktion	Schreibt die gesamte Leserkonfiguration in den Leser..
Syntax	int WriteCompleteConfiguration(bool EEPROM)
Beschreibung	Die Methode schreibt alle Konfigurationsblöcke vom internen Datencontainer in den Leser. Wenn EEPROM true ist, dann werden das EEPROM und das RAM des Lesers adressiert. Andernfalls nur das RAM.
Rückgabewert	Fedm.OK (0) oder Statusbyte der Antwort (>0)
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .

7.1.31. ResetCompleteConfiguration

Funktion	Reset der gesamten Leserkonfiguration.
Syntax	int ResetCompleteConfiguration(bool EEPROM)
Beschreibung	Die Methode stellt alle Konfigurationswerte im Leser zurück auf die Werkseinstellungen im internen EEPROM und RAM, wenn der Parameter EEPROM=true oder nur im RAM, wenn EEPROM=false.
Rückgabewert	Fedm.OK (0) oder Statusbyte der Antwort (>0)
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .

7.1.32. ApplyConfiguration

Funktion	Aktualisiert die Konfiguration im Leser.
Syntax	int ApplyConfiguration(bool EEPROM)
Beschreibung	<p>Die Methode aktualisiert die Leserkonfiguration im EEPROM und RAM (EEPROM=true) oder nur im RAM (EEPROM=false), nachdem mit SetConfigPara einzelne Parameterwerte im Datencontainer des Leserobjektes modifiziert wurden. Die Methode schreibt nur die Blöcke in den Leser, die modifiziert wurden.</p> <p>Mit dieser Methode realisiert man den schnellsten Transfer der Konfigurationsparameter. Setzt aber zuvor das Lesen der gesamten Konfiguration voraus.</p>
Rückgabewert	Fedm.OK (0) oder Statusbyte der Antwort (>0)
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .

7.1.33. GetLastError

Funktion	Gibt den letzten Fehlercode zurück.
Syntax	int GetLastError()
Beschreibung	Gibt den letzten Fehlercode zurück.
Rückgabewert	Fehlercode (<0)
Exceptions	Keine

7.1.34. GetLastErrorStatus

Funktion	Gibt den letzten Statuscode zurück.
Syntax	Byte GetLastErrorStatus()
Beschreibung	Gibt den Statuscode der letzten Kommunikation zurück.
Rückgabewert	Statuscode (>=0)
Exceptions	Keine

7.1.35. GetErrorText

Funktion	Abfrage eines Fehlertextes
Syntax	string GetErrorText(int errorCode)
Beschreibung	Gibt einen kurzen Fehlertext, der über den Fehlercode referenziert wird, zurück.
Rückgabewert	Ein Fehlertext
Exceptions	Keine

7.1.36. GetStatusText

Funktion	Abfrage eines Statustextes
Syntax	string GetStatusText(byte statusCode)
Beschreibung	Gibt einen kurzen Statustext, der über den Statuscode referenziert wird, zurück.
Rückgabewert	Ein Statustext
Exceptions	Keine

7.1.37. GetData

Funktion	Holt ein Datum aus einem Datencontainer.
Syntax	<code>int GetData(string id, out bool data)</code> <code>int GetData(string id, out byte data)</code> <code>int GetData(string id, out byte[] data)</code> <code>int GetData(string id, out uint data)</code> <code>int GetData(string id, out long data)</code> <code>int GetData(string id, out string data)</code> <code>int GetData(int address, out byte[] data, int length, int memID)</code>
Beschreibung	<p>Methode zum Lesen eines Wertes aus einem Datencontainer.</p> <p>Der Speicherort des Datums wird durch den Parameter <i>id</i> bestimmt.</p> <p>Die zuletzt aufgeführte, nur selten notwendige Variante adressiert das Datum über die Adresse (address) und die Speicher-ID (memID).</p> <p>Tritt ein Fehler auf, gibt die Methode einen Fehlercode (< 0) zurück.</p>
Rückgabewert	Fedm.OK oder einen Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.1.38. SetData

Funktion	Schreibt ein Datum in einem Datencontainer.
Syntax	<code>int SetData(string id, bool data)</code> <code>int SetData(string id, byte data)</code> <code>int SetData(string id, byte[] data)</code> <code>int SetData(string id, uint data)</code> <code>int SetData(string id, long data)</code> <code>int SetData(string id, string data)</code>
Beschreibung	<p>Methode zum Setzen eines Wertes in einem Datencontainer.</p> <p>Der Speicherort des Datums wird durch den Parameter <i>id</i> bestimmt.</p> <p>Tritt ein Fehler auf, gibt die Methode einen Fehlercode (< 0) zurück.</p>
Rückgabewert	Fedm.OK oder einen Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.1.39. GetConfigPara

Funktion	Holt ein Parameterwert aus einem Datencontainer.
Syntax	<code>int GetConfigPara(string id, out bool data, bool EEPROM)</code> <code>int GetConfigPara (string id, out byte data, bool EEPROM)</code> <code>int GetConfigPara (string id, out byte[] data, bool EEPROM)</code> <code>int GetConfigPara (string id, out uint data, bool EEPROM)</code> <code>int GetConfigPara (string id, out long data, bool EEPROM)</code> <code>int GetConfigPara (string id, out string data, bool EEPROM)</code>
Beschreibung	<p>Methode zum Lesen eines Parameterwertes aus einem Datencontainer. Der Parameter <i>id</i> enthält den Parameternamen aus dem Namespace OBID.ReaderConfig. Der letzte Parameter gibt an, ob der Speicherort das EEPROM (true) oder das RAM (false) sein soll.</p> <p>Tritt ein Fehler auf, gibt die Methode einen Fehlercode (< 0) zurück.</p>
Rückgabewert	Fedm.OK oder einen Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.1.40. SetConfigPara

Funktion	Schreibt einen Parameterwert in einen Datencontainer.
Syntax	<code>int SetConfigPara (string id, bool data, bool EEPROM)</code> <code>int SetConfigPara (string id, byte data, bool EEPROM)</code> <code>int SetConfigPara (string id, byte[] data, bool EEPROM)</code> <code>int SetConfigPara (string id, uint data, bool EEPROM)</code> <code>int SetConfigPara (string id, long data, bool EEPROM)</code> <code>int SetConfigPara (string id, string data, bool EEPROM)</code>
Beschreibung	<p>Methode zum Setzen eines Parameterwertes in einem Datencontainer. Der Parameter <i>id</i> enthält den Parameternamen aus dem Namespace OBID.ReaderConfig. Der letzte Parameter gibt an, ob der Speicherort das EEPROM und RAM (true) oder nur das RAM (false) sein soll.</p> <p>Wenn der Parameterwert geändert wurde, gibt die Methode Fedm.MODIFIED (1) zurück. Falls der Parameterwert unverändert bleibt, wird Fedm.OK (0) zurückgegeben.</p> <p>Tritt ein Fehler auf, gibt die Methode einen Fehlercode (< 0) zurück.</p>
Rückgabewert	Fedm.MODIFIED oder Fedm.OK oder einen Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.1.41. TestConfigPara

Funktion	Testet einen Parameternamen für den eingestellten Lesertyp.
Syntax	int TestConfigPara (string id)
Beschreibung	<p>Methode zum Testen eines Parameternamens für die Verwendung mit dem eingestellten Lesertyp. Wird der Parametername von dem eingestellten Lesertyp nicht unterstützt, gibt die Funktion Fedm.ERROR_UNSUPPORTED_NAMESPACE zurück.</p> <p>Der Parameter <i>id</i> enthält den Parameternamen aus dem Namespace OBID.ReaderConfig.</p> <p>Tritt ein Fehler auf, gibt die Methode einen Fehlercode (< 0) zurück.</p>
Rückgabewert	Fedm.OK oder einen Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.1.42. GetByteContainer

Funktion	Gibt ein Byte-Array mit der kompletten Leser-Konfiguration zurück.
Syntax	byte[] GetByteContainer(int arrayID)
Beschreibung	Diese Methode gibt ein Byte-Array mit der kompletten Leser-Konfiguration zurück. Die Konfiguration des Lesers bleibt unverändert. Als <i>arrayID</i> sind die Konstanten RFC_EEDATA_MEM oder RFC_RAMDATA_MEM erlaubt. Diese Konstanten sind in der Struktur Fedm definiert. Das Byte-Array hat eine Länge von 1024 byte.
Rückgabewert	Ein Byte-Array mit der Leser-Konfiguration oder null, wenn ein Fehler auftrat.
Exceptions	FedmException
Beispiel	

7.1.43. SetByteContainer

Funktion	Überschreibt die komplette Leser-Konfiguration in einem Datencontainer.
Syntax	int SetByteContainer(int arrayID, byte[] array)
Beschreibung	Diese Methode überschreibt die komplette Leser-Konfiguration in einem Datencontainer mit dem Inhalt des Byte-Arrays <i>array</i> . Die Konfiguration des Lesers bleibt unverändert. Als <i>arrayID</i> sind die Konstanten RFC_EEDATA_MEM oder RFC_RAMDATA_MEM erlaubt. Diese Konstanten sind in der Struktur Fedm definiert. Das Byte-Array hat eine Länge von 1024 byte.
Rückgabewert	Fehlercode (< 0) oder Fedm.OK
Exceptions	FedmException
Beispiel	

7.1.44. GetTagList

Funktion	Gibt ein Dictionary mit TagHandler-Objekten zurück.
Syntax	Dictionary<string, FedmlscTagHandler>GetTagList()
Beschreibung	Diese Methode gibt die zuvor mit der Methode TagInventory erstellte TagHandler-Liste als Dictionary zurück.
Rückgabewert	Leeres oder gefülltes Dictionary.
Exceptions	FedmException
Beispiel	

7.1.45. GetTagHandler, GetSelectedTagHandler

Funktion	Gibt ein TagHandler-Objekt zurück.
Syntax	FedmlscTagHandler GetTagHandler(string uid) FedmlscTagHandler GetSelectedTagHandler()
Beschreibung	Diese Methoden geben einen zuvor mit der Methode TagInventory oder TagSelect erstellten TagHandler zurück.
Rückgabewert	TagHandler oder null.
Exceptions	FedmException
Beispiel	

7.1.46. CreateNonAddressedTagHandler

Funktion	Erzeugt ein TagHandler-Objekt für non-addressed Kommunikation.
Syntax	FedmlscTagHandler CreateNonAddressedTagHandler(uint TagHandlerType)
Beschreibung	<p>Diese Methode erzeugt einen TagHandler vom Typ <i>TagHandlerType</i> zur anschließenden non-addressed Kommunikation mit einem Transponder.</p> <p>Alle TagHandlerTypen sind in der Klasse FedmlscTagHandler angegeben, wobei nicht jeder für non-addressed Kommunikation spezifiziert ist. Nähere Informationen dazu kann nur das Datenblatt des Transponder-Herstellers geben.</p> <p>Wichtige Anmerkung: es kann immer nur ein TagHandler für non-addressed Kommunikation erzeugt werden. Jeder Aufruf von CreateNonAddressedTagHandler zerstört den vorherigen NonAddressedTagHandler!</p>
Rückgabewert	TagHandler oder null.
Exceptions	FedmException
Beispiel	

7.1.47. GetTableItem

Funktion	Gibt ein Eintrag aus der Tabelle zurück.		
Syntax	GetTableItem(int idx, int tableID)		
Beschreibung	Diese Methode gibt einen Eintrag aus der Tabelle <i>tableID</i> zurück, der über den Paramter <i>idx</i> referenziert wird.		
	Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Tabelleneintrag oder null, wenn ein Fehler auftrat.		
Exceptions	FedmException		
Beispiel			

7.1.48. SetTableItem

Funktion	Setzt einen Eintrag in der Tabelle.		
Syntax	SetTableItem(int idx, int tableID, FedmTableItem item)		
Beschreibung	Diese Methode setzt einen Eintrag in der Tabelle <i>tableID</i> , der über den Paramter <i>idx</i> referenziert wird. Die Daten des Transponders werden nicht verändert.		
	Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung		X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.49. GetTable

Funktion	Gibt eine Kopie der Tabelle zurück.		
Syntax	FedmTableItem[] GetTable(int tableID)		
Beschreibung	Diese Methode gibt eine Kopie der Tabelle <i>tableID</i> zurück. Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Tabellenkopie oder null, wenn ein Fehler auftrat.		
Exceptions	FedmException		
Beispiel			

7.1.50. SetTable

Funktion	Überschreibt eine Tabelle.		
Syntax	int SetTableItem(int idx, FedmTableItem[] item)		
Beschreibung	Diese Methode überschreibt in einer Tabelle im Leserobjekt am Index <i>idx</i> mit <i>item</i> den Inhalt. Die Daten des Transponders werden nicht verändert. Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung		X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.51. GetTableSize

Funktion	Gibt die Größe der Tabelle zurück.		
Syntax	int GetTableSize(int tableID)		
Beschreibung	Diese Methode gibt die Größe der Tabelle <i>tableID</i> zurück. Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Größe der Tabelle.		
Exceptions	FedmException		
Beispiel			

7.1.52. SetTableSize

Funktion	Setzt die Größe der Tabelle.		
Syntax	(1) int SetTableSize(int tableID, int size) (2) int SetTableSize(int tableID, int size, int rxDB_BlockCount, int rxDB_BlockSize, int txDB_BlockCount, int txDB_BlockSize)		
Beschreibung	<p>Diese Methoden dimensionieren die Tabelle <i>tableID</i> auf die Größe <i>size</i>. Die übergebene Größe ist gleichbedeutend mit der maximalen Anzahl von gleichzeitig im RF-Feld detektierbaren Transpondern.</p> <p>Die Größe einer Tabelle wird nicht im Konstruktor eingestellt. Deshalb ist es notwendig, die Tabellengrößen vor der ersten Verwendung auf die Anzahl der zu erwartenden Einträge zu setzen.</p> <p>Die Methode (1) dimensioniert die Tabelle und stellt die Puffer für Transponderdaten auf 256 Datenblöcke mit maximal 32 Bytes ein.</p> <p>Die Methode (2) dimensioniert die Tabelle (nur ISO_TABLE) und erlaubt zusätzlich die Dimensionierung der Puffer für Transponderdaten.</p> <p>Die Methoden können mit folgenden Tabellen genutzt werden:</p>		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X ⁽¹⁾	X ^{(1) und (2)}
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.53. GetTableLength

Funktion	Gibt die Anzahl der Einträge einer Tabelle zurück.		
Syntax	int GetTableLength(int tableID)		
Beschreibung	Diese Methode gibt die Anzahl der Einträge der Tabelle <i>tableID</i> zurück. Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Anzahl der Elemente der Tabelle oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.54. SetTableLength

Funktion	Setzt die Anzahl der Einträge für eine Tabelle.		
Syntax	int SetTableLength(int tableID, int length)		
Beschreibung	Diese Methode setzt die Anzahl der Einträge der Tabelle <i>tableID</i> . Für den Fall, dass man mit den Methoden SetTableData oder SetTableItem Tabelleneinträge in der ISO_TABLE vorbereitet, muß man ev. die Länge der gültigen Tabelleneinträge mit dieser Methode setzen. Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung		X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.55. ResetTable

Funktion	Löscht alle Einträge einer Tabelle.		
Syntax	int ResetTable(int tableID)		
Beschreibung	Diese Methode löscht alle Einträge einer Tabelle. Die mit <i>SetTableSize</i> eingestellt Größe der Tabelle bleibt unverändert. Nach einem Aufruf der Methode <i>ResetTable</i> wird nur der Inhalt der Tabelle gelöscht und die Anzahl gültiger Einträge (TableLength) auf 0 gesetzt.		
	Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.56. GetTableData

Funktion	Überladene Methode zum Lesen eines Tabellenwertes.		
Syntax	int GetTableData(int idx, int tableID, int dataID, out bool data) int GetTableData(int idx, int tableID, int dataID, out byte data) int GetTableData(int idx, int tableID, int dataID, int blockNr, out byte[] data) int GetTableData(int idx, int tableID, int dataID, out uint data) int GetTableData(int idx, int tableID, int dataID, out long data) int GetTableData(int idx, int tableID, int dataID, out string data) int GetTableData(int idx, int tableID, int dataID, int blockNr, out string data)		
Beschreibung	Diese Methode liest einen Wert aus einer Tabelle vom Index <i>idx</i> . Tritt ein Fehler auf wird ein Fehlercode zurückgegeben. Zur genauen Verwendung der Methode siehe 10.5.3. Konstanten für dataID . Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.57. SetTableData

Funktion	Überladene Methode zum Setzen eines Tabellenwertes.		
Syntax	int SetTableData(int idx, int tableID, int dataID, bool data) int SetTableData(int idx, int tableID, int dataID, byte data) int SetTableData(int idx, int tableID, int dataID, int blockNr, byte[] data) int SetTableData(int idx, int tableID, int dataID, uint data) int SetTableData(int idx, int tableID, int dataID, long data) int SetTableData(int idx, int tableID, int dataID, string data) int SetTableData(int idx, int tableID, int dataID, int blockNr, string data)		
Beschreibung	<p>Diese Methode setzt einen Wert in einer Tabelle am Index <i>idx</i>. Tritt ein Fehler auf, wird ein Fehlercode zurückgegeben.</p> <p>Zur genauen Verwendung der Methode siehe 10.5.3. Konstanten für dataID.</p> <p>Die Methode kann mit folgenden Tabellen genutzt werden:</p>		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung		X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.58. VerifyTableDataBlocks

Funktion	Methode zum Verifizieren der gesendeten mit den empfangenen Datenblöcken.		
Syntax	int VerifyTableDataBlocks(int idx, int tableID, int dataID, int bockNr, int blockCnt)		
Beschreibung	<p>Die internen Tabellen vom Typ FedmTableItem haben getrennte Speicher für empfangene und gesendete Transponderdaten. Dies gestattet die Verifikation der gesendeten mit den empfangen Daten. Das Tabellenattribut blockSize, welches die Anzahl der Bytes in jedem Datenblock darstellt, wird intern benutzt. Deshalb muss die BlockSize vorher gesetzt sein (z.B. durch das Lesen eines Datenblocks). Sind die Inhalte der Datenblöcke gleich, dann gibt die Methode Fedm.OK zurück, anderenfalls Fedm.ERROR_VERIFY.</p> <p>Die Methode kann mit folgenden Tabellen genutzt werden:</p>		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung		X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.59. FindTableIndex

Funktion	Überladene Methode um einen Tabellenindex zu erhalten.		
Syntax	int FindTableIndex(int startIdx, int tableID, long dataID, bool data) int FindTableIndex(int startIdx, int tableID, long dataID, byte data) int FindTableIndex(int startIdx, int tableID, long dataID, uint data) int FindTableIndex(int startIdx, int tableID, long dataID, long data) int FindTableIndex(int startIdx, int tableID, long dataID, string data)		
Beschreibung	Die Methode sucht, basierend auf den Kriterien der übergebenen Parameter, einen Tabelleneintrag. Falls ein Tabelleneintrag gefunden wurde, gibt die Methode einen Null-basierenden Index zurück, anderenfalls Fedm.ERROR_NO_TABLE_DATA. Zur genauen Verwendung der Methode siehe 10.5.3. Konstanten für dataID . Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Index oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel			

7.1.60. AddEventListener

Funktion	Überladene Methode zur Anmeldung einer Ereignisbehandlungsroutine.	
Syntax	void AddEventListener (FelscListener l, int evt) void AddEventListener(FeUsbListener l, int evt)	
Beschreibung	<p>Diese Methode wird benutzt, um eine Ereignisbehandlungsroutine zu implementieren. Für jedes in der untenstehenden Tabelle aufgelistete Ereignis muß AddEventListener separat aufgerufen werden. Dabei kann für jeden Listener (das Empfängerobjekt) jeder Event nur einmal angemeldet werden.</p> <p>Die in den folgenden Tabellen aufgelisteten Ereignis-IDs sind möglich:</p> <p>FelscListener:</p>	
	Event ID⁶	Erläuterung
	TRANSCIVE_STRING_EVENT	Ein String mit Datum und Uhrzeit wird je für das Sende- und Empfangsprotokoll an den Listener gesendet.
	SEND_STRING_EVENT	Ein String mit Datum und Uhrzeit wird für das Sendeprotokoll an den Listener gesendet.
	RECEIVE_STRING_EVENT	Ein String mit Datum und Uhrzeit wird für das Empfangsprotokoll an den Listener gesendet.
	SCANNER_PRT_EVENT	<p>Ein Byte-Array mit den Empfangsdaten wird an den Listener gesendet.</p> <p>Die Anmeldung dieses Events startet intern einen kontinuierlichen Empfangsprozess für Daten, die ein Leser im Scanmodus ausgibt.</p>
	SEND_PRT_EVENT	Ein Byte-Array mit den Sendedaten wird an den Listener gesendet.
	RECEIVE_PRT_EVENT	Ein Byte-Array mit den Empfangsdaten wird an den Listener gesendet

⁶ s. Struktur FelscListenerConst

	FeUsbListener:	
	Event ID⁷	Erläuterung
	FEUSB_CONNECT_EVENT	Die neue Verbindung eines USB-Lesers wird an den Listener gemeldet.
	FEUSB_DISCONNECT_EVENT	Das Entfernen eines USB-Lesers wird an den Listener gemeldet.
Exceptions	FedmException	
Querverweis	7.6.FelscListener , 7.7.FeUsbListener	
Beispiel	<pre> using System; using OBID; class MyClass : FeIscListener { // Überschriebene Methode aus FeIscListener // werden bei Ereignissen aufgerufen public void OnSendProtocol(FedmIscReader reader, string sendProtocol) { Console.WriteLine(sendProtocol); } void OnSendProtocol(FedmIscReader reader, byte[] sendProtocol) { // an dieser Stelle kann eine Behandlungsmethode für das Byte-Array // implementiert werden } public void OnReceiveProtocol(FedmIscReader reader, string receiveProtocol) { Console.WriteLine(receiveProtocol); } void OnReceiveProtocol(FedmIscReader reader, byte[] receiveProtocol) { // an dieser Stelle kann eine Behandlungsmethode für das Byte-Array // implementiert werden } MyClass() { try { int readerType; this.reader = new FedmIscReader(); reader.ConnectCOMM(1); // Verbindung zum COM-Leser herstellen reader.FindBaudRate(); // Einstellen der Baudrate // Anmelden der Ereignisbehandlungsroutinen reader.AddEventListener(this, FeIscListenerConst.SEND_STRING_EVENT); reader.AddEventListener(this, FeIscListenerConst.RECEIVE_STRING_EVENT); reader.SendProtocol(0x65); // Ermitteln der SoftwareVersion // Achtung! Die Protokolle werden angezeigt! readerType = reader.GetReaderType(); // Leser-Typ holen Console.WriteLine("Lesertyp: "); } catch { } } } </pre>	

⁷ s. Struktur FeUsbListenerConst

```
        Console.WriteLine(readerType);
        Console.ReadLine();
        // Abmelden der Ereignisbehandlungsroutinen
        reader.RemoveEventListener(this, FeIscListenerConst.SEND_STRING_EVENT);
        reader.RemoveEventListener(this, FeIscListenerConst.RECEIVE_STRING_EVENT);
    }
    catch (FedmException e)
    {
        Console.WriteLine(e.Message);
    }
    catch (FePortDriverException e)
    {
        Console.WriteLine(e.Message);
    }
    catch (FeReaderDriverException e)
    {
        Console.WriteLine(e.Message);
    }
}

/// <summary>
/// Der Haupteinstiegspunkt für die Anwendung.
/// </summary>
[STAThread]
static void Main(string[] args)
{
    new MyClass();
}

public FedmIscReader reader;
}
```

7.1.61. RemoveEventListener

Funktion	Meldet eine zuvor installierte Ereignisbehandlungsmethode ab.
Syntax	void RemoveEventListener(FelscListener l, int evt) void RemoveEventListener(FeUsbListener l, int evt)
Beschreibung	Diese Methode wird benutzt, um eine Ereignisbehandlungsroutine abzumelden. Weitere Informationen findet man in AddEventListener .
Exceptions	FedmException
Beispiel	Ein ausführliches Beispiel befindet sich bei der Beschreibung der Methode AddEventListener .

7.1.62. StartAsyncTask

Funktion	Methode startet einen asynchronen Task.		
Syntax	int StartAsyncTask(int TaskID, FedmTaskListener listener, FedmTaskOption opt)		
Beschreibung	<p>Mit dieser Funktion wird ein asynchroner Task gestartet. Ein asynchroner Task ist ein interner Thread, der z.B. ein Inventory-Kommando an den Leser sendet und für eine Zeit Timeout auf die Antwort wartet. Die Signalisierung der Antwortdaten bzw. der Abbruchbedingung an die Applikation erfolgt mit dem Aufruf eines Delegates.</p> <p>Das Taskverhalten wird im Parameter <i>iTaskID</i> spezifiziert. Drei Aufgaben sind z.Zt. definiert:</p>		
	Task	TaskID	Anmerkungen
	Einmaliger Inventory	ID_FIRST_NEW_TAG	<p>Ein Task kann nur gestartet werden, wenn folgende Option in der Firmware des Lesers integriert ist: Das Leserprotokoll [0xB0][0x01] Inventory muß in seinem Mode-Byte ein optionales NOTIFY-Flag unterstützen.</p> <p>Nach dem Empfang der Leserprotokolls innerhalb der vorgegebenen Zeit, ruft der Task den Delegaten OnNewTag auf und beendet sich anschließend selbständig. Kommt es zu einer Zeitüberschreitung, wird der Delegat aufgerufen und der Status 0x01 (kein Transponder im Lesefeld) übermittelt und der Task beendet. Im Fehlerfall wird der Task immer sofort beendet und der Delegate übergibt den Fehlercode.</p> <p>Unterstützt werden die drei Schnittstellen seriell, USB und TCP/IP, wobei die Schnittstellen vor dem Starten des Tasks geöffnet sein müssen. Der selbständige Verbindungsaufbau per TCP/IP vom Leser oder einem geeigneten Konverter zur Übermittlung der Daten ist nicht möglich.</p>
	Repetierender Inventory	ID EVERY_NEW_TAG	<p>Es gelten die Bedingungen des einmaligen Inventory mit folgendem Unterschied:</p> <p>Der repetierende Inventory definiert eine zyklische Aufgabe, die nur durch CancelAsyncTask beendet werden kann. Ein Zyklus entspricht einem einmaligen Inventory und endet in einer Warteschleife, bis der nächste Zyklus von der Applikation durch den Aufruf von TriggerAsyncTask erneut angestoßen wird. Durch die Applikations-seitige Triggerung wird sichergestellt, dass eine Applikation Zeit für die Entgegennahme und Bearbeitung der Inventarisierungsdaten erhält.</p>

	Empfang von Notifications	ID_NOTIFICATION	<p>Ein Task sollte nur gestartet werden, wenn der Notification-Mode in der Firmware des Lesers integriert und aktiviert ist. Unterstützt wird nur die Kommunikation über TCP/IP. Mögliche Verbindungsoptionen sind (s. Systemhandbuch zum Leser):</p> <ul style="list-style-type: none"> - Temporärer Verbindungsaufbau durch den Leser für die Dauer der Datenübertragung - Dauerhafter Verbindungsaufbau durch den Leser (in Planung) - Dauerhafter Verbindungsaufbau durch den Host (in Planung) <p>Der Task definiert eine endlose Aufgabe, die nur durch CancelAsyncTask beendet werden kann, bzw. im Fehlerfall während der Initialisierungsphase, nach dem Aufruf des Delegaten OnNewNotification, sofort beendet wird.</p> <p>Der Task wartet auf den Empfang der Buffered-Read-Mode Daten und ruft anschließend den Delegat auf. Nach der Rückkehr des Delegats können sofort wieder Daten vom Leser entgegengenommen werden.</p> <p>Bei Übertragungsfehlern wird der Delegate mit dem Fehlercode aufgerufen und anschließend die Empfangsprozedur fortgesetzt. Wenn die Keep-Alive Option aktiviert ist (voreingestellt), dann wird eine Unterbrechung der Netzwerkverbindung erkannt, der empfangende Socket geschlossen und anschließend neu initialisiert. Dadurch ist sichergestellt, dass der RFID-Leser nach der Wiederherstellung der Verbindung erneut eine Verbindung aufbauen kann.</p> <p>Der Zeitpunkt eines Verbindungsabbaus durch Keep-Alive wird wie folgt berechnet:</p> $\text{IdleTime} + \text{RepeatCount} * \text{IntervalTime}$ <p>Die voreingestellten Werte sind:</p> <ul style="list-style-type: none"> - IdleTime ist 500ms - RepeatCount ist 5 (Window XP) oder 10 (Vista or 7) - IntervalTime ist 500ms <p>Hinweis: je nach Einstellung des Lesers können in kürzesten Abständen sehr viele Daten vom Leser verschickt werden. Ohne Handshake-Mechanismen (s. Systemhandbuch zum Leser) können u.U. Daten verloren gehen, wenn der Host für die Quantität der Notifications nicht geeignet ist.</p>
	Alle für den Task und für der Delegaten relevanten Daten sind in der Struktur FedmTaskOption zusammengefasst.		
Rückgabewert	Index oder ein Fehlercode (< 0)		
Exceptions	FedmException		
Beispiel (VB.NET)	<pre>taskOpt = New FedmTaskOption() taskOpt.IPPort = "192.168.1.1" taskOpt.NotifyWithAck = 0 taskOpt.Timeout = 1000 ` 1000 ms reader.StartAsyncTask(FedmTaskOption.ID_NOTIFICATION, Me, taskOpt)</pre>		

7.1.63. CancelAsyncTask

Funktion	Ein Inventarisierungs- bzw. Notificationtask wird beendet
Syntax	int CancelAsyncTask()
Beschreibung	<p>Mit dieser Funktion wird ein asynchroner Task sofort beendet. Allerdings wird nur der interne Thread beendet. Der Task im Leser kann nicht unterbrochen werden.</p> <p>Den einmaligen Inventory (gestartet mit TaskID = ID_FIRST_NEW_TAG) sollte man normalerweise nicht mit dieser Funktion beenden. Den repetierenden Inventory (gestartet mit TaskID = ID_EVERY_NEW_TAG) sollte man dann mit dieser Funktion beenden, wenn der Delegat zurückkehrt und der interne Thread auf den nächsten Trigger wartet. So ist sichergestellt, dass der Task im Leser beendet ist und er wieder Leserprotokolle bearbeiten kann.</p> <p>Notificationtasks müssen immer mit dieser Funktion beendet werden.</p> <p>Zur Vermeidung von Deadlocks wird der Task nicht beendet, wenn der Ausführungspfad des Tasks innerhalb des Dellegaten liegt. In diesem Fall kehrt die Funktion sofort mit dem Rückgabewert FEISC_ERR_TASK_BUSY (-4084) zurück und die Applikation muß CancelAsyncTask solange aufrufen, bis der Rückgabewert nicht mehr -4084 ist. Applikations-seitig muss sichergestellt werden, dass die Callback-Funktion immer zuverlässig zurückkehrt.</p>
Exceptions	FedmException
Beispiel	

7.1.64. TriggerAsyncTask

Funktion	Der nächste Zyklus im Inventarisierungstask wird angestoßen
Syntax	void TriggerAsyncTask()
Beschreibung	<p>Mit dieser Funktion wird der nächste Inventory-Zyklus im asynchronen Task angestoßen. Der asynchrone Task muß zuvor mit der TaskID=ID_EVERY_NEW_TAG gestartet sein.</p> <p>Aufgerufen wird diese Funktion immer dann, nachdem der Delegat verlassen wurde. Ohne diesen Aufruf bleibt ein Task mit repetierender Funktion in einer Warteschleife hängen.</p>
Exceptions	FedmException
Beispiel	

7.2. FedmIsoTableItem, FedmBrmTableItem

7.2.1. Datenfelder in FedmISOTableItem

In der folgenden Tabelle sind alle (public und private) Datenfelder dokumentiert. Wenn kein Vermerk angebracht wurde, hat man direkten Zugriff auf das Datenfeld.

Eingehende Informationen zu jedem Datenfeld finden sich im jeweiligen Systemhandbuch des OBID®-Readers.

Datenfeld	Relevanz	Beschreibung
Kategorie: Transponder spezifische Informationen		
uid[]	alle	Seriennummer des Transponders
transponderType	alle	Typ des Transponders (s. Anhang Systemhandbuch zum OBID®-Reader)
DsflID	ISO 15693	Data Storage Format Identifier
trInfo	ISO 14443-A	Transponder Information
optInfo	ISO 14443-A	Optionale Information
protolInfo	ISO 14443-B	Protocol Info Byte
chipID	STM SR176 and SRlxx (ISO 14443 classic-pro reader)	siehe Systemhandbuch zum OBID®-Reader
IDDT	EPC Class1 Gen2, ISO 18000-3M3	Identifier Data Type (siehe Systemhandbuch zum OBID®-Reader)
class1Gen2PC[]	EPC Class1 Gen2, ISO 18000-3M3	siehe Systemhandbuch zum OBID®-Reader
AFI	ISO 15693	Application Family Identifier
memSize	ISO 15693	Memory Size
ICRef	ISO 15693	IC Referenz
FSCI	ISO 14443-4	siehe Systemhandbuch zum OBID®-Reader
FWI	ISO 14443-4	siehe Systemhandbuch zum OBID®-Reader
DSI	ISO 14443-4	siehe Systemhandbuch zum OBID®-Reader
DRI	ISO 14443-4	siehe Systemhandbuch zum OBID®-Reader
NAD	ISO 14443-4	siehe Systemhandbuch zum OBID®-Reader
CID	ISO 14443-4	siehe Systemhandbuch zum OBID®-Reader
productCode	ISO 14443-4 ASK CTx	siehe Systemhandbuch zum OBID®-Reader
fabCode	ISO 14443-4 ASK CTx	siehe Systemhandbuch zum OBID®-Reader
appCode	ISO 14443-4 ASK CTx	siehe Systemhandbuch zum OBID®-Reader
embedderCode	ISO 14443-4 ASK CTx	siehe Systemhandbuch zum OBID®-Reader
verlog	Innovatron (ISO 14443B')	siehe Systemhandbuch zum OBID®-Reader
config	Innovatron (ISO 14443B')	siehe Systemhandbuch zum OBID®-Reader
atr[]	Innovatron (ISO 14443B')	siehe Systemhandbuch zum OBID®-Reader

Kategorie: Daten des Transponders			
blockSize	alle	Blockgröße (Anzahl Bytes je Datenblock)	
securityStatus[]	ISO 15693	Status-Information zu Datenblöcken	Zugriff mit GetData() und SetData()
rxPubData[]	alle	Datenblöcke nach Lesevorgang	
txPubData[]	alle	Datenblöcke für den nächsten Schreibvorgang	
rxDB_EpcBank[]	EPC Class1 Gen2, ISO 18000-3M3	Datenblöcke nach Lesevorgang	
txDB_EpcBank[]	EPC Class1 Gen2, ISO 18000-3M3	Datenblöcke für den nächsten Schreibvorgang	
rxDB_TidBank[]	EPC Class1 Gen2, ISO 18000-3M3	Datenblöcke nach Lesevorgang	
txDB_TidBank[]	EPC Class1 Gen2, ISO 18000-3M3	Datenblöcke für den nächsten Schreibvorgang	
rxDB_ResBank[]	EPC Class1 Gen2, ISO 18000-3M3	Datenblöcke nach Lesevorgang	
rxDB_ResBank[]	EPC Class1 Gen2, ISO 18000-3M3	Datenblöcke für den nächsten Schreibvorgang	
Kategorie: Informationen zu Antennen und Feldstärken			
antCount	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Anzahl der Antennen, von denen der Tag gelesen wurde	Zugriff mit GetRSSI()
antNumber[]	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Array mit Antennennummern	
antStatus[]	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Array mit Statusinformationen	
antRSSI[]	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Array mit RSSI-Messwerten	
Kategorie: Datenintegrität			
isUid	alle	Zeigt an, dass uid und transponderType gültig sind	
isAFI	ISO 15693	Zeigt an, dass AFI gültig ist	
isSysInfo	ISO 15693	Zeigt an, dass ISO 15693 Systeminfos gültig sind	
isISO14443Info	ISO 14443	Zeigt an, dass ISO 14443 Systeminfos gültig sind	
isRSSI	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Zeigt an, dass Antenneninfos und RSSI-Werte gültig sind	
isBlockSizeSet	alle	Zeigt an, dass blockSize gültig ist	
isSelected	ISO 15693, ISO 14443	Zeigt an, dass dieser Tabelleneintrag den selektierten Transponder repräsentiert	

7.2.2. Datenfelder in FedmBRMTableItem

In der folgenden Tabelle sind alle (public und private) Datenfelder dokumentiert. Wenn kein Vermerk angebracht wurde, hat man direkten Zugriff auf das Datenfeld.

Eingehende Informationen zu jedem Datenfeld finden sich im jeweiligen Systemhandbuch des OBID®-Readers.

Datenfeld	Relevanz	Beschreibung	
Kategorie: Transponder spezifische Informationen			
uid[]	alle	Seriennummer des Transponders	
transponderType	alle	Typ des Transponders (s. Anhang Systemhandbuch zum OBID®-Reader)	
trInfo	ISO 14443-A	Transponder Information	
IDDT	EPC Class1 Gen2, ISO 18000-3M3	Identifier Data Type (siehe Systemhandbuch zum OBID®-Reader)	
class1Gen2PC[]	EPC Class1 Gen2, ISO 18000-3M3	siehe Systemhandbuch zum OBID®-Reader	
AFI	ISO 15693	Application Family Identifier	
DsfID	ISO 15693	Data Storage Format Identifier	
Kategorie: Daten des Transponders			
blockSize	alle	Blockgröße (Anzahl Bytes je Datenblock)	
dbAddress	alle	Start-Adresse	
blockCount	alle	Anzahl Datenblöcke	
rxPubData[]	alle	Datenblöcke nach Lesevorgang	Zugriff mit GetData()
Kategorie: Informationen zu Antennen und Feldstärken			
antennaNumber	alle	Feld mit Einträgen, von welchen Antennen der Tag gelesen wurde. Alternativ zu antNumber[] und antRSSI[]	
antCount	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Anzahl der Antennen, von denen der Tag gelesen wurde	Alternativ zu antennaNumber Zugriff mit GetRSSI()
antNumber[]	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Array mit Antennennummern	
antRSSI[]	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Array mit RSSI-Messwerten	
Kategorie: sonstige Informationen			
input	alle	Feld mit Einträgen zu gesetzten Eingängen	
status	alle	Status Information zum Lesevorgang	
readerTime	alle	Datum/Uhrzeit im Typ FelscReaderTime	
macAddr	alle	MAC-Adresse	
direction	Gate People Counter	Richtungsinformation	

Kategorie: Datenintegrität		
isUid	alle	Zeigt an, dass uid und transponderType gültig sind
isAntNr		Zeigt an, dass Antenneninformationen in antennaNumber gültig sind
isRSSI	ISO 15693, EPC Class1 Gen2, ISO 18000-3M3	Zeigt an, dass RSSI-Werte gelesen wurden
isDB	alle	Zeigt an, dass Datenblöcke gelesen wurden
isTimer	alle	Zeigt an, dass die Zeit gültig ist
isDate	alle	Zeigt an, dass das Datum gültig ist
isInput	alle	Zeigt an, dass Informationen zu gesetzten Eingängen gültig sind
isMacAddr	alle	Zeigt an, dass die MAC-Adresse gültig ist
isDirection	Gate People Counter	Zeigt an, dass die Richtungsinformation gültig ist

7.2.3. GetData

Funktion	Überladene Methode zum Lesen eines Tabellenwertes.		
Syntax	int GetData(int dataID, out bool data) int GetData(int dataID, out byte data) int GetData(int dataID, int blockNr, out byte[] data) int GetData(int dataID, out uint data) int GetData(int dataID, out long data) int GetData(int dataID, out string data) int GetData(int dataID, int blockNr, out string data)		
Beschreibung	Diese Methode liest einen Wert aus der Tabelle. Tritt ein Fehler auf wird ein Fehlercode zurückgegeben. Zur genauen Verwendung der Methode siehe 10.5.3. Konstanten für dataID .		
	Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	Keine		
Beispiel			

7.2.4. SetData

Funktion	Überladene Methode zum Setzen eines Tabellenwertes.		
Syntax	int SetData(int dataID, bool data) int SetData(int dataID, byte data) int SetData(int dataID, int blockNr, byte[] data) int SetData(int dataID, uint data) int SetData(int dataID, long data) int SetData(int dataID, string data) int SetData(int dataID, int blockNr, string data)		
Beschreibung	Diese Methode setzt einen Wert in der Tabelle. Tritt ein Fehler auf, wird ein Fehlercode zurückgegeben. Zur genauen Verwendung der Methode siehe 10.5.3. Konstanten für dataID . Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung		X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)		
Exceptions	Keine		
Beispiel			

7.2.5. GetRSSI

Funktion	Methode gibt RSSI-Werte zurück.		
Syntax	Dictionary<byte, FedmlscRssItem> GetRSSI()		
Beschreibung	Diese Methode gibt aus der Tabelle die RSSI-Werte in Form eines Dictionary zurück. Der Schlüssel ist die Antennennummer, der Wert ein Objekt mit dem RSSI-Wert zu dieser Antenne. Die Methode kann mit folgenden Tabellen genutzt werden:		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung	X	X
Rückgabewert	Dictionary mit RSSI-Werten oder im Fehlerfallein leeres Dictionary		
Exceptions	Keine		
Beispiel			

7.2.6. VerifyDataBlocks

Funktion	Methode zur Verifizierung von gesendeten mit empfangen Datenblöcken.		
Syntax	int VerifyDataBlocks(int blockNr, int blockCnt)		
Beschreibung	<p>Die Tabellen vom Typ FedmTableItem haben getrennte Speicher für empfangene und gesendete Transponderdaten. Dies gestattet die Verifikation der gesendeten mit den empfangen Daten. Das Tabellenattribut blockSize, welches die Anzahl der Bytes in jedem Datenblock darstellt, wird intern benutzt. Deshalb muss die BlockSize vorher gesetzt sein (z.B. durch das Lesen eines Datenblocks). Sind die Inhalte der Datenblöcke gleich, dann gibt die Methode Fedm.OK zurück, andernfalls Fedm.ERROR_VERIFY.</p> <p>Die Methode kann mit folgenden Tabellen genutzt werden:</p>		
	tableID	BRM_TABLE	ISO_TABLE
	Unterstützung		X
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0).		
Exceptions	Keine		
Beispiel			

7.2.7. IsDataValid

Funktion	Methode zur Prüfung der Gültigkeit von Tabellendaten.
Syntax	bool IsDataValid(int dataID)
Beschreibung	Überschriebene Methode, um ein Datenelement auf seine Gültigkeit zu überprüfen.
Rückgabewert	True, wenn der Datenwert gültig ist.
Exceptions	Keine
Beispiel	

7.2.8. GetIdentifier

Funktion	Methode zur Identifizierung einer Tabelle.
Syntax	string GetIdentifier()
Beschreibung	Die Methode gibt ein String-Objekt mit der Kennung der Tabelle zurück.
Rückgabewert	„ISO“ oder „BRM“
Exceptions	Keine
Beispiel	

7.3. FedmlscFunctionUnit

7.3.1. FedmlscFunctionUnit

Funktion	Erzeugt eine neue Instanz der Klasse FedmlscFunctionUnit (Konstruktor).
Syntax	FedmlscFunctionUnit(FedmlscReader, int FUType)
Beschreibung	<p>Es wird ein FunctionUnit-Objekt erzeugt. Nur mit einem FunctionUnit-Objekt können die Protokollfunktionen ausgeführt werden.</p> <p>Dem Konstruktor muß die Instanz eines Leserobjektes und dem Typ der Funktionseinheit übergeben werden.</p>
Rückgabewert	Wenn ein FunctionUnit-Objekt fehlerfrei erstellt werden konnte, wird die neue Instanz der Klasse FedmlscFunctionUnit zurückgeliefert.
Exceptions	Im Fehlerfall wirft die Methode die Ausnahme FedmException .
Beispiel	<pre>using OBID; ... try { fu = new FedmlscFunctionUnit(reader, FedmlscFunctionUnit.FU_TYPE_MUX); } catch (FedmException e) { Console.WriteLine("Ausnahme beim Erzeugen eines neuen Objektes: " + e.Message); } private FedmlscReader reader; private FedmlscFunctionUnit fu;</pre>

7.3.2. GetFUType

Funktion	Gibt Typnummer der Funktionseinheit zurück.
Syntax	int GetFUType()
Beschreibung	Gibt die Typnummer der Funktionseinheit zurück. Die Typnummern sind im Anhang unter 10.5.1. Allgemeine Konstanten aufgelistet.
Rückgabewert	Typnummer.
Exceptions	Keine
Beispiel	

7.3.3. GetLastError

Funktion	Gibt den letzten Fehlercode zurück.
Syntax	int GetLastError()
Beschreibung	Gibt den letzten Fehlercode zurück.
Rückgabewert	Fehlercode (<0)
Exceptions	Keine

7.3.4. GetErrorText

Funktion	Abfrage eines Fehlertextes
Syntax	string GetErrorText(int errorCode)
Beschreibung	Gibt einen kurzen Fehlertext, der über den Fehlercode referenziert wird, zurück.
Rückgabewert	Ein Fehlertext
Exceptions	Keine

7.3.5. SendProtocol

Funktion	Die zentrale Kommunikationsmethode.
Syntax	int SendProtocol(byte cmd)
Beschreibung	Der Protokollverkehr wird mit der Methode <i>SendProtocol</i> durchgeführt. Dieser wird nur das Steuerbyte für das gewählte Protokoll übergeben. Alle für den Protokolltransfer benötigten Daten werden aus dem Datencontainer TmpData entnommen. Deshalb muss sichergestellt sein, dass alle benötigten Parameter vorher eingestellt sind.
Rückgabewert	Ein Integer mit dem Fehler-(< 0) oder Statuscode(≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	Eine ausführliche Beschreibung zur Verwendung dieser Methode befindet sich in 8.3. Kommandos für eine Funktionseinheit

7.3.6. GetData

Funktion	Holt ein Datum aus einem Datencontainer.
Syntax	<code>int GetData(string id, out bool data)</code> <code>int GetData(string id, out byte data)</code> <code>int GetData(string id, out uint data)</code> <code>int GetData(string id, out long data)</code> <code>int GetData(string id, out string data)</code>
Beschreibung	Methode zum Lesen eines Wertes aus einem Datencontainer. Der Speicherort des Datums wird durch den Parameter <i>id</i> bestimmt. Tritt ein Fehler auf, gibt die Methode einen Fehlercode (< 0) zurück.
Rückgabewert	Fedm.OK oder einen Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.3.7. SetData

Funktion	Schreibt ein Datum in einem Datencontainer.
Syntax	<code>int SetData(string id, bool data)</code> <code>int SetData(string id, byte data)</code> <code>int SetData(string id, uint data)</code> <code>int SetData(string id, long data)</code> <code>int SetData(string id, string data)</code>
Beschreibung	Methode zum Setzen eines Wertes in einem Datencontainer. Der Speicherort des Datums wird durch den Parameter <i>id</i> bestimmt. Tritt ein Fehler auf, gibt die Methode einen Fehlercode (< 0) zurück.
Rückgabewert	Fedm.OK oder einen Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.3.8. AddChild

Funktion	Fügt eine Funktionseinheit als Child-Objekt hinzu.
Syntax	int AddChild(int outNr, FedmIsFunctionUnit child)
Beschreibung	<p>Fügt die Funktionseinheit <i>child</i> als Child-Objekt der internen Liste nachfolgender Funktionseinheiten am Ausgang <i>outNr</i> hinzu. Ein zuvor an <i>outNr</i> gespeichertes Child-Objekt wird überschrieben.</p> <p>Hinweis: Die Verwaltung von Child-Objekten bleibt der Applikation vorbehalten.</p>
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.3.9. DeleteChild

Funktion	Entfernt eine Funktionseinheit als Child-Objekt aus der Liste.
Syntax	int DeleteChild(int outNr)
Beschreibung	<p>Entfernt die Funktionseinheit als Child-Objekt am Ausgang <i>outNr</i> aus der internen Liste nachfolgender Funktionseinheiten.</p> <p>Hinweis: Die Verwaltung von Child-Objekten bleibt der Applikation vorbehalten.</p>
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.3.10. GetChild

Funktion	Gibt eine Funktionseinheit als Child-Objekt zurück.
Syntax	int GetChild(int outNr)
Beschreibung	<p>Gibt die Funktionseinheit als Child-Objekt am Ausgang <i>outNr</i> aus der internen Liste nachfolgender Funktionseinheiten zurück.</p> <p>Hinweis: Die Verwaltung von Child-Objekten bleibt der Applikation vorbehalten.</p>
Rückgabewert	Fedm.OK oder ein Fehlercode (< 0)
Exceptions	FedmException
Beispiel	

7.4. FedmlscPeopleCounter

7.4.1. GetCounterValues

Funktion	Abfrage der Zählerstände										
Syntax	long[] GetCounterValues()										
Beschreibung	Diese Methode ermittelt die aktuellen Zählerstände und gibt diese als Array zurück.										
Rückgabewert	<p>Array mit den vier Zählerständen.</p> <table border="1"> <thead> <tr> <th>Index</th><th>Zähler</th></tr> </thead> <tbody> <tr> <td>0</td><td>Zähler 1 von Radar Detektor 1</td></tr> <tr> <td>1</td><td>Zähler 2 von Radar Detektor 1</td></tr> <tr> <td>2</td><td>Zähler 1 von Radar Detektor 2</td></tr> <tr> <td>3</td><td>Zähler 2 von Radar Detektor 2</td></tr> </tbody> </table>	Index	Zähler	0	Zähler 1 von Radar Detektor 1	1	Zähler 2 von Radar Detektor 1	2	Zähler 1 von Radar Detektor 2	3	Zähler 2 von Radar Detektor 2
Index	Zähler										
0	Zähler 1 von Radar Detektor 1										
1	Zähler 2 von Radar Detektor 1										
2	Zähler 1 von Radar Detektor 2										
3	Zähler 2 von Radar Detektor 2										
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .										
Beispiel	<pre> FedmlscReader reader; ... // Verbindungsaufbau mit internem ReadReaderInfo() reader.ConnectTCP("192.168.10.10", 10001); ... long[] values = null; FedmlscPeopleCounter pc = null; // Liste mit People Counter holen Dictionary<byte, FedmlscPeopleCounter> mapPC = reader.GetPeopleCounterMap(); // People Counter mit Busadresse 1 holen bool ret = mapPC.TryGetValue(1, out pc); if (ret) { try { // Zählerstände holen values = pc.GetCounterValues(); } catch (Exception ex) { // einen Fehler abfangen } } </pre>										

7.4.2. SetCounterValues

Funktion	Setzen der Zählerstände
Syntax	<pre>int SetCounterValues(long radar1Counter1, long radar1Counter2, long radar2Counter1, long radar2Counter2)</pre>
Beschreibung	Diese Methode setzt alle Zählerstände auf die übergebenen Werte.
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder Statusbyte der Antwort (≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>FedmlscReader reader; ... // Verbindungsaufbau mit internem ReadReaderInfo() reader.ConnectTCP("192.168.10.10", 10001); ... int status = 0; FedmlscPeopleCounter pc = null; // Liste mit People Counter holen Dictionary<byte, FedmlscPeopleCounter> mapPC = reader.GetPeopleCounterMap(); // People Counter mit Busadresse 1 holen bool ret = mapPC.TryGetValue(1, out pc); if (ret) { try { // alle Zählerstände auf 0 setzen status = pc.SetCounterValues(0, 0, 0, 0); } catch (Exception ex) { // einen Fehler abfangen } }</pre>

7.4.3. SetOutputsOn

Funktion	Setzen der digitalen Ausgänge
Syntax	<pre>int SetOutputsOn(int holdTime1, int holdTime2, int holdTime3)</pre>
Beschreibung	<p>Diese Methode setzt bis zu drei digitale Ausgänge für die Zeit holdTimeX. Wenn holdTimeX auf 0 gesetzt wird, dann wird der betreffende Ausgang nicht aktiviert. Wenn holdTimeX auf 65535 gesetzt wird, dann wird der betreffende Ausgang dauerhaft aktiviert.</p> <p>Der Wertebereich von holdTimeX ist 0...65535, wobei die Aktivierungszeit in Stufen von 100ms einstellbar ist.</p>
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder Statusbyte der Antwort (≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>FedmlscReader reader; ... // Verbindungsaufbau mit internem ReadReaderInfo() reader.ConnectTCP("192.168.10.10", 10001); ... int status =0; FedmlscPeopleCounter pc = null; // Liste mit People Counter holen Dictionary<byte, FedmlscPeopleCounter> mapPC = reader.GetPeopleCounterMap(); // People Counter mit Busadresse 1 holen bool ret = mapPC.TryGetValue(1, out pc); if (ret) { try { // Ausgänge 1 für 1 Sekunde und 3 für 2 Sekunden setzen status = pc.SetOutputsOn(10, 0, 20); } catch (Exception ex) { // einen Fehler abfangen } }</pre>

7.4.4. SetOutputsOff

Funktion	Zurücksetzen der digitalen Ausgänge
Syntax	<code>int SetOutputsOff(bool off1, bool off2, bool off3)</code>
Beschreibung	Diese Methode setzt bis zu drei digitale Ausgänge zurück.
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder Statusbyte der Antwort (≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>FedmlscReader reader; ... // Verbindungsaufbau mit internem ReadReaderInfo() reader.ConnectTCP("192.168.10.10", 10001); ... int status =0; FedmlscPeopleCounter pc = null; // Liste mit People Counter holen Dictionary<byte, FedmlscPeopleCounter> mapPC = reader.GetPeopleCounterMap(); // People Counter mit Busadresse 1 holen bool ret = mapPC.TryGetValue(1, out pc); if (ret) { try { // Ausgänge 1 und 3 werden zurückgesetzt status = pc.SetOutputsOff(true, false, true); } catch (Exception ex) { // einen Fehler abfangen } }</pre>

7.4.5. SetOutputsFlashing

Funktion	Setzen der digitalen Ausgänge
Syntax	<code>int SetOutputsFlashing(int frequencyOut1, int holdTime1, int frequencyOut2,int holdTime2, int frequencyOut3,int holdTime3)</code>
Beschreibung	<p>Diese Methode setzt bis zu drei digitale Ausgänge für die Zeit holdTimeX in einen mit der Frequenz frequencyOutX blinkenden Zustand. Wenn holdTimeX auf 0 gesetzt wird, dann wird der betreffende Ausgang nicht aktiviert. Wenn holdTimeX auf 65535 gesetzt wird, dann wird der betreffende Ausgang dauerhaft aktiviert.</p> <p>Der Wertebereich von holdTimeX ist 0...65535, wobei die Aktivierungszeit in Stufen von 100ms einstellbar ist.</p> <p>Der Wertebereich für frequencyOutX ist 1, 2, 4, 8 und entspricht der Frequenz in Hz.</p>
Rückgabewert	Einen Integer mit dem Fehlercode (< 0) oder Statusbyte der Antwort (≥ 0).
Exceptions	Im Fehlerfall wirft die Methode eine der Ausnahmen FedmException , FePortDriverException , FeReaderDriverException .
Beispiel	<pre>FedmlscReader reader; ... // Verbindungsaufbau mit internem ReadReaderInfo() reader.ConnectTCP("192.168.10.10", 10001); ... int status =0; FedmlscPeopleCounter pc = null; // Liste mit People Counter holen Dictionary<byte, FedmlscPeopleCounter> mapPC = reader.GetPeopleCounterMap(); // People Counter mit Busadresse 1 holen bool ret = mapPC.TryGetValue(1, out pc); if (ret) { try { // Ausgang 1 mit 1Hz für 1 Sekunde und // Ausgang 3 mit 2Hz für 2 Sekunden setzen status = pc.SetOutputsFlashing(1, 10, 0, 0, 2, 20); } catch (Exception ex) { // einen Fehler abfangen } }</pre>

7.5. FeHexConvert

In der Klasse **HeHexConvert** sind nützliche Funktionen zur Konvertierung von Daten und Bearbeitung der Zugriffskonstanten zusammengefasst. Alle Funktionen sind statisch, können also aus jedem Kontext heraus verwendet werden.

7.5.1. ByteArrayToHexStringWithSpaces

Funktion	Konvertierungsfunktion
Syntax	static System.String ByteArrayToHexStringWithSpaces(byte[] in)
Beschreibung	Die Funktion konvertiert ein Byte-Array in eine Zeichenkette. Dabei wird jedes Byte in zwei Hex-Zeichen (a-f, A-F, 0-9) umgewandelt und als Separator ein Leerzeichen zwischen je zwei Zeichen eingebaut.
Rückgabewert	Zeichenkette
Exceptions	Keine
Beispiel	Übergabe : 0x11, 0x22, 0xF0, 0x5E Rückgabe : "11 22 F0 5E"

7.5.2. ByteArrayToHexString

Funktion	Überladene Konvertierungsfunktion
Syntax	static System.String ByteArrayToHexString(byte[] in) static string ByteArrayToHexString(byte[] in, int start, int count)
Beschreibung	Die erste Funktion konvertiert ein Byte-Array in eine Zeichenkette. Dabei wird jedes Byte in zwei Hex-Zeichen (a-f, A-F, 0-9). Die zweite Funktion nimmt für die Konvertierung nur die <i>count</i> Byte ab dem Index <i>start</i> . Die erste Funktion ist die Umkehrfunktion zu HexStringToByteArray .
Rückgabewert	Zeichenkette
Exceptions	Keine
Beispiel	Übergabe : 0x11, 0x22, 0xF0, 0x5E Rückgabe : "1122F05E"

7.5.3. ByteToHexString

Funktion	Konvertierungsfunktion
Syntax	static string ByteToHexString(byte in)
Beschreibung	Die Funktion konvertiert das Byte in eine Zeichenkette mit zwei Hex-Zeichen (a-f, A-F, 0-9). Umkehrfunktion zu HexStringToByte .
Rückgabewert	Zeichenkette
Exceptions	Keine
Beispiel	Übergabe : 0x11 Rückgabe : "11"

7.5.4. IntegerToHexString

Funktion	Konvertierungsfunktion
Syntax	static string IntegerToHexString(int in)
Beschreibung	Die Funktion konvertiert den int-Wert in eine Zeichenkette, wobei jedes Byte des Integer in je zwei Hex-Zeichen (a-f, A-F, 0-9) konvertiert wird. Umkehrfunktion zu HexStringToInteger .
Rückgabewert	Zeichenkette
Exceptions	Keine
Beispiel	Übergabe : 287502430 Rückgabe : "1122F05E"

7.5.5. LongToHexString

Funktion	Konvertierungsfunktion
Syntax	static string LongToHexString(long in)
Beschreibung	Die Funktion konvertiert den long-Wert in eine Zeichenkette, wobei jedes Byte des Long in je zwei Hex-Zeichen (a-f, A-F, 0-9) konvertiert wird. Umkehrfunktion zu HexStringToLong .
Rückgabewert	Zeichenkette
Exceptions	Keine
Beispiel	Übergabe : 1234813534658031710 Rückgabe : "1122F05E1122F05E"

7.5.6. HexStringToByte

Funktion	Konvertierungsfunktion
Syntax	static byte HexStringToByte(string str)
Beschreibung	Die Funktion konvertiert eine Zeichenkette, bestehend aus den Hex-Zeichen 0-9, a-f, A-F, in einen int-Wert. Die Zeichenkette muß eine gerade Anzahl Zeichen haben und darf aus maximal 8 Zeichen bestehen. Umkehrfunktion zu ByteToHexString .
Rückgabewert	Byte-Wert
Exceptions	System.ArgumentException
Beispiel	Übergabe : "11" Rückgabe : 0x11

7.5.7. HexStringToByteArray

Funktion	Überladene Konvertierungsfunktion
Syntax	static byte[] HexStringToByteArray(String str) static byte[] HexStringToByteArray(string str, int start, int count)
Beschreibung	Die erste Funktion konvertiert einen String in ein Byte-Array. Dabei werden jeweils zwei Hex-Zeichen (a-f, A-F, 0-9) in ein Byte umgewandelt. Die zweite Funktion nimmt für die Konvertierung nur die <i>count</i> Zeichen ab <i>start</i> . Die erste Funktion ist die Umkehrfunktion zu HexStringToByteArray .
Rückgabewert	Byte-Array
Exceptions	System.ArgumentException
Beispiel	Übergabe : "1122F05E" Rückgabe : 0x11, 0x22, 0xF0, 0x5E

7.5.8. HexStringToInteger

Funktion	Konvertierungsfunktion
Syntax	static int HexStringToInteger(string str)
Beschreibung	Die Funktion konvertiert eine Zeichenkette, bestehend aus den Hex-Zeichen 0-9, a-f, A-F, in einen int-Wert. Die Zeichenkette muß eine gerade Anzahl Zeichen haben und darf aus maximal 8 Zeichen bestehen. Umkehrfunktion zu IntegerToHexString .
Rückgabewert	int-Wert
Exceptions	System.ArgumentException
Beispiel	Übergabe : "1122F05E" Rückgabe : 287502430

7.5.9. HexStringToLong

Funktion	Konvertierungsfunktion
Syntax	static long HexStringToLong(string str)
Beschreibung	Die Funktion konvertiert eine Zeichenkette, bestehend aus den Hex-Zeichen 0-9, a-f, A-F, in einen long-Wert. Die Zeichenkette muß eine gerade Anzahl Zeichen haben und darf aus maximal 16 Zeichen bestehen. Umkehrfunktion zu LongToHexString .
Rückgabewert	long-Wert
Exceptions	System.ArgumentException
Beispiel	Übergabe : "1122F05E1122F05E" Rückgabe : 1234813534658031710

7.5.10. isHexString

Funktion	Testfunktion für String
Syntax	static bool isHexString(string str)
Beschreibung	Die Funktion überprüft, ob die übergebene Zeichenkette ausschließlich aus den Hex-Zeichen a-f, A-F und 0-9 besteht.
Rückgabewert	True, wenn nur die oben genannten Zeichen enthalten sind, andernfalls False.
Exceptions	Keine
Beispiel	

7.5.11. GetMemIDofID

Funktion	Funktion für Zugriffskonstante
Syntax	static int GetMemIDofID(string ID)
Beschreibung	Die Funktion gibt die Memory-ID aus der Zugriffskonstanten <i>ID</i> zurück.
Rückgabewert	int
Exceptions	Keine
Querverweis	5.6.2. Zugriffskonstanten
Beispiel	Übergabe : FEDM_ISC_EE_COM_BUSADR ("03 03 01 00 01 00 00") Rückgabe : 3

7.5.12. GetByteCntOfID

Funktion	Funktion für Zugriffskonstante
Syntax	static int GetByteCntOfID(string ID)
Beschreibung	Die Funktion gibt die Anzahl der Bytes, aus dem ein Parameter besteht, aus der Zugriffskonstanten <i>ID</i> zurück.
Rückgabewert	int
Exceptions	Keine
Querverweis	5.6.2. Zugriffskonstanten
Beispiel	Übergabe : FEDM_ISC_EE_COM_BUSADR ("03 03 01 00 01 00 00") Rückgabe : 1

7.5.13. GetAdrOfID

Funktion	Funktion für Zugriffskonstante
Syntax	static int GetAdrOfID(string ID)
Beschreibung	Die Funktion gibt die Anfangsadresse (Index im Datencontainer) eines Parameters aus der Zugriffskonstanten <i>ID</i> zurück.
Rückgabewert	int
Exceptions	Keine
Querverweis	5.6.2. Zugriffskonstanten
Beispiel	Übergabe : FEDM_ISC_EE_COM_BUSADR ("03 03 01 00 01 00 00") Rückgabe : 1

7.6.FelscListener

7.6.1. OnSendProtocol

Funktion	Überladene Eventmethoden für Ereignisse						
Syntax	void OnSendProtocol(FedmlscReader reader, byte[] sendProtocol); void OnSendProtocol(FedmlscReader reader, string sendProtocol);						
Beschreibung	<p>Die Methoden werden aufgerufen, wenn man mit der Methode AddEventHandler der Leserklasse FedmlscReader Ereignisse angemeldet hat, die im Zusammenhang mit einem Sendeprotokoll stehen. Auf folgende Ereignisse wird eine der überladenen Methoden aufgerufen:</p> <table> <tr> <td>TRANSCIVE_STRING_EVENT</td><td>OnSendProtocol(..., string sendProtocol)</td></tr> <tr> <td>SEND_STRING_EVENT</td><td>OnSendProtocol(..., string sendProtocol)</td></tr> <tr> <td>SEND_PRT_EVENT</td><td>OnSendProtocol(..., byte[] sendProtocol)</td></tr> </table>	TRANSCIVE_STRING_EVENT	OnSendProtocol(..., string sendProtocol)	SEND_STRING_EVENT	OnSendProtocol(..., string sendProtocol)	SEND_PRT_EVENT	OnSendProtocol(..., byte[] sendProtocol)
TRANSCIVE_STRING_EVENT	OnSendProtocol(..., string sendProtocol)						
SEND_STRING_EVENT	OnSendProtocol(..., string sendProtocol)						
SEND_PRT_EVENT	OnSendProtocol(..., byte[] sendProtocol)						
Rückgabewert	Kein						
Exceptions	Keine						
Beispiel	s. Beispiel in 7.1.60. AddEventListener						

7.6.2. OnReceiveProtocol

Funktion	Überladene Eventmethoden für Ereignisse						
Syntax	void OnReceiveProtocol(FedmlscReader reader, byte[] receiveProtocol); void OnReceiveProtocol(FedmlscReader reader, string receiveProtocol);						
Beschreibung	<p>Die Methoden werden aufgerufen, wenn man mit der Methode AddEventHandler der Leserklasse FedmlscReader Ereignisse angemeldet hat, die im Zusammenhang mit einem Empfangsprotokoll stehen. Auf folgende Ereignisse wird eine der überladenen Methoden aufgerufen:</p> <table> <tr> <td>TRANSCIVE_STRING_EVENT</td><td>OnReceiveProtocol(..., string receiveProtocol)</td></tr> <tr> <td>RECEIVE_STRING_EVENT</td><td>OnReceiveProtocol(..., string receiveProtocol)</td></tr> <tr> <td>RECEIVE_PRT_EVENT</td><td>OnReceiveProtocol(..., byte[] receiveProtocol)</td></tr> </table>	TRANSCIVE_STRING_EVENT	OnReceiveProtocol(..., string receiveProtocol)	RECEIVE_STRING_EVENT	OnReceiveProtocol(..., string receiveProtocol)	RECEIVE_PRT_EVENT	OnReceiveProtocol(..., byte[] receiveProtocol)
TRANSCIVE_STRING_EVENT	OnReceiveProtocol(..., string receiveProtocol)						
RECEIVE_STRING_EVENT	OnReceiveProtocol(..., string receiveProtocol)						
RECEIVE_PRT_EVENT	OnReceiveProtocol(..., byte[] receiveProtocol)						
Rückgabewert	Kein						
Exceptions	Keine						
Beispiel	s. Beispiel in 7.1.60. AddEventListener						

7.7.FeUsbListener

7.7.1. OnConnectReader

Funktion	Eventmethode für Ereignis
Syntax	void OnConnectReader(int deviceHandle, long deviceID);
Beschreibung	<p>Die Methode wird aufgerufen, wenn man mit der Methode AddEventHandler der Leserklasse FedmlscReader das Ereignis FEUSB_CONNECT_EVENT angemeldet hat und ein USB-Leser an den PC angeschlossen wird.</p> <p>Dieses Verfahren ist sehr hilfreich, um einer Applikation die Verfügbarkeit des USB-Lesers anzuzeigen.</p> <p>Im ersten Übergabeparameter wird ein Device-Handle übergeben. Dieser ist nur von Bedeutung, wenn man mit der Klasse FeUsb mehrere USB-Leser verwaltet.</p> <p>Im zweiten Übergabeparameter befindet sich die Seriennummer des USB-Lesers.</p>
Rückgabewert	Kein
Exceptions	Keine
Beispiel	s. äquivalentes Beispiel in 7.1.60. AddEventListener

7.7.2. OnDisconnectReader

Funktion	Eventmethode für Ereignis
Syntax	void OnDisconnectReader(int deviceHandle, long deviceID);
Beschreibung	<p>Die Methode wird aufgerufen, wenn man mit der Methode AddEventHandler der Leserklasse FedmlscReader das Ereignis FEUSB_DISCONNECT_EVENT angemeldet hat und ein USB-Leser vom PC abgezogen wird.</p> <p>Dieses Verfahren ist sehr hilfreich, um einer Applikation die Verfügbarkeit des USB-Lesers anzuzeigen.</p> <p>Im ersten Übergabeparameter wird ein Device-Handle übergeben. Dieser ist nur von Bedeutung, wenn man mit der Klasse FeUsb mehrere USB-Leser verwaltet.</p> <p>Im zweiten Übergabeparameter befindet sich die Seriennummer des USB-Lesers.</p>
Rückgabewert	kein
Exceptions	Keine
Beispiel	s. äquivalentes Beispiel in 7.1.60. AddEventListener

7.8.FedmTaskListener

7.8.1. OnNewTag

Funktion	Methode signalisiert die Verfügbarkeit neuer Transponderdaten in der ISO-Tabelle
Syntax	void OnNewTag(int error);
Beschreibung	<p>Die Methode wird aufgerufen, wenn ein Leser einen Inventarisierungsvorgang abgeschlossen und die Daten in die ISO-Tabelle eingefügt wurden.</p> <p>Der Zugriff auf die Transponderdaten in der Tabelle erfolgen analog dem Zugriff nach einem [0xB0][0x01] Inventory Befehl. Mehr Informationen unter 8.2.2. Beispiele für die Verwendung der ISO Tabelle mit [0xB0] Protokollen.</p> <p>Der Empfang von Notifications muß zuvor mit der Methode StartAsyncTask eingerichtet werden.</p>
Querverweis	7.1.62. StartAsyncTask
Exceptions	Keine

7.8.2. OnNewNotification

Funktion	Methode signalisiert die Verfügbarkeit neuer Transponderdaten in der BRM-Tabelle
Syntax	void OnNewNotification(int error, string ip, uint portNr);
Beschreibung	<p>Die Methode wird aufgerufen, wenn ein Leser eine Notification gesendet und die Daten in die BRM-Tabelle eingefügt wurden.</p> <p>Der Zugriff auf die Transponderdaten in der Tabelle erfolgt analog dem Zugriff nach einem [0x22] Read Buffer Befehl. Mehr Informationen unter 8.2.4. Kommandos für Buffered Read Mode.</p> <p>Der Parameter <i>ip</i> ist die IP-Adresse des Lesers und <i>portNr</i> die lokale Portnummer, über den die Notification empfangen wurde.</p> <p>Der Empfang von Notifications muß zuvor mit der Methode StartAsyncTask eingerichtet werden.</p>
Querverweis	7.1.62. StartAsyncTask
Exceptions	Keine

7.8.3. OnNewReaderDiagnostic

Funktion	Methode signalisiert den Empfang neuer Reader Diagnostic Daten
Syntax	void OnNewReaderDiagnostic(int error, string ip, uint portNr);
Beschreibung	<p>Die Methode wird aufgerufen, wenn ein Leser Reader Diagnostic Daten gesendet hat und die Daten im Datencontainer TempData abgelegt wurden.</p> <p>Der Zugriff auf die Diagnostic-Daten erfolgt mit:</p> <pre>byte[] data; GetData(FEDM_ISC_TMP_DIAG_DATA, out data);</pre> <p>Zur Interpretation der Diagnose-Daten ist das Systemhandbuch des Lesers erforderlich. Die Diagnose-Daten werden mit dem Modebyte 0x01 generiert.</p> <p>Der Parameter <i>ip</i> ist die IP-Adresse des Lesers und <i>portNr</i> die lokale Portnummer, über den die Notification empfangen wurde.</p> <p>Der Empfang von Notifications muß zuvor mit der Methode StartAsyncTask eingerichtet werden.</p>
Querverweis	7.1.62. StartAsyncTask
Exceptions	Keine

7.8.4. OnNewPeopleCounterEvent

Funktion	Methode signalisiert den Empfang neuer Zählerstände eines People Counters
Syntax	void OnNewPeopleCounterEvent(uint counter1, uint counter2, uint counter3, uint counter4, string ip, uint portNr, uint busAddress);
Beschreibung	<p>Die Methode wird aufgerufen, wenn ein People Counter einen internen Zähler inkrementiert hat und der Leser dies gesendet hat.</p> <p><i>counter 1..4</i> sind die Zählerstände.</p> <p>Der Parameter <i>ip</i> ist die IP-Adresse des Lesers und <i>portNr</i> die lokale Portnummer, über den die Notification empfangen wurde.</p> <p>Der Parameter <i>busAddress</i> ist die Busadresse des People Counter.</p> <p>Der Empfang von People Counter Events muß zuvor mit der Methode StartAsyncTask eingerichtet werden.</p>
Querverweis	7.1.62. StartAsyncTask
Exceptions	Keine

7.8.5. OnNewSAMResponse

Funktion	Methode signalisiert die Verfügbarkeit der SAM-Daten
Syntax	void OnNewSAMResponse(int error, byte[] responseData);
Beschreibung	Die Methode wird aufgerufen, wenn der adressierte SAM im Leser die angeforderten Daten zurückgibt (error = 0).
Querverweis	7.1.23. SendSAMCommand
Exceptions	Keine

7.8.6. OnNewApduResponse

Funktion	Methode signalisiert die Verfügbarkeit der APDU-Daten
Syntax	void OnNewApduResponse(int error);
Beschreibung	Die Methode wird aufgerufen, wenn der adressierte Transponder die angeforderten APDU-Daten zurückgibt (error = 0).
Querverweis	7.1.21. SendTclApdu , SendTclPing , SendTclDeselect
Exceptions	Keine

7.8.7. OnNewQueueResponse

Funktion	Methode signalisiert die Verfügbarkeit der Antwort-Daten einer Command Queue Operation
Syntax	void OnNewQueueResponse(int error);
Beschreibung	Die Methode wird aufgerufen, wenn die angeforderten Antwort-Daten vom Leser gesendet wurden (error = 0).
Querverweis	7.1.22. SendCommandQueue
Exceptions	Keine

7.9. FedmException

Funktion	Ausnahme-Klasse für FedmIsCReader.
Beschreibung	Es wird eine neue Instanz dieser Klasse im Fehlerfall geworfen.

7.10. FedmPortDriverException

Funktion	Ausnahme-Klasse für FedmIsCReader.
Beschreibung	Es wird eine neue Instanz dieser Klasse im Fehlerfall geworfen.

7.11. FedmReaderDriverException

Funktion	Ausnahme-Klasse für FedmIsCReader.
Beschreibung	Es wird eine neue Instanz dieser Klasse im Fehlerfall geworfen.

8. Beispiele für die Verwendung der Methode `SendProtocol`

Die Methode *SendProtocol* der Leserklasse bzw. FU-Klasse ist von zentraler Bedeutung für den Protokolltransfer. Aus diesem Grund wird für jedes Steuerbyte⁸ ein Beispiel aufgeführt, das verdeutlichen soll, welche Daten mit welchen Zugriffskonstanten vor jedem Protokolltransfer in Datencontainer zu speichern sind und welche Daten nach dem Protokolltransfer zur Verfügung stehen. Mit manchen Protokollen können unterschiedliche Daten transferiert werden. In einem solchen Fall wird nur ein exemplarisches Beispiel aufgeführt.

Alle Zugriffskonstanten sind in der Struktur **FedmIscReaderID** bzw. **FedmIscFunctionUnitID** enthalten und sollten eingehend zusammen mit der im Systemhandbuch zum Leser bzw. Funktionseinheit angeführten Erklärung der Protokolldaten studiert werden.

Auf die Auswertung der Rückgabewerte der Methoden, bzw. dem Abfangen der Ausnahmen wird hier aus Gründen der Übersichtlichkeit verzichtet. Sie sollte allerdings in Applikationen immer erfolgen. Dies gilt insbesondere für die Methode `SendProtocol(..)`

In den unten aufgeführten Beispielen wird stillschweigen davon ausgegangen, dass die Leserklasse **FedmIscReader** und die Strukturen **FedmIscReaderID** und **FedmIscReaderConst** sowie **FedmIscFunctionUnit** und **FedmIscFunctionUnitID** eingebunden sind:

```
using OBID.FedmIscReader;  
using OBID.FedmIscReaderID;  
using OBID.FedmIscReaderConst;  
  
using OBID.FedmIscFunctionUnit;  
using OBID.FedmIscFunctionUnitID;
```

Als Leserobjekt sei `reader` definiert:

```
FedmIscReader reader = new FedmIscReader;
```

Als FU-Objekt sei `fu` definiert:

```
FedmIscFunctionUnit fu = new FedmIscFunctionUnit;
```

⁸ nicht alle Steuerbytes werden von jedem Leser unterstützt. Weitere Informationen zu den unterstützten Steuerbytes entnimmt man dem Systemhandbuch des jeweiligen Lesers

8.1. Grundkommandos für den Leser

[Steuerbyte] Protokoll	Beispiel ⁹
[0x18] Destroy	<pre> byte mode = 0; // Mode (z.Zt. immer 0) byte epcLen = 0; // Anzahl Bytes in EPC string epc; // EPC string pw; // Passwort // hole die Daten z.B. aus Eingabefeldern // ermittle die Länge des EPC epcLen = epc.Length reader.SetData(FEDM_ISC_TMP_EPC_DESTROY_MODE, (byte)0); reader.SetData(FEDM_ISC_TMP_EPC_DESTROY_LEN, epcLen); reader.SetData(FEDM_ISC_TMP_EPC_DESTROY_PASSWORD, pw); reader.SetData(FEDM_ISC_TMP_EPC_DESTROY_EPC, epc); reader.SendProtocol(0x18); </pre>
[0x1A] Halt	<pre> reader.SendProtocol(0x1A); </pre>
[0x1B] Reset QUIET Bit	<pre> reader.SendProtocol(0x1B); </pre>
[0x1C] EAS	<pre> reader.SendProtocol(0x1C); </pre>
[0x21]Read Buffer (nur ID ISC.LR200 und ID ISC.LR2000)	<pre> byte dataSets = 1; // Anzahl angeforderter Datensätze byte trData = 0; // Datensatzstruktur byte recSets = 0; // Anzahl Datensätze in Protokoll reader.SetData(FEDM_ISCLR_TMP_BRM_SETS, dataSets); reader.SendProtocol(0x21); // lese Datenblöcke von Transponder mit Buffered Read Mode reader.GetData(FEDM_ISCLR_TMP_BRM_TRDATA, out trData); reader.GetData(FEDM_ISCLR_TMP_BRM_RECSETS, out recSets); // Alle weiteren Transponderdaten sind in BRMTable enthalten. Beispiel für Datenzugriffe in // 8.2.4. Kommandos für Buffered Read Mode </pre>
[0x22]Read Buffer (für alle Leser mit Buffered Read Mode, außer ID LR200)	<pre> uint dataSets = 1; // Anzahl angeforderter Datensätze byte trData = 0; // Datensatzstruktur uint recSets = 0; // Anzahl Datensätze in Protokoll reader.SetData(OBID.ReaderCommand._0x22.Req.DATA_SETS, dataSets); reader.SendProtocol(0x22); // lese Datenblöcke von Transponder mit Buffered Read Mode reader.GetData(OBID.ReaderCommand._0x22.Rsp.TR_DATA1, out trData); reader.GetData(OBID.ReaderCommand._0x22.Rsp.DATA_SETS, out recSets); // Alle weiteren Transponderdaten sind in BRMTable enthalten. Beispiel für Datenzugriffe in // 8.2.4. Kommandos für Buffered Read Mode </pre>

⁹ alle Beispiele sind in C# ausgeführt

[Steuerbyte] Protokoll	Beispiel ⁹
[0x31] Read Data Buffer Info	<pre> uint tabSize = 0; // Größe des Datenpuffers uint tabStart = 0; // Startadresse für ersten Datensatz uint tabLen = 0; // Anzahl Datensätze in Datenpuffer reader.SendProtocol(0x31); reader.GetData(OBID.ReaderCommand._0x31.Rsp.TAB_SIZE, out tabSize); reader.GetData(OBID.ReaderCommand._0x31.Rsp.TAB_START, out tabStart); reader.GetData(OBID.ReaderCommand._0x31.Rsp.TAB_LEN, out tabLen); </pre>
[0x32] Clear Data Buffer	<pre> reader.SendProtocol(0x32); </pre>
[0x33] Initialize Buffer	<pre> reader.SendProtocol(0x33); </pre>
[0x34] Force Notify Trigger	<pre> reader.SendProtocol(0x34); </pre>
[0x52] Baud Rate Detection	<pre> reader.SendProtocol(0x52); </pre>
[0x55] Start Flash Loader	<pre> reader.SendProtocol(0x55); </pre>
[0x63] CPU Reset	<pre> reader.SendProtocol(0x63); </pre>
[0x64] System Reset	<pre> byte cMode = 0; // LRU1000 RF-Controller (1 für LRU1000 AC-Controller) reader.SetData(OBID.ReaderCommand._0x64.Req.MODE, cMode); reader.SendProtocol(0x64); </pre>
[0x65] Get Software Version	<pre> string softVer; // Software-Version als String reader.SendProtocol(0x65); reader.GetData(FEDM_ISC_TMP_SOFTVER, out softVer); </pre>
[0x66] Get Reader Info	<pre> string sInfo; // Reader Info als String reader.SetData(OBID.ReaderCommand._0x66.Req.MODE, (uint)0); // identisch mit [0x65] //reader.SetData(ReaderCommand._0x66.Req.MODE, (uint)1); // AC-Controller reader.SendProtocol(0x66); reader.GetData(OBID.ReaderCommand._0x66.Rsp.READER_INFO, sInfo); </pre>
[0x69] RF Reset	<pre> reader.SendProtocol(0x69); </pre>
[0x6A] RF ON/OFF	<pre> byte RF = 1; // RFON reader.SetData(OBID.ReaderCommand._0x6A.Req.RF_OUTPUT, RF); reader.SendProtocol(0x6A); </pre>
[0x6C] Set Noise Level	<pre> uint NLMin = 500; // minimaler Noise Level uint NLAvg = 1000; // mittlerer Noise Level uint NLMax = 1500; // maximaler Noise Level reader.SetData(FedmlscReaderID.FEDM_ISC_TMP_NOISE_LEVEL_MIN, NLMin); reader.SetData(FedmlscReaderID.FEDM_ISC_TMP_NOISE_LEVEL_AVG, NLAvg); reader.SetData(FedmlscReaderID.FEDM_ISC_TMP_NOISE_LEVEL_MAX, NLMax); reader.SendProtocol(0x6C); </pre>

[Steuerbyte] Protokoll	Beispiel ⁹
[0x6D] Get Noise Level	<pre> uint NLMin = 0; // minimaler Noise Level uint NLAvg = 0; // mittlerer Noise Level uint NLMax = 0; // maximaler Noise Level reader.SendProtocol(0x6D); reader.GetData(OBID.ReaderCommand._0x6D.Rsp.NL_MIN, out NLMin); reader.GetData(OBID.ReaderCommand._0x6D.Rsp.NL_AVG, out NLAvg); reader.GetData(OBID.ReaderCommand._0x6D.Rsp.NL_MAX, out NLMax); </pre>
[0x6E] Reader Diagnostic	<pre> byte diagMode = 1; // Diagnose-Modus string sData; // Diagnose-Daten als String reader.SetData(OBID.ReaderCommand._0x6E.Req.MODE, diagMode); reader.SendProtocol(0x6E); reader.GetData(OBID.ReaderCommand._0x6E.Rsp.DIAGNOSTIC_DATA, sData); </pre>
[0x6F] Base Antenna Tuning	<pre> reader.SendProtocol(0x6F); // der Long-Range-Reader wechselt in einen Spezialmodus // nur mit einem Reset kann dieser Modus verlassen werden </pre>
[0x71] Set Output	<pre> // Beispiel 1 aus Systemhandbuch ID ISC.M01 reader.SetData(OBID.ReaderCommand._0x71.Req.OUTPUT_STATE, (uint)0); // OS-Bytes // zurücksetzen reader.SetData(OBID.ReaderCommand._0x71.Req.OutputState.OUT1, (byte)0x01); // Ausgang 1 // aktivieren reader.SetData(OBID.ReaderCommand._0x71.Req.OutputState.LED_GREEN, (byte)0x10); // LED // grün aus reader.SetData(OBID.ReaderCommand._0x71.Req.OutputState.LED_RED, (byte)0x01); // LED // rot an reader.SetData(OBID.ReaderCommand._0x71.Req.OutputState.BEEPER, (byte)0x11); // Piepser // alternierend an reader.SetData(OBID.ReaderCommand._0x71.Req.OUTPUT_STATE_FLASH, (uint)0); // OSF- // Bytes zurück. reader.SetData(OBID.ReaderCommand._0x71.Req.OutputStateFlash.BEEPER, (byte)0x01); // // Piepser mit 4Hz reader.SetData(OBID.ReaderCommand._0x71.Req.OS_TIME, (uint)5); // 500ms Aktivzeit Piepser // und LEDs reader.SetData(OBID.ReaderCommand._0x71.Req.OUT_TIME, (uint)3); // Ausgang 1 300ms aktiv reader.SendProtocol(0x71); </pre>

[Steuerbyte] Protokoll	Beispiel ⁹
[0x72] Set Output	<pre>// Beispiel aus dem Systemhandbuch ID ISC.LRU1000 reader.SetData(OBID.ReaderCommand._0x72.Req.MODE, (byte)0x00); // set mode to 0 reader.SetData(OBID.ReaderCommand._0x72.Req.OUT_N, (byte)0x03); // activate 3 outputs reader.SetData(OBID.ReaderCommand._0x72.Req.No1.OUT_NUMBER, (byte)0x01); // output 1 reader.SetData(OBID.ReaderCommand._0x72.Req.No1.OUT_TYPE, (byte)0x00); // type: general output reader.SetData(OBID.ReaderCommand._0x72.Req.No1.State.MODE, (byte)0x03); // alternating reader.SetData(OBID.ReaderCommand._0x72.Req.No1.State.FREQUENCY, (byte)0x01); // 4 Hz reader.SetData(OBID.ReaderCommand._0x72.Req.No1.OUT_TIME, (uint)5); // 500 ms reader.SetData(OBID.ReaderCommand._0x72.Req.No2.OUT_NUMBER, (byte)0x01); // relais 1 reader.SetData(OBID.ReaderCommand._0x72.Req.No2.OUT_TYPE, (byte)0x04); // type: relais reader.SetData(OBID.ReaderCommand._0x72.Req.No2.State.MODE, (byte)0x02); // switching off reader.SetData(OBID.ReaderCommand._0x72.Req.No2.State.FREQUENCY, (byte)0x00); // unchanged reader.SetData(OBID.ReaderCommand._0x72.Req.No2.OUT_TIME, (uint)2); // 200 ms reader.SetData(OBID.ReaderCommand._0x72.Req.No3.OUT_NUMBER, (byte)0x02); // relais 2 reader.SetData(OBID.ReaderCommand._0x72.Req.No3.OUT_TYPE, (byte)0x04); // type: relais reader.SetData(OBID.ReaderCommand._0x72.Req.No3.State.MODE, (byte)0x01); // switching on reader.SetData(OBID.ReaderCommand._0x72.Req.No3.State.FREQUENCY, (byte)0x00); // unchanged reader.SetData(OBID.ReaderCommand._0x72.Req.No3.OUT_TIME, (uint)10); // 1000 ms reader.SendProtocol((0x72);</pre>
[0x74] Get Input	<pre>// Beispiel für ID ISC.LR2500-B bool in1 = false; // Eingang 1 bool in2 = false; // Eingang 2 bool in3 = false; // Eingang 3 reader.SendProtocol(0x74); reader.GetData(OBID.ReaderCommand._0x74.Rsp.Inputs.IN1, out in1); reader.GetData(OBID.ReaderCommand._0x74.Rsp.Inputs.IN2, out in2); reader.GetData(OBID.ReaderCommand._0x74.Rsp.Inputs.IN3, out in3);</pre>
[0x75] Adjust Antenna	<pre>int antValue = 0; // Antennen-Spannung reader.SendProtocol(0x75); reader.GetData(FEDM_ISCM_TMP_ANTENNA_VALUE, out antValue);</pre>

[Steuerbyte] Protokoll	Beispiel ⁹
<p>[0x80] Read Configuration</p> <p>und</p> <p>[0x81] Write Configuration</p>	<pre>// das Beispiel zeigt das Lesen und Zurückschreiben eines Blocks der Leserkonfiguration byte cfgAdr = 2; // Adresse des Konfigurationsblocks bool eeProm = true; // Konfigurationsdaten aus/in EEPROM des Lesers byte busAddress; // Busadresse des ISC.LR-Lesers aus Block 2 reader.SetData(OBID.ReaderCommand._0x80.Req.CFG_ADDRESS, (byte)0x00); //alles zurücksetzen reader.SetData(OBID.ReaderCommand._0x80.Req.CfgAddr.ADDRESS, cfgAdr); // Adresse setzen reader.SetData(OBID.ReaderCommand._0x80.Req.CfgAddr.LOCATION, eeProm); // Speicherort auf EEPROM setzen reader.SendProtocol(0x80); // Konfigurationsdaten lesen //Busadresse übernehmen reader.GetConfigPara(OBID.ReaderConfig.HostInterface.Serial.BusAddress, out busAddress); reader.SetData(OBID.ReaderCommand._0x81.Req.CFG_ADDRESS, (byte)0x00); //alles zurücks. reader.SetData(OBID.ReaderCommand._0x81.Req.CfgAddr.ADDRESS, cfgAdr); //Adresse setzen reader.SetData(OBID.ReaderCommand._0x81.Req.CfgAddr.LOCATION, eeProm); // Speicherort auf EEPROM setzen reader.SendProtocol(0x81); // Konfigurationsdaten zurückschreiben</pre>
[0x83] Set Default Configuration	<pre>reader.SetData(OBID.ReaderCommand._0x83.Req.CFG_ADDRESS, (byte)0x00); //alles zurücksetzen reader.SetData(OBID.ReaderCommand._0x83.Req.CfgAddr.ADDRESS, (byte)0x02); //Adresse setzen reader.SetData(OBID.ReaderCommand._0x83.Req.CfgAddr.LOCATION, false); // RAM wählen reader.SetData(OBID.ReaderCommand._0x83.Req.CfgAddr.MODE, false); // nur Block 2 auf Default reader.SendProtocol(0x83); // Konfigurationsdaten von Block 2 in RAM auf Default setzen</pre>
[0x85] Set System Timer	<pre>reader.SetData(OBID.ReaderCommand._0x85.Req.TIMER_HOUR, (uint)16); // 16 Uhr reader.SetData(OBID.ReaderCommand._0x85.Req.TIMER_MINUTE, (uint)20); // 20 Minuten reader.SetData(OBID.ReaderCommand._0x85.Req.TIMER_MILLISECONDS, (uint)2000); // 2000 Millisekunden reader.SendProtocol(0x85); // Timer setzen</pre>
[0x86] Get System Timer	<pre>uint hour = 0; // Stunden uint minute = 0; // Minuten uint milliSec = 0; // Millisekunden reader.SendProtocol(0x86); // Timer auslesen reader.GetData(OBID.ReaderCommand._0x86.Req.TIMER_HOUR, out hour); //Stunden übernehmen reader.GetData(OBID.ReaderCommand._0x86.Req.TIMER_MINUTE, out minute); // Minuten übernehmen reader.GetData(OBID.ReaderCommand._0x86.Req.TIMER_MILLISECONDS, out milliSec); // Millisekunden übernehmen</pre>

[Steuerbyte] Protokoll	Beispiel ⁹
[0x87] Set System Date	<pre> reader.SetData(OBID.ReaderCommand._0x87.Req.DATE_CENTURY, (uint)20); // 20. Jahrhundert reader.SetData(OBID.ReaderCommand._0x87.Req.DATE_YEAR, (uint)4); // Jahr 04 im Jahrhundert reader.SetData(OBID.ReaderCommand._0x87.Req.DATE_MONTH, (uint)9); // September reader.SetData(OBID.ReaderCommand._0x87.Req.DATE_DAY, (uint)15); // 15. September reader.SetData(OBID.ReaderCommand._0x87.Req.DATE_TIMEZONE, (uint)0); // z.Zt. ungenutzt reader.SetData(OBID.ReaderCommand._0x87.Req.TIME_HOUR, (uint)12); // Stunden reader.SetData(OBID.ReaderCommand._0x87.Req.TIME_MINUTE, (uint)00); // Minuten reader.SetData(OBID.ReaderCommand._0x87.Req.TIME_MILLISECONDS, (uint)0); // Millisekunden (inkl. Sekunden) reader.SendProtocol(0x87); // Datum und Uhrzeit setzen </pre>
[0x88] Get System Date	<pre> byte cCentury = 0; // Jahr byte cYear = 0; // Jahr im Jahr byte cMonth = 0; // Monat byte cDay = 0; // Tag byte cTimezone = 0; // Zeitzone (z.Zt. ungenutzt) byte cHour = 0; // Stunde byte cMinute = 0; // Minute uint uiMilliSec = 0; // Millisekunden reader.SendProtocol(0x88); // Datum und Uhrzeit auslesen reader.GetData(OBID.ReaderCommand._0x88.Req.DATE_CENTURY, out cCentury); // Jahr reader.GetData(OBID.ReaderCommand._0x88.Req.DATE_YEAR, out cYear); // Jahr im Jahr reader.GetData(OBID.ReaderCommand._0x88.Req.DATE_MONTH, out cMonth); // Monat reader.GetData(OBID.ReaderCommand._0x88.Req.DATE_DAY, out cDay); // Tag im Monat reader.GetData(OBID.ReaderCommand._0x88.Req.DATE_TIMEZONE, out cTimezone); // z.Zt. ungenutzt reader.GetData(OBID.ReaderCommand._0x88.Req.TIME_HOUR, out cHour); // Stunden reader.GetData(OBID.ReaderCommand._0x88.Req.TIME_MINUTE, out cMinute); // Minuten reader.GetData(OBID.ReaderCommand._0x88.Req.TIME_MILLISECOND, out uiMilliSec); // Millisekunden </pre>
[0x8A] Read Configuration und [0x8B] Write Configuration	<pre> // das Beispiel zeigt das Lesen und Zurückschreiben eines Blocks der Leserkonfiguration uint CfgAdr = 1; // Adresse des Konfigurationsblocks byte BusAdress; // Busadresse des ISC.LRU3000-Lesers aus Block 1 reader.SetData(OBID.ReaderCommand._0x8A.Req.DEVICE, (byte)0x02); // RF-Controller reader.SetData(OBID.ReaderCommand._0x8A.Req.BANK, (byte)0x01); // Bank Main reader.SetData(OBID.ReaderCommand._0x8A.Req.MODE, (byte)0x00); // alles zurücksetzen reader.SetData(OBID.ReaderCommand._0x8A.Req.Mode.LOCATION, true); // EEPROM reader.SetData(OBID.ReaderCommand._0x8A.Req.CFG_ADDRESS, CfgAdr); // Konfigurations- Adr. reader.SetData(OBID.ReaderCommand._0x8A.Req.CFG_N, (byte)1); // 1 Konfigurationsblock reader.SendProtocol(0x8A); // Konfigurationsdaten lesen </pre>

[Steuerbyte] Protokoll	Beispiel ⁹
	<pre> // z.B. Busadresse übernehmen reader.GetConfigPara(OBID.ReaderConfig.HostInterface.Serial.BusAddress, out BusAdr); // Parameteränderung mit reader.SetConfigPara(ReaderConfig..., ...); reader.SetData(OBID.ReaderCommand._0x8B.Req.DEVICE, (byte)0x02); // RF-Controller reader.SetData(OBID.ReaderCommand._0x8B.Req.BANK, (byte)0x01); // Bank Main reader.SetData(OBID.ReaderCommand._0x8B.Req.MODE, (byte)0x00); // alles zurücksetzen reader.SetData(OBID.ReaderCommand._0x8B.Req.Mode.LOCATION, true); // EEPROM reader.SetData(OBID.ReaderCommand._0x8B.Req.CFG_ADDRESS, CfgAdr); // Konfigurations- Adr. reader.SetData(OBID.ReaderCommand._0x8B.Req.CFG_N, (byte)1); // 1 Konfigurationsblock reader.SendProtocol(0x8B); // Konfigurationsdaten zurückschreiben </pre>

[Steuerbyte] Protokoll	Beispiel ⁹
[0x8C] Set Default Configuration	<pre> reader.SetData(OBID.ReaderCommand._0x8C.Req.DEVICE, (byte)0x02); // RF-Controller reader.SetData(OBID.ReaderCommand._0x8C.Req.BANK, (byte)0x01); // Bank Main reader.SetData(OBID.ReaderCommand._0x8C.Req.MODE, (byte)0x00); // alles zurücksetzen reader.SetData(OBID.ReaderCommand._0x8C.Req.Mode.LOCATION, true); // EEPROM reader.SetData(OBID.ReaderCommand._0x8C.Req.CFG_ADDRESS, (uint)1); // Konfigurations-Adr. reader.SetData(OBID.ReaderCommand._0x8C.Req.CFG_N, (byte)1); // 1 Konfigurationsblock reader.SendProtocol(0x8C); // Konfigurationsdaten in CFG1 zurücksetzen </pre>
[0xA0] Reader Login	<pre> string passWord; // Leserpasswort // hole das Passwort z.B. aus einem Eingabefeld reader.SetData(OBID.ReaderCommand._0xA0.Req.PASSWORD, passWord); // Passwort setzen reader.SendProtocol(0xA0); // Passwort an Leser schicken </pre>
[0xA2] Write Mifare Keys	<pre> string key; // Mifare-Key // hole den Mifare-Key z.B. aus einem Eingabefeld reader.SetData(OBID.ReaderCommand._0xA2.Req.KEY_TYPE, (byte)0); reader.SetData(OBID.ReaderCommand._0xA2.Req.KEY_ADR, (byte)0); reader.SetData(OBID.ReaderCommand._0xA2.Req.KEY, key); reader.SendProtocol(0xA2); // Mifare-Key an Leser schicken </pre>
[0xA3] Write AES/DES Keys	<pre> string key; // Key // hole den Mifare-Key z.B. aus einem Eingabefeld reader.SetData(OBID.ReaderCommand._0xA3.Req.MODE, (byte)0); // RAM reader.SetData(OBID.ReaderCommand._0xA3.Req.KEY_INDEX, (byte)0); reader.SetData(OBID.ReaderCommand._0xA3.Req.AUTHENTICATION_MODE, (byte)0); //DESFire native TDES reader.SetData(OBID.ReaderCommand._0xA3.Req.KEY_LEN, key.length); reader.SetData(OBID.ReaderCommand._0xA3.Req.KEY, key); reader.SendProtocol(0xA3); // Key an Leser schicken </pre>
[0xAD] Write Reader Authent Key	<pre> string key; // Authent-Key // hole den Authent-Key z.B. aus einem Eingabefeld reader.SetData(OBID.ReaderCommand._0xAD.Req.KEY_TYPE, (byte)2); // AES256 reader.SetData(OBID.ReaderCommand._0xAD.Req.KEY_LEN, (byte)32); reader.SetData(OBID.ReaderCommand._0xAD.Req.KEY, key); reader.SendProtocol(0xAD); // Authent-Key an Leser schicken </pre>

[Steuerbyte] Protokoll	Beispiel ⁹
<p>[0xB0] ISO Mandatory and Optional Commands</p>	<pre>// das Beispiel zeigt den [0x01] Inventory reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x01); // Inventory reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x01.Req.MODE, (byte)0x00); // kein More-Flag reader.SendProtocol(0xB0); // die Inventory-Daten liegen im internen ISO-Table-Objekt. Beispiel für Datenzugriffe in 8.2.2. Beispiele für die Verwendung der ISO Tabelle mit [0xB0] Protokollen</pre>
<p>[0xB1] ISO15693 Customer and Proprietary Commands</p> <p>(TagHandler-Klassen bieten ein vollständiges undeinfacheres API)</p>	<pre>// das Beispiel zeigt den [0xA2] Set EAS // alle anderen 0xB1-Kommandos entsprechend string snr = new string(); // für Seriennummer byte isoError = 0; // für ISO-Fehlercode reader.SetData(FEDM_ISC_TMP_B1_CMD, (byte)0xA2); // Set EAS reader.SetData(FEDM_ISC_TMP_B1_MFR, (byte) ISO_MFR_PHILIPS); //Hersteller Code reader.SetData(FEDM_ISC_TMP_B1_MODE, (byte) ISO_MODE_ADR); // addressed mode // ... Seriennummer z. B. aus Textfeld holen und in snr speichern reader.SetData(FEDM_ISC_TMP_B1_REQ_UID, snr); int status = reader.SendProtocol(0xB1); if(status == 0x95) { // hole ISO-Fehlercode reader.GetData(FEDM_ISC_TMP_B1_ISO_ERROR, out isoError); }</pre>
<p>[0xB2] ISO14443 Special Commands</p> <p>[0x2B] ISO14443-4 Transponder Info</p> <p>(TagHandler-Klassen bieten ein vollständiges undeinfacheres API)</p>	<pre>byte cFSCI = 0; byte cFWI = 0; byte cDSI = 0; byte cDRI = 0; byte cNad = 0; byte cCid = 0; reader.SetData(FEDM_ISC_TMP_B2_CMD, (byte)0x2B); // ISO14443-4 Transponder Info int iStatus = reader.SendProtocol(0xB2); // Transponder muß zuvor mit [0x25] Select selektiert sein if(iStatus == 0x00) { // Tabellenindex des selektierten Transponders ermitteln int ildx = reader.FindTableIndex(0, FEDM_ISC_ISO_TABLE, DATA_IS_SELECTED, true); if(ildx >= 0) { // hole Transponderdaten reader.GetTableData(ildx, FEDM_ISC_ISO_TABLE, DATA_FSCI, out cFSCI) reader.GetTableData(ildx, FEDM_ISC_ISO_TABLE, DATA_FWI, out cFWI) reader.GetTableData(ildx, FEDM_ISC_ISO_TABLE, DATA_DSI, out cDSI) reader.GetTableData(ildx, FEDM_ISC_ISO_TABLE, DATA_DRI, out cDRI) reader.GetTableData(ildx, FEDM_ISC_ISO_TABLE, DATA_NAD, out cNad) reader.GetTableData(ildx, FEDM_ISC_ISO_TABLE, DATA_CID, out cCid) } }</pre>

[Steuerbyte] Protokoll	Beispiel ⁹
<p>[0xB2] ISO14443 Special Commands</p> <p>[0xB0] Authent Mifare</p> <p>(TagHandler-Klassen bieten ein vollständiges undeinfacheres API)</p>	<pre> byte dbAddress = 0; // Adresse des ersten Datenblocks byte keyType = 0; // Keytype für Authentifikation byte keyAdr = 0; // EEPROM-Adresse des Keys im Leser byte keyLocation = 0; // Location of the Authent-Key (0: Reader; 1: Protocol) string key = "000000000000"; // Authent-Key reader.SetData(FEDM_ISC_TMP_B2_CMD, (byte)0xB0); // Authent Mifare reader.SetData(FEDM_ISC_TMP_B2_MODE, (byte)0x00); // Mode-Byte löschen reader.SetData(FEDM_ISC_TMP_B2_MODE, (byte)FEDM_ISC_ISO_MODE_SEL); //selected reader.SetData(FEDM_ISC_TMP_B2_REQ_KEY_TYPE, keyType); reader.SetData(FEDM_ISC_TMP_B2_REQ_DB_ADR, dbAddress); reader.SetData(FEDM_ISC_TMP_B2_MODE_KL, keyLocation); if(keyLocation == 0) reader.SetData(FEDM_ISC_TMP_B2_REQ_KEY_ADR, keyAdr); else reader.SetData(FEDM_ISC_TMP_ISO14443A_KEY, key); reader.SendProtocol(0xB2); </pre>
<p>[0xB2] ISO14443 Special Commands</p> <p>[0xB1] Authent my-d</p> <p>(TagHandler-Klassen bieten ein vollständiges undeinfacheres API)</p>	<pre> byte keyAdrTag = 5; // Adresse des Keys auf dem Transponder byte keyAdrSam = 2; // Adresse des Keys im Authentifikationsmodul byte cntAdr = 3; // Adresse des Authentifikationszählers byte authSeq = 0; // Authentifikationssequenz reader.SetData(FEDM_ISC_TMP_B2_CMD, (byte)0xB1); // Authent my-d reader.SetData(FEDM_ISC_TMP_B2_MODE, (byte) FEDM_ISC_ISO_MODE_SEL); // selected reader.SetData(FEDM_ISC_TMP_B2_REQ_KEY_ADR_TAG, keyAdrTag); reader.SetData(FEDM_ISC_TMP_B2_REQ_KEY_ADR_SAM, keyAdrSam); reader.SetData(FEDM_ISC_TMP_B2_REQ_AUTH_COUNTER_ADR, cntAdr); reader.SetData(FEDM_ISC_TMP_B2_REQ_KEY_AUTH_SEQUENCE, authSeq); reader.SendProtocol(0xB2); </pre>
<p>[0xB2] ISO14443 Special Commands</p> <p>[0xB2] Authent Mifare Ultralight C</p> <p>(TagHandler-Klassen bieten ein vollständiges undeinfacheres API)</p>	<pre> byte keyIndex = 0; // reader key index for authentication reader.SetData(FEDM_ISC_TMP_B2_CMD, (byte)0xB2); // Authent Mifare Ultralight C reader.SetData(FEDM_ISC_TMP_B2_MODE, (byte)0x00); // clear mode byte reader.SetData(FEDM_ISC_TMP_B2_MODE, (byte)FEDM_ISC_ISO_MODE_SEL); //selected reader.SetData(FEDM_ISC_TMP_B2_REQ_KEY_INDEX, keyIndex); reader.SendProtocol(0xB2); </pre>

[Steuerbyte] Protokoll	Beispiel ⁹
<p>[0xB2] ISO14443 Special Commands</p> <p>[0x30] Mifare Value Commands</p> <p>(TagHandler-Klassen bieten ein vollständiges und einfacheres API)</p>	<pre> byte mfCmd = 0x01; // Mifare Command byte dbAdr = 0x05; // Datenblock-Adresse byte[] opValue = new byte[4]; // OP_VALUE byte destAdr = 0x05; // Zieladresse opValue[0] = 0x00; opValue[1] = 0x00; opValue[2] = 0x00; opValue[3] = 0x03; reader.SetData(FEDM_ISC_TMP_B2_CMD, (byte)0x30); // Mifare Value Commands reader.SetData(FEDM_ISC_TMP_B2_MODE, (byte) FEDM_ISC_ISO_MODE_SEL); // selected reader.SetData(FEDM_ISC_TMP_B2_REQ_MF_CMD, mfCmd); reader.SetData(FEDM_ISC_TMP_B2_REQ_DB_ADR, dbAdr); reader.SetData(FEDM_ISC_TMP_B2_REQ_OP_VALUE, opValue); reader.SetData(FEDM_ISC_TMP_B2_REQ_DEST_ADR, destAdr); reader.SendProtocol(0xB2); </pre>

8.2. Tabellen orientierte Kommandos für den Leser

Tabellen orientierte Kommandos können nur zum Einsatz kommen, wenn die jeweilige Tabelle (Tabelle für Host Kommandos bzw. Tabelle für Buffered Read Mode) zuvor dimensioniert wurde (s. [5.1.1. Initialisierung](#)).

Wenn Transponder mit mehr als 256 Datenblöcken im Host-Mode verwendet werden, muss die Tabelle zuvor mit der Methode SetTableSize (2) vorbereitet werden (s. [7.1.52. SetTableSize](#)).

8.2.1. Besonderheiten des addressed mode

Fast alle Host-Commands können im addressed mode verwendet werden. In diesem Fall muß die Seriennummer – manchmal auch unified identifier (UID) genannt – in das Protokoll eingesetzt werden. In früheren Versionen der Bibliothek wurden nur UIDs mit einer Länge von 8 Byte unterstützt. Inzwischen ermöglicht ein Flag im Mode-Byte (UID_LF) abweichende UID-Längen. Ist das UID_LF Flag gesetzt, muß die Länge der UID im Protokoll eingetragen werden.

Nachfolgendes Beispiel zeigt exemplarisch einen [0xB0][0xB23] Read Multiple Blocks:

```
// setze UID für Addressed Mode (UID darf bis zu 96 Byte enthalten)
reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.UID, sUid);
reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddMode.UID_LEN, ucUidLen); // Anzahl Byte der UID

reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x23); // Command Read Multiple Blocks
reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen
reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.Mode.ADR, (byte)0x01); // Addressed Mode
reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.Mode.UID_LF, true); // UID_LF Flag
reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.DBN, (byte)0x01); // ein Datenblock lesen
reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.DB_ADR, ucDBAdr); //setze Datenblock-Adresse

reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder
```

8.2.2. Beispiele für die Verwendung der ISO Tabelle mit [0xB0] Protokollen

[Steuerbyte] Protokoll	Beispiel ¹⁰
[0x01] Inventory für HF-Transponder: - Philips I-CODE1 - Texas Instruments Tag-it HF - ISO15693 - ISO14443A - ISO14443B - EPC (Electronic Product Code) - Philips I-CODE UID - Innovision Jewel - ISO 18000-3M3 für UHF-Transponder: - ISO18006-6-B	<pre>byte trType = 0; // für Transpondertyp string snr; // für Seriennummer (auch EPC) string header; // für EPC Header string domain; // für EPC DomainManager-Feld string object ; // für EPC ObjektClass-Feld string epc; // für EPC ("Header.DomainManager.ObjectClass.SerialNumber") reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x01); // Command Inventory reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x01.Req.MODE, (byte)0x00); // kein More-Flag // Tabellenlänge auf 0 setzen und Tabelleninhalt komplett löschen reader.ResetTable(ISO_TABLE); reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder</pre>

¹⁰ alle Beispiele sind in C# ausgeführt

[Steuerbyte] Protokoll	Beispiel ¹⁰
- EM4222 - EPC Class0/0+ - EPC Class1 Gen1 - EPC Class1 Gen2	<pre> // Alle Transponderdaten sind in der Tabelle enthalten for(int cnt=0; cnt< reader.GetTableLength(ISO_TABLE); ++cnt) { // hole Transpondertyp reader.GetTableData(cnt, ISO_TABLE, DATA_TRTYPE, out trType); switch(trType) { case 0x00: // Philips I-CODE1 case 0x01: // Texas Instruments Tag-it HF case 0x03: // ISO15693 case 0x04: // ISO14443A case 0x05: // ISO14443B case 0x07: // I-Code UID case 0x08: // Innovision Jewel case 0x09: // ISO 18000-3M3 case 0x81: // ISO18000-6-B case 0x83: // EM4222 case 0x84: // EPC Class1 Gen2 case 0x88: // EPC Class0/0+ case 0x89: // EPC Class1 Gen1 // hole Seriennummer als String reader.GetTableData(cnt, ISO_TABLE, DATA_SNR, out snr); break; case 0x06: // EPC (Electronic Product Code) // hole EPC-Felder reader.GetTableData(cnt, ISO_TABLE, DATA_EPC_HEADER, out header); reader.GetTableData(cnt, ISO_TABLE, DATA_EPC_DOMAIN, out domain); reader.GetTableData (cnt, ISO_TABLE, DATA_EPC_OBJECT, out object); reader.GetTableData (cnt, ISO_TABLE, DATA_EPC_SNR, out snr); // oder hole EPC-Feld als kompletten String reader.GetTableData (cnt, ISO_TABLE, DATA_EPC, out epc); break; } } </pre>
[0x02] Stay Quiet	<pre> string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x02.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x02); //Command Stay Quiet reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x02.Req.MODE, (byte)0x00); //Mode-Byte zurücks. reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x02.Req.Mode.ADR, (byte)0x01); //Addr. Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x22] Lock Multiple Blocks	<pre> /* <u>Achtung</u>: mit diesem ISO Command werden Datenblöcke unwiderruflich schreibgeschützt!! string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // setze Seriennummer für Addressed Mode </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
	<pre> reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x22.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x22); // Command Lock Multiple Blocks reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x22.Req.MODE, (byte)0x00); // Mode- Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x22.Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x22.Req.NormAddrMode.DBN, (byte)0x01); // einen Datenblock sperren reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x22.Req.NormAddrMode.DB_ADR, (byte)0x00); // setze Datenblock-Adresse reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
<p>[0x23] Read Multiple Blocks (normale Adressierung)</p>	<pre> byte[] dataBlock; // Puffer für einen Datenblock byte dbAddress = 5; // Datenblock-Adresse 5 string snr ; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x23); // Command Read Multiple Blocks reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.Mode.ADR, (byte)0x01); // Addr. Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.NormAddrMode.DBN, (byte)0x01); // ein Datenblock lesen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.NormAddrMode.DB_ADR, dbAddress); // setze Datenblock-Adresse reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0,ISO_TABLE,DATA_SNR, snr); if(idx < 0) return; // hole die Größe des Datenblocks (Blocksize) byte blockSize; reader.GetTableData(idx, ISO_TABLE, DATA_BLOCKSIZE, out blockSize); // ... mach was mit der Blocksize // hole einen Datenblock (dataBlock wird nur blockSize Datenbytes enthalten) reader.GetTableData(idx, ISO_TABLE, DATA_RxDB, dbAddress, out dataBlock); // ... mach was mit dem Datenblock </pre>
<p>[0x23] Read Multiple Blocks (erweiterte Adressierung)</p>	<pre> byte[] dataBlock; // Puffer für einen Datenblock uint dbAddress = 5; // Datenblock-Adresse 5 string snr ; // für Seriennummer string sPw; // für Access Passwort // ... Seriennummer z. B. aus Textfeld holen // ... Passwort z. B. aus Textfeld holen und in sPw speichern // // setze Seriennummer (> 8 Byte zulässig) für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.UID_LEN, snr.Length/2); // Länge der UID in Byte reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x23); // Command </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
	<pre> Read Multiple Blocks reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.Mode.ADR, (byte)0x01); // Addr. Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.Mode.EXT_ADR, true);// erweiterter Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.Mode.UID_LF, true); // nur falls UID-Länge != 8 reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.BANK, (byte)0x00);// Speicherbank zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.Bank.Number, (byte)0x03); //Bank User Memory reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.Bank.ACCESS_ FLAG, true); // mit Access Passwort reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.ACCESS_PW_L ENGTH, (byte)sPw.Length/2);// Länge PW reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.ACCESS_PW, sPw); // setze Passwort reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.DB_ADR, dbAddress); // Datenblock-Adresse reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x23.Req.ExtAddrMode.DBN, (byte)0x01); // ein Datenblock lesen reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0,ISO_TABLE,DATA_SNR, snr); if(idx < 0) return; // hole die Größe des Datenblocks (Blocksize) byte blockSize; reader.GetTableData(idx, ISO_TABLE, DATA_BLOCKSIZE, out blockSize); // ... mach was mit der Blocksize // hole einen Datenblock (dataBlock wird nur blockSize Datenbytes enthalten) reader.GetTableData(idx, ISO_TABLE, DATA_RxDB, dbAddress, out dataBlock); // ... mach was mit dem Datenblock </pre>
<p>[0x24] Write Multiple Blocks (normale Adressierung)</p>	<pre> /* das Beispiel zeigt den [0x24] Write Multiple Blocks. Im Addressed Mode muß zuvor ein [0x01] Inventory ausgeführt sein. Achtung: wenn noch kein [0x23] Read Multiple Blocks ausgeführt wurde, dann ist die Blocksize auf 4 voreingestellt. Unterstützt der Transponder im Lesefeld aber eine andere Blocksize, muß diese zuerst in der Tabelle für diesen Transponder gesetzt werden!! Man kann mit GetTableData(..., DATA_IS_BLOCK_SIZE_SET, ...) abfragen, ob die Blocksize bereits mit [0x23] Read Multiple Blocks gelesen wurde. Mit dem folgenden Aufruf kann man die Blockgröße in der Tabelle setzen, wenn man explizit nur Schreiben will und die Blockgröße (z.B. 8) bekannt ist: SetTableData(idx, ISO_TABLE, DATA_BLOCK_SIZE, (byte)8); */ byte[] dataBlock; // Puffer für einen Datenblock byte adAddress = 5; // Datenblock-Adresse 5 string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // ... Datenblock z. B. aus einem Textfeld holen und in dataBlock speichern // Tabellenindex der Seriennummer ermitteln </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
	<pre> int idx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x24); // Command Read Multiple Blocks reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24.Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24.Req.NormAddrMode.DBN, (byte)0x01); // einen Datenblock schreiben reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24.Req.NormAddrMode.DB_ADR, dbAddress); // setze Datenblock-Adresse reader.SetTableData(idx, ISO_TABLE, DATA_BLOCK_SIZE, (byte)0x08); // setze Blocksize auf 8 // schreibe einen Datenblock (mit einer Blocksize von 8 Bytes!) in Tabelle reader.SetTableData(idx, ISO_TABLE, DATA_TxDB, ucDBAdr, dataBlock); reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
<p>[0x24] Write Multiple Blocks (erweiterte Adressierung)</p>	<pre> /* das Beispiel zeigt den [0x24] Write Multiple Blocks. Im Addressed Mode muß zuvor ein [0x01] Inventory ausgeführt sein. Achtung: wenn noch kein [0x23] Read Multiple Blocks ausgeführt wurde, dann ist die Blocksize auf 4 voreingestellt. Unterstützt der Transponder im Lesefeld aber eine andere Blocksize, muß diese zuerst in der Tabelle für diesen Transponder gesetzt werden!! Man kann mit GetTableData(..., DATA_IS_BLOCK_SIZE_SET, ...) abfragen, ob die Blocksize bereits mit [0x23] Read Multiple Blocks gelesen wurde. Mit dem folgenden Aufruf kann man die Blockgröße in der Tabelle setzen, wenn man explizit nur Schreiben will und die Blockgröße (z.B. 8) bekannt ist: SetTableData(ildx, ISO_TABLE, DATA_BLOCK_SIZE, (byte)8); */ byte[] dataBlock; // Puffer für einen Datenblock uint dbAddress = 5; // Datenblock-Adresse 5 string snr; // für Seriennummer string sPw; // für Access Passwort // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // ... Passwort z. B. aus Textfeld holen und in sPw speichern // ... Datenblock z. B. aus einem Textfeld holen und in dataBlock speichern // Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.UID_LEN, snr.Length / 2); // Länge der UID in Byte reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x24); // Command Read Multiple Blocks reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.Mode.EXT_ADR, true); // erweiterter Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.Mode.UID_LF, true); // nur falls UID-Länge != 8 reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.BANK, (byte)0x00); // Speicherbank zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.Bank.NUMBER, (byte)0x03); // Bank User Memory reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.Bank.ACCESS_ FLAG, true); // mit Access Password reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.ACCESS_PW_L ENGTH, (byte)sPw.Length/2); // Länge PW reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.ACCESS_PW, sPw); // setze Passwort reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.DB_ADR, dbAddress); // Datenblock-Adresse reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x24 Req.ExtAddrMode.DBN, (byte)0x01); // einen Datenblock schreiben reader.SetTableData(idx, ISO_TABLE, DATA_BLOCK_SIZE, (byte)0x08); // setze Blocksize auf 8 // schreibe einen Datenblock (mit einer Blocksize von 8 Bytes!) in Tabelle reader.SetTableData(idx, ISO_TABLE, DATA_TxDB, ucDBAdr, dataBlock); </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
	reader. SendProtocol (0xB0); // Kommunikation mit Leser/Transponder

[Steuerbyte] Protokoll	Beispiel ¹⁰
[0x25] Select	<pre> string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x25); // Command Select reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x25] Select mit Option Card Information für ISO14443 Transponder	<pre> string snr; // für Seriennummer byte format = 0; // Format Byte aus dem Antwortprotokoll // ... Seriennummer z. B. aus Textfeld holen und in sSnr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x25); // Command Select reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Req.Mode.CINF, true); // CINF-Flag reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // das Format Byte ist in TempData gespeichert reader.GetData(OBID.ReaderCommand._0xB0.SubCmd._0x25.Rsp.FORMAT; format); // Format // die Card Information sind im TMPDATA_MEM ab Index 2048 gespeichert // die Struktur und Länge der Daten müssen dem Systemhandbuch entnommen werden // der prinzipielle Zugriff sieht so aus: // byte[] cardInfo; // int length = s. Systemhandbuch // reader.GetData(2048, cardInfo, length, TMPDATA_MEM); </pre>
[0x26] Reset to Ready	<pre> string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x26.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x26); // Command Reset to Ready reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x26.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x26.Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x27] Write AFI	<pre> string snr; // für Seriennummer byte afi = 0; // für AFI // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // ... AFI z. B. aus Eingabefeld holen und in afi speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x27.Req.UID, snr); // Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, snr); </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
	<pre> if(idx < 0) return; // schreibe AFI in Tabelle reader.SetTableData(idx,ISO_TABLE, DATA_AFI, afi); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x27);// Command Write AFI reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x27.Req.MODE, (byte)0x00);// Mode-Byte zurücksetzen reader.setData(OBID.ReaderCommand._0xB0.SubCmd._0x27.Req.Mode.ADR, (byte)0x01);//Addr. Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x28] Lock AFI	<pre> string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x28.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x28);// Command Lock AFI reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x28.Req.MODE, (byte)0x00);// Mode-Byte löschen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x28.Req.Mode.ADR, (byte)0x01);// Addr. Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x29] Write DSFID	<pre> string snr; // für Seriennummer byte dsfid = 0; // für DSFID // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // ... DSFID z. B. aus Eingabefeld holen und in dsfid speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x29.Req.UID, snr); // Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0,ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // schreibe DSFID in Tabelle reader.SetTableData(idx, ISO_TABLE, DATA_DSfid, dsfid); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x29);// Command Write DSFID reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x29.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x29.Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x2A] Lock DSFID	<pre> string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2A.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x2A);// Command Lock DSFID reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2A.Req.MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2A.Req.Mode.ADR, (byte)0x01); // Addr. Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>
[0x2B] Get System Information	<pre> byte dsfid = 0; // für DSFID byte afi = 0; // für AFI </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
	<pre> byte[] ucMemSize = {0, 0}; // für Memory-Size byte icRef = 0; // für IC-Reference string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2B.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x2B); // Command Get System Information reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2B.Req.MODE, (byte)0x00); // Mode- Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2B.Req.Mode.ADR, (byte)0x01); // Addressed Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // hole AFI reader.GetTableData(idx, ISO_TABLE, DATA_AFI, out afi); // ... mach was mit AFI // ... hole alle anderen Daten nach dem gleichen Muster </pre>
[0x2C] Get Multiple Block Security Status	<pre> byte secStatus; // für Security Status string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // setze Seriennummer für Addressed Mode reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2C.Req.UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2C.Req.DBN, (byte)0x05); // 5 Datenblöcke reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2C.Req.DB_ADR, (byte)0x00); // setze 1. Datenblock-Adresse reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0x2C); // Get Multiple Block Security Status reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2C.Req.MODE, (byte)0x00); // Mode- Byte zurücksetzen reader.SetData(OBID.ReaderCommand._0xB0.SubCmd._0x2C.Req.Mode.ADR, (byte)0x01); // Addr. Mode reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // hole von den Blöcken 0..4 den Security Status for(int cnt=0; cnt<5; ++cnt) { reader.GetTableData(idx, ISO_TABLE, DATA_SEC_STATUS, cnt, out secStatus); // ... mach was mit secStatus } </pre>
[0xA0] Read Config Block nur für I-Code 1	<pre> byte[] configBlock; // Puffer für einen Datenblock (Blocksize ist immer 4) byte cbAddress = 0; // Datenblock-Adresse 0 string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen // setze Seriennummer für Addressed Mode reader.SetData(FEDM_ISC_TMP_B0_REQ_UID, snr); </pre>

[Steuerbyte] Protokoll	Beispiel ¹⁰
	<pre> reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0xA0); // Command Read Configuration Block reader.SetData(FEDM_ISC_TMP_B0_MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(FEDM_ISC_TMP_B0_MODE_ADR, (byte)0x01); // Addressed Mode reader.SetData(FEDM_ISC_TMP_B0_REQ_CB_ADR, cbAddress); // setze Datenblock-Adresse reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder // Alle Transponderdaten sind in der Tabelle enthalten // zuerst Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // hole den Datenblock reader.GetTableData(idx, ISO_TABLE, DATA_RxCB, cdAddress, out configBlock); // ... mach was mit dem Datenblock </pre>
<p>[0xA1] Write Config Block</p> <p>nur für I-Code 1</p>	<pre> /* Achtung: Mit diesem ISO Command kann man die Konfiguration eines Transponders so verändern, dass er unbrauchbar wird!! */ byte[] configBlock; // Puffer für einen Datenblock (Blocksize ist immer 4) byte cbAddress = 0; // Datenblock-Adresse 0 string snr; // für Seriennummer // ... Seriennummer z. B. aus Textfeld holen und in snr speichern // ... Datenblock z. B. aus einem Textfeld holen und in configBlock speichern // Tabellenindex der Seriennummer ermitteln int idx = reader.FindTableIndex(0,ISO_TABLE, DATA_SNR, snr); if(idx < 0) return; // setze Seriennummer für Addressed Mode reader.SetData(FEDM_ISC_TMP_B0_REQ_UID, snr); reader.SetData(OBID.ReaderCommand._0xB0.SUB_COMMAND, (byte)0xA1); // Command Write Multiple Block reader.SetData(FEDM_ISC_TMP_B0_MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(FEDM_ISC_TMP_B0_MODE_ADR, (byte)0x01); // Addr. Mode reader.SetData(FEDM_ISC_TMP_B0_REQ_CB_ADR, cbAddress); // setze Datenblock-Adresse // schreibe einen Datenblock in Tabelle reader.SetTableData(idx, ISO_TABLE, DATA_TxCB, cbAddress, configBlock); reader.SendProtocol(0xB0); // Kommunikation mit Leser/Transponder </pre>

8.2.3. Beispiele für die Verwendung der ISO Tabelle mit [0xB3] Protokollen

[Steuerbyte] Protokoll	Beispiel ¹¹
<p>[0x18] Kill</p> <p>für UHF-Transponder:</p> <ul style="list-style-type: none"> - EPC Class1 Gen1 - EPC Class1 Gen2 	<pre>// Achtung: mit diesem ISO Command werden Transponder unwiederruflich zerstört!! string sEpc; // für EPC string sPw; // für Kill Password byte cEpcLen = 0; // Länge der EPC in Byte byte cPwLen = 0; // Länge des Kill Password // ... EPC z. B. aus Textfeld holen und in sEpc speichern, dito mit der Länge // ... Passwort z. B. aus Textfeld holen und in sPw speichern, dito mit der Länge // Tabellenindex der EPC ermitteln int ildx = reader.FindTableIndex(0, ISO_TABLE, DATA_SNR, sEpc); // setze EPC für Addressed Mode reader.SetData(FEDM_ISC_TMP_B3_REQ_EPC, sEpc); reader.SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, cEpcLen); // Länge der EPC in Byte reader.SetData(FEDM_ISC_TMP_B3_CMD, (byte)0x18); // Command Kill reader.SetData(FEDM_ISC_TMP_B3_MODE, (byte)0x00); // Mode-Byte zurücksetzen reader.SetData(FEDM_ISC_TMP_B3_MODE_ADR, (byte)0x01); // Addressed Mode reader.SetData(FEDM_ISC_TMP_B3_MODE_EPC_LF, true); // EPC Längen Flag reader.SetData(FEDM_ISC_TMP_B3_KILL_PW_LENGTH, cPwLen); // Länge Passwort reader.SetData(FEDM_ISC_TMP_B3_KILL_PW, sPw); // Passwort reader.SendProtocol(0xB3); // Kommunikation mit Leser/Transponder</pre>
<p>[0x22] Lock Multiple Blocks</p> <p>für UHF-Transponder:</p> <ul style="list-style-type: none"> - EPC Class1 Gen1 - EPC Class1 Gen2 	<pre>// Achtung: mit diesem ISO Command können Datenblöcke unwiederruflich schreibgeschützt werden!! string sEpc; // für EPC string sLockData; // für optionale Lockdaten string sPw; // für optionales Access Password byte cEpcLen = 0; // Länge der EPC in Byte byte cTrType = 0; // Transpondertyp byte cLockDataLen = 0; // Länge der Lockdaten byte cPwLen = 0; // Länge des optionalen Access Password // ... EPC z. B. aus Textfeld holen und in sEpc speichern, dito mit der Länge // ... Lockdaten z. B. aus Textfeld holen und in sLockData speichern, dito mit der Länge // ... Passwort z. B. aus Textfeld holen und in sPw speichern, dito mit der Länge // Tabellenindex der EPC ermitteln int ildx = FindTableIndex(0, ISO_TABLE, DATA_SNR, sEpc); // Transpondertyp ermitteln GetTableData(ildx, ISO_TABLE, DATA_TRTYPE, out cTrType); // setze EPC für Addressed Mode SetData(FEDM_ISC_TMP_B3_REQ_EPC, sEpc); SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, cEpcLen); // Länge der EPC in Byte SetData(FEDM_ISC_TMP_B3_CMD, (byte)0x22); // Command Lock SetData(FEDM_ISC_TMP_B3_MODE, (byte)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B3_MODE_ADR, (byte)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_EPC_LF, true); // EPC Längen Flag SetData(FEDM_ISC_TMP_B3_REQ_TR_TYPE, cTrType); // Transpondertyp SetData(FEDM_ISC_TMP_B3_LOCK_DATA_LENGTH, cLockDataLen); // Länge Lockdaten</pre>

¹¹ alle Beispiele sind in C# ausgeführt

[Steuerbyte] Protokoll	Beispiel ¹¹
	<pre> SetData(FEDM_ISC_TMP_B3_LOCK_DATA, sLockData); // Lockdaten SetData(FEDM_ISC_TMP_B3_ACCESS_PW_LENGTH, cPwLen); // Länge Passwort if(cPwLen > 0) SetData(FEDM_ISC_TMP_B3_ACCESS_PW, sPw); // Passwort SendProtocol(0xB3); // Kommunikation mit Leser/Transponder </pre>
<p>[0x24] Write Multiple Blocks</p> <p>für UHF-Transponder: - EPC Class1 Gen2</p>	<pre> /* das Beispiel zeigt den [0x24] Write Multiple Blocks. Im Addressed Mode muß zuvor ein [0x01] Inventory ausgeführt sein. <u>Achtung:</u> wenn noch kein [0xB0][0x23] Read Multiple Blocks ausgeführt wurde, dann ist die Blocksize auf 4 voreingestellt. Unterstützt der Transponder im Lesefeld aber eine andere Blocksize, muß diese zuerst in der Tabelle für diesen Transponder gesetzt werden!! Man kann mit GetTableData(.., FEDM_ISC_DATA_IS_BLOCK_SIZE_SET, ..) abfragen, ob die Blocksize bereits mit [0xB0][0x23] Read Multiple Blocks gelesen wurde. */ byte[][] cDB; // Puffer für Daten (1. Dimension für Blocknummer, 2. Dimension für // Blockdaten) string sEpc; // für EPC string sPw; // für optionales Access Password byte cEpcLen = 0; // Länge der EPC in Byte byte cPwLen = 0; // Länge des optionalen Access Password // ... EPC z. B. aus Textfeld holen und in sEpc speichern, dito mit der Länge // ... Passwort z. B. aus Textfeld holen und in sPw speichern, dito mit der Länge // ... Datenblock z. B. aus einem Textfeld holen und in cDB speichern // Tabellenindex der EPC ermitteln int idx = FindTableIndex(0, ISO_TABLE, DATA_SNR, sEpc); // setze EPC für Addressed Mode SetData(FEDM_ISC_TMP_B3_REQ_UID, sEpc); SetData(FEDM_ISC_TMP_B3_REQ_EPC_LEN, cEpcLen); // Länge der EPC in Byte SetData(FEDM_ISC_TMP_B3_CMD, (byte)0x24); // Command Read Multiple Blocks SetData(FEDM_ISC_TMP_B3_MODE, (byte)0x00); // Mode-Byte zurücksetzen SetData(FEDM_ISC_TMP_B3_MODE_ADR, (byte)0x01); // Addressed Mode SetData(FEDM_ISC_TMP_B3_MODE_EPC_LF, true); // EPC Längen Flag SetData(FEDM_ISC_TMP_B3_MODE_EXT_ADR, true); // 2 Byte DB-Adresse SetData(FEDM_ISC_TMP_B3_BANK_ACCESS_FLAG, true); // mit Access Password SetData(FEDM_ISC_TMP_B3_BANK_BANK_NR, (byte)0x01); // EPC Speicherbank SetData(FEDM_ISC_TMP_B3_REQ_DBN, (byte)0x06); // sechs Datenblöcke schreiben SetData(FEDM_ISC_TMP_B3_REQ_DB_ADR_EXT, (uint)0); // erste Datenblock-Adresse SetData(FEDM_ISC_TMP_B3_REQ_DB_SIZE, (byte)0x02); // Größe Datenblock // setze Blocksize in Tabelle auf 2 SetTableData(idx, ISO_TABLE, DATA_BLOCK_SIZE, (byte)2); // schreibe Datenblöcke in Tabelle for(int iAdr=0; iAdr<6; ++iAdr) SetTableData(idx, ISO_TABLE, DATA_TxDB, iAdr, cDB[iAdr]); SendProtocol(0xB3); // Kommunikation mit Leser/Transponder </pre>

8.2.4. Kommandos für Buffered Read Mode

[Steuerbyte] Protokoll	Beispiel ¹²
[0x21] Read Buffer	<pre> // das Beispiel zeigt das Lesen von Datensätzen mit Seriennummer, Datenblock und Timer-Wert byte dataSets = 1; // Anzahl angeforderter Datensätze byte recSets = 0; // Anzahl Datensätze in Protokoll byte[] dataBlock; // Puffer für einen Datenblock FelscReaderTime time = 0; // für Timer-Wert string snr; // für Seriennummer bool snrFlag = false; // Flag für Seriennummer in Datensatz bool dbFlag = false; // Flag für Datenblock in Datensatz bool timerFlag = false; // Flag für Timer in Datensatz FedmBrmTableItem item; // ein Tabelleneintrag mit Daten für einen Transponder reader.SetData(FEDM_ISCLR_TMP_BRM_SETS, dataSets); reader.SendProtocol(0x21); // lese Datenblöcke von Transponder mit Buffered Read Mode reader.GetData(FEDM_ISCLR_TMP_BRM_TRDATA_SNR, out snrFlag); reader.GetData(FEDM_ISCLR_TMP_BRM_TRDATA_DB, out dbFlag); reader.GetData(FEDM_ISCLR_TMP_BRM_TRDATA_TIME, out timerFlag); reader.GetData(FEDM_ISCLR_TMP_BRM_RECSETS, out recSets); // Alle Transponderdaten sind in der Tabelle enthalten for(int cnt=0; cnt< reader.GetTableLength(BRM_TABLE); cnt++) { item = (FedmBrmTableItem) reader.GetTableItem(cnt, BRM_TABLE); if(snrFlag) // hole Seriennummer item.GetData(DATA_SNR, out snr); if(dbFlag) // hole Datenblock 1 item.GetData(DATA_RxDB, out dataBlock); if(timerFlag) // hole Timer-Wert time = item.GetReaderTime(); } </pre>
[0x22] Read Buffer	<pre> // das Beispiel zeigt das Lesen von Datensätzen mit Seriennummer, Datenblöcken, Timer-Wert, // Datumsfeld und Antennennummer uint uiDataSets = 1; // Anzahl angeforderter Datensätze uint uiRecSets = 0; // Anzahl Datensätze in Protokoll byte cAnt; // Antennennummer byte cInput = 0; // Input-Byte byte cState = 0; // Status-Byte FelscReaderTime time = 0; // für Timer-Wert byte cSize; // für Blockgröße eines Datenblocks uint uiDBN = 0; // für Anzahl der Datenblöcke string sSnr; // für Seriennummer string sDB; // für Datenblöcke bool bSNR = false; // Flag (in TR-DATA1) für Seriennummer in Datensatz bool bDB = false; // Flag (in TR-DATA1) für Datenblock in Datensatz bool bANT = false; // Flag (in TR-DATA1) für Antennennummer in Datensatz bool bTime = false; // Flag (in TR-DATA1) für Uhrzeit in Datensatz bool bDate = false; // Flag (in TR-DATA1) für Datumsfeld in Datensatz </pre>

¹² alle Beispiele sind in C# ausgeführt

[Steuerbyte] Protokoll	Beispiel ¹²
	<pre> bool bExt = false; // EXTENSION-Flag (in TR-DATA1) im Datensatz: zeigt an, dass ein // zweites TR-DATA Byte im Protokoll enthalten ist, welches weitere // Flags enthält bool blnput = false; // Flag (in TR-DATA2) für Input und Status im Datensatz FedmBrmTableItem item; // ein Tabelleneintrag mit Daten für einen Transponder SetData(OBID.ReaderCommand._0x22.Req.DATA_SETS, uiDataSets); SendProtocol(0x22); // lese Daten von Transponder mit Buffered Read Mode GetData(OBID.ReaderCommand._0x22.Req.TrData1.UID, out bSNR); GetData(OBID.ReaderCommand._0x22.Req.TrData1.DB, out bDB); GetData(OBID.ReaderCommand._0x22.Req.TrData1.ANT, out bANT); GetData(OBID.ReaderCommand._0x22.Req.TrData1.TIME, out bTime); GetData(OBID.ReaderCommand._0x22.Req.TrData1.DATE, out bDate); GetData(OBID.ReaderCommand._0x22.Req.TrData1.EXT, out bExt); GetData(OBID.ReaderCommand._0x22.Req.TrData2.INPUT, out Input); GetData(OBID.ReaderCommand._0x22.Rsp.DATA_SETS, out uiRecSets); // Alle Transponderdaten sind in der Tabelle enthalten for(int iCnt=0; iCnt<GetTableLength(FEDM_ISC_BRM_TABLE); iCnt++) { item = (FedmBrmTableItem) reader.GetTableItem(cnt, BRM_TABLE); if(bSNR) // hole Seriennummer item.GetData(cnt, DATA_SNR, out sSnr); if(bDB) // hole alle Datenblöcke { // hole Anzahl der Datenblöcke item.GetData(iCnt, DATA_DBN, out uiDBN); // hole die Blockgröße item.GetData(iCnt, DATA_BLOCK_SIZE, out cSize); // hole Datenblöcke for(int i=0; i<uiDBN; ++i) { item.GetData(iCnt, DATA_RxDB, i, out sDB); // tu was mit den Datenblöcken } } if(bANT) // hole Antennennummer item.GetData(iCnt, DATA_ANT_NR, out cAnt); if(bTime bDate) // hole Datum und/oder Uhrzeit time = item.GetReaderTime(); if(bExt && blnput) // hole Input- und Status-Byte { GetData(DATA_INPUT, out clnput); GetData(DATA_STATE, out cState); } } </pre>

8.3. Kommandos für eine Funktionseinheit

Die Funktion *SendProtocol* ist von zentraler Bedeutung für den Protokolltransfer. Aus diesem Grund wird für jedes Steuerbyte ein Beispiel aufgeführt, das verdeutlichen soll, welche Daten mit welchen Zugriffskonstanten vor jedem Protokolltransfer im Datencontainer zu speichern sind und welche Daten nach dem Protokolltransfer zur Verfügung stehen.

Alle Zugriffskonstanten sind in der Struktur *FedmlscFunctionUnitID* aufgelistet und sollten eingehend zusammen mit der im Systemhandbuch zur Funktionseinheit angeführten Erklärung der Protokoll Daten studiert werden.

Auf die Auswertung der Rückgabewerte der Funktionen wird hier aus Gründen der Übersichtlichkeit verzichtet. Sie sollte allerdings in Applikationen immer erfolgen.

[Steuerbyte] Protokoll		Beispiel ¹³
ID ISC.DAT	[0xC0] Get Firmware Version	<pre>string sFirmware; // Puffer für Firmware Informationen fu.SendProtocol(0xC0); fu.GetData(FEDM_ISC_FU_TMP_SOFTVER, out sFirmware);</pre>
	[0xC1] CPU Reset	<pre>fu.SendProtocol(0xC1);</pre>
	[0xC2] Set Capacities	<pre>fu.SetData(FEDM_ISC_FU_TMP_DAT_ANT_VAL_C1, (byte)0xAB); // Kapazität 1 fu.SetData(FEDM_ISC_FU_TMP_DAT_ANT_VAL_C2, (byte)0x9F); // Kapazität 2 fu.SendProtocol(0xC2);</pre>
	[0xC3] Get Antenna Values	<pre>string sAntValues; // Puffer für Tuningwerte fu.SendProtocol(0xC3); fu.GetData(FEDM_ISC_FU_TMP_DAT_ANT_VAL, out sAntValues);</pre>
	[0xC4] Set Outputs	<pre>fu.SetData(FEDM_ISC_FU_TMP_DAT_OUT, (byte)1); // Ausgang 1 wird geschaltet fu.SendProtocol(0xC4);</pre>
	[0xC5] Re-Tuning	<pre>fu.SendProtocol(0xC5);</pre>
	[0xC6] Start Tuning	<pre>fu.SendProtocol(0xC6);</pre>
	[0xC8] Store Settings	<pre>fu.SendProtocol(0xC8);</pre>
	[0xC9] Detect	<pre>fu.SendProtocol(0xC9);</pre>
	[0xCA] Set Address	<pre>byte cAdr = 2; // neue Adresse fu.SetData(FEDM_ISC_FU_TMP_DAT_NEW_ADR, cAdr); // neue Adresse für Funktionseinheit fu.SendProtocol(0xCA); // neue Adresse wird wirksam fu.SetData(FEDM_ISC_FU_TMP_DAT_ADR, cAdr); // neue Adresse für nachfolgende // Protokolle übernehmen</pre>
	[0xCB] Set Mode	<pre>fu.SetData(FEDM_ISC_FU_TMP_DAT_MODE, (byte)1); // Mode 1 fu.SendProtocol(0xCB);</pre>
	[0xDC] Detect	<pre>fu.SendProtocol(0xDC);</pre>

¹³ alle Beispiele sind in C# ausgeführt

[Steuerbyte] Protokoll	Beispiel ¹³
ID ISC.ANT.UMUX	[0xDD] Select Channel fu. SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH1, (byte)1); // Ausgang für Eingang 1 auf 1 fu. SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH2, (byte)8); // Ausgang für Eingang 2 auf 8 fu. SendProtocol (0xDD);
	[0xDE] CPU Reset fu. SendProtocol (0xDE);
	[0xDF] Get Firmware Version string sFirmware; // Puffer für Firmware Informationen fu. SendProtocol (0xDF); fu. GetData (FEDM_ISC_FU_TMP_SOFTVER, out sFirmware);
	[0xDC] Detect/Get Power byte[] Power = new byte[5]; // Puffer für Power Informationen byte UMuxState = 0; // Statusbyte der Antwort fu. SetData (FEDM_ISC_FU_TMP_FLAGS, (byte)0); // immer auf 0 fu. SendProtocol (0xDC); fu. GetData (FEDM_ISC_FU_TMP_UMUX_POWER, Power); fu. GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, out UMuxStatus);
	[0xDD] Select Channel byte UMuxState = 0; // Statusbyte der Antwort fu. SetData (FEDM_ISC_FU_TMP_FLAGS, (byte)0); // immer auf 0 fu. SetData (FEDM_ISC_FU_TMP_MUX_OUT_CH1, (byte)1); // selektiere Ausgang 1 fu. SendProtocol (0xDD); fu. GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, out UMuxStatus);
	[0xDE] CPU Reset byte UMuxState = 0; // Statusbyte der Antwort fu. SetData (FEDM_ISC_FU_TMP_FLAGS, (byte)0); // immer auf 0 fu. SendProtocol (0xDE); fu. GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, out UMuxStatus);
	[0xDF] Get Firmware Version byte[] Firmware = new byte[7]; // Puffer für Firmware Informationen byte UMuxState = 0; // Statusbyte der Antwort fu. SetData (FEDM_ISC_FU_TMP_FLAGS, (byte)0); // immer auf 0 fu. SendProtocol (0xDF); fu. GetData (FEDM_ISC_FU_TMP_SOFTVER, out Firmware); fu. GetData (FEDM_ISC_FU_TMP_UMUX_LAST_STATE, out UMuxStatus);

9. Beispiel für die Verwendung von TagHandler-Klassen

```
using OBID;
using OBID.TagHandler;

...
FedmIscReaderReader = new FedmIscReader();
Reader.SetTableSize(10); // max 10 Transponder je Inventory

...
// Verbindungsaufbau mit Leser
Reader.ConnectUSB(0);

...
int back = 0;
int tagDriver = 0; // 9 für MIFARE DESFire
byte BlockSize = 0;
byte[] Data = null;
Dictionary<string, FedmIscTagHandler> TagList;
Dictionary<string, FedmIscTagHandler>.ValueCollection listTagHandler;
FedmIscTagHandler TagHandler = null;

// Inventory command mit Standard-Optionen
TagList = Reader.TagInventory(true, 0x00, 1);

if (TagList.Count > 0)
{
    listTagHandler = TagList.Values;
    foreach (FedmIscTagHandler tagHandler in listTagHandler)
    {
        if (tagHandler != null)
        {
            TagHandler = tagHandler;

            // Transponder selektieren:
            // notwendig für ISO 14443 (tagDriver ev. vorgeben (s. Systemhandbuch zum Leser))
            // optional für ISO 15693
            // nicht für EPC Class 1 Gen 2
            TagHandler = Reader.tagSelect(TagHandler, tagDriver);

            else if (TagHandler is FedmIscTagHandler_ISO15693)
            {
                FedmIscTagHandler_ISO15693 thIso = (FedmIscTagHandler_ISO15693)TagHandler;

                // Daten lesen und wieder zurückschreiben
                back = thIso.ReadMultipleBlocksWithSecStatus(4, 4, out BlockSize, out Data);
                back = thIso.WriteMultipleBlocks(4, 4, 4, Data);
            }
            else if (TagHandler is FedmIscTagHandler_ISO14443_4_MIFARE_DESFire)
            {
                FedmIscTagHandler_ISO14443_4_MIFARE_DESFire thIso =
                    (FedmIscTagHandler_ISO14443_4_MIFARE_DESFire)TagHandler;

                // Versionsinformationen auslesen
                // das interne Interface IFlexSoftCrypto wird verwendet
                back = thIso.IFlexSoftCrypto.GetVersion((byte)0, out Data);
            }
        }
    }
}
```

```
else if(TagHandlerisFedmIscTagHandler_EPC_Class1_Gen2)
{
    FedmIscTagHandler_EPC_Class1_Gen2 thGen2 = (FedmIscTagHandler_EPC_Class1_Gen2)TagHandler;

    // neue EPC auf Transponder schreiben (ohne Passwort)
    thGen2.WriteEPC("0102030405060708090A0B0C", "");
}
}
```

10. Anhang

10.1. Liste der Fehlercodes

Alle nachfolgend aufgelisteten Konstanten sind in der Struktur Fedm deklariert.

Fehler-Konstante	Wert	Beschreibung
MODIFIED	1	Kein Fehler aufgetreten, Container wurde modifiziert
OK	0	Kein Fehler aufgetreten
ERROR_BLOCK_SIZE	-101	Die Blockgröße in der Zugriffskonstanten ist falsch
ERROR_BIT_BOUNDARY	-102	Die Bitgrenze in der Zugriffskonstanten ist falsch
ERROR_BYTE_BOUNDARY	-103	Die Bytegrenze in der Zugriffskonstanten ist falsch
ERROR_ARRAY_BOUNDARY	-104	Die Arraygrenze eines Datencontainers wurde überschritten
ERROR_BUFFER_LENGTH	-105	Die Länge des Datenpuffers ist nicht ausreichend
ERROR_PARAMETER	-106	Ein Übergabeparameter ist unbekannt
ERROR_STRING_LENGTH	-107	Der übergebene String ist zu lang
ERROR_ODD_STRING_LENGTH	-108	Der übergebene String enthält eine ungerade Anzahl Zeichen
ERROR_NO_DATA	-109	Keine Daten im Protokoll
ERROR_NO_READER_HANDLE	-110	Kein Reader-Handle gesetzt
ERROR_NO_PORT_HANDLE	-111	Kein Port-Handle gesetzt
ERROR_UNKNOWN_CONTROL_BYTE	-112	Unbekanntes Steuerbyte
ERROR_UNKNOWN_MEM_ID	-113	Unbekannte Speicher-ID
ERROR_UNKNOWN_POLL_MODE	-114	Unbekannter Pollmode
ERROR_NO_TABLE_DATA	-115	Keine Daten in einer Tabelle
ERROR_UNKNOWN_ERROR_CODE	-116	Unbekannter Fehlercode
ERROR_UNKNOWN_COMMAND	-117	Unbekanntes Kommando
ERROR_UNSUPPORTED	-118	Funktion wird nicht unterstützt
ERROR_NO_MORE_MEM	-119	Kein Programmspeicher mehr verfügbar
ERROR_NO_READER_FOUND	-120	Kein Leser gefunden
ERROR_NULL_POINTER	-121	Der übergebene Zeiger ist NULL
ERROR_UNKNOWN_READER_TYPE	-122	Unbekannter Lesertyp
ERROR_UNSUPPORTED_READER_TYPE	-123	Funktion kann für diesen Lesertyp nicht

Fehler-Konstante	Wert	Beschreibung
		verwendet werden
ERROR_UNKNOWN_TABLE_ID	-124	Unbekannte Tabellenkonstante
ERROR_UNKNOWN_LANGUAGE	-125	Unbekannte Sprachenkonstante
ERROR_NO_TABLE_SIZE	-126	Tabelle hat Größe 0
ERROR_SENDBUFFER_OVERFLOW	-127	Sendepuffer ist voll
ERROR_VERIFY	-128	Daten sind ungleich
ERROR_OPEN_FILE	-129	Fehler beim Öffnen der Datei
ERROR_SAVE_FILE	-130	Fehler beim Speichern der Datei
ERROR_UNKNOWN_TRANSPONDER_TYPE	-131	unbekannter Transpondertyp
ERROR_READ_FILE	-132	Fehler beim Lesen der Datei
ERROR_WRITE_FILE	-133	Fehler beim Schreiben in Datei
ERROR_UNKNOWN_EPC_TYPE	-134	unbekannter EPC-Typ
ERROR_UNSUPPORTED_PORT_DRIVER	-135	Funktion unterstützt nicht den aktiven Kommunikations-Treiber
ERROR_UNKNOWN_ADDRESS_MODE	-136	unbekannter Adressmodus
ERROR_ALREADY_CONNECTED	-137	Leserobjekt ist bereits mit einem Port verbunden
ERROR_NOT_CONNECTED	-138	Leserobjekt ist nicht mit einem Port verbunden
ERROR_NO_MODULE_HANDLE	-139	Kein Handle eines Lesermoduls gefunden
ERROR_EMPTY_MODULE_LIST	-140	Die Modulliste ist leer
ERROR_MODULE_NOT_FOUND	-141	Modul nicht in Modulliste gefunden
ERROR_DIFFERENT_OBJECTS	-142	Laufzeitobjekte sind nicht identisch
ERROR_NOT_AN_EPC	-143	Seriennummer des Transponders ist keine EPC
ERROR_OLD_LIB_VERSION	-144	Veraltete Bibliothek erkannt (Fehlercode für Java/.NET-Bibliotheken)
ERROR_WRONG_READER_TYPE	-145	Falscher Lesertyp
ERROR_CRC	-146	CRC-Fehler in Datei
ERROR_CFG_BLOCK_PREVIOUSLY_NOT_READ	-147	Konfigurationsblock muss erst gelesen werden
ERROR_UNSUPPORTED_CONTROLLER_TYPE	-148	Nicht unterstützter Controller-Typ
ERROR_VERSION_CONFLICT	-149	Versionskonflikt mit einer oder mehrerer abhängiger Bibliotheken
ERROR_UNSUPPORTED_NAMESPACE	-150	Namespace wird vom Lesertyp nicht unterstützt.
ERROR_TASK_STILL_RUNNING	-151	Asynchroner Task ist noch am laufen
ERROR_TAG_HANDLER_NOT_IDENTIFIED	-152	TagHandler-Typ konnte nicht identifiziert werden
ERROR_UNVALID_IDD_LENGTH	-153	Längenangabe für IDD ist außerhalb des

Fehler-Konstante	Wert	Beschreibung
		zulässigen Bereichs
ERROR_UNVALID_IDD_FORMAT	-154	Formateinstellung für IDD ist falsch
ERROR_UNKNOWN_TAG_HANDLER_TYPE	-155	Unbekannter TagHandler-Typ
ERROR_UNSUPPORTED_TRANSPONDER_TYPE	-156	Transponder- oder Chiptyp wird nicht unterstützt
ERROR_CONNECTED_WITH_OTHER_MODULE	-157	Nur TCP/IP: eine Verbindung zum gleichen Leser wurde bereits von einem anderen Modul aufgebaut.
ERROR_INVENTORY_NO_TID_IN_UID	-158	Inventory mit Rückgabe von UID = EPC + TID, aber TID fehlt
ERROR_NO_XML_FILE	-200	Datei ist kein XML-Dokument
ERROR_NO_OBID_TAG	-201	Datei enthält kein Element 'OBID'
ERROR_NO_CHILD_TAG	-202	Kein Sub-Element vorhanden
ERROR_TAG_NOT_FOUND	-203	Element nicht in Dokument
ERROR_DOC_NOT_WELL_FORMED	-204	XML-Dokument nicht wohlgeformt
ERROR_NO_TAG_VALUE	-205	Kein Element-Inhalt vorhanden
ERROR_NO_TAG_ATTRIBUTE	-206	Kein Attribute vorhanden
ERROR_DOC_FILE_VERSION	-207	Ungültige Dokument-Version
ERROR_DOC_FILE_FAMILY	-208	Dokument für andere Leserfamilie
ERROR_DOC_FILE_TYPE	-209	Falscher Dateityp
ERROR_WRONG_CONTROLLER_TYPE	-210	Falscher Controller-Typ
ERROR_WRONG_MEM_BANK_TYPE	-211	Falsche Speicherbank

10.2. Unterstützte OBID® Leser

Leser	Anmerkungen
ID ISC.M02	
ID ISC.MR/PR100	alle Schnittstellen
ID ISC.PRH100/PRH101	alle Schnittstellen
ID ISC.MR/PR101	alle Schnittstellen
ID ISC.MR102	alle Schnittstellen
ID ISC.PRH102	alle Schnittstellen
ID ISC.PRHD102	alle Schnittstellen
ID ISC.MR200	alle Schnittstellen
ID ISC.LR200	
ID ISC.LR1002	alle Schnittstellen
ID ISC.LR2000	alle Schnittstellen
ID ISC.LR2500-A	alle Schnittstellen
ID ISC.LR2500-B	alle Schnittstellen
ID ISC.MU02	
ID ISC.MRU102	alle Schnittstellen
ID ISC.MRU200	alle Schnittstellen
ID ISC.LRU1000	alle Schnittstellen
ID ISC.LRU2000	alle Schnittstellen
ID ISC.LRU3000	alle Schnittstellen
ID CPR.02	
ID CPR.M02	alle Schnittstellen
ID CPR.04	alle Schnittstellen
ID CPR30.xx	alle Schnittstellen
ID CPR40.xx	alle Schnittstellen
ID CPR44.xx	alle Schnittstellen
ID CPR46.xx	alle Schnittstellen
ID CPR50.xx	alle Schnittstellen
ID CPR52.xx	alle Schnittstellen
ID MAX50.xx	alle Schnittstellen

10.3. Unterstützte Transponder

Die Unterstützung von Transpondertypen ist zunächst einmal abhängig von der im Leser implementierten Firmware. Genaue Auskunft erhält man im Systemhandbuch des Lesers.

Nachfolgend sind die Transpondertypen aufgeführt, die zum Zeitpunkt der Entwicklung der Bibliothek bekannt waren.

Transponder	Typkennung	Anmerkung
I-CODE 1	0x00	HF-Transponder
Tag-it	0x01	HF-Transponder
ISO15693	0x03	HF-Transponder
ISO14443-A	0x04	HF-Transponder
ISO14443-B	0x05	HF-Transponder
EPC	0x06	HF-Transponder (EPC-Typen 1..4)
I-CODE UID	0x07	HF-Transponder
Jewel	0x08	HF-Transponder
ISO 18000-3M3	0x09	HF-Transponder
STMicroelectronics SR176	0x0A	HF-Transponder
STMicroelectronics SRIxx	0x0B	HF-Transponder
Microchip MCRFxxx	0x0C	HF-Transponder
Innovatron (ISO 14443B')	0x10	HF-Transponder
ASK CTx	0x11	HF-Transponder
ISO18000-6-A	0x80	UHF-Transponder
ISO18000-6-B	0x81	UHF-Transponder
EM4222	0x83	UHF-Transponder
EPC Class1 Generation 2	0x84	UHF-Transponder
EPC Class0/0+	0x88	UHF-Transponder
EPC Class1 Generation 1	0x89	UHF-Transponder

10.4. TCP-Status

Informationen zum einzelnen Status finden sich im Internet unter dem Suchbegriff *Transmission Control Protocol*

TCP-Status	Wert
CLOSED	1
LISTEN	2
SYN_SENT	3
SYN_RCVD	4
ESTABLISHED	5
FIN_WAIT1	6
FIN_WAIT2	7
CLOSE_WAIT	8
CLOSING	9
LAST_ACK	10
TIME_WAIT	11

10.5. Liste der Konstanten

Alle aufgeführten Konstanten sind in der Struktur FedmISCRConst definiert.

10.5.1. Allgemeine Konstanten

Konstante	Beschreibung
TYPE_...	Lesertyp entsprechend dem Protokoll [0x65] Software Version
NAME_...	Leserbezeichnung entsprechend dem Systemhandbuch
TR_TYPE_...	Kennung des Transpondertyps entsprechend dem Anhang im Systemhandbuch
EPC_TYPE_...	Kennung der EPC-Typen; (EPC = Electronic Product Code)
FU_TYPE_...	Kennung der Funktionseinheiten (s. Klasse FedmIsFunctionUnit)

10.5.2. Konstanten für tableID

Konstante	Beschreibung
BRM_TABLE	Tabellen-ID für BRM-Tabelle
ISO_TABLE	Tabellen-ID für ISO-Tabelle

10.5.3. Konstanten für dataID

Konstante	Beschreibung/Verwendung																																
DATA_TRTYPE	Transpondertyp																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex		X		X		
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex		X		X																													
DATA_SNR	Seriennummer																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X		X	X	SetTableData					X	X	FindTableIndex					X	X
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X		X	X																											
SetTableData					X	X																											
FindTableIndex					X	X																											

Konstante	Beschreibung/Verwendung																																
DATA_RxDB DATA_RxDB_EPC_BANK DATA_RxDB_TID_BANK DATA_RxDB_RES_BANK <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	Datenblöcke aus Empfangsprotokoll Hinweis: GetTableData und SetTableData (nur ISO-Table) für Datenblöcke verwenden !! <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData			X			X	FindTableIndex						
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X			X																											
SetTableData			X			X																											
FindTableIndex																																	
DATA_TxDB DATA_TxDB_EPC_BANK DATA_TxDB_TID_BANK DATA_TxDB_RES_BANK <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	Datenblöcke für Sendeprotokoll Hinweis: GetTableData und SetTableData (nur ISO-Table) für Datenblöcke verwenden !! <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData			X			X	FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X			X																											
SetTableData			X			X																											
FindTableIndex																																	
DATA_TIMER <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Timer-Wert aus Empfangsprotokoll [0x22] Read Buffer. Alternativ kann auch die Methode getReaderTime von FedmBrmTableItem verwendet werden. <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X	X			SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
X																																	
	bool	byte	byte[]	uint	long	string																											
GetTableData			X	X																													
SetTableData																																	
FindTableIndex																																	
DATA_RxCB <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	Konfigurations-Datenblock aus Empfangsprotokoll Hinweis: GetTableData und SetTableData für Datenblöcke verwenden !! <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData			X			X	FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X			X																											
SetTableData			X			X																											
FindTableIndex																																	
DATA_TxCB <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	Konfigurations-Datenblock für Sendeprotokoll (nur für ISO-Tabelle) Hinweis: GetTableData und SetTableData für Datenblöcke verwenden !! <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData			X			X	FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X			X																											
SetTableData			X			X																											
FindTableIndex																																	
DATA_AFI <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	AFI aus [0xB0] [0x2B] Get System Information <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData		X		X		X	FindTableIndex		X		X		
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData		X		X		X																											
FindTableIndex		X		X																													
DATA_DSFD	DSFID aus Empfangsdaten [0xB0] [0x01] Inventory																																

Konstante	Beschreibung/Verwendung																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData		X		X		X	FindTableIndex		X		X		
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData		X		X		X																											
FindTableIndex		X		X																													
DATA_TRINFO	Transponder Info (nur für ISO14443-4 Transponder) aus Empfangsdaten [0xB0] [0x01] Inventory																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex		X		X		
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex		X		X																													
DATA_OPTINFO	Optional Info (nur für ISO14443A Transponder) aus Empfangsdaten [0xB0] [0x01] Inventory																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
DATA_PROTOINFO	Protocol Info (nur für ISO14443B Transponder) aus Empfangsdaten [0xB0] [0x01] Inventory																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
DATA_FSCI	Max. Frame Size (nur für ISO14443-4 Transponder) aus Empfangsdaten [0xB2] [0x2B] ISO14443-4 Transponder Info																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
DATA_FWI	Frame Waiting Time (nur für ISO14443-4 Transponder) aus Empfangsdaten [0xB2] [0x2B] ISO14443-4 Transponder Info																																
<table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	

Konstante	Beschreibung/Verwendung																																
<div>DATA_DSI</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<div>Devisor Send Integer (nur für ISO14443-4 Transponder) aus Empfangsdaten [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
<div>DATA_DRI</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<div>Devisor Receive Integer (nur für ISO14443-4 Transponder) aus Empfangsdaten [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
<div>DATA_NAD</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<div>Node Address (nur für ISO14443-4 Transponder) aus Empfangsdaten [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
<div>DATA_CID</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<div>Card Identifier (nur für ISO14443-4 Transponder) aus Empfangsdaten [0xB2] [0x2B] ISO14443-4 Transponder Info</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
<div>DATA_SEC_STATUS</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<div>Security Status aus Empfangsdaten [0xB0] [0x23] Read Multiple Blocks</div> <div>Hinweis: GetTableData und SetTableData für Datenblöcke verwenden !!</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData			X			X	FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X			X																											
SetTableData			X			X																											
FindTableIndex																																	
<div>DATA_BLOCK_SIZE</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<div>Blocksize aus Empfangsdaten [0xB0] [0x23] Read Multiple Blocks</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>SetTableData</td><td></td><td>X</td><td></td><td>X</td><td></td><td>X</td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X			SetTableData		X		X		X	FindTableIndex						
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X																													
SetTableData		X		X		X																											
FindTableIndex																																	

Konstante	Beschreibung/Verwendung																																
<div>DATA_ MEM_SIZE</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Speichergröße aus [0xB0] [0x2B] Get System Information</div> <div><table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X			X																											
SetTableData																																	
FindTableIndex																																	
<div>DATA_ IC_REF</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>IC-Referenz aus [0xB0] [0x2B] Get System Information</div> <div><table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td></tr></table></div>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex		X		X		
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex		X		X																													
<div>DATA_ DB_ADR</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table></div>	BRM-Table	ISO-Table	X		<div>Datenblock-Adresse aus Empfangsprotokoll [0x22] Read Buffer</div> <div><table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
X																																	
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X		X																											
SetTableData																																	
FindTableIndex																																	
<div>DATA_ DBN</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table></div>	BRM-Table	ISO-Table	X		<div>Anzahl Datenblöcke aus Empfangsprotokoll [0x22] Read Buffer</div> <div><table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	byte	byte[]	uint	long	string	GetTableData		X	X		X	X	SetTableData							FindTableIndex						
BRM-Table	ISO-Table																																
X																																	
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X		X	X																											
SetTableData																																	
FindTableIndex																																	
<div>DATA_ IS_BLOCK_SIZE_SET</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Flag, ob Blocksize mit [0xB0] [0x23] Read Multiple Blocks gesetzt wurde</div> <div><table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>SetTableData</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td></tr><tr><td>FindTableIndex</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	byte	byte[]	uint	long	string	GetTableData	X	X	X	X			SetTableData	X	X		X			FindTableIndex	X					
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData	X	X	X	X																													
SetTableData	X	X		X																													
FindTableIndex	X																																
<div>DATA_ IS_SELECTED</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Flag, ob Transponder selektiert ist. Wird mit [0xB0][0x25] Select gesetzt</div> <div><table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>SetTableData</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td></tr><tr><td>FindTableIndex</td><td>X</td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	byte	byte[]	uint	long	string	GetTableData	X	X	X	X			SetTableData	X	X		X			FindTableIndex	X					
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData	X	X	X	X																													
SetTableData	X	X		X																													
FindTableIndex	X																																
<div>DATA_ IS_ISO14443_4_INFO</div> <div><table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table></div>	BRM-Table	ISO-Table		X	<div>Flag, ob Daten mit [0xB2] [0x2B] ISO14443-4 Transponder Info gesetzt wurden</div> <div><table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td>X</td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>SetTableData</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div>		bool	byte	byte[]	uint	long	string	GetTableData	X	X	X	X			SetTableData	X	X		X			FindTableIndex						
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData	X	X	X	X																													
SetTableData	X	X		X																													
FindTableIndex																																	

Konstante	Beschreibung/Verwendung																																
<div>DATA_EPC</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<div>EPC; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory. Der EPC als String mit allen Feldern in der Darstellung: "xx.xxxxxx.xxxxxx.xxxxxx" (Header.DomainManager.ObjectClass.Seriennummer).</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td>X</td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData							FindTableIndex						X
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData			X			X																											
SetTableData																																	
FindTableIndex						X																											
<div>DATA_EPC_TYPE</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td></td><td>X</td></tr></table>	BRM-Table	ISO-Table		X	<div>EPC-Typ; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory. Der EPC-Typ wird aus dem Feld EPC-Header ermittelt.</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X			SetTableData							FindTableIndex		X		X		
BRM-Table	ISO-Table																																
	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData		X	X	X																													
SetTableData																																	
FindTableIndex		X		X																													
<div>DATA_EPC_HEADER</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<div>Feld EPC Header; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td>X</td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData				X		X	SetTableData							FindTableIndex				X		
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData				X		X																											
SetTableData																																	
FindTableIndex				X																													
<div>DATA_EPC_DOMAIN</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<div>Feld EPC-DomainManager; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData					X	X	SetTableData							FindTableIndex					X	X
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData					X	X																											
SetTableData																																	
FindTableIndex					X	X																											
<div>DATA_EPC_OBJECT</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<div>Feld EPC-ObjectClass; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData					X	X	SetTableData							FindTableIndex					X	X
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData					X	X																											
SetTableData																																	
FindTableIndex					X	X																											
<div>DATA_EPC_SNR</div> <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td>X</td></tr></table>	BRM-Table	ISO-Table	X	X	<div>Feld EPC-Seriennummer; (EPC = Electronic Product Code) aus Empfangsdaten [0xB0] [0x01] Inventory</div> <table><tr><th></th><th>bool</th><th>byte</th><th>byte[]</th><th>uint</th><th>long</th><th>string</th></tr><tr><td>GetTableData</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td>X</td><td>X</td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData					X	X	SetTableData							FindTableIndex					X	X
BRM-Table	ISO-Table																																
X	X																																
	bool	byte	byte[]	uint	long	string																											
GetTableData					X	X																											
SetTableData																																	
FindTableIndex					X	X																											

Konstante	Beschreibung/Verwendung																																		
DATA_ DATE <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Datumsfeld aus Empfangsprotokoll [0x22] Read Buffer <table><tr><td></td><td>FelscReaderTime</td><td></td><td></td><td></td><td></td></tr><tr><td>GetReaderTime</td><td>X</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		FelscReaderTime					GetReaderTime	X																						
BRM-Table	ISO-Table																																		
X																																			
	FelscReaderTime																																		
GetReaderTime	X																																		
DATA_ANT_NR <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Antennennummer aus Empfangsprotokoll [0x22] Read Buffer <table><tr><td></td><td>bool</td><td>byte</td><td>byte[]</td><td>uint</td><td>long</td><td>string</td></tr><tr><td>GetTableData</td><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td>X</td><td></td><td>X</td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X	X	X		X	SetTableData							FindTableIndex		X		X				
BRM-Table	ISO-Table																																		
X																																			
	bool	byte	byte[]	uint	long	string																													
GetTableData		X	X	X		X																													
SetTableData																																			
FindTableIndex		X		X																															
FEDM_ISC_DATA_INPUT <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Input-Byte aus Empfangsprotokoll [0x22] Read Buffer <table><tr><td></td><td>bool</td><td>byte</td><td>byte[]</td><td>uint</td><td>long</td><td>string</td></tr><tr><td>GetTableData</td><td></td><td>X</td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X		X	X	X	SetTableData							FindTableIndex								
BRM-Table	ISO-Table																																		
X																																			
	bool	byte	byte[]	uint	long	string																													
GetTableData		X		X	X	X																													
SetTableData																																			
FindTableIndex																																			
FEDM_ISC_DATA_STATE <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Status-Byte aus Empfangsprotokoll [0x22] Read Buffer <table><tr><td></td><td>bool</td><td>byte</td><td>byte[]</td><td>uint</td><td>long</td><td>string</td></tr><tr><td>GetTableData</td><td></td><td>X</td><td></td><td>X</td><td>X</td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData		X		X	X	X	SetTableData							FindTableIndex								
BRM-Table	ISO-Table																																		
X																																			
	bool	byte	byte[]	uint	long	string																													
GetTableData		X		X	X	X																													
SetTableData																																			
FindTableIndex																																			
FEDM_ISC_DATA_MAC_ADR <table><tr><th>BRM-Table</th><th>ISO-Table</th></tr><tr><td>X</td><td></td></tr></table>	BRM-Table	ISO-Table	X		Status-Byte aus Empfangsprotokoll [0x22] Read Buffer <table><tr><td></td><td>bool</td><td>byte</td><td>byte[]</td><td>uint</td><td>long</td><td>string</td></tr><tr><td>GetTableData</td><td></td><td></td><td>X</td><td></td><td></td><td>X</td></tr><tr><td>SetTableData</td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>FindTableIndex</td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>		bool	byte	byte[]	uint	long	string	GetTableData			X			X	SetTableData							FindTableIndex								
BRM-Table	ISO-Table																																		
X																																			
	bool	byte	byte[]	uint	long	string																													
GetTableData			X			X																													
SetTableData																																			
FindTableIndex																																			

10.6. Änderungshistorie

V4.04.01

- Fehlerkorrektur in TagHandler-Klasse **FedmlscTagHandler_EPC_Class1_Gen2**: Korrektur bei der Berechnung der Passwortlänge in den Methoden Kill und Lock
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

V4.04.00

- Unterstützung für 64-Bit .NET Framework
- Neuer Namespace **OBID.ReaderCommand** enthält strukturiert alle Command-Parameter (diese Sammlung ersetzt nicht die Konstanten in **FedmlscReaderID**, sondern wird als bessere, weil übersichtlichere Alternative empfohlen)
- Neue TagHandler-Klasse: **FedmlscTagHandler_ISO18000_3M3**
- Neue Methoden in der TagHandler-Klasse **FedmlscTagHandler_EPC_Class1_Gen2**:
 1. GetTagModelNumber
 2. GetMaskDesignerID
 3. GetMaskDesignerName
 4. IsUidWithTid
 5. IsExtendedPC_W1
 6. GetExtendedProtocolControlW1
- Klasse **FedmlscReader**: Command [0x6E]: Unterstützung für Mode 0x21
- Klasse **FedmBrmTableItem**: Neues Element: class1Gen2XPC_W1 (Extended PC Word 1)
- Klasse **FedmlsoTableItem**: Neues Element: class1Gen2XPC_W1 (Extended PC Word 1)
- Struktur **FedmlscReaderInfo**: Neue Elemente zur Ausgabe der Versionsnummer der Embedded ACC Applikation: *AccEmbAppSwVer* und *AccEmbAppDevVer*
- Bugfix für Command [0x77] Get People Counter Values: Ausgabe von counter2 korrigiert.
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

V4.03.00

- Unterstützung für den neuen Leser: **ID CPR46.xx**
- Unterstützung für neue Transponder: Innovatron (ISO 14443B') und ASK CTx
- Neue TagHandler-Klassen:
 1. **FedmlscTagHandler_ISO14443_Innovatron**
 2. **FedmlscTagHandler_ISO14443_3_ASK_CTx**
- Klasse **FedmlscReader**:
 1. Unterstützung des People Counter im Notification Mode.

- 2. Neue überladene *SetTableSize* Methode zum zusätzlichen Modifizieren der Größe der Datenpuffer für Transponderdaten.
- 3. Neue, überladene Methoden *ConnectCOMM* und *ConnectUSB* für gesicherte Datenübertragung.
- Klasse **FedmBrmTableItem**:
 - 1. fast alle Tabellen-Elemente sind jetzt public
 - 2. Richtungsinformation von People Counter wird gespeichert
 - 3. Neue Datenelemente: IDDT, AFI und DSFID
- Klasse **FedmlsoTableItem**: fast alle Tabellen-Elemente sind jetzt public
- Klasse **FelscReaderTime**: Datumsformat geändert für Kompatibilität mit ISO 8601
- Interface **FedmTaskListener**: neue Methode *OnNewPeopleCounterEvent*
- Bugfix in *ReadCompleteBank* von **FedmlscTagHandler_EPC_Class1_Gen2**: Wiederholung der Daten nach Vielfachen von 166 Bytes
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Umbenennung von Namespaces:

Bisher	Neu
OperatingMode.xxMode.DataSource. MifareAppID	OperatingMode.xxMode.DataSource. Mifare.Classic.AppID
OperatingMode.xxMode.DataSource. MifareKeyAddress	OperatingMode.xxMode.DataSource. Mifare.Classic.KeyAddress
OperatingMode.xxMode.DataSource. MifareKeyType	OperatingMode.xxMode.DataSource. Mifare.Classic.KeyType

Anm: xxMode steht für NotificationMode oder ScanMode

V4.02.00

- Erstes Release für das .NET Framework 4.0
- Verbesserte Threadsicherheit
- Unterstützung für den neuen Leser: **ID ISC.LR1002**
- Klasse **FedmlscReader**:
 - 1. Umbenennung der Methode *GetNonAddressedTagHandler* in *CreateNonAddressedTagHandler*.
 - 2. *Dispose*-Methode hinzugefügt
- **TagHandler-Klasse für ISO 14443-4 Mifare DESFire mit FlexSoft- und SAM-Crypto**: Fehlerkorrekturen in *SetConfiguration* mit Werten 1 und 2 in Parameter option
- **ISO 14443-3 bzw. -4 Transponder**: Optimiertes Select-Verfahren in der Methode *TagSelect*.
- **EPC Class 1 Gen 2**:
 - 1. Unterstützung für non-addressed mode.
 - 2. Fehlermeldung (-158) vor Ausführung von Transponder-Commands im addressed-mode, wenn die TID nicht Bestandteil der UID ist, aber UID = EPC + TID konfiguriert ist.
- **TagHandler-Klasse für EPC Class 1 Gen 2**:

1. ISO-Errorcode wird in der Klasse gespeichert.
 2. Methode GetTidOfUid gibt TID zurück, auch wenn EPC-Länge = 0 ist.
- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
 - Anbindung an Log-Manager

V4.00.07

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Neue Methode in TagHandler-Klasse für EPC Class1 Gen2: Lock mit vereinfachter Parameterfolge

V4.00.03

- Fehlerkorrektur in TagHandler-Klasse für EPC Class1 Gen2 in Methode ReadCompleteBank

V4.00.02

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Keep-Alive-Option für Notification-Tasks, aktiviert in der Hilfsklasse FedmTaskOption
- Prüfung auf doppelte UIDs in der Methode FedmlscReader::TagInventory
- TagHandler für EPC Class1 Gen2: neue Methode ReadCompleteBank
- Fehlerkorrektur in der Methode FedmlscReader::ReadCompleteConfiguration für ID ISC.LR2500-A und ID ISC.LRU3000
- Nur Windows: Abhängigkeit von MFC- und CRT-Bibliotheken nach MS11-025¹⁴

V4.00.01

- Fehlerkorrektur in TagHandler-Klasse für EPC Class1 Gen2 in Methode ReadMultipleBlock: EPC-Bank, TID-Bank und RES-Bank werden unterstützt

V4.00.00

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Unterstützung für den neuen Leser: **ID ISC.LR2500-A**
- Unterstützung für UIDs bis 96 Bytes
- Unterstützung für UHF-Konfiguration UID = EPC + TID

¹⁴ Microsoft Sicherheitsbulletin Artikel-ID: 2538218 vom 14. Juni 2011

- Die Organisation der Leserkonfiguration oberhalb CFG63 ist für den **ID ISC.LRU3000** mit der Firmwareversion V2.0.0 verändert worden und nicht mehr kompatibel zur Vorversion. Diese SDK-Version integriert die notwendigen Anpassungen und ist damit inkompatibel für Firmwareversionen < V2.0.0. Innerhalb der Reader-Klassen findet keine Überprüfung bzgl. der Inkompatibilität statt. Diese muss Applikations-seitig erfolgen.

Die Tabelle gibt Klarheit bzgl. der Kompatibilitäten:

LRU3000-Firmware	SDK-Version	ISOStart-Version	XML-Konfigurationsdatei
< 2.00.00	<= 3.03.01	<= 8.03.02	muss mit ISOStart <= 8.03.02 erstellt sein
>= 2.00.00	>= 4.00.00	>= 9.00.00	muss mit ISOStart >= 9.00.00 erstellt sein

- Parallele Nutzung einer TCP/IP-Verbindung von mehreren Leserobjekten ist nicht mehr möglich. Der neue Fehlercode -157 wird zurückgegeben, wenn ConnectTCP mit der gleichen IP-Adresse und Port ein zweites Mal, aber von einem anderen Reader-Modul aufgerufen wird
- Die Methode Disconnect der Leserklasse **FedmlscReader** hat eine neue Signatur: der Rückgabetyyp void wurde in int geändert, damit ein positiver Rückgabewert anzeigen kann, wenn die Verbindung nicht korrekt geschossen werden konnte. Der positive Rückgabewert entspricht dem letzten Status der Verbindung. Es wird empfohlen, die Codezeilen genau zu untersuchen, die diese Methode aufrufen.
- Neue Methode in der Leserklasse **FedmlscReader**: GetTcpConnectionState
- Unterstützung für [0x74] Input Event im Notification-Mode für die Leser **ID CPR50** und **ID MAX50**
- TagHandler-Klasse für EPC Class1 Gen2 mit neuen Methoden: GetProtocolControl, GetEpcOfUid, GetTidOfUid
- Erweiterung in der Klasse **FedmTaskOption** um Keep-Alive Parameter für den Notification-Task. Die Keep-Alive-Option ist aktiviert. Wenn die Keep-Alive Option aktiviert ist (empfohlen), dann wird eine Unterbrechung der Netzwerkverbindung erkannt, der empfangende Socket geschlossen und anschließend neu initialisiert. Dadurch ist sichergestellt, dass der RFID-Leser nach der Wiederherstellung der Verbindung erneut eine Verbindung aufbauen kann.

V3.03.01

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen
- Unterstützung für die neuen Leser: **ID ISC.MRU102**
- Fehlerkorrekturen in TagHandler-Klassen
- Erstes Release für Windows CE

V3.03.00

- Update der Namespaces und Zugriffskonstanten für Leserkonfigurationen

- Unterstützung für die neuen Leser: **ID ISC.LR2500-B**, **ID ISC.MR102**, **ID CPR30.xx** und **ID ISC.CPR52.xx**

V3.02.09

- Transponderklassen (TagHandler) zur effizienten Programmierung für Transponder unterschiedlicher Standards (ISO 15693, ISO 14443, EPC Class 1 Gen 2) oder mit Herstellerspezifischen Commands
- Namespace OBID.TagHandler enthält alle TagHandler-Klassen
- Neue Methoden in der Leserklasse FedmlscReader:
 1. Synchrones SAM-Command (SendSAMCommand)
 2. TagInventory
 3. TagSelect
 4. GetTagList
 5. GetTagHandler
 6. GetNonAddressedTagHandler
 7. GetSelectedTagHandler

V3.02.04

- Neue Methode SetTableSize in der Reader-Klasse **FedmlscReader** zur Anpassung der Datenpuffer in der Tabelle für Host-Kommandos (ISO_TABLE). Mit dieser Methode kann man die Anzahl der Datenblöcke und die Blockgröße einstellen und die Limitierung auf 256 Datenblöcke aufheben.

V3.02.01

- Unterstützung für HF-Gates mit People Counter **ID ISC.ANT1690/600-GPC** und **ID ISC.ANT1700/740-GPC**
- Neue Konfigurationsparameter im Package de.feig.ReaderConfig.
- Ausgabe von RSSI-Messungen mit UHF-Reader **ID ISC.LRU3x00**.

V3.01.06

- Unterstützung für neue Leser: **ID ISC.LRU3000**, **ID CPR44.xx**, **ID MAX50.xx**
- Neue Konfigurationsparameter im Package OBID.ReaderConfig.
- Option zur Verschlüsselung der Protokolle mit Hilfe der openssl Bibliothek in der Version 0.9.8l (s. [5.3.3. Sicherheit in der Datenübertragung](#)).
- Anpassungen in Klasse FedmlscReaderInfo für neue Optionen mit Protokoll [0x66] Reader Info.

V3.00.13

- Fehlerbehebung in der Klasse **FedmlscReader** ermöglicht dem Garbage Collector die vollständige Speicherfreigabe.
- Neue Methode **GetReport()** in der Klasse **FedmlscReaderInfo**.

V3.00.07

- Unterstützung für neuen Leser: **ID CPR50.xx**.
- Unterstützung für Transpondertyp NXP MIFARE DESFire.
- Neue Methoden in **FedmlscReader** für OBID®*classic-pro* Leser
 - **SendSAMComand**
 - **SendTclApdu**
 - **SendTclPing**
 - **SendTclDeselect**
 - **SendCommandQueue**
 - **AddCommandToQueue**
- Neue Methoden in **FedmlscReader** für alle OBID® Leser
 - **TransferReaderCfgToXmlFile**
 - **TransferXmlFileToReaderCfg**
- Neue Klasse **FedmCprApdu** für asynchrone ISO14443-4 T=CL Protokolle mit OBID®*classic-pro* Lesern.
- Neue Klasse **FedmCprCommandQueue** für OBID®*classic-pro* Leser zum asynchronen Ausführen von [0xBC] Command Queue.
- Änderungen in der Leserklasse **FedmlscReader**:
 - Die Methoden **ConnectUSB** und **ConnectTCP** führen intern ein **ReadReaderInfo** aus und lesen damit wichtige Eigenschaften vom Leser aus.
 - Die Methode **ConnectCOMM** kann (empfohlen), gesteuert durch einen neuen Parameter, intern ein **FindBaudRate** und, bei erfolgreichem Detekt des Lesers, ein **ReadReaderInfo** ausführen. Dadurch wird der Lesertyp intern gesetzt.
 - Die Methode **SendProtocol(0x72)** verwendet intern modifizierte Definitionen der Konstanten **FEDM_ISC_TMP_0x72_OUT_TYPE_1...FEDM_ISC_TMP_0x72_OUT_TYPE_8**: Bisher adressierten diese Konstanten ein Bit. Jetzt werden drei Bits adressiert. Daraus resultiert, dass der OUT-TYPE 'Relay' jetzt mit 0x04 und nicht mehr mit 0x01 angegeben werden muss. Siehe auch [8.1. Grundkommandos für den Leser](#). Diese Änderung ist für alle Lesertypen anzuwenden, die das Protokoll [0x72] Set Output unterstützen.

V3.00.00

- Unterstützung für neue Leser: **ID ISC.MRU200**, **ID ISC.PRHD102** und **ID CPR40.xx**.

- Folgende ältere Leser werden nicht mehr unterstützt: ID ISC.M01 und ID ISC.LR100.
- Unterstützung für UHF-Multiplexer **ID ISC.ANT.UMUX**.
- Unterstützung für Transpondertyp EPC Class1 Gen2 HF.
- Automatisches Detektieren von Versionskonflikten mit abhängigen Bibliotheksdateien.
- Neue Methoden in **FedmlscReader**
 - ApplyConfiguration
 - ReadCompleteConfiguration
 - WriteCompleteConfiguration
 - ResetCompleteConfiguration
 - ReadReaderInfo
- Neue Klasse **FedmlscReaderInfo** sammelt wichtige Informationen zum angeschlossenen Leser.
- Zur Verbesserung der Übersichtlichkeit der Zugriffskonstanten für Leserkonfigurationen wurden diese im Namespace OBID.ReaderConfig strukturiert. Dadurch entfallen die Strukturen FedmlscReaderID_MR200, FedmlscReaderID_LR200, FedmlscReaderID_LR2000, FedmlscReaderID_LRU1000, FedmlscReaderID_LRU2000 sowie die Zugriffskonstanten für OBID *i-scan*® Short- und Mid-Range Leser und OBID® *classic-pro* Leser in der Struktur FedmlscReaderID.
- Neue überladene Methoden Get/SetConfigPara zur Modifikation der im Namespace OBID.ReaderConfig organisierten Konfigurationsparameter.
- Das Schreiben der Leserkonfiguration ist nur noch für Konfigurationsblöcke möglich, die zuvor gelesen wurden, mit der Ausnahme, dass nach der Übernahme von Konfigurationsparametern aus einer XML-Datei das Schreiben in den Leser immer möglich ist.

V2.05.01

- Modifizierte Lizenzbestimmung
- Unterstützung für den Leser ID ISC.LRU2000. Die Konfigurationskonstanten sind im Interface **FedmlscReaderID_LRU2000** zusammengestellt
- Erweiterungen für den UHF-Leser ID ISC.LRU1000 für die Konfiguration im Interface **FedmlscReaderID_LRU1000**
- Neue Methoden in der Leserklasse **FedmlscReader** zur Unterstützung von asynchronen Tasks (nur ab .NET 2.0).
- Neues Interface **FedmTaskListener** (nur ab .NET 2.0).
- Neue Propertyklasse **FedmTaskOption** (nur ab .NET 2.0).

V2.04.00

- Unterstützung für .NET 2.0

- Signierung der Assembly-Dateien OBIDISC4NETnative.DLL und OBIDISC4NET.DLL für .NET 2.0 mit einem starken Namen
- Neue allgemeine Konstanten für den UHF-Leser LRU1000:

Bezeichner	Kommentar
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN1_MASK_LGT	Konstanten für Selection Mask in der Konfiguration für den Transpondertyp EPC Class 1 Gen 1
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN1_MASK_START_PTR	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN1_MASK	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_LGT	Konstanten für Selection Mask in der Konfiguration für den Transpondertyp EPC Class 1 Gen 2
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MODE	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MODE_TRUNC	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MODE_BANK	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_START_PTR	
FEDM_ISC_LRU1000_EE_SELmask_EPC_CL1_GEN2_MASK_MSB	
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK_LGT	Konstanten für Selection Mask in der Konfiguration für den Transpondertyp ISO18000-6-B
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK_MODE	
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK_START_PTR	
FEDM_ISC_LRU1000_EE_SELmask_ISO18000_6_B_MASK	

V2.03.05

- Unterstützung für den Leser ID ISC.LR2000. Die Konfigurationskonstanten sind in der Struktur **FedmlscReaderID_LR2000.cs** zusammengestellt
- Erweiterungen für den UHF-Leser ID ISC.LRU1000 für die Konfiguration in der Struktur **FedmlscReaderID_LRU1000.cs**
- Unterstützung für neue Transponder-Typen: HF-Transponder Innovision Jewel und UHF-Transponder EPC Class0/0+
- Erweiterungen für ISO 14443A Transponder:
 1. [0xB2][0x30] Mifare Value Commands
 2. [0xB0][0x25] Select unterstützt Card Information
- Neue Kommunikationsfunktionen in **FedmlscReader**:
 1. sendProtocol (byte cmdByte, string RequestData, out string ResponseData)
 2. sendTransparent (string SndProtocol, bool CalcChecksum, out string RecProtocol)
- Unterstützung für neues Protokoll: [0x72] Set Output:
- Neue allgemeine Konstanten:

Bezeichner	Kommentar
FEDM_ISC_TMP_B0_MODE_CINF	Flag Card Information im Mode-Byte für [0xB0][0x25] Select
FEDM_ISC_TMP_B0_MODE_WR_NE	Flag Write-Erase im Mode-Byte für [0xB0][0x24] Write Multiple Blocks

Bezeichner	Kommentar
FEDM_ISC_TMP_B0_RSP_FORMAT	Format Byte im Antwortprotokoll von [0xB0][0x25] Select, wenn das CINF-Flag gesetzt ist
FEDM_ISC_TMP_B2_REQ_MF_CMD	Parameter für [0xB2][0x30] Mifare Value Commands
FEDM_ISC_TMP_B2_REQ_OP_VALUE	
FEDM_ISC_TMP_B2_REQ_DEST_ADR	
FEDM_ISC_TMP_ADV_BRM_TRDATA2	2. Byte von TR-DATA im Antwortprotokoll von [0x22] Read Bufer
FEDM_ISC_TMP_ADV_BRM_TRDATA2_...	Flags im 2. Byte von TR-DATA im Antwortprotokoll von [0x22] Read Bufer
FEDM_ISC_TMP_0x72_OUT...	Konstanten für [0x72] Set Output

- Geänderte allgemeine Konstanten:

Alter Bezeichner	Neuer Bezeichner
FEDM_ISC_TMP_ADV_BRM_TRDATA1	FEDM_ISC_TMP_ADV_BRM_TRDATA
FEDM_ISC_TMP_ADV_BRM_TRDATA1_...	FEDM_ISC_TMP_ADV_BRM_TRDATA_...

V2.03.02

- Erweiterungen für den Leser ID ISC.MR200
- Fehlerkorrektur in FedmIsCReaderConst. Alle Applikationen mit einem Verweis auf OBIDISC4NET müssen neu kompiliert werden!

V2.03.00

- Unterstützung für USB-Leser mit neuen Protokollen
- Unterstützung für neuen Leser: ID ISC.MR200
- Erweiterungen für den Leser ID ISC.LRU1000
- Unterstützung für die Funktionseinheiten ID ISC.ANT.MUX und ID ISC.DAT
- Unterstützung neuer UHF Transpondertypen
- Neue Funktionen zum Versenden von Protokollen: SendProtocol und SendTransparent
- Integration neuer Protokolle
- Änderung in Aufrufparametern der Funktionen:

Klasse	alte Funktion	neue Funktion
FedmBrmTableItem	GetData(string, out int)	GetData(string, out uint)
FedmBrmTableItem	GetData(long, out int)	GetData(long, out uint)
FedmBrmTableItem	SetData(string, int)	SetData(string, uint)
FedmBrmTableItem	SetData(long, int)	SetData(long, uint)
FedmIsoTableItem	GetData(string, out int)	GetData(string, out uint)

Klasse	alte Funktion	neue Funktion
FedmIsoTableItem	GetData(long, out int)	GetData(long, out uint)
FedmIsoTableItem	SetData(string, int)	SetData(string, uint)
FedmIsoTableItem	SetData(long, int)	SetData(long, uint)

V1.00.00

- Erste Version