

C++ Class Library

ID FEDM

Version 4.06.02

Part C.MAX

Software-Support for OBID[®] myAXXESS Reader

Operating System	Target		Notes
	32-Bit	64-Bit	
Windows XP	X	(X)	with 64-Bit OS: only with 32-Bit Runtime Environment
Windows Vista / 7 / 8	X	X	
Windows CE	X	-	
Linux	X	(X)	with 64-Bit OS: only with 32-Bit Runtime Environment
Apple Mac OS X	-	-	On request

Note

© Copyright 2009-2014 by FEIG ELECTRONIC GmbH
Lange Straße 4
D-35781 Weilburg-Waldhausen
eMail: obid-support@feig.de

This manual supercedes all previous editions.
The information contained in this manual is subject to change without notice.

Copying of this document, and giving it to others and the use or communication of the contents thereof, are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design.

The information contained in this manual has been gathered with all due care and to the best of our knowledge. FEIG ELECTRONIC GmbH assumes no liability for the accuracy and completeness of the data in this manual. In particular, FEIG ELECTRONIC GmbH cannot be held liable for consequential damages resulting from inaccurate or incomplete information. Since even with our best efforts this document may still contain mistakes, please contact us should you find any errors.

FEIG ELECTRONIC GmbH assumes no responsibility for the use of any information contained in this manual and makes no representation that they free of patent infringement. FEIG ELECTRONIC GmbH does not convey any license under its patent rights nor the rights of others.

The installation instructions given in this manual are based on advantageous boundary conditions. FEIG ELECTRONIC GmbH does not give any guarantee promise for perfect function of an OBID®-system in cross surroundings.

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

Linux® is a registered Trademark of Linus Torvalds.

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.

Licensing agreement for use of the software

This is an agreement between you and FEIG ELECTRONIC GmbH (hereafter "FEIG") for use of the ID FEDM C++ Class Library and the present documentation, hereafter called licensing material. By installing and using the software you agree to all terms and conditions of this agreement without exception and without limitation. If you are not or not completely in agreement with the terms and conditions, you may not install the licensing material or use it in any way. This licensing material remains the property of FEIG ELECTRONIC GmbH and is protected by international copyright.

§1 Object and scope of the agreement

1. FEIG grants you the right to install the licensing material provided and to use it under the following conditions.
2. You may install all components of the licensing material on a hard disk or other storage medium. The installation and use may also be done on a network fileserver. You may create backup copies of the licensing material.
3. FEIG grants you the right to use the documented program library for developing your own application programs or program libraries, and you may sell the runtime files `FedmlscMyAxxessVCxx.dll`¹ and `libFedmlscMyAxxess.so.x.y.z`² without licensing fees under the stipulation that these application programs or program libraries are used to control devices and/or systems which are developed and/or sold by FEIG.

§2. Protection of the licensing material

1. The licensing material is the intellectual property of FEIG and its suppliers. It is protected in accordance with copyright, international agreements and relevant national statutes where it is used. The structure, organization and code of the software are a valuable business secret and confidential information of FEIG and its suppliers.
2. You agree not to change, modify, translate, reverse develop, decompile, disassemble the program library or the documentation or in any way attempt to discover the source code of this software.
3. To the extent that FEIG has applied protection marks, such as copyright marks and other legal restrictions in the licensing material, you agree to keep these unchanged and to use them unchanged in all complete or partial copies which you make.
4. The transmission of licensing material in part or in full is prohibited unless there is an explicit agreement to the contrary between you and FEIG. Application programs or program libraries which are created and sold in accordance with §1 Par. 3 of this Agreement are excepted.

§3 Warranty and liability limitations

1. You agree with FEIG that it is not possible to develop EDP programs such that they are error-free for all application conditions. FEIG explicitly makes you aware that the installation of a new program can affect already existing software, including such software that does not run at the same time as the new software. FEIG assumes no liability for direct or indirect damages, for consequential damages or special damages, including lost profits or lost savings. If you want to ensure that no already installed program will be affected, you should not install the present software.
2. FEIG explicitly notes that this software makes it possible for irreversible settings and adaptations to be made on devices which could destroy these devices or render them unusable. FEIG assumes no liability for such actions, regardless of whether they are carried out intentionally or unintentionally.
3. FEIG provides the software „as is“ and without any warranty. FEIG cannot guarantee the performance or the results you obtain from using the software. FEIG assumes no liability or guarantee that the protection rights of third parties are not violated, nor that the software is suitable for a particular purpose.
4. FEIG call explicit attention the licensed material is not designed with components and testing for a level of reliability suitable for use in or in connection with surgical implants or as critical components in any life support systems whose failure to perform can reasonably be expected to cause significant injury to a human.
To avoid damage, injury, or death, the user or application designer must take reasonably prudent steps to protect against system failures.

¹ xx represents the version number of the dependent MFC library

² x.y.z represents the actual version number

§4 Concluding provisions

1. This Agreement contains the complete licensing terms and conditions and supercedes any prior agreements and terms. Changes and additions must be made in writing.
2. If any provision this agreement is declared to be void, or if for any reason is declared to be invalid or of no effect, the remaining provisions shall be in no manner affected thereby but shall remain in full force and effect. Both parties agree to replace the invalid provision with one which comes closest to its original intention.
4. This agreement is subject to the laws of the Federal Republic of Germany. Place of jurisdiction is Frankfurt a. M.

Contents:

Licensing agreement for use of the software	3
Contents:	5
1. Overview.....	8
1.1. Principles.....	8
1.2. Class-Library	9
2. Revisions since the previous version.....	11
2.1. Limitation of Release-Version	11
3. Installation.....	12
3.1. 32- and 64-Bit Windows XP/Vista/7/8.....	12
3.2. Windows CE	13
3.3. 32- and 64-Bit Linux	13
3.4. Source Code.....	14
3.5. Dependencies.....	14
4. Integration into application projects.....	15
5. Installation on the target computer	16
5.1. 32-Bit Libraries on a 32- and 64-Bit Windows.....	16
5.2. 64-Bit Libraries on a 64-Bit Windows	17
5.3. 32-Bit Windows CE	18
5.4. 32- and 64-Bit Linux	19
6. Class description.....	20
6.1. FedmIscMyAxxessReader	20
6.1.1. Implemented tables	20
6.1.2. Methods (public)	21
6.1.2.1. FedmIscMyAxxessReader	21
6.1.2.2. GetReaderObject.....	22
6.1.2.3. GetLastError	23
6.1.2.4. GetErrorText	23
6.1.2.5. GetIDDLenght / GetIDDDFormat	24
6.1.2.6. SetDateFormat	25
6.1.2.7. GetDateFormat.....	26
6.1.2.8. ClearTable	27
6.1.2.9. SerializeTableIn	28

6.1.2.10. SerializeTableOut	29
6.1.2.11. StartEventHandler	30
6.1.2.12. StopEventHandler.....	32
6.1.2.13. AddTableItem (for native C/C++ interface)	33
6.1.2.14. AddTableItem (for 'naturally speaking' interface).....	34
6.1.2.15. GetTableItem (for native C/C++ interface).....	35
6.1.2.16. GetTableItem (for 'naturally speaking' interface)	36
6.1.2.17. RemoveTableItem	37
6.1.2.18. ModifyTableItem (for native C/C++ interface)	38
6.1.2.19. ModifyTableItem (for 'naturally speaking' interface)	39
6.1.2.20. WriteTables	40
6.1.2.21. ReadTable	41
6.2. Working with the Reader classes	42
6.2.1. Important initializations	42
6.2.2. Example 1: Transfer tables into Reader using the C++ API	43
6.2.3. Example 2: Transfer tables into Reader using serialization of CSV-File.....	45
7. Appendix	46
7.1. List of constants	46
7.1.1. Internal Constants.....	46
7.1.2. Table Constants.....	46
7.1.3. Event Constants	46
7.1.4. IDD Format Constants	47
7.1.5. Date Format Constants.....	47
7.1.6. Filetype Constants for Serialization.....	47
7.2. Conventions for 'naturally speaking' interface	48
7.2.1. Timezone Table Item Conventions.....	48
7.2.2. Holiday Table Item Conventions	49
7.2.3. Access Table Item Conventions.....	49
7.2.4. Event Table Item Conventions	50
7.3. CSV-File structure	51
7.3.1. Access Table	51
7.3.2. Timezone Table	51
7.3.3. Holiday Table	52
7.3.4. Event Table.....	52
7.4. XML-File structure.....	53
7.4.1. Access Table	53
7.4.2. Timezone Table	53
7.4.3. Holiday Table	54
7.4.4. Event Table.....	54
7.5. Revision history	55

Notes concerning the documentation for this library

This manual describes a software library which is also available as annotated source code. For this reason we have limited the documentation to what is absolutely necessary for understanding the functionality and use of the classes. It is assumed that the user of this library reads the source code and becomes familiar with the details using this document, the header files and the included comments.

To understand the internal program sequences you will also have to refer to the system manuals for whichever OBID® readers and function libraries you are using.

FEIG ELECTRONIC GmbH does not repeat the same information about OBID® readers in different manuals or use cross-references to certain pages in a different document. This is necessary due to the constant updating of manuals, and it prevents confusion caused by information in out-of-date documents. The user of this library is therefore well advised to check regularly that he has the most current manuals. If not, these can of course be obtained whenever needed from FEIG ELECTRONIC GmbH.

Important notes:

You are only permitted to use this library if you have first agreed to the license conditions on the back of this page.

Anyone is free to modify source code. Therefore you should work only with libraries you have received directly from FEIG ELECTRONIC GmbH. In any case further transmission of the source code is prohibited.

1. Overview

This documentation is the third part of the documentation for the ID FEDM Class Library and describes the API for OBID® myAXXESS Readers. The concept and base classes are described in document number H10102-xe-ID-B and the reader classes for OBID i-scan® and OBID® *classic-pro* Readers in document number H10202-xe-ID-B.

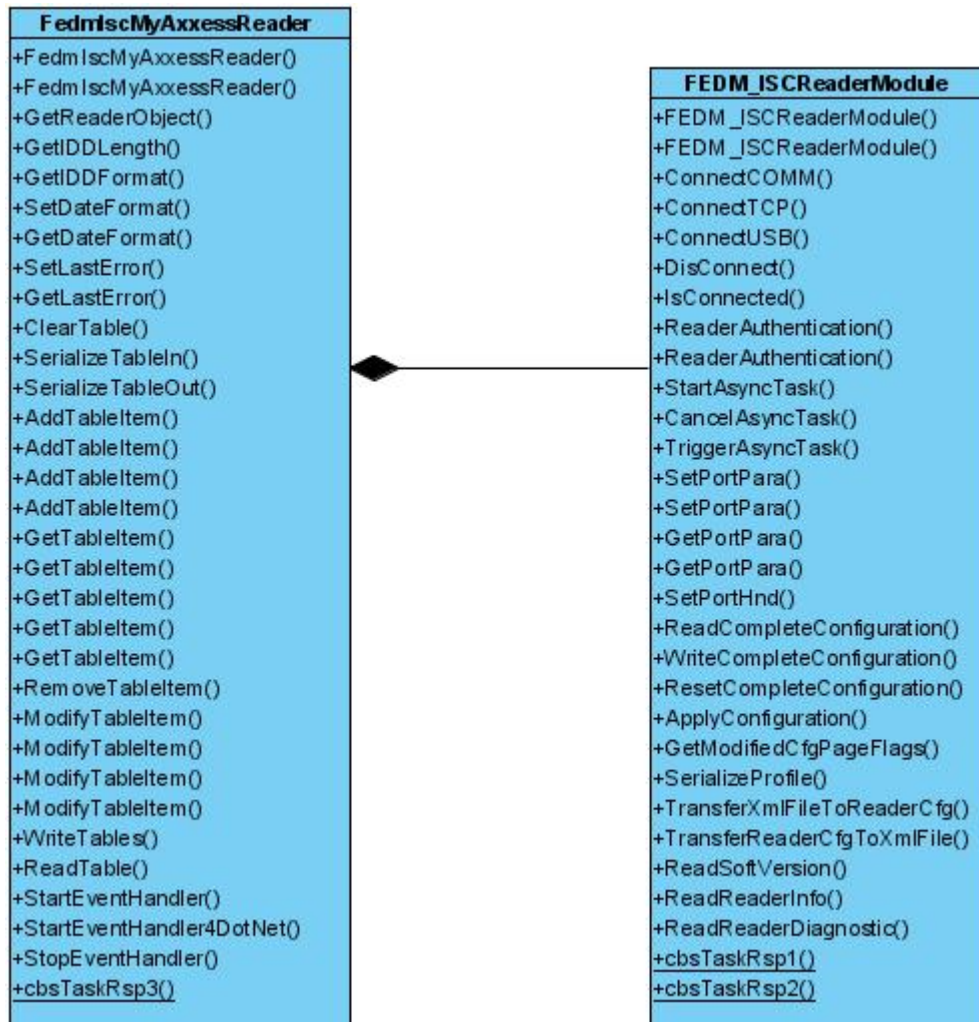
1.1. Principles

Access Control solutions based on OBID® myAXXESS technology provides an opportunity to connect a huge amount of OBID® myAXXESS Reader at a Local Area Network (LAN) under control of a host application. Each OBID® myAXXESS Reader can work online or offline. All necessary tables are stored persistent. Optionally, an event mechanism can be installed to notify a host application about access events.

Access conditions are handled with three tables: an Access-Table, a Timezone-Table and a Holiday-Table. The detailed structure of each table is documented in the designated system manual.

1.2. Class-Library

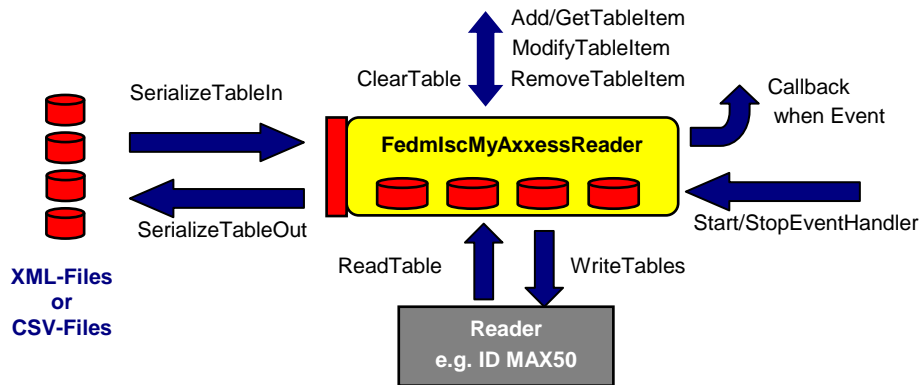
Support for OBID® myAXXESS Readers consists of the class FedmlscMyAxxessReader which includes the reader class FEDM_ISCReaderModule, the representation of the physical myAXXESS-Reader.



The operation list of class FEDM_ISCReaderModule illustrates the main scope of this aggregated class: managing basic Reader configuration and establishing a connection.

The main class FedmlscMyAxxessReader integrates the access tables and realizes an easy API for the access table management. The operation list can be classified into four groups:

1. methods for table exchange with the OBID® myAXXESS Reader
2. methods for table modification
3. methods for serialization of tables
4. methods for event notification



Group 2 and 3 are alternatively. With the table modification methods each table item can be accessed. This interface is ideal for the use together with databases inside a homogeneous application. The serialization of tables allows the co-operation with applications written in different computer languages.

Additionally, the serialization option provides an easy solution for backup of tables.

2. Revisions since the previous version

- First release of 64-Bit library for Windows.

2.1. Limitation of Release-Version

The serialization of XML-Files will be realized in future.

3. Installation

Normally, this package is shipped together with other libraries in a Software Development Kit (SDK). Copy the SDK into a directory of your choice.

The files of this library package can be found in the sub-directory fedm-classlib.

Please refer to the manual Part B.ISC (H10202-xe-ID-B.DOC) to get detailed information about the installation.

3.1. 32- and 64-Bit Windows XP/Vista/7/8

After the installation the directory fedm-classlib contains additionally to the core library files the following files:

Files in sub-directory bin\myaxxess\vcxxx\x86	Description
FedmlscMyAxxessVC80.dll/.lib	with Visual Studio 2005 compiled library (release version), compatible with MFC-Version 8.0
FedmlscMyAxxessVC80d.dll/.lib	with Visual Studio 2005 compiled library (debug version), compatible with MFC-Version 8.0
FedmlscMyAxxessVC90.dll/.lib	with Visual Studio 2008 compiled library (release version), compatible with MFC-Version 9.0
FedmlscMyAxxessVC90d.dll/.lib	with Visual Studio 2008 compiled library (debug version), compatible with MFC-Version 9.0
FedmlscMyAxxessVC100.dll/.lib	with Visual Studio 2010 compiled library (release version), compatible with MFC-Version 10.0
FedmlscMyAxxessVC100d.dll/.lib	with Visual Studio 2010 compiled library (debug version), compatible with MFC-Version 10.0
FedmlscMyAxxessVC110.dll/.lib	with Visual Studio 2012 compiled library (release version), compatible with MFC-Version 11.0
FedmlscMyAxxessVC110d.dll/.lib	with Visual Studio 2012 compiled library (debug version), compatible with MFC-Version 11.0
Files in sub-directory bin\myaxxess\vcxxx\x64	Description
FedmlscMyAxxessVC110.dll/.lib	with Visual Studio 2012 compiled library (release version), compatible with MFC-Version 11.0
FedmlscMyAxxessVC110d.dll/.lib	with Visual Studio 2012 compiled library (debug version), compatible with MFC-Version 11.0

3.2. Windows CE

After the installation the directory fedm-classlib contains additionally to the core library files the following files:

Files in sub-directory	Description
bin\myaxxess	
FedmlscMyAxxessCE.dll/.lib	with Visual Studio compiled library, compatible with the Windows CE platform

3.3. 32- and 64-Bit Linux

After the installation the directory fedm-classlib contains additionally to the core library file the following file:

Files in sub-directory	Description
bin/myaxxess	
libFedmlscMyAxxess.so.x.y.z	with GCC compiled library for Linux

x.y.z represents the version number of the library

The compiled library is linked against LibC V6 and LibStdC++ V6

Create a symbolic link to the library file libFedmlscMyAxxess.so.x.y.z in the directory /usr/lib:

```
cd /usr/lib
```

```
ln -sf /<Verzeichnis>/libFedmlscCore.so.x.y.z libFedmlscMyAxxess.so.x
```

```
ln -sf /<Verzeichnis>/libFedmlscCore.so.x libFedmlscMyAxxess.so
```

```
ldconfig
```

3.4. Source Code

After the installation the source code of the library can be found in the directory src.

Files in sub-directory src\myaxxess	Description
files in sub-directory core	core classes documented in Part B.ISC (H10202-xe-ID-B.pdf)
FedmlscMyAxxessTables.h	Table declarations for OBID i-scan® and OBID® classic-pro reader family with myAxxess functionality
FedmlscMyAxxessReader.h/.cpp	Main class for OBID i-scan® and OBID® classic-pro reader family with myAxxess functionality
FedmlscMyAxxess_Xml.h	XML-Tag names for myAxxess XML-Files
FedmlscMyAxxess_XmlParser.h/.cpp	XML-Parser for myAxxess XML-Files
FedmlscMyAxxess_CsvParser.h/.cpp	CSV-Parser for myAxxess CSV-Files

3.5. Dependencies

The library file FedmlscMyAxxess depends on the library file FedmlscCore with the same version number. It is a condition precedent to install both libraries always in the same version number.

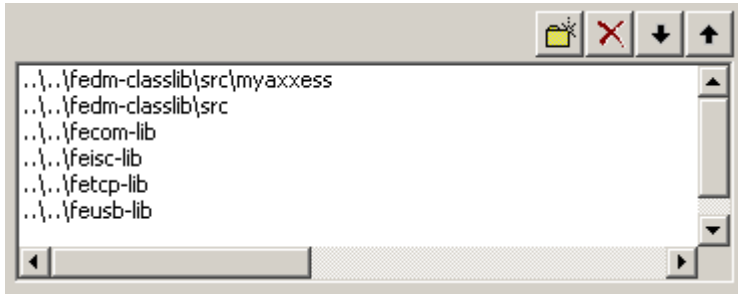
The FEDM class library depends on the function libraries for the communications interfaces (FECOM, FEUSB, FETCP) and the reader family (FEISC, FEFU, FETCL). These libraries are part of the respective SDK and must be installed on the target system.

The class FEDM_ISCReader contains a method *EvalLibDependencies* for the verification of the version numbers of the dependent function libraries. It is recommended to invoke this method once in the application after the program started.

4. Integration into application projects

All preprocessor definitions are compiled in the include file `FedmlscMyAxxessReader.h` for using the compiled libraries and it is necessary to link this include file for the compiler and, only for Windows, `FedmlscMyAxxessVCxx.lib` (Release-Version) or `FedmlscMyAxxessVCxxd.lib` (Debug-Version) for the linker.

As the FEDM class library depends on FExxx function libraries, the following include directories are important to add to the compiler settings:



Other settings are not necessary.

Important: Visual Studio projects have to link the MFC dynamically.

5. Installation on the target computer

Together with the application files, the runtime file of the class libraries FedmlscMyAxxess and FedmlscCore and the runtime files of the function libraries FECOM, FEUSB, FETCL, FETCP, FEISC and FEFU must be installed on the target computer.

5.1. 32-Bit Libraries on a 32- and 64-Bit Windows

It is recommended to keep the library files in the directory of the application. This avoids version conflicts with later installations which also install these library files, but possibly different versions.

The library files FedmlscMyAxxessVCxxx.dll and FedmlscCoreVCxxx.dll depend on newer MFC libraries which are usually not present on the target computer. Therefore, they must be installed. So-called merge modules are provided with Visual Studio which can be incorporated in a Setup project and which install the MFC libraries. The following merge modules are necessary for the FedmlscCore/FedmlscMyAxxess libraries:

Library File	MFC Version	Merge Modules
FedmlscMyAxxessVC80.dll	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
FedmlscMyAxxessVC90.dll	Version 9.0 (9.0.30729.6161 s. MS11-025 ²)	Microsoft_VC90_MFC_x86.msm Microsoft_VC90_CRT_x86.msm policy_9_0_Microsoft_VC90_MFC_x86.msm policy_9_0_Microsoft_VC90_CRT_x86.msm
FedmlscMyAxxessVC100.dll	Version 10.0 (10.0.30319.460 s. MS11-025 ³)	Microsoft_VC100_MFC_x86.msm Microsoft_VC100_CRT_x86.msm
FedmlscMyAxxessVC110.dll	Version 11.0 (11.0.51106.1)	Microsoft_VC110_MFC_x86.msm Microsoft_VC110_CRT_x86.msm

Alternatively, the installation of the Visual C++ Runtime Libraries can be realized with the download site of Microsoft. For each version of MFC you can find a file called vcredist_x86.exe for download.

1st Note: The file vcredist_x86.exe must be of version of at least the specified number above.

2nd Note: Merge Modules can only be updated with Windows Update.

¹ Microsoft Security Bulletin Article-ID: 2538218 from Juni 14, 2011

² Microsoft Security Bulletin Article-ID: 2538243 from Juni 14, 2011

³ Microsoft Security Bulletin Article-ID: 2542054 from Juni 14, 2011

3rd Note: Debug versions of FEDM, marked with a **d** at the end of the file name (e.g. FedmlscCoreVCxxx**d**.dll and FedmlscMyAxxessVCxxx**d**.dll) cannot be installed on target computers. The reason is that every merge module does install only the release version of a MFC library.

5.2. 64-Bit Libraries on a 64-Bit Windows

It is recommended to keep the library files in the directory of the application. This avoids version conflicts with later installations which also install these library files, but possibly different versions.

The library files FedmlscMyAxxessVCxxx.dll and FedmlscCoreVCxxx.dll depend on newer MFC libraries which are usually not present on the target computer. Therefore, they must be installed. So-called merge modules are provided with Visual Studio which can be incorporated in a Setup project and which install the MFC libraries. The following merge modules are necessary for the FedmlscCore/FedmlscMyAxxess libraries:

Library File	MFC Version	Merge Modules
FedmlscMyAxxessVC110.dll	Version 11.0 (11.0.51106.1)	Microsoft_VC110_MFC_x64.msm Microsoft_VC110_CRT_x64.msm

Alternatively, the installation of the Visual C++ Runtime Libraries can be realized with the download site of Microsoft. For each version of MFC you can find a file called vcredist_x64.exe for download.

1st Note: The file vcredist_x64.exe must be of version of at least the specified number above.

2nd Note: Merge Modules can only be updated with Windows Update.

3rd Note: Debug versions of FEDM, marked with a **d** at the end of the file name (e.g. FedmlscCoreVC1xx**d**.dll and FedmlscMyAxxessVC1xx**d**.dll) cannot be installed on target computers. The reason is that every merge module does install only the release version of a MFC library.

5.3. 32-Bit Windows CE

Together with the application files, the runtime file of the class libraries FedmlscMyAxxess and FedmlscCore and the runtime files of the function libraries FECOM, FEUSB, FETCL, FETCP, FEISC and FEFU must be installed on the target computer.

It is recommended to keep the library files in the directory of the application. This avoids version conflicts with later installations which also install these library files, but possibly different versions.

5.4. 32- and 64-Bit Linux

The installation on the target is equivalent to [3.3. 32- and 64-Bit Linux](#).

The installation instructions for the dependent function libraries can be found in the respective manuals.

6. Class description

The ID FEDM class library undergoes a continuous adaptation process. We will make every effort to maintain the documented status. Changes are still however possible.

6.1. FedmlscMyAxxessReader

The class FedmlscMyAxxessReader inherits all necessary methods for Reader management and builds an easy to use API for access table management so that you can work with an object from this class.

It is recommended to create an instance of this class for every OBID® myAXXESS Reader.

6.1.1. Implemented tables

Table	Description
m_CrcList	Contains a CRC List for table verification. This list is for internal use.
m_Metadata	Contains general information about the Timezone Table, Holiday Table and Access Table. This list is for internal use.
m_TimezoneTable	List of Timezone Entries callable by index starting with 1.
m_HolidayTable	List of Holiday Entries.
m_AccessTable	List of Access Entries.
m_EventTable	List of Event Datasets.

6.1.2. Methods (public)

6.1.2.1. FedmIscMyAxxessReader

Function	Constructor		
Syntax	FedmIscMyAxxessReader(unsigned char ucIDDLenght, unsigned char ucIDDFormat = FEDM_MYAXXESS_IDD_FORMAT_NUM)		
Description	<p>Constructor of FedmIscMyAxxessReader class.</p> <p>The identification characteristic for access control applications is the Identifier Data (IDD) located in the access table. Each IDD must have the same length and this information must be set with <i>ucIDDLenght</i>. A minimum length of 1 is required and a maximum length of 16 is possible.</p> <p><i>ucIDDFormat</i> is the serialization format in XML-/CSV-Files used for all IDDs (the default is numerical format):</p>		
	IDD-Format	Description	Example
	ASCII	IDD can contain any ASCII characters: 0-9, a-z, A-Z and special characters	MyIDD_0123
	Hexadecimal	IDD can contain only hexadecimal characters: 0-9, a-f, A-F	FA3359
	Numerical	IDD consists of only numeric characters: 0-9	16397145
	<p>The calculation of IDD-Length depends on the IDD-Format:</p> <p><u>ASCII</u>: max. 64 chars (<= 64 Byte)</p> <p><u>Hex</u>: max. 128 chars (<= 64 Byte)</p> <p><u>Num</u>: 0..18.446.744.073.709.551.615 (<= 8 Byte)</p>		
Cross-Reference	7.1.4. IDD Format Constants		
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error }</pre>		

6.1.2.2. GetReaderObject

Function	Returns the pointer to the reader class
Syntax	FEDM_ISCReaderModule* GetReaderObject(void)
Description	This method returns the aggregated main reader class which has the control of the physical Reader.
Return value	Pointer to FEDM_ISCReaderModule.
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... FEDM_ISCReaderModule* pReader = m_pMyAxxessReader->GetReaderObject(); pReader->ConnectTCP("192.168.1.10", 10001); pReader->ReadReaderInfo();</pre>

6.1.2.3. GetLastError

Function	Returns the last error code from the reader class
Syntax	int GetLastError(void)
Description	Returns the last error code from the reader class. The error codes are documented in the manual Part A (H10102-xe-ID-B.pdf) of the core library.
Return value	Error code (<0) or OK (=0).
Example	

6.1.2.4. GetErrorText

Function	Returns an error text
Syntax	char* GetErrorText(int iErrorCode)
Description	Returns an error text for the transferred error code. The error codes and texts are documented in the manual Part A (H10102-xe-ID-B.pdf) of the core library.
Return value	Error text.
Example	

6.1.2.5. GetIDDLength / GetIDDFormat

Function	Gets the <i>ucIDDFormat</i> and <i>ucIDDLength</i>
Syntax	unsigned char GetIDDLength(void) unsigned char GetIDDFormat(void)
Description	These methods return the format and length used for all IDD's set in the constructor.
Return value	The return values are <i>ucIDDFormat</i> and <i>ucIDDLength</i> .
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; unsigned char ucIDDFormat; unsigned char ucIDDLength; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... ucIDDFormat = m_pMyAxxessReader-> GetIDDFormat(); ucIDDLength = m_pMyAxxessReader-> GetIDDLength();</pre>

6.1.2.6. SetDateFormat

Function	Sets the Date Format		
Syntax	int SetDateFormat (unsigned int uiDateType)		
Description	The function is used to set the date format, necessary for table serialization. <i>uiDateType</i> contains the date format (the default is ISO 8601 compatible format):		
	Date-Format	Description	Example
	ISO 8601	ISO 8601 is an international standard for date and time representations issued by the International Organization for Standardization (ISO)	2009-10-22
	German	The all-numeric form for dates is in the order day-month-year, using a period as the separator.	22.10.2009
	US	In the United States, dates are traditionally written in the month-day-year order	10/22/2009
Cross-Reference	7.1.5. Date Format Constants		
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.		
Example	<pre> ... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... if((m_pMyAxxessReader->SetDateFormat(FEDM_MYAXXESS_DATE_FORMAT_GER)) != 0) { // Code here for error } ... </pre>		

6.1.2.7. GetDateFormat

Function	Gets the Date Format
Syntax	unsigned char GetDateFormat(void)
Description	The function returns the date format, necessary for table serialization.
Return value	The return value is the <i>uiDateType</i> .
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; unsigned int ucDateType; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... ucDateType = m_pMyAxxessReader->GetIDateFormat();</pre>

6.1.2.8. ClearTable

Function	Clears one Table
Syntax	int ClearTable(unsigned int uiTableID)
Description	<p>The method clears the internal table specified by <i>uiTableID</i>. Only Timezone Table, Holiday Table, Access Table and Event Table are possible.</p> <p>For <i>uiTableID</i> see 7.1.2. Table Constants.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... if((m_pMyAxxessReader->ClearTable(FEDM_MYAXXESS_TIMEZONE_TABLE)) != 0) { // Code here for error }</pre>

6.1.2.9. SerializeTableIn

Function	Serialize in of one table
Syntax	int SerializeTableIn(char* szFilename, unsigned int uiFileType=FEDM_MYAXXESS_FILETYPE_XML)
Description	<p>The method reads one table from XML- or CVS-Format into the internal table. The internal table is deleted before.</p> <p><i>szFilename</i> is the name of the file to be read. It must include the directory, if the file is not located in the same directory as the application.</p> <p><i>uiFileType</i> is the type of file to be read. For <i>uiFileType</i> see 7.1.6. Filetype Constants for Serialization.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre> ... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... if((m_pMyAxxessReader->SerializeTableIn("HolidayTable.csv", FEDM_MYAXXESS_FILETYPE_CSV)) != 0) { // Code here for error } ... </pre>

6.1.2.10. SerializeTableOut

Function	Serialize out of one table
Syntax	<pre>int SerializeTableOut(unsigned int uiTableID, char* szFilename, unsigned int uiFileType=FEDM_MYAXXESS_FILETYPE_XML)</pre>
Description	<p>The method writes one internal table in XML- or CSV-Format.</p> <p><i>uiTableID</i> is the table identifier. Only Timezone Table, Holiday Table, Access Table and Event Table are possible. For <i>uiTableID</i> see 7.1.2. Table Constants.</p> <p><i>szFilename</i> is the name of the file to be written. It must include the directory, if the file is not located in the same directory as the application.</p> <p><i>uiFileType</i> is the type of file to be written. For <i>uiFileType</i> see 7.1.6. Filetype Constants for Serialization.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... if((m_pMyAxxessReader->SerializeTableOut(FEDM_MYAXXESS_ACCESS_TABLE, "AccessTable.csv", FEDM_MYAXXESS_FILETYPE_CSV)) != 0) { // Code here for error }</pre>

6.1.2.11. StartEventHandler

Function	Initialization of event handling															
Syntax	<pre>int StartEventHandler(unsigned int uiPortNr, void* pAny, LPFN_FEDM_MYAXXESS_EVENT_CB cbEvent, LPFN_FEDM_MYAXXESS_KEEP_ALIVE_CB cbKeepAlive, int iAuthentType, char* sAuthentKey)</pre>															
Description	<p>The method initializes an event handler for different events from an OBID® myAXXESS Reader. The event handler is based on two callback methods: <i>cbEvent</i> will be called if an access event is sent. <i>cbKeepAlive</i> will be called periodically if the keep alive option in the Reader is enabled. The Reader sends a [0x6E] Reader Diagnostic protocol with actual error and table information.</p> <p><i>uiPortNr</i> is the IP port number where to listen for event notifications.</p> <p><i>pAny</i> is a pointer to anything and will be transferred back as the first parameter in the callback function. Normally, the this pointer is transferred to call inside the static callback function a member method of this.</p> <p>If the Reader is working in crypto mode, the library needs the authentication password <i>sAuthentKey</i> and the type of authentication <i>iAuthentType</i> to handle the authentication with the Reader for each event.</p> <p>Following settings are supported:</p> <table><tr><th>Mode</th><th>iAuthentType</th><th>sAuthentKey</th></tr><tr><td>no Authentication</td><td>-1</td><td>NULL</td></tr><tr><td>AES 128 Bit</td><td>0</td><td>16 Byte Key</td></tr><tr><td>AES 192 Bit</td><td>1</td><td>24 Byte Key</td></tr><tr><td>AES 256 Bit</td><td>2</td><td>32 Byte Key</td></tr></table> <p>Note that only one event handler can be started for each reader object.</p>	Mode	iAuthentType	sAuthentKey	no Authentication	-1	NULL	AES 128 Bit	0	16 Byte Key	AES 192 Bit	1	24 Byte Key	AES 256 Bit	2	32 Byte Key
Mode	iAuthentType	sAuthentKey														
no Authentication	-1	NULL														
AES 128 Bit	0	16 Byte Key														
AES 192 Bit	1	24 Byte Key														
AES 256 Bit	2	32 Byte Key														
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.															
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... int back = m_pMyAxxessReader->StartEventHandler(10020, this, cbsEvent, cbsKeepAlive, -1, NULL); if(back) { // Code here for error } ...</pre>															

```
int MyClass::cbsEvent( void* pAny, int iError, FEDM_ISC_MYAXXESS_EVENT_TABLE_ITEM* pItem,  
                      unsigned char& ucAction, char* cIPAdr, int iPortHnd );  
  
{  
    if((MyClass*)pAny == NULL)  
        return 0;  
    ...  
    ...  
}  
...  
int MyClass::cbsKeepAlive( void* pAny, int iError, unsigned int uiErrorFlags,  
                          unsigned int uiTableSize, unsigned int uiTableLength )  
  
{  
    if((MyClass*)pAny == NULL)  
        return 0;  
    ...  
    ...  
}
```

6.1.2.12. StopEventHandler

Function	The event handler is removed
Syntax	int StopEventHandler()
Description	The method stops the installed event handler.
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre>... #include "FedmIscMyAxxessReader.h" int back = m_pMyAxxessReader->StopEventHandler(); if(back) { // Code here for error }</pre>

6.1.2.13. AddTableItem (for native C/C++ interface)

Function	Overloaded method to add one item to a table
Syntax	<pre>int AddTableItem(FEDM_ISC_MYAXXESS_TIMEZONE_TABLE_ITEM* pItem) int AddTableItem(FEDM_ISC_MYAXXESS_HOLIDAY_TABLE_ITEM* pItem) int AddTableItem(FEDM_ISC_MYAXXESS_ACCESS_TABLE_ITEM* pItem)</pre>
Description	<p>The method adds one item at the end of an internal table.</p> <p><i>pItem</i> is the item to add.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... FEDM_ISC_MYAXXESS_HOLIDAY_TABLE_ITEM pHolidayItem; pHolidayItem.ucHoliday_Year = 9; pHolidayItem.ucHoliday_Month = 1; pHolidayItem.ucHoliday_Day = 1; if((m_pMyAxxessReader->AddTableItem(&pHolidayItem)) != 0) { // Code here for error }</pre>

6.1.2.14. AddTableItem (for 'naturally speaking' interface)

Function	Add one item to a table
Syntax	int AddTableItem(unsigned int uiTableID, char* sItem)
Description	<p>The method adds one item at the end of an internal table.</p> <p><i>uiTableID</i> is the table identifier. Only Timezone Table, Holiday Table and Access Table are possible. For <i>uiTableID</i> see 7.1.2. Table Constants.</p> <p><i>sItem</i> is the new table item content in form of semicolon separated identifier=value string.</p> <p>For conventions of <i>sItem</i> see 7.2. Conventions for 'naturally speaking' interface.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre> ... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... char sHolidayItem [] = "Holiday=2009-08-11"; if((m_pMyAxxessReader->AddTableItem(FEDM_MYAXXESS_HOLIDAY_TABLE, sHolidayItem)) != 0) { // Code here for error } </pre>

6.1.2.15. GetTableItem (for native C/C++ interface)

Function	Overloaded method to get one item of a table
Syntax	<pre> int GetTableItem(unsigned int uiIndex, FEDM_ISC_MYAXXESS_TIMEZONE_TABLE_ITEM* pItem) int GetTableItem(unsigned int uiIndex, FEDM_ISC_MYAXXESS_HOLIDAY_TABLE_ITEM* pItem) int GetTableItem(unsigned int uiIndex, FEDM_ISC_MYAXXESS_ACCESS_TABLE_ITEM* pItem) int GetTableItem(FEDM_ISC_MYAXXESS_EVENT_TABLE_ITEM* pItem) </pre>
Description	<p>The method returns one item of an internal table, identified by the zero-based index <i>uiIndex</i>. For Timezone Table, Holiday Table and Access Table the item with the selected index is returned. Only for the Event Table the oldest item is returned and deleted from the Event Table.</p> <p><i>pItem</i> is used to keep requested item.</p>
Return value	In case of error the function returns <0, otherwise the index of next table item to be read >0 is returned. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre> ... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... FEDM_ISC_MYAXXESS_HOLIDAY_TABLE_ITEM HolidayItem; if((m_pMyAxxessReader ->GetTableItem(1, &HolidayItem)) != 0) { // Code here for error } </pre>

6.1.2.16. GetTableItem (for 'naturally speaking' interface)

Function	Get one item of a table
Syntax	int GetTableItem(unsigned int uiTableID, unsigned int uiIndex, char* sItem)
Description	<p>The method returns one item of an internal table, identified by the zero-based index <i>uiIndex</i>. For Timezone Table, Holiday Table and Access Table the item with the selected index is returned. Only for the Event Table the index is ignored and the oldest item is returned and deleted from the Event Table.</p> <p><i>uiTableID</i> is the table identifier. Only Timezone Table, Holiday Table, Access Table and Event Table are possible. For <i>uiTableID</i> see 7.1.2. Table Constants.</p> <p><i>sItem</i> is used to return the item content as a semicolon separated identifier=value string For conventions of <i>sItem</i> see 7.2. Conventions for 'naturally speaking' interface.</p>
Return value	In case of error the function returns <0, otherwise the index of next table item to be read >0 is returned. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre> ... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... char sHolidayItem[] = "Holiday= 12/31/2009"; if((m_pMyAxxessReader->GetTableItem(FEDM_MYAXXESS_HOLIDAY_TABLE, 1, sHolidayItem)) != 0) { // Code here for error } </pre>

6.1.2.17. RemoveTableItem

Function	Remove one item of a table
Syntax	int RemoveTableItem(unsigned int uiTableID, unsigned int uiIndex)
Description	<p>The method removes one item of an internal table, identified by the zero-based index <i>uiIndex</i>.</p> <p><i>uiTableID</i> is the table identifier. Only Holiday Table and Access Table are possible. For <i>uiTableID</i> see 7.1.2. Table Constants.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... if((m_pMyAxxessReader->RemoveTableItem(FEDM_MYAXXESS_HOLIDAY_TABLE, 1)) != 0) { // Code here for error }</pre>

6.1.2.18. ModifyTableItem (for native C/C++ interface)

Function	Overloaded method to modify one item of a table
Syntax	<pre> int ModifyTableItem(unsigned int uiIndex, FEDM_ISC_MYAXXESS_TIMEZONE_TABLE_ITEM* pItem) int ModifyTableItem(unsigned int uiIndex, FEDM_ISC_MYAXXESS_HOLIDAY_TABLE_ITEM* pItem) int ModifyTableItem(unsigned int uiIndex, FEDM_ISC_MYAXXESS_ACCESS_TABLE_ITEM* pItem) </pre>
Description	<p>The method modifies one item to an internal table, identified by the zero-based index <i>uiIndex</i>.</p> <p><i>pItem</i> is the new item to replace the old item.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre> ... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... FEDM_ISC_MYAXXESS_HOLIDAY_TABLE_ITEM HolidayItem; HolidayItem.ucHoliday_Year = 9; HolidayItem.ucHoliday_Month = 1; HolidayItem.ucHoliday_Day = 1; if((m_pMyAxxessReader->ModifyTableItem(1, &HolidayItem)) != 0) { // Code here for error } ... </pre>

6.1.2.19. ModifyTableItem (for 'naturally speaking' interface)

Function	Modify one item of a table
Syntax	int ModifyTableItem(unsigned int uiTableID, unsigned int uiIndex, char* sItem)
Description	<p>The function modifies one item in an internal table, identified by the zero-based index <i>uiIndex</i>.</p> <p><i>uiTableID</i> is the table identifier. Only Holiday Table and Access Table are possible. For <i>uiTableID</i> see 7.1.2. Table Constants.</p> <p><i>sItem</i> is the new item in form of a semicolon separated identifier=value string to replace the old item.</p>
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre> ... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... char sHolidayItem [] = "Holiday=09-08-11"; if((m_pMyAxxessReader->ModifyTableItem(1, sHolidayItem)) != 0) { // Code here for error } </pre>

6.1.2.20. WriteTables

Function	Write Tables into the Reader
Syntax	int WriteTables()
Description	The method writes the Timezone Table, Holiday Table and Access Table into the OBID®-Reader. Metadata and CRC List are calculated.
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	see 6.2.2. Example 1: Transfer tables

6.1.2.21. ReadTable

Function	Read one table out of the Reader
Syntax	int ReadTable(unsigned int uiTableID);
Description	The method reads one complete table out of the OBID®-Reader. <i>uiTableID</i> is the table identifier. For <i>uiTableID</i> see 7.1.2. Table Constants.
Return value	In case of error the function returns <0, otherwise 0. The list of error codes can be found in the appendix of manual FEDM Part.A.
Example	<pre>... #include "FedmIscMyAxxessReader.h" ... FedmIscMyAxxessReader* m_pMyAxxessReader; ... m_pMyAxxessReader = new FedmIscMyAxxessReader(4, FEDM_MYAXXESS_IDD_FORMAT_HEX); if(m_pMyAxxessReader == NULL) { // Code here for error } ... if (m_pAxxess->ReadTable(FEDM_MYAXXESS_TIMEZONE_TABLE)) { // Code here for error }</pre>

6.2. Working with the Reader classes

6.2.1. Important initializations

Before using the class `FedmlscMyAxxessReader` for the first time, some initializing must be performed:

1. Reader type	<p>The reader type must be set in the reader class with one of two options:</p> <ol style="list-style-type: none">1. The call of the method <code>GetReaderObject()->ReadReaderInfo()</code> after a successful connection (recommended).2. Set of reader type with the method: <code>GetReaderObject()->SetReaderType(..)</code> The constants of all reader types are listed in the file <code>FEDM_ISC.h</code>.
2. IDD-Length	Must be set once with constructor
3. IDD-Format	Can be set once with constructor. The default setting is: numerical format.
4. Date-Format	Must be set with <code>SetDateFormat</code> and can be changed at run-time. The default setting is: ISO 8601 compatible format.

6.2.2. Example 1: Transfer tables into Reader using the C++ API

```
...
#include "fetcp.h"
#include "FedmIscMyAxxessReader.h"

// create instance with default IDD-Format and IDD-Length of 4 bytes
FedmIscMyAxxessReader m_MyAxxessReader(4);
FEDM_ISC_READER_INFO* pReaderInfo = NULL;

int back = m_MyAxxessReader.GetReaderObject()->ConnectTCP( "192.168.1.1", 10001 );
if ( back < 0 )
{
    // Code here for error
}

// read reader infos and set reader type for internal use
pReaderInfo = m_MyAxxessReader.GetReaderObject()->ReadReaderInfo();
if (pReaderInfo == NULL)
{
    // Code here for error
}

// define timezones
FEDM_ISC_MYAXXESS_TIMEZONE_TABLE_ITEM      TimezoneItem;
TimezoneItem.ucDays = 7;
TimezoneItem.ucEndDate_Day = 8;
TimezoneItem.ucEndDate_Month = 9;
TimezoneItem.ucEndDate_Year = 10;
TimezoneItem.ucEndTime_Hour = 11;
TimezoneItem.ucEndTime_Minute = 12;
TimezoneItem.ucStartDate_Day = 13;
TimezoneItem.ucStartDate_Month = 1;
TimezoneItem.ucStartDate_Year = 14;
TimezoneItem.ucStartTime_Hour = 15;
TimezoneItem.ucStartTime_Minute = 16;
int iTimezoneNum = 1; // Set number of timezones

// generate timezone table
for (int iTimezoneCnt = 0; iTimezoneCnt < iTimezoneNum; iTimezoneCnt ++ )
{
    if ((m_MyAxxessReader.AddTableItem(&TimezoneItem)) != 0)
    {
        // Code here for error
    }
}

// define one holiday
FEDM_ISC_MYAXXESS_HOLIDAY_TABLE_ITEM HolidayItem;
HolidayItem.ucHoliday_Year = 9;
```

```
HolidayItem.ucHoliday_Month = 1;
HolidayItem.ucHoliday_Day = 2;
int iHolidayNum = 1; // Set number of holidays

// generate holiday table
for (int iHolidayCnt = 0; iHolidayCnt < iHolidayNum; iHolidayCnt++)
{
    if ((m_MyAxxessReader.AddTableItem(&HolidayItem)) != 0)
    {
        // Code here for error
    }
}

// define one access item with IDD = 16397145
FEDM_ISC_MYAXXESS_ACCESS_TABLE_ITEM    AccessItem;
unsigned int uiIDD = 16397145;
// convert numeric number into hex array. FEDM_ConvXXXXToYYY-functions are part of the core library
FEDM_ConvUIntToHexUChar(uiIDD, AccessItem.ucIDD, 4);
// resulting array values:
// AccessItem.ucIDD[0] = 0x00;
// AccessItem.ucIDD[1] = 0xFA;
// AccessItem.ucIDD[2] = 0x33;
// AccessItem.ucIDD[3] = 0x59;
AccessItem.uiTimezones = 9;
int iAccessNum = 1; // Set number of access items

// generate access table
for (int iAccessCnt = 0; iAccessCnt < iAccessNum; iAccessCnt++)
{
    if ( m_MyAxxessReader.AddTableItem(&pAccessItem) != 0 )
    {
        // Code here for error
    }
}

// write all tables into reader
if (m_MyAxxessReader.WriteTables())
{
    // Code here for error
}
```

6.2.3. Example 2: Transfer tables into Reader using serialization of CSV-File

```
...
#include "fetcp.h"
#include "FedmIscMyAxxessReader.h"

// create instance with default IDD-Format and IDD-Length of 4 bytes
FedmIscMyAxxessReader m_MyAxxessReader(4);
// read reader infos and set reader type for internal use
FEDM_ISC_READER_INFO* pReaderInfo = NULL;

int back = m_MyAxxessReader.GetReaderObject()->ConnectTCP( "192.168.1.1", 10001 );
if ( back < 0 )
{
    // Code here for error
}

// read reader infos and set reader type for internal use
pReaderInfo = m_MyAxxessReader.GetReaderObject()->ReadReaderInfo();
if (pReaderInfo == NULL)
{
    // Code here for error
}

/ read timezone table from file (date format is determined automatically)
if (m_MyAxxessReader.SerializeTableIn( "TiemzoneTable.csv", FEDM_MYAXXESS_FILETYPE_CSV) != 0)
{
    // Code here for error
}

// read holiday table from file (date format is determined automatically)
if (m_MyAxxessReader.SerializeTableIn( "HolidayTable.csv", FEDM_MYAXXESS_FILETYPE_CSV) != 0)
{
    // Code here for error
}

// read access table from file (IDD format must be numerical, IDD length must be 4)
if (m_MyAxxessReader.SerializeTableIn( "AccessTable.csv", FEDM_MYAXXESS_FILETYPE_CSV) != 0)
{
    // Code here for error
}

// write all tables into reader
if (m_MyAxxessReader.WriteTables())
{
    // Code here for error
}
```

7. Appendix

7.1. List of constants

All constants listed here are defined in FedmIscMyAxxessReader.h.

7.1.1. Internal Constants

Constant	Description
FEDM_MYAXXESS_TABLE_VERSION	Version of MyAxxess Tables.
FEDM_MYAXXESS_UPDATE_START	Command byte to start the update.
FEDM_MYAXXESS_UPDATE_END	Command byte to end the update.
FEDM_MYAXXESS_WRITE_TABLE	Command byte to write a table.
FEDM_MYAXXESS_RESET_TABLE	Command byte to reset a table.
FEDM_MYAXXESS_READ_TABLE	Command byte to read a table.
FEDM_MYAXXESS_NOTIFY	Command byte for notify messages.
FEDM_MYAXXESS_MAX_BLOCKSIZE	max. data size for Reader communication over USB or TCP.
FEDM_MYAXXESS_SERIAL_BLOCKSIZE	max. data size for Reader communication over serial port.

7.1.2. Table Constants

Constant	Description
FEDM_MYAXXESS_CRC_LIST	Table-ID for CRC List
FEDM_MYAXXESS_METADATA	Table-ID for Metadata
FEDM_MYAXXESS_TIMEZONE_TABLE	Table-ID for Timezone Table
FEDM_MYAXXESS_HOLIDAY_TABLE	Table-ID for Holiday Table
FEDM_MYAXXESS_ACCESS_TABLE	Table-ID for Access Table
FEDM_MYAXXESS_EVENT_TABLE	Table-ID for Event Table

7.1.3. Event Constants

Constant	Description
FEDM_MYAXXESS_EVENT_ACCESS	Event-ID for Access Events
FEDM_MYAXXESS_EVENT_IO	Event-ID for IO Events
FEDM_MYAXXESS_EVENT_ALARM	Event-ID for Alarm Events

Constant	Description
FEDM_MYAXXESS_EVENT_REQUEST	Event-ID for Request Events

7.1.4. IDD Format Constants

Constant	Description
FEDM_MYAXXESS_IDD_FORMAT_ASCII	ASCII format for IDD
FEDM_MYAXXESS_IDD_FORMAT_NUM	numeric format for IDD
FEDM_MYAXXESS_IDD_FORMAT_HEX	hexadecimal format for IDD

7.1.5. Date Format Constants

Constant	Description
FEDM_MYAXXESS_DATE_FORMAT_ISO8601	Date formatted in ISO8601 Standard (2009-12-31)
FEDM_MYAXXESS_DATE_FORMAT_GER	Date formatted in German Standard (31.12.2009)
FEDM_MYAXXESS_DATE_FORMAT_US	Date formatted in US Standard (12/31/2009)

7.1.6. Filetype Constants for Serialization

Constant	Description
FEDM_MYAXXESS_FILETYPE_XML	Data is serialized in XML.
FEDM_MYAXXESS_FILETYPE_CSV	Data is serialized in CSV.

7.2. Conventions for 'naturally speaking' interface

All conventions listed here are used for the 'naturally speaking' interface.

The general conversation is: "identifier=value; identifier=value;..."

7.2.1. Timezone Table Item Conventions

Conventions for the Timezone Table are:

"Days=value;DateFrom=value;TimeFrom=value;DateTo=value;TimeTo=value"

Identifier	Value
Days	Days separated by + Mo+Tu+We+Th+Fr+Sa+Su
DateFrom	Date separated by -, . or / <ul style="list-style-type: none">• ISO8601 Standard (2009-12-31)• German Standard (31.12.2009)• US Standard (12/31/2009) The Standard is selected automatically by separator.
DateTo	Date separated by -, . or /.
TimeFrom	Time in 24 hours separated by : seconds are optional and will be ignored.
TimeTo	Time in 24 hours separated by : seconds are optional and will be ignored.

Example:

"Days=We+Th+Fr;DateFrom=2009-10-30;TimeFrom=07:35;DateTo=2009-03-01;TimeTo=20:00"

7.2.2. Holiday Table Item Conventions

Conventions for the Holiday Table are:

“Holiday=value ”

Identifier	Value
DateTo	Date separated by -, . or /. The Standard is selected automatically by separator.

Example in ISO 8601:

“Holiday= 2009-08-11”

7.2.3. Access Table Item Conventions

Conversation for the Access Table are:

“IDD=value;Timezones=value;Reserved=value” – Reserved is optional

Identifier	Value
IDD	Interpreted in the defined format. See 7.1.4. IDD Format Constants.
Timezones	Indexes into the Timezone Table separated by + 1+2+3+4+...
Reserved	ignored

Example:

“IDD=12345;Timezones=1+2+3;Reserved=0”

7.2.4. Event Table Item Conventions

Conversation for the Event Table are:

"EventID=value;IDD=value;Date=value;Time=value;Input=value;Action=value;Error=value;Source=value"

Identifier	Value
EventID	See 7.1.3. Event Constants.
IDD	Interpreted in the defined format. See 7.1.4. IDD Format Constants.
Date	Date separated by -, . or /. The Standard is selected automatically by separator.
Time	Time in 24 hours separated by :
Input	Flagfield of Input Events
Action	Flagfield of Actions
Error	Flagfield of Errors
Source	IP-Address of the Event 192.168.3.1

Example:

"EventID=1;IDD=12345;Date=31.12.2009;Time=12:34:09;Input=0;Action=1;Error=0;Source=192.168.3.1"

7.3. CSV-File structure

All Comma Separated Value files (CSV) must follow rules for the internal structure.

7.3.1. Access Table

Content	Example	Notes
<Table Identifier>	AccessTable;	
IDD-Format;<value>;	IDD-Format;ASCII;	values are ASCII, Num, Hex
IDD-Length;<value>;	IDD-Length;8;	<u>ASCII</u> : max. 16 chars (<= 16 Byte) <u>Num</u> : 0..18.446.744.073.709.551.615 (<= 8 Byte) <u>Hex</u> : max. 32 chars (<= 16 Byte)
<table header>	Number;IDD;Timezones;Restriction;	
<No>;<IDD>;<timezones>;0	1;1111;1+2+6;0;	table data
<No>;<IDD>;<timezones>;0	2;2222;1+2+6;0;	table data
...	...	
<No>;<IDD>;<timezones>;0	5;5555;1;0;	table data

7.3.2. Timezone Table

Content	Example
<Table Identifier>	TimezoneTable;
<table header>	Number;Mo;Tu;We;Th;Fr;Sa;Su;DateFrom;TimeFrom;DateTo;TimeTo;
<No>;<timezone record>;	1;1;1;1;1;0;0;01.01.2009;08:00;31.12.2009;18:00;
<No>;<timezone record>;	2;0;1;1;1;1;0;0;01.02.2009;08:00;31.12.2009;18:00;
...	...
<No>;<timezone record>;	5;0;0;0;0;1;0;0;01.05.2009;08:00;31.12.2009;18:00;

7.3.3. Holiday Table

Content	Example
<Table Identifier>	HolidayTable;
<table header>	Number;Holiday;
<No>;<holiday record>;	1;01.01.2009;
<No>;<holiday record>;	2;01.02.2009;
...	...
<No>;<holiday record>;	5;05.01.2009;

7.3.4. Event Table

Content	Example
<Table Identifier>	EventTable;
IDD-Format;<value>;	IDD-Format;Num; values are ASCII, Num, Hex
<table header>	Number;EventID;IDD;Date;Time;Input;Status;Error;Source;
<No>;<Event record>;	1;1;12345678;01.01.2009;12:01:02;;1;0;192.168.10.10;
<No>;<Event record>;	2;1;12345679;01.02.2009;12:05:02;;1;0;192.168.10.10;
...	...
<No>;<Event record>;	5;1;12345670;05.01.2009;12:09:02;;1;0;192.168.10.10;

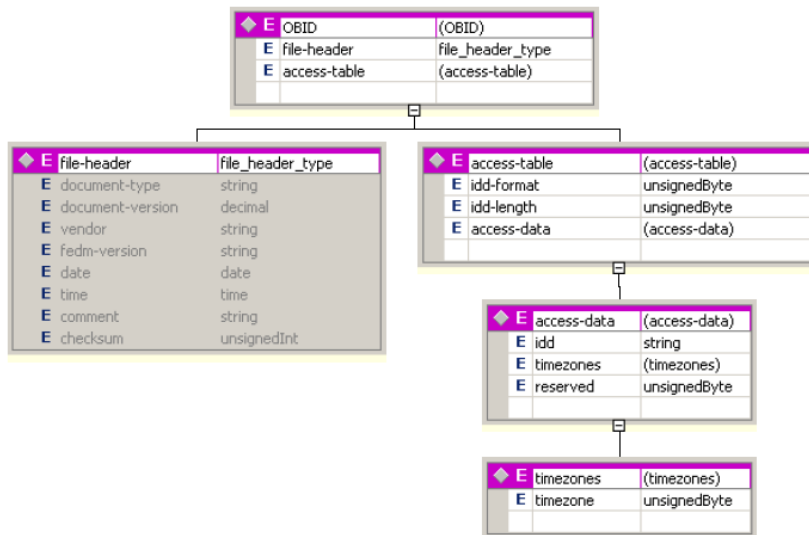
Note: Event Record depends on Data Layout. If one of the elements IDD, Timestamp (Date and Time), Status (Status and Error) or Input is not part of a record, the values are empty strings.

The examples below have no input values.

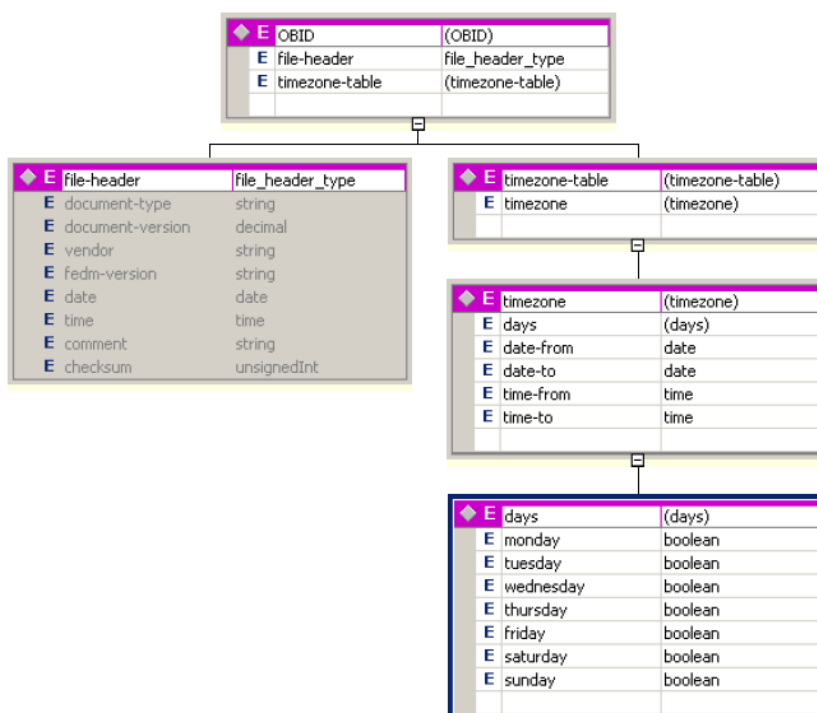
7.4. XML-File structure

All XML files must follow rules for the internal structure. For each XML filetype an XML Schema file is available.

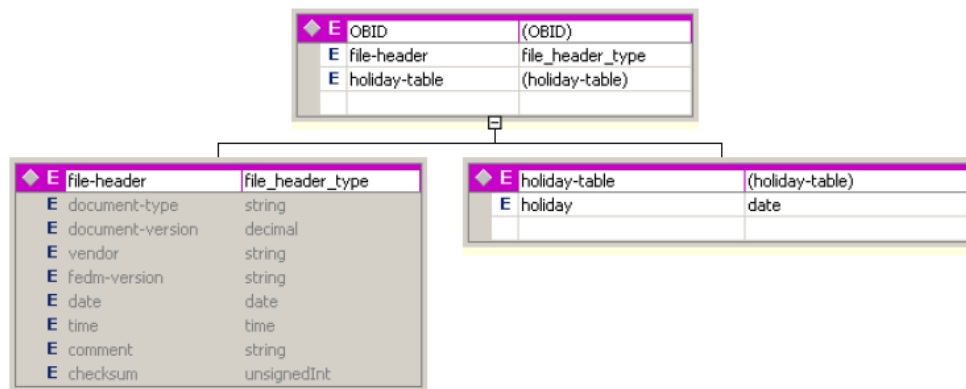
7.4.1. Access Table



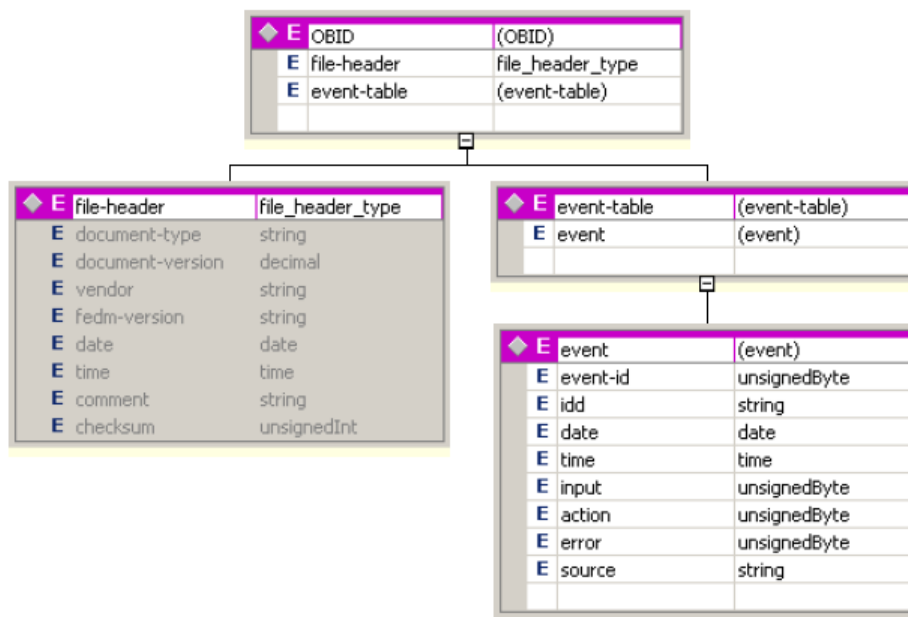
7.4.2. Timezone Table



7.4.3. Holiday Table



7.4.4. Event Table



7.5. Revision history

V4.00.00

- Keep-Alive option for listener socket for receiving access events. The listener socket is closed automatically after a break of the network cable or after loss of power and is recovered again. This ensures the reliability of the network connection.

V3.03.00

- No modifications in this part, but modified base libraries.

V3.01.00

- The API of the main class FedmIscMyAxxessReader must be changed because this class is not longer derived from reader class FEDM_ISCReaderModule. The reader class is now aggregated. This has the consequence that every call of a base class member of FedmIscMyAxxessReader must be extended by the following blue marked construct:

```
// MyAxxReader is of type FedmIscMyAxxessReader  
MyAxxReader.GetReaderObject()->ConnectTCP("192.168.3.10", 10001)
```

- Support for IDD's with up to 64 Byte
- New method **GetErrorText** in class FedmIscMyAxxessReader.
- Support for secured data transmission with the base class method:

```
// MyAxxReader is of type FedmIscMyAxxessReader  
MyAxxReader.GetReaderObject()->ReaderAuthentication(...)
```

After successful authentication, the Reader is switched to crypto mode and the library communicates with enciphered protocols.

StartEventHandler must be extended with two new parameters to handle authentication for each event notification.

V3.00.14

- This is the first Release Version
- Modifications since the Beta-Version:
 1. Structure of Event Table is modified
 2. CSV file format of Event Table is modified
 3. Event Handling with callback functions is implemented