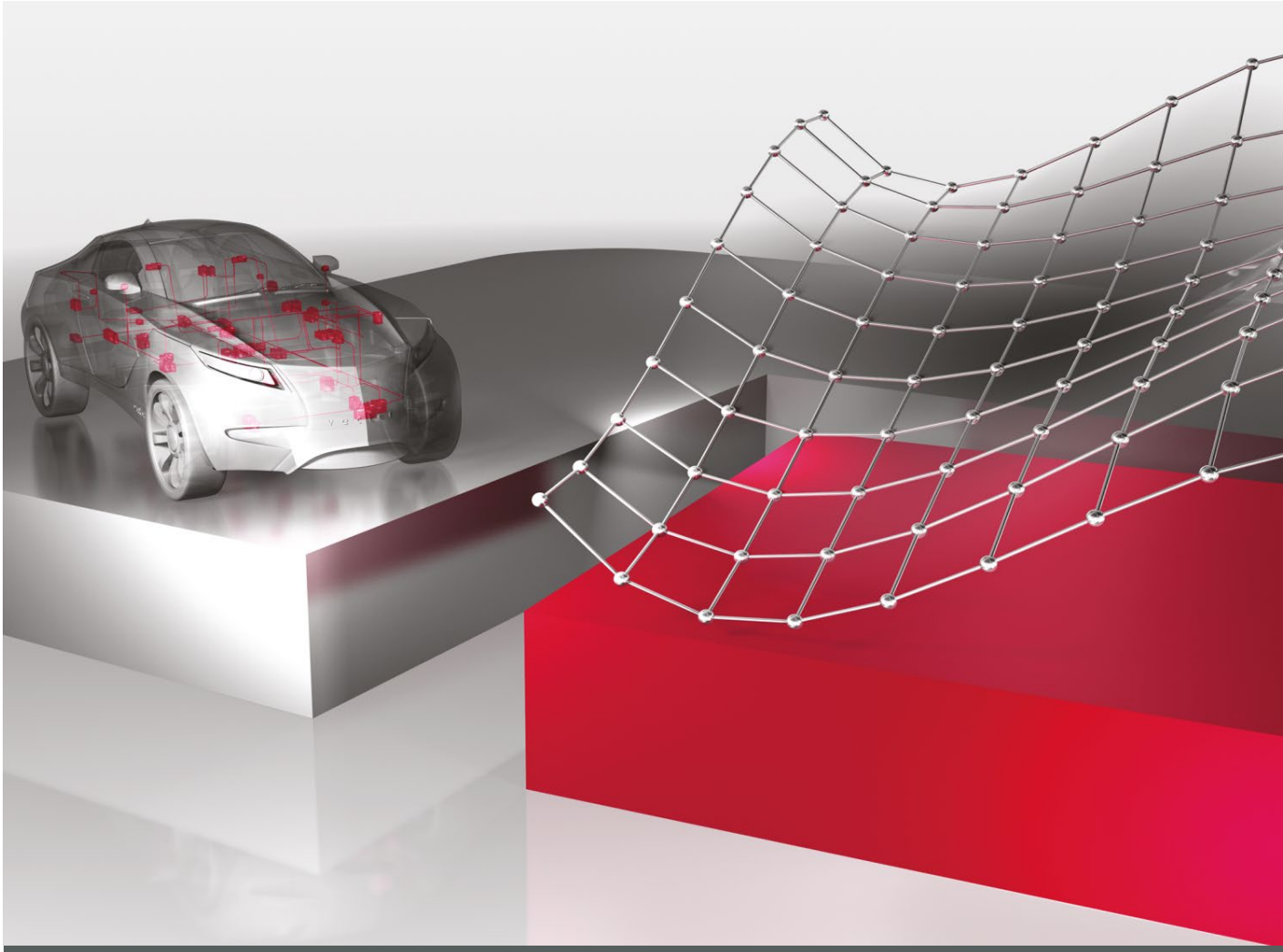
VECTOR



MDF4 Lib

Product Information

Table of Contents

V2.2 7/2019 – Valid for MDF4 Lib as of version 1.7.

1 Overview

1.1 Introduction

MDF (Measurement Data Format) is a binary file format for measurement data that was developed by Vector in 1991 in cooperation with Robert Bosch GmbH. After the automotive industry quickly adopted the MDF format as a defacto standard, a revised Version 4.0 was released as an official ASAM standard in 2009. Current state is Version 4.1 which is available since 2012; a new version 4.2 is planned for 2019.

MDF is primarily used to save measurement data when measuring and calibrating ECUs. The compact binary format facilitates very good performance in both writing and reading, even with enormous amounts of data. MDF files contain the raw measurement data acquired during a measurement and all supplemental information needed to interpret the raw data.

The measurement data can be saved with different, equidistant or non-equidistant sampling rates. The extendability of MDF, especially starting with version 4.x, makes it very easy to store other information within the same file, e.g. to log measurement conditions.

1.2 Application Areas

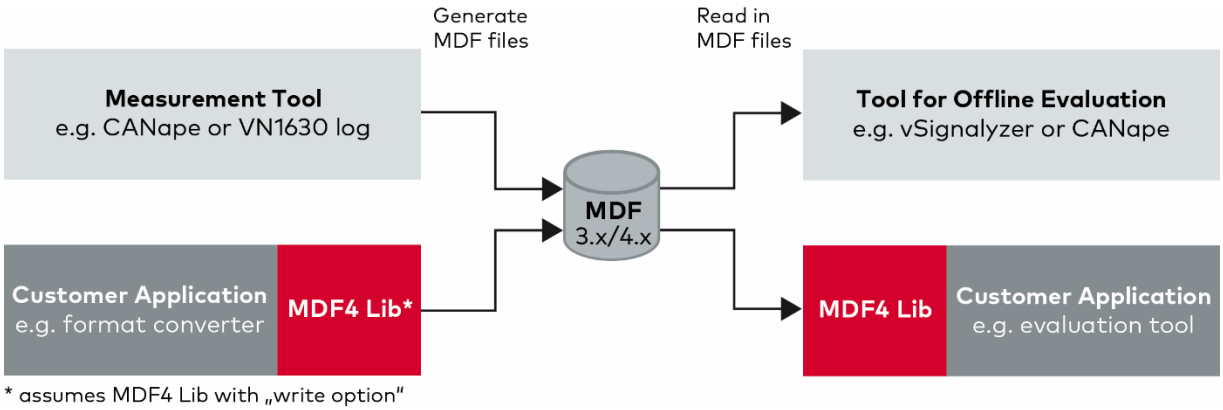
MDF4 Lib is a powerful function library; in its base variant you can use it to validate and sort MDF files as well as read them into your own applications. The new ASAM-standardized MDF4 is supported in addition to the widely used MDF3 format. The library offers a convenient interface, which you can use to access signal data and supplemental information in an MDF file, regardless of the specific MDF version (3.x/4.x).

The interface is available both for C++ and Python on Windows and Linux and for .NET on Windows. The base variant of the MDF4 Lib can be extended by a "write option" that makes it possible not only to read, but also to generate MDF files. As is the case with reading, all MDF versions are also supported in writing, but when writing an older MDF version certain features may not be available in the selected version. Currently, the MDF4 Lib only supports writing of sorted MDF files (offline use case).

Later on, * will designate the functionality that requires a "write option" and ** the functionality that is only available in Windows.

1.3 Overview of Advantages

- Easy to use function library for generating*, validating**, sorting** and reading MDF files
- Version-independent access to MDF3 and MDF4 formats
- Quick and memory-optimized opening of multiple MDF files
- Efficient reading of signal data for sorted files
- Easy generation of MDF objects and writing of signal data for the offline use case*
- Convenient access to various types of MDF meta information
- Seamless data transfer from Vector tools and data loggers, which offer MDF as an output or export format
- Use of a tested and practice proven standard component reduces effort for training, development, testing and maintenance



* assumes MDF4 Lib with „write option"

Figure 1: Function library MDF4 Lib integrates measurement data formats like MDF3 and MDF4 easily and quickly in your applications.

2 Features and Advantages

Multiple MDF files may be opened or generated* starting from a central File Manager Object. Related sub-objects might also be read or written, depending on the desired information. The specific objects and their methods are accessed via interface pointers. This lets you evaluate the MDF features independent of the MDF version.

For quick access to signal values, the MDF files can be sorted** before opening. In writing*, the MDF4 Lib only supports generation of already sorted MDF files, i.e. only records for one sampling rate may be written at a time (offline use case).

MDF4 Lib can be used to validate** MDF files, i.e. check them for possible format errors or inconsistencies, e.g. before being imported into an archiving system. The object orientation and hierarchical structuring of the interface (API) lets users quickly learn and intuitively apply the available functionality. A detailed online Help and sample projects with documented source code permit quick training without requiring detailed knowledge of the MDF format. IntelliSense - which is available in modern development environments such as MS Visual Studio - also supports you in quick and convenient coding.

The generation of objects "on demand" and their management by "reference counting" permit efficient and resource-economic implementations. In C++, when the supplied "smart pointer" class is used, it is even possible to omit an explicit release of the objects. This approach helps to avoid resource leaks. It also reduces the scope and complexity of your code, which generally improves its maintainability and readability.

All library calls are thread-safe. The library thus may be used in a multi-threading use case, e.g. to evaluate each MDF file in a separate thread. Additional optimizations may be enabled for this case.

For diagnostics, the logging capabilities of MDF4 Lib can be used to log any error and warning messages or optionally all library calls. In addition to logging these messages to a file, they alternatively may be a collected by a callback routine, e.g. for custom evaluation or display.

If the writing* of an MDF file is terminated unexpectedly, e.g. due to power off or a tool crash, it might no longer be possible to properly finalize the file. The MDF4 Lib marks such an MDF file as "unfinalized" according to ASAM COMMON MDF 4.1.1. During read-in, the MDF4 Lib recognizes MDF files that are marked as unfinalized. MDF4 Lib might be able to finalize** such a file at a later time, depending on the tool that generated it or the type of finalization steps that are missing.

The MDF4 Lib is available both as Unicode and as Multi-Byte String (MBCS) variant on Windows. You can use the variant that matches your project settings to avoid time-consuming and cumbersome string conversions. (Strings are exclusively processed in Unicode UTF-8 format in Linux).

2.1 Supported MDF Versions

- MDF 2.0 to MDF 2.1
- MDF 3.0 to MDF 3.3
- MDF 4.0 to MDF 4.1

Because of the special properties of the MDF format, MDF files of a future version 3.x/4.x will also be readable, provided that newly introduced features do not explicitly prevent this. Of course, access to the new features is not possible until the MDF4 Lib is updated.

3 Functional Features for Reading MDF Files

You can read the following MDF properties:

- Format information
- Standard and user-specific file comments
- Start time stamp of measurement
- File history
- Channel groups
- Channel properties such as name, comment, source information, units, conversion rules and hierarchical structures as well as array properties (e.g. axes assignments)
- Attachments (properties and data) including saving the attachment to a new file

- Events (time stamp and properties) such as trigger events
- All conversion rule types specified in MDF 3.3 and MDF 4.1 are supported
- Conversion of raw values into physical values/strings
- Source information for channels and channel groups

3.1 Functions for Reading the Signal Data

- Convenient iteration over all samples by auto-increment
- Explicit navigation in the time series for sorted files with positioning by index or time stamp.
- Reading of the time stamp in seconds or nanoseconds
- Reading of signal values as raw or physical value
- Numeric values can be read as DOUBLE or INT64, depending on the data type (avoids rounding errors)
- String values are converted to the relevant variant (Unicode/MBCS)
- Raw values of complex data types may be queried as a byte array
- Reading of a value array (time series) with a single call
- Support of signals with variable length (feature of MDF4.0)
- Querying of the invalidation bit (feature of MDF4.0)
- Reading of compressed data (feature of MDF 4.1)

4 Functional Features for Generating MDF Files*

You can write the following MDF features to a newly generated MDF file:

- Start time stamp of measurement
- Channel groups
- Channels and their properties including their hierarchical structures
- Array channels including record layout and axes assignments
- Source information for channels and channel groups
- Conversion rules (all types specified in MDF 3.3 and MDF 4.1)
- Comments for the measurement file and all other objects that support comments
- Attachments (as external reference or as embedded data)
- Events such as trigger events

4.1 Functions for Writing the Signal Data*

- Setting of individual signal values in the current sample
- Setting of invalidation bits (MDF4 feature from MDF 4.0)
- As an alternative: efficient writing of the entire record assembled by your individual code
- Support of signals with variable length (MDF4 feature from MDF 4.0)
- Compression of signal data (MDF4 feature from MDF 4.1)
- Sorted writing (only offline use case, i.e. only records for one sampling rate may be written at a time)
- Monitoring of file size

5 Functional Features for Changing MDF Files*

You can change the following MDF features when modifying an existing MDF file:

- Start time stamp of measurement
- Add or change comments (except in file history)
- Add new channel groups
- Add new channels
- Add source information for channels and channel groups
- Add conversion rules
- Add and remove attachments
- Add signal data to channel groups that do not contain any samples yet

Functions for writing the signal data are listed in section 4.1.

6 Other Functions**

- Unsorted MDF files can be sorted before opening. This enables direct access to a certain index or time instant within a time series.
- MDF files can be validated before opening them. Errors and warnings are reported via a callback interface.
- MDF4 Lib recognizes MDF files that were marked as "unfinalized" during writing*. It might be able to finalize such a file, depending on the tool used to generate them and the required finalization steps.

7 More Information

7.1 Maintenance Contract

When purchasing the MDF4 Lib, it is possible to purchase a maintenance contract. Within the one-year period of the maintenance contract, updated versions of the MDF4 Lib can be obtained at no additional cost. An existing maintenance contract may be extended by one year.

7.2 Scope of Delivery for Windows

- C++ function library as Windows-DLL in four variants (Unicode/MBCS and 32-/64-bit), C++ sample applications as Visual Studio projects with documented source code and an online Help with introduction, tutorials and detailed interface description
- .NET interface as Windows assembly build upon the C++ library, including a related C# sample application as Visual Studio project and a dedicated online Help function
- Python interface build upon the C++ library as a Python library (.pyd), including related examples as py files and a dedicated online Help function

7.3 Scope of Delivery for Linux

- C++ function library as a Shared Library (.so) in the variant Unicode UTF-8 64-bit, C++ sample applications as CMake project with documented source code and an online Help with introduction, tutorials and detailed interface description
- Python interface build upon the C++ library as a Shared Library (.so), including related examples as py files and a dedicated online Help function

7.4 System Requirements for Windows

The following system requirements must be satisfied to use the MDF4 Lib on Windows:

- C++ Interface: Microsoft Visual C++ Version 6.0 or higher
- .NET Interface: Microsoft .NET 4.0 framework or higher and respective .NET language (e.g. Microsoft Visual C#)

- Python interface: Python 3.4 or higher, and NumPy in a version matching the Python version
- Windows 10/8.1/8/7
- For access to XML features with MDF4, in addition the Microsoft MSXML Parser V6 is required

7.5 System Requirements for Linux

To use MDF4 Lib on Linux, the following system requirements must be met:

- C++ Interface: GNU Compiler GCC C/C++ Version 4.8.3 or higher
- Linux system with x86_64 architecture
- Required system libraries: libc6, libstdc++6, libpthread, libdl
- For access to XML features in MDF4, the XML Parser libxml2 is also required

VECTOR

Get More Information

Visit our website for:

- News
- Products
- Demo software
- Support
- Training classes
- Addresses

www.vector.com