VECTOR

MDF4 Lib

Product Information

vector.com

Contents

V2.6 2026-08

Valid for MDF4 Lib as of version 2.0.

©2026 Vector Informatik GmbH. Any distribution or copying prohibited.

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 de-facto standard, a revised version 4.0 was released as an official ASAM standard in 2009. Current state is version 4.2 which is available since 2019, version 4.3 is foreseen for 2024.

MDF is primarily used to save measurement data in the automotive industry. 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 extendibility 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 and easy-to-use function library for creating, modifying, validating, sorting, and reading MDF in your own applications. It provides a convenient interface to access signal data and supplemental information in an MDF file, regardless of the specific MDF version. It supports the ASAM standard MDF4 as well as the older format MDF3. 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 purchasing a “write option", so that in addition to reading, also the generation and modification of MDF files is possible. As with reading, all MDF versions are supported in writing, although when writing an older MDF version some features may not be available in the selected version. MDF4 Lib only supports writing sorted MDF files.

Later on, * will designate the functionality that requires a “write option”.

1.3 Overview of Advantages

- Easy to use function library for creating*, modifying*, 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
- Transformation* of a MDF 4.x file to MDF 4.1 with loss-less compression of all signal data
- Transformation* of a MDF 4.x file to MDF 4.2 with column-oriented storage of signal data
- Extraction or removal of a time range from a MDF 4.x file
- Seamless data transfer from Vector tools and data loggers, which offer MDF as an output or export format
- The use of a tested and field proven standard component reduces effort for training, development, testing and maintenance

©2026 Vector Informatik GmbH. Any distribution or copying prohibited.

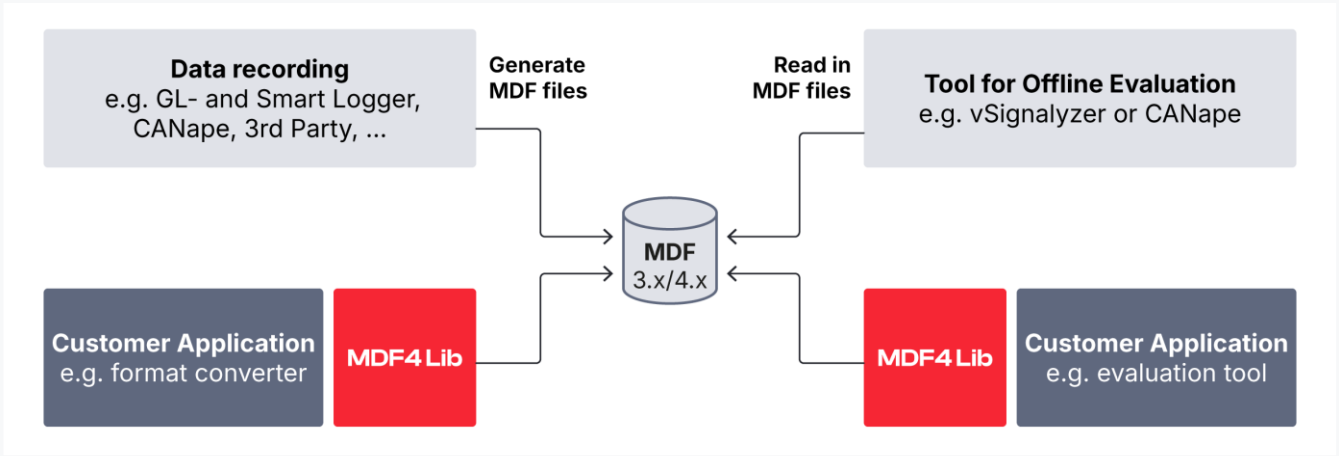

FIGURE 1: Function library MDF4 Lib integrates measurement data formats MDF3 and MDF4 easily and quickly in your applications.

2. Features and Advantages

Starting from a central File Manager object, multiple MDF files can be opened or generated* in parallel. Related sub-objects might also be read or written, depending on the desired information. The elements and data in the file are accessed via objects (Python, .NET) or interface pointers (C++) and their properties and methods. This lets you evaluate the MDF features independent of the MDF version without the need for case distinctions in the code.

For quick access to signal values, the MDF files can be sorted before opening. When writing*, the library always creates sorted MDF files. Starting with MDF4 Lib version 2.0 more than one sampling rate can be written at a time.

Validation or checking of MDF files for possible format errors is easily possible, e.g. before import into an archiving system.

Thanks to the object orientation and hierarchical structuring of the interface (API), the available functionality can be quickly learned and intuitively used. An extensive online help and sample projects with documented source code allow a quick orientation without detailed knowledge of the MDF format. IntelliSense - which is available in modern development environments such as MS Visual Studio - also supports you in coding quickly and conveniently.

The generation of objects “on demand” and their management by “reference counting” enable an efficient and resource saving implementation. To avoid resource leaks in C++, an explicit release of objects can be omitted when using the supplied "smart pointer" class. This also reduces the size and complexity of your code, which usually improves maintainability and readability.

All library calls are thread-safe. Additional optimizations can be enabled for the optional multi-threading use case. This allows parallel processing of tasks, e.g. for evaluating each MDF file in a separate thread.

For error diagnosis, any error and warning messages or optionally all library calls can be logged to a file using the logging options. Alternatively, the messages can also be recorded via callback e.g. for your own evaluation or display.

If the writing* of an MDF file using MDF4 Lib is terminated unexpectedly, e.g. due to power off or a tool crash, it might no longer be possible to properly finalize the file. In this case, the uncomplete MDF file will be marked as “unfinalized” according to ASAM COMMON MDF 4.1.1. During read-in, the MDF4 Lib recognizes this and is capable to finalize such a file later. Depending on the type of finalization steps that are missing, this also is possible for MDF file created by 3rd party tools

In addition, the library offers a comfortable method to re-write a MDF file, for instance to minimize its file size (remove unused ranges in the file, loss-less compression of signal data for MDF 4.1 or higher), or to optimize data storage for faster read access (column-oriented storage of signal data for MDF 4.2 or higher).

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).

©2026 Vector Informatik GmbH. Any distribution or copying prohibited.

2.1 Supported MDF Versions

> MDF 2.0 to MDF 2.1
> MDF 3.0 to MDF 3.3
> MDF 4.0 to MDF 4.2

Because of the special properties of the MDF format, MDF files of a future version 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 4.2 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, including new compression algorithms foreseen for MDF 4.3)
> Reading of data with column-oriented storage (feature of MDF 4.2)
> Reading of event signals (feature of MDF 4.2)

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 4.2)
> 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

©2026 Vector Informatik GmbH. Any distribution or copying prohibited.

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, including new compression algorithms foreseen for MDF 4.3)
> Writing of signal data with column-oriented storage (MDF4 feature from MDF 4.2)
> Sorted writing (starting with MDF4 Lib 2.0 multiple sampling rates 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
> Comments (except in file history)
> Add new channel groups and channels
> Change various properties of channel groups and channels
> Add source information for channels and channel groups
> Add conversion rules
> Add, remove, or replace attachments
> Add signal data to channel groups that do not contain any samples yet
> Add signal to an existing channel group that already contains samples using column-oriented storage (MDF4 feature from MDF 4.2)

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.
> MDF files can be re-written. This can serve to close "gaps", i.e. remove unused byte ranges in the file to minimize the file size. Optionally, a time range can be extracted or removed from the file. In addition, the storage of the signal data can be changed, e.g. by using loss-less compression (MDF 4.1 or higher) or column-oriented storage (MDF 4.2 or higher).

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

©2026 Vector Informatik GmbH. Any distribution or copying prohibited.

- .NET interface build upon the C++ library as separate Windows assemblies for .NET Framework and .NET Core, 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), as well as a Python package build upon this for even more coding comfort, 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), as well as a Python package build upon this for even more coding comfort, 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 (other compilers are possible but not tested)
- .NET Interface: Microsoft .NET Framework 4.0 or higher and respective .NET language (e.g. Microsoft Visual C#)
- .NET Core Interface: Microsoft .NET Core 3.1 or higher and respective .NET language (e.g. Microsoft Visual C#)
- Python interface: Python 3.7 (64 bit) or higher, and NumPy in a version matching the Python version
- Windows 11/10/8 (64 bit)
- 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
- Python interface: Python 3.7 (64 bit) or higher, and NumPy in a version matching the Python version
- 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

©2026 Vector Informatik GmbH. Any distribution or copying prohibited.

MORE INFORMATION

Visit our website for:

- News
- Products
- Demo software
- Support
- Training and Workshops
- Contact addresses

vector.com

©2026 Vector Informatik GmbH. Any distribution or copying prohibited.