Vlinder Software

1340 avenue de Gaudarville
Québec, Québec
Canada G2G 2K6

RTIMDB: Vlinder Software's Real-Time In-Memory Database

Integration guide

Vlinder Software

1340 avenue de Gaudarville
Québec, Québec
Canada G2G 2K6

Copyright notice

Disclaimers

This document is provided "AS-IS" for information purposes only. No guarantees, representations and/or warranties, either express or implied, are given for the contents, nor its accuracy, freedom from errors or fitness for any particular purpose.

Vlinder Software specifically disclaims any liability with respect to this document and no obligations, contractual or otherwise, can arise from this document.

Vlinder Software may modify the contents of this document at any time, without prior notice.

It is possible that use of the technical matter published in this document may require the permission of the proprietor of one or more patents. You are entirely response for identifying and where necessary obtaining a licence under such patents should you choose to use any such technical matter. Vlinder Software has no responsibility in this regard and shall not be liable for any loss or damage suffered in relation to an infringement of any third party patent as a result of such use.

Vlinder Software

1340 avenue de Gaudarville
Québec, Québec
Canada G2G 2K6

1 Introduction

Vlinder Software's Real-Time In-Memory Database was designed for integration for embedded systems communicating using one of the popular SCADA protocols, such as DNP3. As such, the principal use-cases considered in the design consist of the device receiving control requests (select, operate, direct-operate, write), freeze requests (immediate freeze, freeze-and-clear, freeze-at-time[1]) and read requests.

A typical device would have a few hundred data points, each of which represented with a type and an index. Some systems may give unique names to their data points. However, RTIMDB currently only supports accessing points through their type and index.

1 The database does not manage time itself: if the device receives a freeze-at-time request, it should buffer the request and translate it to an immediate freeze at the given time.

Vlinder Software

1340 avenue de Gaudarville
Québec, Québec
Canada G2G 2K6

2 Table of Contents

Vlinder Software

1340 avenue de Gaudarville
Québec, Québec
Canada G2G 2K6

3 Details

We assume the following:

1. `int` is at least 32 bits wide;
2. the alignment requirements for double are sufficient for all intrinsic types in the system.

Although this list may not be exhaustive, we try to avoid all other assumptions about the system.

4 Compiling RTIMDB

RTIMDB is written in ISO C++11 and uses features that are not available in some older compilers.

The preferred way to compile RTIMDB is by generating the Makefiles, Solution files, or other files for your build system using CMake.

When generating the configuration for RTIMDB, the following values need to be filled in:

Configuration value	Description	Default value
`RTIMDB_MAX_CONCURRENT_TRANSACTIONS`	Maximum number of concurrent read transactions (for freezes during polls)	5
`RTIMDB_POINT_COUNT`	Total number of points in the system (maximum)	200
`RTIMDB_ALLOW_EXCEPTIONS`	Should exceptions be thrown from the library, on error	ON

If configured to support exceptions, no-throw versions of all functions are available by appending std::nothrow to the parameter list. When configured to not allow exceptions, the no-throw versions become the default versions. The no-throw versions of the functions return an error code.

If you do wish to compile RTIMDB “by hand”, you should do the following:

1. copy rtimdb_config.hpp.in to rtimdb_config.hpp;
2. open rtimdb_config.hpp in an editor;
3. fill in the values for the above-mentioned configuration values.

4.1 Compiler warnings in Microsoft Visual Studio

The Microsoft C++ compiler will emit two types of warnings: warning number 4251 and warning number 4996. Both can be safely ignored in the vast majority of cases.

Warning C4251: '<member-name>' : class '<template-type>' needs to have dll-interface to be used by clients of class '<type-name>': this warning may be safely ignored if the compiler settings for optimization and alignment are the same for both the RTIMDB DLL and the client code. In other cases, the size and layout of template-type objects may be different in the client code vs. the DLL's code.

If you prefer, there are two work-arounds for this case: either compile the RTIMDB library as a static library, or create an instance of the template, specialized as specified in the warning message, and export it from the DLL. This latter option is not portable to non-Windows platforms, which is why we prefer not implementing it.

We recommend making sure that compiler settings are compatible throughout your product's code-base.

Warning C4996: '<function-name>': Function call with parameters that may be unsafe - this call relies on the caller to check that the passed values are correct (and other variants): Microsoft has "deprecated" a number of standard functions. The use of the term "deprecated" is unfortunate, as Microsoft does not have the authority to actually deprecate parts of an ISO standard. They provide non-portable, "safer" alternatives. However, these "safer" alternatives generally do not provide for better safety and are not portable to other platforms.

It is true that the standard algorithms rely on the programmer to ascertain the validity of the arguments to the function – else their behaviour is generally undefined. Static code analysis, systematic code review and separate security review of all Vlinder Software code ascertains the correctness of the code.

While annoying, this warning may therefore be ignored.

4.2 Compiler settings with GCC

If during compilation, GCC complains that C++11 is needed (i.e. fails to compile in one of its headers because that header is restricted to C++11): you need use a version of GCC that has support for C++11 (i.e. 4.8 or later) and tell it to use it on the command-line by adding "-std=c++11" to its arguments. The CMake-generated Makefiles will do this automatically.

Vlinder Software

1340 avenue de Gaudarville
Québec, Québec
Canada G2G 2K6

5 Using the RTIMDB API

The database expects to be used in two phases: a *configuration* phase and an *execution* phase. During the configuration phase, you insert the data points for your device into the database and register any observers and filters you may want to register.

5.1 Rules for filters

The database allows for one filter per data point. The filter will be given the action being performed (write, select, operate, direct operate, freeze or freeze-and-clear) the old value and the new value. Installing a filter on a point removes the default filter for that point. Default filters are:

- allow write only for strings and datasets;
- allow freeze and freeze-and-clear on counters and analog inputs;
- allow select, operate and direct operate on binary outputs and analog outputs.

You are advised to make your filters purely functional: if the filter for any point change in a transaction returns false, the transaction is rolled back. If filters have side-effects, those side-effects will not be rolled back, which may result in unwanted behaviour (e.g. your application thinking an action has taken place while it hasn't).

Calls to `Database::update` are not filtered: `Database::update` is intended to be used by the application internally to advise the library of a change in the status (i.e. value) of a point. The control methods are intended for use by a protocol. It is therefore assumed that your application has pre-approved the transaction taking place, and it will therefore take place.