**README libIEC61850**

This file is part of the documentation of **libIEC61850**. More documentation can be found online at [http://libiec61850.com](http://libiec61850.com/).

The API documentation can be found here:

* C API: <https://support.mz-automation.de/doc/libiec61850/c/latest/>
* .NET API: <https://support.mz-automation.de/doc/libiec61850/net/latest/>

Also consider to review the examples to understand how to use the library.

Content:

* [Overview](https://stackedit.io/app#overview)
* [Features](https://stackedit.io/app#features)
* [Examples](https://stackedit.io/app#examples)
* [Building and running the examples](https://stackedit.io/app#building-and-running-the-examples-with-the-provided-makefiles)
* [Building the library with TLS support](https://stackedit.io/app#building-the-library-with-tls-support)
* [Installing the library and the API headers](https://stackedit.io/app#installing-the-library-and-the-api-headers)
* [Building on Windows with GOOSE support](https://stackedit.io/app#building-on-windows-with-goose-support)
* [Building with the cmake build script](https://stackedit.io/app#building-with-the-cmake-build-script)
* [Using the log service with sqlite](https://stackedit.io/app#using-the-log-service-with-sqlite)
* [C# API](https://stackedit.io/app#c-api)
* [Experimental Python bindings](https://stackedit.io/app#experimental-python-bindings)
* [Licensing](https://stackedit.io/app#commercial-licenses-and-support)
* [Contributing](https://stackedit.io/app#contributing)
* [Third-party contributions](https://stackedit.io/app#third-party-contributions)

**Overview**

libiec61850 is an open-source (GPLv3) implementation of an IEC 61850 client and server library implementing the protocols MMS, GOOSE and SV. It is implemented in C (according to the C99 standard) to provide maximum portability. It can be used to implement IEC 61850 compliant client and server applications on embedded systems and PCs running Linux, Windows, and MacOS. Included is a set of simple example applications that can be used as a starting point to implement own IEC 61850 compliant devices or to communicate with IEC 61850 devices. The library has been successfully used in many commercial software products and devices.

For commercial projects licenses and support is provided by MZ Automation GmbH. Please contact [info@mz-automation.de](mailto:info@mz-automation.de) for more details on licensing options.

**Features**

The library support the following IEC 61850 protocol features:

* MMS client/server, GOOSE (IEC 61850-8-1)
* Sampled Values (SV - IEC 61850-9-2)
* Support for buffered and unbuffered reports
* Online report control block configuration
* Data access service (get data, set data)
* online data model discovery and browsing
* all data set services (get values, set values, browse)
* dynamic data set services (create and delete)
* log service  
  \*\* flexible API to connect custom data bases  
  \*\* comes with sqlite implementation
* MMS file services (browse, get file, set file, delete/rename file)  
  \*\* required to download COMTRADE files
* Setting group handling
* GOOSE and SV control block handling
* TLS support
* C and C#/.NET API

**Examples**

The examples are built automatically when CMake is used to build the library.

NOTE: Most examples are intended to show a specific function of the library. They are designed to show this function as simple as possible and may miss error handling that has to be present in real applications!

**Building and running the examples with the provided makefiles**

In the project root directory type

make examples

If the build succeeds you can find a few binary files in the projects root directory. You can also find a binary version of the library (“libiec61850.a”) in the “build” directory.

Run the sample applications in the example folders. E.g.:

cd examples/server\_example\_basic\_io

sudo ./server\_example\_basic\_io

on the Linux command line.

You can test the server examples by using a generic client or the provided client example applications.

**Building the library with TLS support**

Download, unpack, and copy mbedtls-2.16 into the third\_party/mbedtls folder.

NOTE: The current version support mbedtls version 2.16. When you download the source archive from <https://tls.mbed.org/> you have to rename the extracted folder to “mbedtls-2.16”.

In the main libiec61850 folder run

make WITH\_MBEDTLS=1

When using CMake the library is built automatically with TLS support when the folder third\_party/mbedtls/mbedtls-2.16 is present.

**Installing the library and the API headers**

The make and cmake build scripts provide an install target. This target copies the API header files and the static library to a single directory for the headers (INSTALL\_PREFIX/include) and the static library (INSTALL\_PREFIX/lib). With this feature it is more easy to integrate libiec61850 in an external application since you only have to add a simple include directory to the build tool of your choice.

This can be invoked with

make install

The default install directory for the make build script is “.install”.

You can modify this by setting the INSTALL\_PREFIX environment variable (e.g.):

make INSTALL\_PREFIX=/usr/local install

For the cmake build script you have to provide the CMAKE\_INSTALL\_PREFIX variable

**Building on windows with GOOSE support**

To build the library and run libiec61850 applications with GOOSE support on Windows (7/8/10) the use of a third-party library (winpcap) is required. This is necessary because current versions of Windows have no working support for raw sockets. You can download winpcap here ([http://www.winpcap.org](http://www.winpcap.org/)).

1. Download and install winpcap. Make sure that the winpcap driver is loaded at boot time (you can choose this option at the last screen of the winpcap installer).
2. Reboot the system (you can do this also later, but you need to reboot or load the winpcap driver before running any llibiec61850 applications that use GOOSE).
3. Download the winpcap developers pack from here (<http://www.winpcap.org/install/bin/WpdPack_4_1_2.zip>)
4. Unpack the zip file. Copy the folders Lib and Include from the WpdPack directory in the third\_party/winpcap directory of libiec61850

**Building with the cmake build script**

With the help of the cmake build script it is possible to create platform independent project descriptions and let cmake create specific project or build files for other tools like Make or Visual Studio.

If you have cmake installed fire up a command line (cmd.exe) and create a new subdirectory in the libiec61850 folder. Change to this subdirectory. Then you can invoke cmake. As an command line argument you have to supply a “generator” that is used by cmake to create the project file for the actual build tool (in our case Visual Studio 2015).

cmake -G "Visual Studio 14 2015" ..

will instruct cmake to create a “solution” for Visual Studio 2015. The resulting project files will be 32 bit.

To build 64 bit libraries the “Win64” generator option has to be added.

cmake -G "Visual Studio 14 2015 Win64" ..

Note: The “…” at the end of the command line tells cmake where to find the main build script file (called CMakeLists.txt). This should point to the folder libiec61850 which is in our case the parent directory (…).

Depending on the system you don’t have to provide a generator to the cmake command.

To select some configuration options you can use ccmake or cmake-gui.

**Using the log service with sqlite**

The library provides support for the IEC 61850 log service. It provides an abstract interface for a logging database. Included is a driver for using sqlite for logging. This driver can be seen as an example on how to use the abstract logging interface.

You can use the driver by including the src/logging/drivers/sqlite/log\_storage\_sqlite.c file into your application build.

On Ubuntu Linux (and simpilar Linux distributions) it is enough to install the sqlite dev packages from the standard repository. For other OSes (e.g. Windows) and cross-compiling it is recomended to download the amalagation source code (from <https://www.sqlite.org/download.html>) of sqlite and copy them to the third\_party/sqlite folder.

On windows the cmake skript will detect the sqlite source code and also creates the example project for logging.

**C# API**

A C#/.NET wrapper and examples and Visual Studio/MonoDevelop project files can be found in the dotnet folder. The examples and the C# wrapper API can be build and run on .NET or Mono.

**Experimental Python bindings**

The experimental Python binding can be created using SWIG with cmake.

To enable the bindings you have to select the phyton configuration option with ccmake of cmake-gui.

We don’t provide any support for the Python bindings!

**Commercial licenses and support**

Support and commercial license options are provided by MZ Automation GmbH. Please contact [info@mz-automation.de](mailto:info@mz-automation.de) for more details.

**Contributing**

If you want to contribute to the improvement and development of the library please send me comments, feature requests, bug reports, or patches. For more than trivial contributions I require you to sign a Contributor License Agreement. Please contact [info@libiec61850.com](mailto:info@libiec61850.com).

Please don’t send pull requests before signing the Contributor License Agreement! Such pull requests may be silently ignored.

**Third-party contributions**

* The Mac OS X socket and ethernet layer has been kindly contributed by Michael Clausen, HES-SO Valais-Wallis, [http://www.hevs.ch](http://www.hevs.ch/)