# License C++ library

**v1.1.1**

# Table of contents



# Overview

**License** C++ library version **1.1.1** offers a licensing feature for C++ applications. The library provides functions for generating a token (base on username and MAC address of the computer), creating a license based on it, and validating the license with control of its validity time. **AES256** encryption method is implemented. To embed licensing in a C++ application, generate a token, then based on the token and the required number of days generate a license string (key). License checking in the application is performed by a single library function. The library **doesn't have any third-party dependencies** and provide simple interface. It compatible with Linux and Windows.

# Versions

**Table 1** - Library versions.

| Version | Release date | What's new |
|---|---|---|
| 1.0.0 | 31.01.2022 | First version. |
| 1.1.0 | 28.09.2023 | - Hardware dependency moved to MAC address from CPU ID.<br>- Documentation updated. |
| 1.1.1 | 05.01.2024 | - Code style issues fixed.<br>- Documentation updated.<br>- Only AES256 algorithm is used. |

# Library files

The library is supplied as source code only. The **License** library is a CMake project. Library files:

```
CMakeLists.txt ----------- Main library CMake file
README.md ---------------- Documentation
src ---------------------- Folder with library source code
    CMakeLists.txt ------- Library CMake file
    License.cpp ---------- Library source code file
    License.h ------------ License class header file
    LicenseVersion.h ----- Header file with License class version
    LicenseVersion.h.in -- CMake service file to version file
    AES.cpp -------------- AES encryption source code file
    AES.h ---------------- AES encryption header file
examples ----------------- Folder with examples
    CMakeLists.txt ------- CMake file for examples
    CombinedTest --------- Folder with License library test
        MakeLists.txt ---- Application CMake file
        main.cpp --------- Application source code file
    LicenseGenerator ----- Folder with license generator
        MakeLists.txt ---- Application CMake file
        main.cpp --------- Application source code file
    LicenseTest ---------- Folder with license validator
        MakeLists.txt ---- Application CMake file
        main.cpp --------- Application source code file
    TokenGenerator ------- Folder with token generator
        MakeLists.txt ---- Application CMake file
        main.cpp --------- Application source code file
```

# License class description

## License class declaration

**License** interface class declared in **License.h** file. Class declaration:

```
class License
{
public:
    /// Get License class version.
    static std::string getVersion();

    /// Create token based on hardware and user name.
    std::string getToken(std::string userName);

    /// Extract information from a token.
    bool getTokenInfo(std::string token, std::string &userName,
                      std::string &macAddress);

    /// Init a license.
    bool getLicense(std::string token, int days, std::string &license);

    /// Extract info from a license.
    bool getLicenseInfo(std::string license, std::string &userName,
                        std::string &macAddress, int &daysLeft);

    /// Check status of license.
    bool checkLicense(std::string license, std::string userName);
};
```

## getVersion method

**getVersion()** method returns string of current class version. Method declaration:

```
static std::string getVersion();
```

Method can be used without **License** class instance:

```
std::cout << "License class version: " << License::getVersion() << std::endl;
```

Console output:

```
License class version: 1.1.1
```

# getToken method

**getToken(...)** generates token string based on username and MAC address of the computer. Method declaration:

```
std::string getToken(std::string userName);
```

| Parameter | Value |
|---|---|
| userName | User name to generate token. |

**Returns:** token string.

# getTokenInfo method

**getTokenInfo(...)** method extracts username and MAC address from token string. Method declaration:

```
bool getTokenInfo(std::string token, std::string &userName, std::string &macAddress);
```

| Parameter | Value |
|---|---|
| token | Token to get information from. |
| userName | Output user name string. |
| macAddress | Output MAC address. |

**Returns:** TRUE if the information extracted or FALSE if not.

# getLicense method

**getLicense(...)** generates license from token string and number of days:

```
bool getLicense(std::string token, int days, std::string &license);
```

| Parameter | Value |
|---|---|
| token | Token string in order to create a license. |
| days | License validity period in days. |
| license | Output license string. |

**Returns:** TRUE if it success or FALSE if not.

## getLicenseInfo method

**getLicenseInfo(...)** extracts information (username, MAC address, days left) license string. Method declaration:

```
bool getLicenseInfo(std::string license, std::string &userName, std::string &macAddress,
int &daysLeft);
```

| Parameter | Value |
|---|---|
| licence | License to get information from. |
| userName | Output username string. |
| macAddress | Output MAC address. |
| daysLeft | Remaining days of license. |

**Returns:** TRUE if the information extracted or FALSE if not.

## checkLicense method

**checkLicense(...)** check license status (valid of not valid) based on license string and username. Method declaration:

```
bool checkLicense(std::string license, std::string userName);
```

**Returns:** TRUE if licence is valid and not expired or FALSE if not valid (invalid license string or expired).

# Usage

To add licensing to your application, you need to implement the following logic. Suppose that you have sent a finished application to a user. The application on the user side should generate a token based on the username (the user must enter the username or the username can be pre-defined). Generating the token inside your application is done as follows:

```
std::string userName = "AnyUserName"; // Set by user.
cr::utils::License lic;
std::string token = lic.getToken(userName);
```

When the token is generated the user will send you the token string. Based on the token string and required number of days, you generate the license string. Generating the license string on your side is done as follows:

```
std::string license = ""; // Variable to store license string.
std::string token =
"0255D7EE2DFCFF806359D19B494283FF4329E8B65BC508C761151FC4DDDB7B4B"; // Example.
int days = 20; // License for 20 days, example.
if (!lic.getLicense(token, days, license))
{
    std::cout << "Invalid token for " <<
    userName << " !" << std::endl;
    return -1;
}
```

When the licence string is generated you send it to the user who enters it in your application. This licence can be saved in any application configuration file. The licence validity check (correct licence and number of licence days) is done inside your application (username and licence string are required) using the following code:

```
std::string license =
"EE8C15F833832FF0BD9EAEA2F5FF3B0C277585F342EE404EE8137A26CDC3FFCEB8778CC0DE3F3A045E1F7AC2
8E0AAC7A3CC3BD0C75C6D567B94F1F7D13CC1B201"; // Example.
std::string userName = "AnyUserName";
cr::utils::License lic;
if (!lic.checkLicense(license, userName))
{
    std::cout << "Invalid license for " <<
    userName << " !" << std::endl;
    return -1;
}
std::cout << "License valid!" << std::endl;
```

# Build and connect to your project

Typical commands to build **License** library:

```
cd License
git submodule update --init --recursive
mkdir build
cd build
cmake ..
make
```

If you want connect **License** library to your CMake project as source code you can make follow. For example, if your repository has structure:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
```

Create **3rdparty** folder in your code repository and copy License repository to **3rdparty** folder. New structure of your repository:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    License
```

Create CMakeLists.txt file in **3rdparty** folder. CMakeLists.txt should contain:

```
cmake_minimum_required(VERSION 3.13)

###############################################################################
## 3RD-PARTY
## dependencies for the project
###############################################################################
project(3rdparty LANGUAGES CXX)

###############################################################################
## SETTINGS
## basic 3rd-party settings before use
###############################################################################
# To inherit the top-level architecture when the project is used as a submodule.
SET(PARENT ${PARENT}_YOUR_PROJECT_3RDPARTY)
# Disable self-overwriting of parameters inside included subdirectories.
SET(${PARENT}_SUBMODULE_CACHE_OVERWRITE OFF CACHE BOOL "" FORCE)

###############################################################################
## CONFIGURATION
## 3rd-party submodules configuration
###############################################################################
SET(${PARENT}_SUBMODULE_LICENSE                         ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_LICENSE)
    SET(${PARENT}_LICENSE                               ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_LICENSE_EXAMPLES                      OFF CACHE BOOL "" FORCE)
endif()

###############################################################################
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
###############################################################################
if (${PARENT}_SUBMODULE_LICENSE)
    add_subdirectory(License)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **License** to your project and excludes examples from compiling. Your repository new structure will be:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    CMakeLists.txt
    License
```

Next you need include **3rdparty** folder in main **CMakeLists.txt** file of your repository. Add string at the end of your main **CMakeLists.txt**:

```
add_subdirectory(3rdparty)
```

Next you have to include License library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} License)
```

Done!

# Example

This example illustrates the process of generating a token, creating a license, and verifying the license status.

```
#include <iostream>
#include "License.h"

int main(void)
{
    std::cout << "LicenseGenerator v" <<
    cr::utils::License::getVersion() << std::endl;

    // Set user name to generate license.
    std::string userName = "";
    std::cout << "Enter the UserName to generate license: ";
    std::cin >> userName;

    // Set license duration.
    int days = 0;
    std::cout << "Enter license duration [days]: ";
    std::cin >> days;

    // Create License object and generate token for license.
    cr::utils::License lic;
    std::string token = lic.getToken(userName);
    std::cout << "Token for " << userName << ": " <<
    token << std::endl;

    // Generate license.
    std::string license = "";
```

```
    if (!lic.getLicense(token, days, license))
    {
        std::cout << "Invalid token for " <<
        userName << " !" << std::endl;
        return -1;
    }

    // Check license.
    if (!lic.checkLicense(license, userName))
    {
        std::cout << "Invalid license for " <<
        userName << " !" << std::endl;
        return -1;
    }

    // Show license.
    std::cout << "License for " << userName <<
    "[" << days << " days]: " << license << std::endl;

    // Wait user input.
    std::cin.get();
    std::cin.get();
    return 1;
}
```