AESCRIPTS.COM
LICENSING LIBRARY/TOOLKIT

version 4.1.4 / April 2025

1. VERSION HISTORY

VERSION	RELEASE DATE	CHANGES
1.0	May 15, 2013	- initial release
1.1	July 12, 2013	- support for differently compiled template libraries
1.2	July 15, 2013	- support for multi-threaded linker settings
1.3	July 30, 2013	- support for Visual Studio 2012
1.4	Sep 3, 2013	- support for Visual Studio 2008 - disabled global optimisations
1.5	Sep 30, 2013	- sample plugin supports Premiere Pro now - better documentation/manual - support for render-only licenses
1.6	Oct 18, 2013	- support for disabled or changing MAC address/NIC - added note about disk cache in CS6+
1.7	Jan 29, 2014	- bugfix for certain MAC/network adapter errors, causing correct license to be seen as invalid
1.7a	Feb 8, 2014	- small bugfix for reading older license files
1.8	April 15, 2014	- support for systems with no ethernet adapters or WiFi only
1.9	May 12, 2014	- Unicode path fixes on Asian systems, static denylist
2.0	July 21, 2014	- dynamic denylist - Premiere Pro compatible UI - code cleanup - Visual Studio 2013 compatibility - license can now be deactivated in the dialog
2.1	Oct 30, 2015	- Visual Studio 2015 compatibility - more flexible license file loading in restricted environments - improved license test AE plugin - MacOS: preferences folder is now "/Library/Application Support"
2.2	Feb 5, 2016	- MacOS: support for libc++ and libstdc++ - improved license test AE plugin - new header: aescriptsLicensing_AEHelpers.h - support for trial licenses (expire after x days)
3.0	July 2017	- big update/rewrite, support for licensing server
3.1	September 2017	- registration dialog now reports if a license does not exist on the server - userID/logged user name bugfix on MacOS

3.2	November 2017	- various bugfixes for "overuse" checks - network adapter detection improved - bugfix for trial days detection - improved error handling
3.3	January 2018	- bugfix for render-only and non single-user licenses
3.4	February 2018	- bugfix for overuse checks - full support for VS2010 (overuse functionality was disabled before for that compiler)
3.5	August 2018	- sanity check for overuse call - support for VS2017 - better error handling
3.6	April 2019	- extended LicenseData with startDate, endDate - field serial in LicenseData expanded to 64 characters - support for time-limited licenses
3.7	August 2019	- support for 16-digit private numbers - automatically try to get floating license if plist is present and in render-only mode - support for reading environment variable for floating license server config - bugfix for servers with space(s) in computername - added check if version of header matches version of library
3.8	September 2019	- support for remote network users who don't have a local home directory - more robust against special characters in user and computer names - support for additional library plist configuration file for license server config - bugfixes for license server communication - improved handling of floating licenses
3.9	May 2020	- blocking of Apple Macbook touchbar network bridges - improved machine ID processing
3.9.1	May 26, 2020	- bugfix for machine id check, could return error -2 (different machine id) on some systems
3.9.2	May 28, 2020	- bugfix for daysSinceFirstStart value
4.0	November 2021	- obsolete function getLicenseFromLicenseServer removed, use requestLicenseFromLicenseServer and dropLicenseFromLicenseServer instead - added description of behaviour for loadLicenseFromFile if no license file is present and floating server is configured - libs now also provided for VS2019 natively - added test license for floating license for test plugin in manual - new function getLicenseDataFromFile - improved wording/description of addBlockedSerial() - MacOS builds now require linking to libcurl - MacOS builds are universal binaries now that support M1/Silicone - MacOS support for libstdc++ was dropped by Apple
4.0.1	December 2021	- bugfix for online license validation - sometimes the request could be rejected by the validation server - support for Visual Studio 2022
4.0.2	January 2022	- overall better error handling - unified error code handling with 'Get Help' links for each error - requestLicenseFromLicenseServer() function now also includes a parameter for product version info - improved license registration dialog

		- new function aescriptsGetLicLibBuildVersion() to get more detailed information about the licensing library version - online validation enabled by default - The defines "AE_LICLIB_VERSION", "AE_LICLIB_BUILD_DATE" and "AE_LICLIB_VERSION_TAG" in the header file have been renamed to "AESCRIPTSLICLIB_VERSION", "AESCRIPTSLICLIB_BUILD_DATE" and "AESCRIPTSLICLIB_VERSION_TAG" - fixed hash license issue
4.0.3	March 2022	- treat empty license files the same as non-existent file for floating licenses - trim whitespaces on entering license - correct error message when trying to use a floating license locally - improved versioning support in registration dialog (now an optional parameter in the showRegistrationDialog() function) - improved online validation server communication - improved backwards compatibility - manual extended with common AE plugin problems - more robust string handling in registration dialog - corrected return code for invalid format of license strings (was -8, is now -1) - simplified licTestPlugin (no longer using sequence data)
4.0.4	April 2022	- bugfix for floating licenses ("invalid format") - improved threading model on Windows - more robust handling of license file saving
4.0.5	August 2022	- support for extended privnums - support for subscription licenses (SUB and ESB) - improved support for grace periods in online activation - various bugfixes for online activation
4.0.6	November 2022	- grace period for online activation is now 90 instead of 30 days - increased compatibility on MacOS with certain Xcode/clang settings - fixed stripping of symbols on MacOS - lowest supported MacOS version is now 10.13
4.0.7	February 2023	- improved stability of overuse check mechanism - fixed release library for VS2022
4.1.0	August 2024	- support for incremental online validation intervals - support for new license type FSU (floating subscription licenses) - support for offline licensing method - new function "aescriptsGetRawLicenseDataFromFile" to get the raw contents of a license file - online activation check interval changed back from 90 to 30 days, will be checked in 30, 30, 90, 180 intervals, then never - overuse check mechanism now terminates correctly in standalone applications
4.1.1	September 2024	- bugfix for name resolution of .local and .localdomain hostnames - improved error checking
4.1.2	October 2024	- bugfix for remote license activation (Error -6 Invalid product header)
4.1.3	October 2024	- bugfix for local license activation when license server is configured
4.1.4	April 2025	- cleanup of licensing sample code and headers - support for newer and older versions of the VS2022 runtime - example for license usage in an AEGP plugin - various security enhancements and bugfixes

2. GENERAL INFORMATION AND DEFINITIONS

License format:

A license string that the user receives when ordering a plugin should look like this:

*PRODUCTID*FIRSTNAME*LASTNAME*111111111TTT#*

PRODUCTID is a custom sequence of characters identifying that plugin or suite

FIRSTNAME is the first name of the user

LASTNAME is the last name of the user

The 111111111 part is a unique numerical serial

TTT is a 3-character string that defines the type of license, currently this can be one of the following:

- SUL: single or multi user license
- BTA: beta license
- EDU: educational license
- REN: render-only license (only works in the command-line renderer)
- FLT: floating license (only works on the floating license server, not directly on client computers)

Directly after the type, there is a number defining the amount of users licensed. The most common type/number combination will be SUL1 for a single user license.

Multi-user licenses will have the number of users indicated after the SUL, for example a 12 user license would be SUL12.

License generation:

The license string is generated on the aescripts.com server when the user orders a plugin.

A correct serial is created from the available user information and a private number that is specific to each plugin and that is only known to the server (and also encoded within the plugin's binary code).

The private number has exactly 8 digits for current products and exactly 16 digits for products issued before version 4.0 of the library. Very old plugins may still use a 6 digit private number, and while these still work, please get in touch with aescripts to have them upgraded to the new format.

License activation / validation:

The user can open the plugin in After Effects and click on the "Register" link available within the interface.

A new window will pop up, allowing the user to enter the license string received from aescripts.com.

The license will then be validated by the plugin by checking the data against its internal private number.

Please note: After Effects CS6 and above use a persistent disk cache system, so if the user registers a plugin and then accesses frames that have been rendered before the registration and cached with a watermark, these frames won't get updated automatically. This is only a display issue and can be fixed by either purging the cache or simply changing any parameter on that frame, so it will get re-rendered.

License storage:

If the license has passed the validation check, the plugin will generate and store a license file on disk.

The location of this file will be:

On MacOS: ***~/Library/Application Support/com.aescripts.[pluginname].lic***
On Windows: ***\users\[username]\AppData\Roaming\aescripts\[pluginname]\[pluginname].lic***

Also, if the registering application (i.e. usually Adobe After Effects) is started with administrator/root rights, the license file will *additionally* be stored in the system preferences for all users:

On MacOS: ***/Library/Application Support/com.aescripts.[pluginname].lic***
On Windows: ***\users\All Users\Application Data\aescripts\[pluginname]\[pluginname].lic***

Please note that older versions of the licensing library on MacOS stored the .lic file under
~/Library/Preferences/com.aescripts.[pluginname].licand
optionally***/Library/Preferences/com.aescripts.[pluginname].lic***
These locatons have however been deprecated by Apple in recent MacOS versions. The licensing library will however still try to look for a .lic file in that folders if it does not find the license in the locations given above. A new registration will however only be stored in the new "Application Support" folder.

Inside the license file, the complete license string is stored as an encrypted byte sequence.

A specific binary machine ID is also stored in the file, tying this license file to the computer it was registered on. Simply copying the license file to another computer therefore won't work.

3. THE LICENSING LIBRARY FOR DEVELOPERS

What is it?

This document is accompanied by a static C++ library for Microsoft Windows and Apple MacOS that allows plugin coders to implement several license-related features easily into their products to be distributed on aescripts.com.

Compatibility:

Supported operating system are Windows 7 and above and MacOS 11.2 or above, in 64bit only.

The supported compilers are Microsoft Visual Studio 2012, 2013, 2015, 2017, 2019 and 2022 for Windows and Xcode 10 and above on MacOS.

The licensing library is provided as a static library (.lib in Windows, .a on Mac), with both Debug and Release builds for the 64bit architecture.

On Windows, there are different versions of this library provided depending on the compilation of the C runtime library as static or DLL (/MT or /MD compiler switch, explained below) as well as for VS2012, VS2013, VS2015, VS2017, VS2019 and VS2022 separately (as these compilers are not 100% binary compatible).

On MacOS, there are only debug and release versions of the library available. The MacOS libs are provided as universal binary .a files that support both the Intel and the M1 CPU architecture.

IMPORTANT: Starting with Visual Studio 2022 version 17.10, Microsoft changed a significant part of the C++ runtime (the constructor of std::mutex), breaking compatibility with the older VS2022 runtimes. A binary compiled with this version will only run when the matching runtime is present on the target system. The licensing library includes a workaround/fix that makes it compatible with old and new versions of the VS2022 runtime, but if you are using other third-party libraries compiled for VS2022, be aware that you can run into crashes (i.e. your plugin won't load in AE/PPro).

IMPORTANT: Apple dropped support for the libstdc++ library and now only supports projects compiled with the libc++ standard library.

Content:

The following files are included in the package:

./include	the header that needs to be included
./lib	the library that needs to be linked to (Debug/Release, Win/Mac)
./licTestPlugin	a simple test plugin for After Effects (CS6 or above)
./tools	some internal command-line tools for writing and reading a license, usually not needed for plugin developers

Header files:

To use the licensing library in your project, include the header file "./include/aescriptsLicensing.h"and link to the appropriate library version. As with any other third-party library, it is very important that the correct version is linked to, else you might run into some strange behaviour and crashes!

There is also an (optional) header file supplied as "./include/aescriptsLicensing_AdobeHelpers.h" that you can additionally include (after the normal header and after the Adobe SDK headers) in an Adobe After Effects plugin project. This file offers an easy to use wrapper around some more advanced functions and it is recommended to use it (see the included licTest plugin for example usage).

Linking on Windows:

On Windows, the following libs are provided in the "./lib/Win/VS20xx" folder:
- "aescriptsLicensing_MD_Debug.lib" (debug, 64bit, dynamicallylinked runtime /MD)
- "aescriptsLicensing_MD_Release.lib" (release, 64bit, dynamicallylinked runtime /MD)
- "aescriptsLicensing_MT_Debug.lib" (debug, 64bit, staticallylinked runtime /MT)
- "aescriptsLicensing_MT_Release.lib" (release, 64bit, staticallylinked runtime /MT)

While it should be clear from the project configuration what mode (debug/release) you are compiling in, the MD/MT thing might not be so obvious:

In Visual Studio, go to your project settings/properties and check under "C/C++ - code generation" what option is defined for the "Runtime Library" paramete:
- "Multi-threaded DLL (/MD)": you should link to "aescriptsLicensing_MD_Release.lib"
- "Multi-threaded (/MT)": you should link to "aescriptsLicensing_MT_Release.lib"
- "Multi-threaded Debug DLL (/MDd)": you should link to "aescriptsLicensing_MD_Debug.lib"
- "Multi-threaded Debug (/MTd)": you should link to "aescriptsLicensing_MT_Debug.lib"

Again, this is nothing specific to the aescript licensing framework, this needs to be considered and adjusted in general for any third-party library you link to!

The main difference between MT and MD linking is that the C runtime library is not linked statically in MD mode, so the MSVCRT runtime DLL needs to be present on the user's system already (which is almost always the case these days, as some Adobe applications also rely on it). In MT mode, all code from the Microsoft C runtime library is also included with the generated plugin, making deployment a bit less error-prone, but increasing the filesize.

For an After Effects plugin, you should most likely always use the MD version because of certain restrictions of Windows and AE!

Debug libraries are only for debug builds on developer machines, make sure you never link a debug library to a plugin/application that you distribute to the end user!

You also need to make sure you link to the right library for your compiler. If you link against the wrong compiler binaries (ie. use VS2012 to link the VS2015 libs or vice versa), you will get a linker error similar to this:
"error LNK2038: mismatch detected for '_MSC_VER': value '1600' doesn't match value '1700'"

In that case, go to the project properties and in the linker settings and change the path to the libs so it fits your compiler.

Furthermore, you will also need to link to two standard Windows system libraries related to network communications: "ws2_32.lib" and "iphlpapi.lib". Simply add them in the project properties under "Linker - Input" (no path necessary, just the names of the libs). If you don't do this, you will get a linker error similar to this:
"error LNK2019: unresolved external symbol __imp_setsockopt referenced in function "int __cdecl reduxfxHelpers::setTimeouts(int,int,int)"

Linking on MacOS:

On MacOS, first of all, you need to make sure you also link to the Apple frameworks of IOKit, CoreFoundation, Cocoa and (since version 4.x of the licensing library) also the system library libcurl.

This is done by adding these libraries/frameworks on the "Build Phases" tab in Xcode:
- IOKit.framework
- CoreFoundation.framework
- Cocoa.framework
- libcurl.tbd

Apple now only supports itws own C++ runtime libraries (libc++), the support for libstdc++ was dropped. Please check the build settings in your Xcode project, for "C++ Standard Libray" under "Apple LLVM - Language - C++" there should be libc++.
As on Windows, make sure you link to the correct version of the library, aescriptsLicensing_Debug.a for debug builds and aescriptsLicensing_Release.a for release builds.

Debug libraries are only for debug builds on developer machines, make sure you never link a debug library to a plugin/application that you distribute to the end user!

If you are getting a compilation error about "_curl_easy_init" or similar functions, you still need to link to libcurl.tbd (see above)!

4. THE LICENSING LIBRARY FUNCTIONS

Library functions:

The basic library consists of the following functions and types, defined in "aescriptsLicensing.h":

```
typedef char licString[128];
typedef char machineString[60];
typedef char verString[11];
typedef char licVerString[6];


// this is a class that holds the decoded user license information
struct aescriptsLicenseData {
	char firstName[30]; // user's first name
	char lastName[30]; // user's last name
	int nofUsers; // number of licensed users
	char licType[10]; // license type (SUL, BTA, EDU, REN, FLT)
	char productID[30]; // product ID
	char serial[64]; // license serial string

	aescriptsLicString license; // complete license string
	aescriptsVerString productVersion; // product version
	aescriptsLicVerString licenseVersion; // version of the license file

	int firstStartTimestamp; // timestamp of first start
	int numberOfDaysSinceFirstStart; // number of days since first start
	bool registered; // true if license is valid
	int overused; // same license active in the network? -1: unknown, 0: no, 1: yes
	bool renderOK; // is rendering allowed with this license/state?

	char startDate[11]; // time-limited licenses: start date in format YYYY-MM-DD
	char endDate[11]; // time-limited licenses: end date in format YYYY-MM-DD

	int lastActivationTryTimestamp; // last time the product was tried to be activated
	int lastActivationTimestamp; // last time the product was successfully activated
	int lastResultCode; // result code of the the last library function call
};

// this function initializes/clears an aescriptsLicenseData object
int aescriptsInitLicenseData(aescriptsLicenseData& _lData);

// load and decode the content of the license file from disk but don't validate it
// the privNum can be set to -1 to load the content without validation,
// but then the return value will be meaningless, only the lData values will be set
int aescriptsGetLicenseDataFromFile(const char* _licenseFileName,
	long long _privNum, aescriptsLicenseData& _lData);

// load license file from disk and use it immediately (in case of floating licenses,
// implicitly request a lease from the floating server)
int aescriptsLoadLicenseFromFile(const char* _licenseFileName, const char* _productID,
	long long _privNum, aescriptsLicenseData& _lData, bool _createTrialLicense,
	bool _renderOnlyMode);

// save license to disk
int aescriptsSaveLicenseToFile(const char* _licenseFileName, const char* _productID,
	aescriptsLicString _lic, aescriptsVerString _productVersion);

// validate license string (usually not needed to be called directly,
// used implicitly by loadLicenseFromFile)
int aescriptsValidateLicense(aescriptsLicString _license, const char* _productID,
	long long _privNum, aescriptsLicenseData& _lData, bool _renderOnlyMode);

// get machine ID (usually not needed to be called directly)
int aescriptsGetMachineId(aescriptsMachineString& _result);

// dynamically add data for a blocked serial
int aescriptsAddBlockedSerial(long long _privNum, int _d1, int _d2, int _d3, int _d4);
```

```
// get overuse state (usually not needed to be called directly,
// used implicitly by validateLicense)
int aescriptsCheckOveruse(aescriptsLicenseData& _lData);

// get server and port for (optional) licensing server
int aescriptsGetLicenseServerConfig(
       aescriptsLicString& _cServer, aescriptsLicString& _cPort,
       aescriptsLicString& _cBackupServer, aescriptsLicString& _cBackupPort);

// get licensing library version (as integer, e.g. 40 meaning its version 4.0)
int aescriptsGetLicLibVersion();

// returns licensing library version as integer (e.g. 40 meaning its version 4.0)
// and detailed build version information in the buildInfo parameter
int aescriptsGetLicLibBuildVersion(aescriptsLicString& buildInfo);

// request a floating license from the license server (usually not needed to be called
// directly, used implicitly by loadLicenseFromFile)
int aescriptsRequestLicenseFromLicenseServer(const char* _productID,
       aescriptsVerString _productVersion, long long _privNum,
       aescriptsLicString& _license);

// drop a floating license from the license server (should be done at unloading/exit
// of the plugin/application)
int aescriptsDropLicenseFromLicenseServer(const char* _productID, long long _privNum);

// this function is only used for special cases by developers,
// as instructed by the licensing library support team
int aescriptsConfigure(int token, int value);
```

IMPORTANT: All of these functions are defined in the "aescripts" namespace, so be sure to prefix the functions with "aescripts::"!

There used to be a function "getLicenseFromLicenseServer", which has been deprecated since version 4.0. Please use the functions "requestLicenseFromLicenseServer" and "dropLicenseFromLicenseServer" instead.

IMPORTANT: All of these functions return an integer value representing a result/error code. The possible codes and their meanings are listed at the end of this document.

"Easy Mode" for including the licensing library in an After Effects effect plugin

1. Add this to the main file of your plugin at the top, right after the includes for the other headers:

```
// this is the private number - it is unique to every plugin and issued by aescripts
#define LIC_PRIVATE_NUM 11223344

// the product ID is a string also issued by aescripts that identifies your plugin
#define LIC_PRODUCT_ID "AESLT4"

// this is the filename (extension .lic gets added automatically)
// under which the license is stored on the disk, can be same as LIC_PRODUCT_ID
#define LIC_FILENAME "AESLT4"

// set this define if your plugin is compiled for beta-testers,
// it will then accept BTA type licenses
#define LIC_BETA

// include the aescripts licensing API
// (should be done *after* including the Adobe AE SDK headers!)
#include "../include/aescriptsLicensing.h"

// include the aescripts licensing Adobe helper API
// (should be done *after* including the Adobe AE headers!)
#include "../include/aescriptsLicensing_AdobeHelpers.h"
```

2. Add this as the first line in your EntryPoint function EntryPointFunc():

```
if (aescripts::checkLicenseAE(cmd, in_data, out_data, aescripts::licenseData) != 0)
return err;
```

3. Add this as part of your registration dialog function (called on PF_Cmd_DO_DIALOG):

```
static PF_Err Register(PF_InData *in_data, PF_OutData *out_data) {
       AEGP_SuiteHandler suites(in_data->pica_basicP);
       string sProductVersion = aescripts::intToString(MAJOR_VERSION) + "." +
              aescripts::intToString(MINOR_VERSION) + "." +
              aescripts::intToString(BUILD_VERSION);
       strncpy(aescripts::licenseData.productVersion, sProductVersion.c_str(),
              min<int>((int)sProductVersion.length(), 10));
       int reg_result = aescripts::showRegistrationDialog(&suites,
              aescripts::licenseData, in_data->appl_id != 'PrMr');
       return PF_Err_NONE;
}
```

4. You can now check any time in your code the variable `aescripts::licenseData.registered`.
If it is set to true, your plugin is registered.

5. It is also recommended to display additional info in your plugin's about box, especially if you are dealing with time-limited licenses. Check the included test plugin or add a check like this to your code:

```
if (strlen(aescripts::licenseData.startDate) > 3) { // we have a time-limited license
       dialogString = "Time limited license valid from " +
              string(aescripts::licenseData.startDate) + " to " +
              string(aescripts::licenseData.endDate);
}
```

6. If you have two or more separate plugins or applications that identify as one product and should there be registered using just one license and license file, please keep the data mentioned under 1. identical for each component.

5. USING THE LICENSING LIBRARY

Generic Usage:

A plugin (or program, since the licensing library is not AE/plugin specific!) can use the following functions to write and read a license:

```
// include only aescriptsLicensing.h, not aescriptsLicensing_AdobeHelpers.h in this case
#include "aescriptsLicensing.h"

// SAVING A LICENSE

// the license string the user entered
aescripts::licString license;

// copy license string the user entered into "license" variable first using strcpy!
// use the test license that was provided in your aescripts.com account
strcpy(license, "AESLT4*JOE*USER*0000000000000000SUL1");

// save license (and set plugin version in the license to 1.0.0)
aescripts::saveLicenseToFile("AESLT4", license, "1.0.0");

// LOADING A LICENSE

// struct to hold the license data information
aescripts::LicenseData lData;

// initialize license data to empty fields on startup
aescriptsInitLicenseData(lData);

// load and check license
int privNum = 11223344;
int result = aescripts::loadLicenseFromFile("AESLT4", privNum, lData, false);
if (result == 0) { // valid license
       ... }
```

The loadLicenseFromFile() function returns an integer value as described at the end of this document.

In case of return code -7 (AESCRIPTSLICLIB_RESULT_TRIAL), it means that the library only found a trial license, which gets created automatically on first usage of the program if the (optional) fourth parameter in loadLicenseFromFile() is set to true (which is the default). This is also the case when not validating the license manually before calling the method, and an invalid license is entered.

This is useful if you want to implement a timed trial:
- you always call loadLicenseFromFile() with the 4th parameter set to true and evaluate the result
- if the result is 0, a valid user license is installed, so plugin is registered
- if the result is -7, a trial license is present.
  You should then check the value of lData.numberOfDaysSinceFirstStart against your desired maximum number of trial days and disable functionality if it exceeds that value
- if the result is < 0, there is an error with the license file

Please also check out the included tools and especially the test plugin for example usage!

After Effects test plugin:

The sources for a cross-platform test plugin (for AE/PPro CS6 or higher) are included with the licensing library. This test plugin is a simple one-slider plugin that blends a layer into a grayscale version of it. It will accept any registration/license key that matches the private number 8748765561386391 (see below for test licenses for this number). You can base your own AE plugin on this project, or you can copy the relevant code segments to your already existing project. Apart from adding the license source files as described above, you have to make sure to adjust the paths to the AE SDK folder to the location on your hard drive in the project settings.

There is a function in the source code for the plugin for determining if a pixel should be overwritten with a specific color when running in non-registered mode (a cross is drawn on the output layer to signify trial/unlicensed mode). You can of course replace that with your own trial watermark code.

The test plugin also contains code (not in the licensing framework, but in the actual plugin code) that dumps the found license information to a debug logfile on the desktop. This is for debugging/testing purposes only and should not be put into a released product.

For compiling this plugin, you might need to adapt the path to the Adobe After Effects SDK. By default it looks for it in this folder relative to the source project: ..\..\..\..\Adobe_AfterEffects_SDK. You can however easily adapt this in the project settings of your compiler/IDE.

Blocking of leaked serials:

Once a valid serial gets leaked/distributed illegally, there is now the option to block this serial in future versions. There are two ways to do this:

1. You send the leaked serial to your contact person at aescripts.com and it will be hard-coded as a blocked serial in the next release of the general licensing library.

2. You can dynamically block a serial in your plugin only. For this you can use this function: `aescripts::addBlockedSerial(long long privNum, int d1, int d2, int d3, int d4);`

You should call this function as early as possible in the code (and in any case before you do any validation). In After Effects or Premiere Pro plugins it is a good idea to put it in the GlobalSetup function call. You will get the necessary parameters (privNum and d1 to d4) based on the serial to be invalidated from your contact at aescripts.com

Overuse checks:

The licensing library contains an internal mechanism for determining if other computers on the same network segment use the same serial. This is for local licenses (so not the floating licenses used in the license server) and intended to detect "overuse" of single user licenses.
So if a plugin detects its own serial is in use at the same time on another computer, the plugin with the newer start timestamp switches to trial mode. Once the other plugin is deactivated, the new plugin will become registered again automatically. The developer or user can not influence this behaviour, this is by design.

You can check the aescripts::licenseData.overused field for the value of 1 to see if a plugin is currently running in "overused" mode. A value of -1 means still unknown, 0 means all ok.

Render-only license:

Some notes about render-only licenses. These licenses can only be used for plugins running on the (GUI-less) AE renderer (which is usually started from the command line). A render-only plugin is identified by license type "REN" (last three letters of the license string or content of "licType" string member in code). As this needs to be checked from within the plugin's render code, you need to add this check to your own source code.

However, the included helper file "aescriptsLicensing_AdobeHelpers.h" already includes this code and if you are calling the `aescripts::checkLicenseAE` command already as mentioned above in "easy mode", you don't need to do anything.

Otherwise, you should compare `aescripts::licenseData` .liccType against "REN" and then check with the SDK function "PF_IsRenderEngine()" if you are running in the AE render engine or not and act appropriately.

License Server and floating license support:

There is also an optional "aescripts License Server" application that customers can use. This application for Windows, MacOS or Ubuntu/CentOS-based Linux distributions can run in the network of the customer and serve floating (i.e. dynamically assigned) licenses for all supported AE products to clients running on Windows or MacOS. Support for these type of licenses has been built into the library, and you don't need any special functionality to use it. Setting up the license server is not documented here, as the client/plugin just needs to know the hostname and port it should connect to (optionally also with a fallback hostname/port should the main server not be available).

There are two ways this information can be set on the client computer:

1. A plist file with the configuration for server name and port can be generated with a tool called "aescriptsLicenseServerTool" that needs to be run once on each client computer.
The parameter syntax for this tool is:
aescriptsLicenseServerTool SERVER:PORT
or in case a fallback server should be used:
aescriptsLicenseServerTool SERVER:PORT|BACKUP_SERVER:BACKUP_PORT
The tool will also check if a connection to the given server is possible.

2. An environment variable "aescriptsLicensingServer" can be set to contain "SERVER:PORT" (optionally in the form "SERVER:PORT|BACKUP_SERVER:BACKUP_PORT" in case a fallback server is used).

If both the plist and the variable are present, the plist file takes preference.

As for the floating licenses themselves, they have the format PRODUCTID@REMOTE, so for example for the test product AESLT, the license string would be "AESLT@REMOTE". If you are using the "easy mode" described above with the aescriptsLicensing_AdobeHelpers.h" file, the user can simply enter "@REMOTE" in the license input field and if a server configuration file is present, the library will automatically try to fetch a license for that product from the licensing server (adding the correct product ID automatically).

IMPORTANT: It does not matter how many instances of one plugin or product are running on one client, there will only be ever one license used per client computer on the server.

If you want to request a floating license manually, you can simply use the `loadLicenseFromFile()` function as you would do for a normal/local license, and it will automatically request it from the server if the license was previously saved in the PRODUCTID@REMOTE format described above.

IMPORTANT: In case no license file at all for a certain product is present (yet) on the current system, but a floating server is configured for that computer, a call to `loadLicenseFromFile()` will also automatically try to request a license lease for that product from the server. The same is true for trial license files when a request is done in the render-only mode and a license server is present.

To manually release/drop a license from the server, you can use this function

```
int dropLicenseFromLicenseServer(const char* productID, long long privNum);
```

IMPORTANT: Floating licenses can be released when the user no longer requires them, but at least when the main applications that uses the licensing library (either indirectly if it is a host using a plugin, or directly if it is an application) exits, you should manually drop the license. If this is not done, the license is released on the server only after a specified timeout.

In case you are using the library in an AEGP plugin, which are general purpose plugins in After Effects that do not appear in the effect selection, you won't get a notification that After Effects has exited. You can then either use a global object's destructor and put the drop license line there, or use the AE SDK function AEGP_RegisterDeathHook() to register a callback function when After Effects quits and call it from there (check the "Commando" example in the AE SDK for how to use hooks).

Command line tools:

There are two command line tools with full source code included that show the basic functionality.

The first is "aescriptsLicenser". The syntax for this tool is:

aescriptsLicenser effectname license version

"effectname": the actual effect name or id - it must not contain spaces!

"license": the license string (format: PRODUCTID*FIRSTNAME*LASTNAME*111111111TTT#)

"version": the version string for the effect (format: x.y.z), default is "1.0.0"

IMPORTANT: If the second parameter (license) is given as "-", the tool assumes it should invalidate/deactivate the license.

Then there is "aescriptsValidator", a validation tool that expects two parameters:

aescriptsValidator effectname privatenumber

The parameters are the "effectname" (see above) and the private number used for creating the license key. This tool is obviously not intended for the end user, but it should be helpful to you to check if a license is okay.

When run, it will print the decoded information for the (encrypted) license file, so you can now check if everything is also decoded correctly, eg. the number of licenses.

Online activation/validation:

Starting with licensing library 4.0, a new online validation mechanism is introduced to validate licenses online and prevent multiple usage of the same license across different users.

This feature is enabled by default, but in case a developer wants to intentionally disable it for a specific product, the following special function need to be called **before** using other functions from the library (like *aescripts::loadLicenseFromFile*):

aescripts::configure(AESCRIPTS_CONFIGURE_DISABLE_ONLINE_ACTIVATION, 1);

The first parameter is a constant defined in the header and defines which internal settings should be changed for the current product/plugin – here it is the support for the online activation module. The second parameter can be set to 0 (inactive/disabled/false) or 1 (active/enabled/true). So the line above will set the disabling of the online activation to true.

6. TESTING THE LICENSING LIBRARY

Testing:

For testing of the library itself, you can open up the source projects of the two tools mentioned above and see if you can compile them.

Use the license that was provided in your aescripts.com account. This license is valid for private number: **11223344**

Do not use the private number above for your own plugins, it is for demonstration/testing purposes only! Please obtain a unique number from aescripts for every one of your plugins.

If you want to install a floating license for the test plugin (AESLT4/11223344) on your floating license server, you can use this code: **AESLT4*JOE*USER*639801012259422489779FLT1**

As an example, use the licenser tool to create a license like this (you will need to use your valid license):

*aescriptsLicenser licTestPlugin AESLT4*JOE*USER*000000000000000000000SUL1*

This should create a license file in the following location:

MacOS: **~/Library/Application Support/com.aescripts.licTestPlugin.lic**

Windows: **\users\[username]\AppData\Roaming\aescripts\licTestPlugin\licTestPlugin.lic**

Then you can call the validator to see if the license correctly validates:

aescriptsValidator AESLT4 11223344

To test with your own plugin and private number, you can generate a test license in your aescripts author admin under **Author** -> **Generate License** and generate a **time-limited license** in your name.

7. RESULT CODES

Below are the common result codes used that every library function call returns.

Codes not in this list are either deprecated or are used internally only and are not exposed to the calling functions.

The function resultCodeToMessage() can be used to retrieve the message text/description associated with a certain result code.

Value:	0
Const:	AESCRIPTSLICLIB_RESULT_OK
Description:	Valid license

Value:	-1
Const:	AESCRIPTSLICLIB_RESULT_INVALID_FORMAT
Description:	Invalid license - The license has an invalid format

Value:	-2
Const:	AESCRIPTSLICLIB_RESULT_WRONG_MACHINE
Description:	Invalid license - The license belongs to a different machine ID

Value:	-3
Const:	AESCRIPTSLICLIB_RESULT_NO_LICFILE
Description:	License file not found

Value:	-4
Const:	AESCRIPTSLICLIB_RESULT_CORRUPTED_LICFILE
Description:	License file is corrupted

Value:	-5
Const:	AESCRIPTSLICLIB_RESULT_GENERIC_ERROR
Description:	Generic error

Value:	-6
Const:	AESCRIPTSLICLIB_RESULT_INVALID_PRODUCT
Description:	There is an issue with the product header

Value:	-7
Const:	AESCRIPTSLICLIB_RESULT_TRIAL
Description:	Trial mode

Value:	-8
Const:	AESCRIPTSLICLIB_RESULT_INVALID_TIMELIMITED_LIC_FORMAT
Description:	The time-limited license has invalid format

Value:	-9
Const:	AESCRIPTSLICLIB_RESULT_INVALID_LIC_TYPE
Description:	The license type not recognized

Value:	-10
Const:	AESCRIPTSLICLIB_RESULT_LIC_VERIFICATION_FAILED
Description:	Verification of the license format failed

Value:	-11
Const:	AESCRIPTSLICLIB_RESULT_LIC_BLOCKED
Description:	The license is no longer valid - license is in blocked list
Value:	-12
Const:	AESCRIPTSLICLIB_RESULT_INVALID_LIC_KEY
Description:	The license key is invalid - private number has invalid length (has to be 6 or 16 for V2 licenses and 8 for V4 licenses)
Value:	-14
Const:	AESCRIPTSLICLIB_RESULT_NETWORK_ADAPTER_ERROR
Description:	Error reading network adapter data
Value:	-15
Const:	AESCRIPTSLICLIB_RESULT_LIC_PERIOD_NOT_STARTED
Description:	License period has not started yet
Value:	-16
Const:	AESCRIPTSLICLIB_RESULT_LIC_PERIOD_ENDED
Description:	License period has ended
Value:	-20
Const:	AESCRIPTSLICLIB_RESULT_INVALID_FLOATING_LICENSE
Description:	Invalid floating license - Please double check the license code
Value:	-21
Const:	AESCRIPTSLICLIB_RESULT_NO_CONNECTION_TO_SERVER
Description:	Cannot connect to server - Please make sure the floating license server is running
Value:	-22
Const:	AESCRIPTSLICLIB_RESULT_FLOATING_LIC_REQUIRES_SERVER
Description:	Floating licenses can only be used with the floating license server
Value:	-23
Const:	AESCRIPTSLICLIB_RESULT_NO_SLOTS_ON_SERVER
Description:	There are no more free slots on the floating license server
Value:	-24
Const:	AESCRIPTSLICLIB_RESULT_NO_LICENSE_ON_SERVER
Description:	The license cannot be found on the floating license server
Value:	-25
Const:	AESCRIPTSLICLIB_RESULT_NO_LEASE_ON_SERVER
Description:	The license you are trying to deactivate is not found on the floating license server
Value:	-26
Const:	AESCRIPTSLICLIB_RESULT_CLIENT_DENIED_BY_SERVER
Description:	Your client IP is denied by the floating license server
Value:	-30
Const:	AESCRIPTSLICLIB_RESULT_ONLINE_INVALID
Description:	Online activation: license invalid

Value: -31
Const: AESCRIPTSLICLIB_RESULT_ONLINE_EXPIRED
Description: Online activation: license expired

Value: -32
Const: AESCRIPTSLICLIB_RESULT_ONLINE_MAX_ACTIVATIONS
Description: Online activation: maximum number of activations reached

Value: -33
Const: AESCRIPTSLICLIB_RESULT_ONLINE_NO_CONNECTION
Description: Cannot connect to online activation server

Value: -34
Const: AESCRIPTSLICLIB_RESULT_ONLINE_INVALID_PAYLOAD
Description: Online activation: invalid payload

Value: -35
Const: AESCRIPTSLICLIB_RESULT_ONLINE_NO_DEVICEID
Description: Online activation: missing device ID - This activation needs to be deactivated in your aescripts.com account

Value: -36
Const: AESCRIPTSLICLIB_RESULT_ONLINE_NO_RECORD
Description: Online activation: no matching activation found

Value: -37
Const: AESCRIPTSLICLIB_RESULT_ONLINE_INVALID_REQUEST
Description: Online activation: missing parameters

Value: -38
Const: AESCRIPTSLICLIB_RESULT_ONLINE_INVALID_RESPONSE
Description: Online activation: invalid response from server - This activation needs to be deactivated in your aescripts.com account

Value: -39
Const: AESCRIPTSLICLIB_RESULT_ONLINE_INVALID_SUBSCRIPTION
Description: "Online activation: subscription is not valid

Value: -40
Const: AESCRIPTSLICLIB_RESULT_OFFLINE_INVALID_CODE
Description: Invalid offline activation code

8. IMPORTANTS THINGS TO CONSIDER FOR AE PLUGINS

Multi-frame rendering (MFR):

Adobe After Effects 2022 and above support multi-frame rendering in plugins. For this to work, the plugin has to follow certain implementation guidelines precisely. If this is not done correctly, memory corruption can occur, resulting in bugs and random crashes that are hard to track down. In the past, there were several cases where crashes in the licensing library occurred that were actually caused by not implementing the proper behaviour.

If you want your plugin to support MFR (or if you explicitly don't want to do it), please be sure to read the following guidelines:
https://ae-plugins.docsforadobe.dev/effect-details/multi-frame-rendering-in-ae.html

Struct member alignment (Windows only):

Many developers base their plugins on the examples Adobe provides in the SDK (and rightfully so).
However, there is a problematic compiler setting for Windows in all of the Visual Studio sample projects which will cause memory corruption or crashes if using many other third-party libraries or even just the Windows API. The setting for "struct member alignment" needs to be set to "default", but Adobe sets it to 4 in their projects, probably for historical reasons. If it is set to 4 in your Visual Studio project, make sure to reset it to the default setting, else memory corruption will occur!
More information about this setting is available here:
https://docs.microsoft.com/en-us/cpp/build/reference/zp-struct-member-alignment?view=msvc-170