MarshallSoft AES

(Advanced Encryption Standard)

Library for C/C++

Programmer's Manual

(AES_4C)

Version 6.0

February 14 . 2022

This software is provided as-is.
There are no warranties, expressed or implied.

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815

Web: http://www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

The **MarshallSoft Advanced Encryption Standard Library for C/C++ (AES4C)** is a toolkit that allows software developers to easily implement strong encryption and decryption into a Windows C/C++, Visual C++, .NET or Visual C# application.

The **MarshallSoft Advanced Encryption Standard Library (MarshallSoft AES)** is a component (DLL) library of functions used to perform encryption and decryption using the 256-bit "Advanced Encryption Standard" (AES) as specified by the U.S. National Institute of Standards and Technology (NIST). See http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

AES is considered "strong encryption" and replaces the previous U.S. encryption standard "Data Encryption Standard" (DES). AES is commonly used by many financial entities such as banks to protect their customer's sensitive information.

Our implementation of the Advanced Encryption Standard (AES) has been verified by running the "Advanced Encryption Standard Algorithm Validation Suite" (AESAVS), which can be found at http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf.

This **MarshallSoft Advanced Encryption Standard Programmers Manual for C/C++** provides information need to compile and run programs in a C/C++ programming environment.

The **MarshallSoft Advanced Encryption Standard DLL's** will work under all versions of Windows (2003-2012/XP/Vista/ Windows7/Windows 8). Both Win32 and Win64 DLL's are included.

AES4C includes several C/C++ example programs, including Visual Studio C++ and Visual Studio, which demonstrate AES encryption and decryption. Borland C++ Builder (BCB) examples are also provided.

The **MarshallSoft Advanced Encryption Standard Library for C/C++** component library supports and has been tested with C/C++, Microsoft Visual C++, Visual Studio .NET Framework (Visual C++ .NET, C# .NET), Visual C++ Express, Borland C/C++, Borland Turbo C++ for Windows, Borland C++ Builder, Watcom C/C++, LCC-Win32, and MinGW C compilers. **AES4C** can also be used with most other C/C++ Windows compilers.

The **MarshallSoft AES** DLLs (aes32.dll and aes64.dll) can also be used from any language (Visual Basic, VB.NET, ACCESS, EXCEL, VBA, Borland Delphi, Visual FoxPro, COBOL, Xbase++, Visual dBase, Microsoft Office, etc.) capable of calling the Windows API.

For the latest version of the **MarshallSoft AES** software, see
http:/www.marshallsoft.com/aes4c.htm

Legalities

It is illegal to possess strong encryption software in some countries in the world. Do not download or use this software if it is illegal to do so in your country.

In addition, this software cannot be sold to countries on the U.S. Embargo List. See
http://www.pmddtc.state.gov/embargoed_countries/index.html

1.1 Features

- Works with 32-bit and 64-bit Windows through Windows 11.
- Implements the 256-bit "**Advanced Encryption Standard**" (Rijndael)
- Implements Diffie-Hellman key exchange.
- Includes cryptographically secure (pseudo) random number generator.
- Supports **ECB** (Electronic Cookbook) mode.
- Supports **CBC** (Cipher Block Chaining) mode.
- Supports **PKCS7** padding.
- **Free** technical support and updates for one year.
- License covers all programming languages.
- Royalty free distribution with a compiled application.
- Evaluation version is fully functional (30 day trial). No unlock code is required.
- Can be used from GUI mode or console mode programs.
- Is fully thread safe.
- Implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Both Win32 and Win64 DLLs are included.
- Is native Windows code but can also be called from managed code.
- Will run on machines with or without .NET installed.
- Works with all versions of Microsoft Visual C/C++ (v4.0 through Visual Studio 2022).
- Works with Microsoft Visual Studio .NET and Visual C#.
- Works with Borland C/C++ (v5.0, v5.5, and Borland C++ Builder [all versions]).
- Works with Microsoft Foundation Class, Watcom v11, MinGW, Digital Mars and LCC-WIN32.
- Does **not** depend on support libraries. Makes calls to core Windows API functions only.
- Can be used with any program (in any language) capable of calling Windows API functions such as Visual Basic, VB.NET, Visual FoxPro, Delphi, Xbase++, dBASE, COBOL, Access or Excel.
- Can be purchased with (or without) source code.
- Updates are free for one year (source code updates are separate).
- Unlimited one-year email and phone tech support.
- Documentation online as well as in printable format.

A selection of C/C++ example programs with full source code is included. Refer to Section 6 for more details on each of the example programs.

Legend:

```
Con = Console mode.
GUI = GUI mode.
VC  = VC 4 through VC 6.
VS  = Visual Studio C/C++ (through VS 2022)
VC/B/L/G/D = For VC 4 through 6, Borland 5.0 & 5.5, LCC, MinGW GCC, Digital Mars
```

```
 1. aesver.c          Displays AES4C version (CON, VC/B/L/G)
 2. crypto.c          Encrypts/decrypts a file (GUI, VC/B/L/G)
 3. decrypt.c         Decrypts a file (CON, VC/B/L/G)
 4. encrypt.c         Encrypts a file (CON, VC/B/L/G)
 5. TestAES.c         Performs AES tests (CON, VC/B/L/G)
 6. TestDH.c          Performs Diffi-Hellman tests (CON, VC/B/L/G)
 7. Password.c        Encrypt/decrypt multiple passwords (GUI, VC/B/L/G)
 8. HashDigest.c      Computes hash digest (CON, VC/B/L/G)
 9. aesver.cs         C# version of vc_aesver (VS, C#)
10. Validate.c        Validates AES against on-line encryption (CON, VC/B/L/G).
11. vc_Validate.cpp   Validates AES against on-line encryption (VS, CON)).
12. vc_Crypto.cpp     Encrypts/decrypts a file (VS, GUI)
13. vc_TestAES.cpp    Performs AES tests (VS, CON)
14. vc_TestDH.cpp     Performs Diffi-Hellman tests (VS, CON)
15. vc_vers.cpp       Displays AES version (VS, CON)
16. vc_HashDigest.cpp Computes hash digest (VS, CON, VC/B/L/G)
17. vc_Password.cpp   Encrypt/decrypt multiple passwords (VS, GUI)
```

1.2 Documentation Set

The complete set of documentation is provided in Adobe PDF format. This is the first manual (AES_4C.PDF) in the set.

- AES_4C Programmer's Manual (AES_4C.PDF)
- AES User's Manual (AES_USR.PDF)
- AES Reference Manual (AES_REF.PDF)

The AES_4C Programmer's Manual (AES_4C.PDF) is the language specific (C/C++) manual and provides information needed to install and compile example programs in a C/C++ environment.

The AES User's Manual (AES_USR.PDF) discusses language independent issues. Information encryption/decryption programming issues as well as technical support, purchasing and license information is provided in the manual.

The AES Reference Manual (AES_REF.PDF) contains details on each individual AES function.

Documentation is also provided online at http:/www.marshallsoft.com/aes4c.htm

1.3 Example Program

The following example demonstrates some of the library functions.

```
#include "aes.h"

int EncryptFile(char *KeyBuffer, char *FileNameIn, char *FileNameOut)
{int Code;
 // attach DLL (must be first AES call)
 Code = aesAttach(AES_KEY_CODE, AES_PKCS7_MASK);
 if(Code<0)
   {printf("ERROR %d: Cannot attach\n", Code);
    return FALSE;
   }
 printf("Will encrypt file in ECB (Electronic Code Book) mode\n");
 Code = aesInitAES((char *)KeyBuffer, NULL, AES_ECB_MODE, AES_ENCRYPT, NULL);
 if(Code<0)
   {printf("aesInitAES fails\n");
    PrintError(Code);
    return FALSE;
   }
 printf("Encrypt file...\n");
 Code = aesEncryptFile(NULL, FileNameIn, FileNameOut);
 if(Code<0)
   {printf("aesEncryptFile fails\n");
    PrintError(Code);
    return FALSE;
   }
 printf("%d bytes encrypted\n", Code);
 return Code;
}
```

1.4 Installation

(1) Before installation of AES4C, a Windows C/C++ compiler should already be installed and tested. In particular, include command line tools when installing a compiler to be able to compile using command line makefiles. Refer to the file, MAKEFILE.TXT, for help with makefiles.

(2) Unzip AES4C60.ZIP (evaluation version) or AESxxxxx.ZIP (registered version: xxxxx is the Customer ID).

(3) Run the installation program SETUP.EXE that will install all AES4C files. SETUP will also copy aes32.dll and aes64.dll to the Windows directory. Note that no DLL registration is required.

All recent WIN32 C/C++ compilers support the "declspec" keyword. Microsoft Visual C++ (version 4.0 and up), Borland (version 5.0 and up), and LCC-Win32 compilers support the "declspec" keyword. Makefiles for Watcom (version 11.0 and up) are available on request.

1.5 Uninstalling

Uninstalling AES4C is very easy. First, delete the AES project directory created when installing AES4C. Second, delete aes32.dll and AES64.DLL from the Windows directory, typically C:\WINDOWS.

1.6 Pricing

A developer license for the **MarshallSoft Advanced Encryption Standard Library** can be purchased for $119 (or $199 with source code to the library DLL). Purchasing details can be found in Section 1.4, "How to Purchase", of the AES User's Manual (AES_USR).

Also see INVOICE.TXT or http://www.marshallsoft.com/order.htm

1.7 Updates

When a developer license is purchased for AES, the developer will be sent a set of registered DLLs plus a license file (AESxxxxx.LIC). The license file can be used to update the registered DLL for a period of one year from purchase. Updates can be downloaded from

http://www.marshallsoft.com/update.htm

After one year, the developer license must be updated to be able to download updates. The license can be updated for

- $33 if the update is ordered within one year of the original purchase (or previous update).
- $55 if the update is ordered between one and three years of the original purchase (or previous update).
- $77 if the update is ordered after three years of the original purchase (or previous update). .

The update price includes technical support for an additional year. Note that the registered DLLs, (aes32. dll and aes64.dll) never expire. If source code was previously purchased, updates to the source code can be purchased for $40 along with the license update.

2 AES Library Overview

The **MarshallSoft Advanced Encryption Standard Library for C/C++ (AES4C)** has been tested on multiple computers running Windows XP through Windows 10 in both 32-bit and 64-bit applications.

The **AES4C** library has been tested with several C/C++ compilers, including Microsoft Visual C++ (all versions including Visual Studio .NET and Visual Studio C#), Borland C/C++, Borland C++ Builder, Turbo C/C++ for Windows, Watcom C/C++, Digital Mars, MinGW C++ and LCC-Win32 C

The SETUP installation program will copy the Lib's and DLL to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the AES4C files are copied to the directory specified (default \AES4C). Three sub-directories are created, as follows:

```
DOCS - All documentation files
APPS - All example code
DLLS - All DLL's
```

2.1 Dynamic Link Library

The **MarshallSoft Advanced Encryption Standard Library** is implemented as Win32 and Win64 dynamic link libraries (DLL). A DLL is characterized by the fact that it need not be loaded until required by an application program and that only one copy of the DLL is necessary regardless of the number of application programs that use it. Contrast this to the traditional static library that is bound to each and every application that uses it at link time.

An important advantage that DLL's have over other "popular" library formats such as VBX or OCX is that DLL's are callable by all Windows applications. Since DLL's are the building blocks of the Windows Operating System, they will not be replaced by a "newer technology".

The following files can be found in the DLL sub-directory when SETUP is run:

```
aes32.dll   - Win32 version of AES
aes64.dll   - Win64 version of AES
```

Note: The LIB files for the above DLLs can be found the in the APPS directory

2.2 Keycode

The AES DLLs, aes32.dll and aes64.dll, each have a keycode encoded within them. The keycode is a 9 or 10-digit decimal number (unless it is 0), and will be found in the file KEYCODE.H. The keycode for the evaluation version is 0. The developer will receive a new key code after registering. The KEYCODE is passed to **aesAttach**.

If an error code (a negative number) is returned when calling **aesAttach**, it means that the keycode in the AES application does not match the keycode in the DLL (error -202), or that the (30 day) evaluation version has expired (error -203).

2.3 Win32 STDCALL and DECLSPEC

AES32 is compiled using the _stdcall and _declspec keywords. This means that AES4C uses the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. There are no leading underscores or trailing "@size" strings added to function names.

Microsoft Visual C++ users can look at the AES32 function names using the dumpbin.exe executable: dumpbin /exports aes32.dll

2.4 Console Mode

AES4C functions can be called from Win32/Win64 console mode programs. A "console mode" program is a Windows command line program running in a command window. Although console mode programs look like DOS programs, they are Win32/Win64 programs that have access to the Win32/Win64 API and the entire Windows address space. Programming using console mode programs reduces the complexity of using GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code.

2.5 Static Linking

The registered user can also statically link aes32.obj (or aes64.obj) with their application, rather than making calls to the DLL's.

For example, to create an application that links aes32.obj statically:

(1) All application code that includes AES.H must define STATIC_LIBRARY before including AES.H

If using Microsoft Developer Studio, make these changes:

(1) To the project file: Do not add aes32.lib to the project file.
(2) To the settings: (See "Build Settings" or "Project/Settings")
(2a) C/C++ Tab: Add STATIC_LIBRARY to "preprocessor definitions:"
(2b) Link Tab: Add aes32.obj to "object/library modules:"
(3) Add #include "aes.h" to all source files that make calls to AES functions.

Alternatively, AES32.C can be edited so that it can be compiled and linked like any other program file. In order to do this, remove all code from AES32.C (provided when the source code is purchased) from

```
#ifndef STATIC_LIBRARY
```

through the following

```
#endif
```

2.6 Calling MarshallSoft AES from C++

Like Windows itself, **MarshallSoft AES** functions are coded in ANSI C, but **AES** functions can be called directly from both ANSI C programs and from C++ programs.

2.7 Adding MarshallSoft AES to an Existing Program

In order to call **AES** functions from an existing program,

(1) Add

```
#include "AES.H"
```

to the application source code,

(2) Link with aes32.lib (for MSVC), aes32bc5.lib (Borland C/C++ and C++ Builder), aes32.lib (Watcom), or aes32lcc.lib (Win32/LCC), and recompile from source.

For Win64, link with aes64.lib rather than aes32.lib

2.8 Error Display

The error message text associated with **AES** error codes can be displayed by calling **aesErrorText**. For example, if 'ErrCode' is returned by a AES function and it is negative (indicating an error), the text associated with the error code can be found by calling **aesErrorTex**t, as demonstrated by the following code segment:

```
void DisplayError(int ErrCode, char *MsgPtr)
{int Len;
 char ErrBuff[128];
 printf("ERROR %d: ", ErrCode);
 if(MsgPtr) printf("%s: ", MsgPtr);
 Len = aesErrorText(ErrCode, (char *)ErrBuff, 127);
 if(Len>0) printf("%s\n", ErrBuff);
 else printf("\n");
}
```

2.9 Explicitly Loading an AES DLL

When an application program runs that makes calls to aes32.dll (or aes64.dll), the Windows operating system will locate aes32.dll (or aes64.dll) by searching the directories as specified by the Windows search path. If the AES DLL is placed in \WINDOWS, it will always be found by Windows.

However, aes32.dll (or aes64.dll) can also be loaded from a specified directory by using the GetProcAddress API function:

```
// define fAttachType
typedef int  (WINAPI *fAttachType)(int, int);

static HMODULE hLib;
#ifdef _WIN64
static char PathToLib[] = "C:\\WINDOWS\\AES64.DLL";
#else
static char PathToLib[] = "C:\\WINDOWS\\AES32.DLL";
#endif

// allocate pointer variable to function
static fAttachType AttachPtr;

// load DLL from selected location
hLib = LoadLibrary(PathToLib);
if((int)hLib==0)
  {printf("Error: Could not load '%s' \n", PathToLib);
   exit(1);
  }
printf("Library '%s' is loaded\n", PathToLib);

// get address of function
AttachPtr  = (fAttachType) GetProcAddress(hLib, (char *)"aesAttach");

// attach the DLL
Code = (*AttachPtr)(AES_KEY_CODE, AES_PKCS7_MASK);

... use AES functions ...

// after finished with AES, unload the DLL
printf("Freeing '%s'\n", PathToLib);
FreeLibrary(hLib);
```

2.10 Targeting a 64-Bit CPU

If a compiler generates 32-bit application code and is running on a 64-bit version of Windows, then compiling and linking is the same as it were on a 32-bit Windows system. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

If a compiler generates 64-bit application code and is running on a 64-bit version of Windows, then the compiler must be reconfigured to generate 32-bit application code if the application will call 32-bit DLL's such as aes32.dll. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

2.10.1 Visual Studio C/C++: Versions 2005 through 2022

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click the "Configuration Manager" button in upper right corner.
Click the drop-down button below "Platform".
Click <New...>, then choose "x86" (Win32).

2.10.2 Visual Studio Visual Basic: Versions 2005 through 2022

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click "Build", then "Configuration Manager".
Click the drop-down button below "Active Solution Platform".
Click <New...>, then change "Any CPU" to "x86".

2.11 64-bit AES

64-bit DLL's may only be used by 64-bit application programs running on 64-bit Windows computers. This means that 64-bit application programs must be linked with aes64.lib instead of aes32.lib.

However, if a compiler generates 32-bit code, the application must be linked with aes32.lib even though it may be running on a 64-bit machine.

There are numerous AES4C 64-bit example programs. 64-bit Visual Studio Visual Studio project files can be found in the APPS sub-directory:

```
vc_Crypto(VS2012)x64.vcxproj   vc_Crypto(VS2012)x64.vcxproj
vc_TesAES(VS202)64.vcxproj     vc_TesAES(VS2012)x64.vcxproj
vc_vers(VS2012)x64.vcxproj     vc_vers(VS2012)x64.vcxproj
vc_Validate(VS2012)x64.vcxproj vc_Validate(VS2012)x64.vcxproj

. . .

vc_Crypto(VS2022)x64.vcxproj   vc_Crypto(VS2022)x64.vcxproj
vc_TesAES(VS2022)64.vcxproj    vc_TesAES(VS2022)x64.vcxproj
vc_vers(VS2022)x64.vcxproj     vc_vers(VS2022)x64.vcxproj
vc_Validate(VS2022)x64.vcxproj vc_Validate(VS2022)x64.vcxproj
```

3 Compiler Issues

AES programs can be compiled using an IDE or command line compiler tools. The following sections provide general compiler information.

3.1 Compiling Using an IDE

All current windows compilers have an "Integrated Development Environment" (IDE) for building application programs in the Windows environment.

Note that not only do IDE's vary between the different compiler manufacturers, but they also vary from version to version for the same compiler.

3.1.1 Compiling Example Programs with an IDE

Most of the example programs can be compiled from the compiler's IDE. For Visual C/C++, "project makefiles" are used since they can be used by all versions of Microsoft Visual C++. When opening the workspace, select "makefiles(.mak)" for the file type.

Alternatively, for Visual C++ v6.0, select "projects (.dsp)" for the file type.

3.1.2 Compiling New Projects with an IDE

All of the IDE's use the concept of a file hierarchy. For example, the AESVER example program files hierarchy in the IDE (for 32-bit) should look like:

```
aesver.exe
   +++ aesver.c
   +++ aes32.lib
```

Replace aes32.lib with aes32bc5.lib if using Borland C++ Builder, and with aes32lcc.lib if using LCC-Win32.

The order of the files is not significant. Refer to Section 4 below for a particular IDE.

3.2 Command Line Tool Setup

Many software developers overlook the power of using command line compilers. There are a number of very significant advantages to using the command line version of a C/C++ compiler. Among these are:

- **Easy of Use**: Once set up, typing a single key can compile one or a thousand programs.
- **Power**: Using the command line allows the use of batch files, automating complicated builds.
- **Compatibility**: Command line makefiles (unlike IDE project files) are normally compatible from one version of a compiler to the next.

In order to compile from the command line, command line compiler tools must be set up properly. Note that an option to install command line tools (or not to install them) is provided during installation of a C/C++ compiler. Refer to the compiler manufacturer's manual for details.

If necessary, the size of the environment table space can be increased by adding

```
SHELL=C:\COMMAND.COM /e:1024 /p
```

to CONFIG.SYS in C:\ and then reboot. Yes, this works for all versions of Windows, including Windows NT, 2000, Windows 8, Windows 7, Vista and XP

For all compilers, the path should point to the compiler's BIN directory. For example, to add "C:\BC50\BIN" to an existing path, use

```
PATH C:\BC50\BIN;%PATH%
```

3.2.1 Microsoft Visual C++

Set LIB and INCLUDE environment variables. For example,

```
SET INCLUDE=C:\MSVC\INCLUDE
SET LIB=C:\MSVC\LIB
```

3.2.2 Borland

Check that TURBOC.CFG, BCC32.CFG, TLINK.CFG, and TLINK32.CFG all have the correct information in them, as they should have when the compiler was installed. For example,

```
-IC:\BC5\INCLUDE
-LC:\BC5\LIB
```

BRCC (the Borland Resource Compiler) doesn't use the *.CFG files. Set the INCLUDE environment variable or BRCC will not be able to find the INCLUDE files (such as WINDOWS.H). For example,

```
SET INCLUDE=C:\BC5\INCLUDE
```

Clear the LIB environment variable (so it is not present when SET is typed at the command line) with

```
SET LIB=
```

3.2.3 Watcom

Set the WATCOM environment variables to point to the compilers include (H) and BIN directories. For example,

```
SET INCLUDE=C:\WC11\H;C:\WC11\H\NT
SET WATCOM=C:\WC11
SET EDPATH=C:\WC11\EDDAT
SET WWINHELP=E:\BINW
```

Watcom makefiles are available on request.

3.2.4 LCC (32-bit or 64-bit)

The LCC environment variables are set like the others. For example,

```
SET INCLUDE=C:\LCC\INCLUDE
SET LIB=C:\LCC\LIB
```

After making the above changes for the compiler, type PATH at the command line prompt to verify the search path, and type SET at the command line prompt to verify the INCLUDE and LIB environment variables.

3.3 Command Line Batch Files

If the compiler installation includes command line tools, then all of the example programs can be compiled directly from the command line. These same compiler commands can also be placed in a batch file.

See AESVER$LCC.BAT for an example of a command line batch file for LCC-Win32. Similarly, command line batch files can be created for all of the example programs.

3.4 Command Line Makefiles

Command line makefiles originated on UNIX systems. They are the standard way that C/C++ programs are constructed in command line environments. The advantage of makefiles (as compared to an integrated development environment) is that all compiler switches are always coded within the makefile and the makefile can be run with a single keystroke.

Command line makefiles are provided for Microsoft, Borland, Watcom, and LCC command line compilers. They can be found in the \APPS sub-directory created when SETUP was run

```
makefiles(borland50).zip     Borland C/C++ 5.0 makefiles.
makefiles(borland55).zip     Borland C/C++ 5.5 makefiles.
makefiles(lcc).zip           LCC project command files.
makefiles(microsoft).zip     Microsoft C/C++ makefiles.
makefiles(watcom11).zip      Watcom C/C++ 11 makefiles.
makefiles(gcc).zip           MinGW GCC project build files.
makefiles(mars).zip          Digital Mars project build files.
```

4.0 Supported Compilers

The **MarshallSoft AES Library for C/C++ (AES4C)** has been tested with and works with all versions of the following compilers:

- Microsoft Visual C/C++ (VC 4 & VC 6)
- Visual Studio C/C++ through Visual Studio 2022
- Visual C#,
- Borland C/C++
- Borland/Embarcadero C++ Builder
- Borland Turbo C/C++
- Watcom C/C++
- MinGW C++
- LCC
- Digital Mars.

Other Windows C/C++ compilers may work as well. Refer also to Section 5, "Compiling Example Programs".

4.1 Microsoft Visual C/C++ (VC 4.0 through 6.0)

Microsoft Visual C++ programs can be compiled from either the command line or from within the Microsoft development environment.

The last Win16 Microsoft compiler is version 1.52. All Microsoft Visual C++ compilers (version 4.0 through 6.0) compile Win32 programs ONLY. AES does not support Win16.

Visual C/C++ project files use file extension ".mak". Visual C/C++ 6.0 project files use file extension ".dsp"

Visual C/C++ makefiles use file extension "._M_" and are contained in the file makefiles(microsoft).zip.

4.2 Microsoft Visual Studio C/C++

AES4C works with all versions of Visual Studio

4.2.1 Microsoft Visual Studio C/C++ 2003 and 2005

Visual Studio 2003 through 2005 project files use file extension ".vcproj"

4.2.2 Microsoft Visual Studio C/C++ 2008

Visual Studio 2008 specific project files end with ".(VS2008).vcproj" for 32-bit applications and ".(VS2008)x64.vcproj" for 64-bit applications. Visual Studio 2008 can compile both 32-bit and 64-bit programs.

4.2.3 Microsoft Visual Studio C/C++ 2010

Visual Studio 2010 specific project files end with ".(VS2010).vcxproj" for 32-bit applications and ".(VS2010)x64.vcxproj" for 64-bit applications.

Note that project files for Visual Studio 2010 end with ".vcxproj" rather than ".vcproj" as in earlier versions of Visual Studio.

4.2.4 Microsoft Visual Studio C/C++ 2012

Visual Studio 2012 specific project files end with ".(VS2012).vcxproj" for 32-bit applications and ".(VS2012)x64.vcxproj" for 64-bit applications.

In order to convert an older project file to VS 2012:

1. Open the older (.dsp, .vcproj, .vcxproj) project file with VS 2012.
2. Let VS 2012 update to 2012 format when prompted.
3. Select "Project", "Properties", "Linker", then "Advanced".
4. Change the "Image Has Safe Exception Handlers" to NO (/SAFESEH:NO)
5. Save project. This will insert the line

```
<ImageHasSafeExceptionHandlers>false</ImageHasSafeExceptionHandlers>
```

into your project file as the last line before `</Link>`

4.2.5 Microsoft Visual Studio C/C++ 2013 thru 2022

Visual Studio 2022 specific project files end with ".(VS2022).vcxproj" for 32-bit applications and ".(VS2022)x64.vcxproj" for 64-bit applications.

Projects can also be converted from VS 2012 as described in Section 4.2.4.

4.3 Microsoft C/C++ Express Edition

The “Express Edition” of Microsoft Visual Studio is available as a free download at
http://www.microsoft.com/express/download/

Open the VC project file as in previous versions of Visual Studio.

4.4 Microsoft Studio Visual C#

AES functions can be called from Microsoft Visual C# (C-sharp) in the same manner as Win32 API functions.

All C# projects end with extension ".csproj". For example,

cs_vers.csproj
ClientCS.csproj

In order to call **AES** functions from an existing C# programs, do the following to the C# source code:

(1) Add

using System.Runtime.InteropServices;

to the application source code.

(2) Add the contents of file `aes_funs.cs` to the application source code after

public class aes : System.Windows.Forms.Form

(3) Add the constants from `aes_cons.cs` to the program as they are needed.

Look at the `cs_vers` and `ClientCS` example projects in the APPS directory.

(4) Set "unsafe" compilation for any functions that calls AES functions. For example

```
private unsafe void button1_Click(object sender, System.EventArgs e)
```

(5) Verify that `AllowUnsafeBlocks` is set to true in the project file `(.vcproj)`. That is,

```
AllowUnsafeBlocks = "true"
```

4.5 Borland C/C++

Borland C/C++ Version 5.0 programs can be compiled from either the command line (using makefiles ending with "._B_") or from within the Borland development environment using Borland v5.0 or above. All Borland 5.0 makefiles are contained in file `makefiles(borland50).zip`.

Borland C/C++ Version 5.5 (which can be downloaded from `http://edn.embarcadero.com/article/20633`) is a free Win32 console mode compiler (no IDE). Makefiles for BC v5.5 end with "._I_", and (like Borland C++ Builder) use ILINK32 rather than TLINK32. Be careful with linker response files (*.RSP) -- they must **NOT** end with a carriage return / line feed!

All Borland 5.5 makefiles are contained in file `makefiles(borland55).zip`.

Borland programs always link with aes32bc5.lib.

4.6 Borland C++ Builder

Borland C++ Builder does not have command line tools, so all programs must be compiled from the Borland C++ Builder integrated environment.

Note that aes32bc5.lib is the LIB file used with Borland C++ Builder. AES32BC5 .LIB can be created from aes32.dll by using the Borland IMPLIB program:

```
IMPLIB aes32bc5.lib aes32.dll
```

The file C-BUILDER.ZIP contains the Borland C++ Builder project makefiles.

4.7 Watcom C/C++

AES4C works with Watcom 11 and Open Watcom (http://www.openwatcom.org).

Watcom C/C++ programs can be compiled from either the command line or from within the Watcom development environment.

All Watcom 11 makefiles use file extension "._W_" and are contained in file `makefiles(watcom11).zip`

Watcom IDE

To create a new project, choose "File", then "New Project". Enter the project name and then choose Win32 as the target. Use the INS (Insert) key to pop up a dialog box into which the project file names are entered.

Select "Options" from the main window, then "C Compiler Switches", then "10", Memory Models and Processor Switches". Check "80386 Stack based calling [-3s]", then check "32-bit Flat model [-mf]".

4.8 LCC-Win32/Win64 C/C++

LCC-Win32/Win64 C programs can be compiled from either the command line or from within the development environment.

LCC-Win32/Win64 is a freeware C compiler developed and distributed by Jacob Navia at

http:/ww.cs.virginia.edu/~lcc-win32/

All LCC-Win32/Win64 makefiles use file extension "._L_" are contained in file makefiles(lcc).zip

4.9 MinGW C/C++

MinGW (Minimalist GNU for Windows) is part of the GNU Compiler Collection (GCC), and GNU Binutils, for use in the development of native Microsoft Windows applications. See http://www.mingw.org

All MinGW GCC makefiles use file extension "._G_" are contained in file makefiles(gcc).zip

4.10 Digital Mars C/C++

All Digital Mars makefiles use file extension "._D_" are contained in file makefiles(mars).zip

5 Compiling Programs

At the current time, Microsoft Visual Studio is the only C/C++ compiler that can generate 64-bit application code.

5.1 Compiling AES Source Code

This section applies only to those who have purchased source code for the **MarshallSoft Advanced Encryption Standard Library.**

aes32.dll and AES64.DLL have been compiled using Microsoft Visual C++, and are callable from applications written using Microsoft, Borland, or Watcom compilers. If AES.C is recompiled using Borland or Watcom compilers, then the resulting aes32.dll can only be used by applications compiled with the same compiler, unless the "_stdcall" and "_declspec" keywords are specified.

Microsoft Visual C++ is used to create aes32.dll and AES32.OBJ (for static linking).

In order to create aes32.dll, type:

```
NMAKE -f AES32._M_
```

In order to create AES32.OBJ (for static linking) type

```
NMAKE -f AES32S._M_
```

Alternatively, AES32.C can be included in a project (along with MSC-VS.C and MSC-STR.C) like any other C file. Before compiling, define the symbol STATIC_LIBRARY.

AES64.DLL has been compiled using Microsoft Visual Studio 2008, and is callable from 64-bit applications programs.

5.2 Compiling Example Programs

The example programs can be compiled by using either the command line compiler or the compiler integrated development environment (IDE). Most compiler vendors provide both IDE and command line tools, although some compilers are command line only (Borland C/C++ 5.5 and LCC-Win32) or IDE only (Borland C/C++ Builder).

Refer also to Section 4, "Supported Compilers".

5.2.1 Compiling Using Visual C++ (VC v4.0, v5.0, and v6.0)

Microsoft Visual C/C++ (v4.0, v5.0, v6.0) compiles only 32-bit programs.

Programs can be compiled with Microsoft Visual C/C++ using either Developer Studio / Visual Studio or the command line compiler. Project makefiles (files ending in ".MAK") are provided for Developer Studio / Visual Studio. Command line makefiles (files ending in "._M_") are provided for use with the command line compiler (e.g.: NMAKE -f AESVER(LIB)32._M_).

MSVC v6.0 project files (files ending in ".DSP") are also provided.

5.2.2 Compiling Using Visual C++ .Net

Microsoft Visual C/C++ .NET compiles only 32-bit programs.

Programs can be compiled with Microsoft Visual C/C++ .NET using either Visual Studio .NET or the command line compiler. Project files (files ending in ".VCXPROJ" or ".VCPROJ") are provided for Visual Studio (e.g.: VC_VERS.VCPROJ).

Command line makefiles (files ending in "._M_") are provided for use with the Visual C++ .NET command line compiler (e.g.: NMAKE -f AESVER(LIB)32._M_).

5.2.3 Compiling Using Visual C#

AES functions can also be called from C# programs in the same manner in which Win32 API DLL functions are called.

5.2.4 Compiling Using Borland C/C++ 5.0

Borland C/C++ 5.0 can compile both 32-bit and 16-bit programs.

Programs can be compiled with Borland C/C++ using either the Borland IDE or the command line compiler. Several Project files (files ending in ".IDE") are provided for the Borland IDE (unzip BC50-IDE.ZIP) and command line makefiles (files ending in "._B_") for the Borland command line compiler. For example (MAKE -f AESVER(LIB)32._B_)

5.2.5 Compiling Using Borland C/C++ 5.5

Borland C/C++ 5.5 is a command line compiler that can compile both 32-bit programs only. Command line makefiles (ending in "._I_") are provided. For example, (MAKE -f AESVER(LIB)32._I_) .

5.2.6 Compiling Using Borland C++ Builder

Borland C++ Builder (BCB) is an IDE that features "drag and drop" forms building (like Delphi and Visual Basic).

BCB compiles 32-bit programs only. Compile the BCB example program BCB_PRJ with BCB_PRJ.MAK if running BCB version 1 through 3, and compile with BCB_PRJ.BPR if running BCB version 4 or above.

The file C-BUILDER.ZIP contains the Borland C++ Builder project makefiles.

5.2.7 Compiling Using Watcom 11 / Open Watcom

Programs can be compiled with Watcom 11 using either the Watcom IDE or the command line compiler. Several command line makefiles (files ending in "._W_") for the Watcom command line compiler are provided. For example (e.g.: WMAKE -f AESVER(LIB)32._W_)

5.2.8 Compiling Using LCC-WIN32

Several command line makefiles (files ending in "._L_") for the LCC command line compiler are provided. For example (e.g.: MAKE -f AESVER(LIB)32._L_)

5.2.9 Compiling Using MinGW

Several command line makefiles (files ending in "._G_") for the GCC command line compiler are provided. For example (e.g.: MAKE -f AESVER(LIB)32._G_)

5.2.10 Compiling Using Digital Mars

Several command line makefiles (files ending in "._D_") for the Digital Mars command line compiler are provided. For example (e.g.: MAKE -f AESVER(OBJ)32._D_)

6 Example Programs

Many of the example programs are written in Win32 console mode. This was done in order to provide the clearest possible code, without the complication and complexity of GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code. Example programs are located in the APPS sub-directory. Refer to Section 4.0 for help with compiling and using makefiles.

See Section 1.1 “Features” above for a list of all example programs in a tabular format.

Project files are classified as follows:

- `*._M_` Microsoft C/C++ makefile (command line).
- `*._B_` Borland C/C++ 5.0 makefile (command line).
- `*._I_` Borland C/C++ 5.5 makefile (command line).
- `*._W_` Watcom C/C++ makefile (command line).
- `*._L_` LCC makefile (command line).
- `*._G_` GCC ++ makefile (command line).
- `*._D_` Digital Mars makefile (command line)
- `*.dsp` Microsoft v6.0 project file.
- `*.vcproj` Microsoft Studio Visual C++ project file.
- `*.vcxproj` Microsoft Visual Studio C++ project file (VS2010 to VS2022).
- `*.csproj` Microsoft Visual Studio C# project file.

Microsoft Visual C/C++ Developer Studio files end in ".MAK" (or ".DSP" for v6.0) and can be loaded with "Open Workspace".

Please refer to the AES User’s Manual (AES_USR.PDF) for basic information on Advanced Encryption Standard protocols.

Note: Contact us at **info@marshallsoft.com** with Subject “AES4C HELP” if you are having problems running the evaluation version with Windows Server.

6.1 aesver

aesver.c is the console mode program that displays the **MarshallSoft AES** library version number, build, and registration string. Compile and run this program to verify that **AES4C** has been correctly installed.

See the COMPILE comment in **aesver.c** for information on compiling.

6.2 TestAES

TestAES.c is console mode example program that demonstrates how to encrypt and decrypt messages.

See the COMPILE comment in **TestAES.c** for information on compiling.

6.3 TestDH

TestAES is a Visual Studio (console mode) example program that demonstrates how to do Diffie-Hellman key exchange.

See the COMPILE comment in **TestDH.c** for information on compiling.

6.4 Crypto

Crypto.c is a Windows (GUI mode) example program that encrypts and/or decrypts a file.

See the COMPILE comment in **Crypto.c** for information on compiling.

6.5 Encrypt

Encrypt.c is a console mode program that encrypts a user specified file.

See the COMPILE comment in **Encrypt.c** for information on compiling.

6.6 Decrypt

Decrypt.c is a console mode program the decrypts a previously encrypted file.

See the COMPILE comment in **Decrypt.c** for information on compiling.

6.7 Password

Password.c is a GUI mode program that manages a set of 5 passwords which are always kept encrypted on disk. A master password is used to access the set of 5 passwords.

See the COMPILE comment in **Password.c** for information on compiling.

6.8 HashDigest

HashDigest.c is a console mode program that computes the SHA 256 hash digest of a data buffer.

See the COMPILE comment in **HashDigest.c** for information on compiling.

6.9 Validate

Validate is a console mode program that validates against an on-line AES encryption tool such as:

```
https://www.javainuse.com/aesgenerator
https://ieasynote.com/tools/aes
https://the-x.cn/en-us/cryptography/Aes.aspx
```

See the COMPILE comment in **Validate.c** for information on compiling.

6.10 vc_vers

vc_vers.cpp is a Visual Studio (GUI mode) example program that displays the DLL Version, Build #, and registration string. Compile and run this program to verify that **AES4C** has been correctly installed.

See the BUILD comment in **vc_vers.cpp** for information on compiling.

6.11 vc_TestAES

vc_TestAES is a Visual Studio (console mode) example program that demonstrates how to encrypt and decrypt messages.

See the BUILD comment in **vc_TestAES.cpp** for information on compiling.

6.12 vc_TestDH

vc_TestDH is a Visual Studio (console mode) example program that demonstrates how to do Diffie-Hellman key exchange.

See the BUILD comment in **vc_TestDH.cpp** for information on compiling.

6.13 vc_Crypto

vc_Crypto is a Visual Studio (GUI mode) example program that encrypts and/or decrypts a file.

See the BUILD comment in **vc_Crypto.cpp** for information on compiling.

6.14 vc_Password

vc_Password is a Visual Studio (GUI mode) example program that manages a set of 5 passwords which are always kept encrypted on disk. A master password is used to access the set of 5 passwords.

See the BUILD comment in **vc_Password.cpp** for information on compiling.

6.15 vc_HashDigest

HashDigest is a Visual Studio (console mode) program that computes the SHA 256 hash digest of a data buffer.

See the BUILD comment in **vc_HashDigest.cpp** for information on compiling.

6.16 vc_Validate

vc_Validate is a Visual Studio (console mode) program that validates against an on-line AES encryption tool such as:

```
https://www.javainuse.com/aesgenerator
https://ieasynote.com/tools/aes
https://the-x.cn/en-us/cryptography/Aes.aspx
```

See the BUILD comment in **vc_Validate.cpp** for information on compiling.

7 Revision History

Version 1.0: March 19, 2013

- The initial public release of AES4C.

Version 2.0: June 4, 2014

- Added aesEncryptWrite() function that encrypts data & writes to a file.
- Added aesReadDecrypt() function that reads an encrypted file & decrypts.
- Added aesSha256() function that computes the SHA-256 data hash.
- Added AES_SHA256_METHOD key generation method to aesMakeUserKey().
- Added support for Visual Studio 2013.
- Added support for Digital Mars C/C++ compiler.
- Added PASSWORD.C and VC_PASSWORD.CPP example programs.

Version 3.0: May 6, 2015

- Replaced function aesSha25() with aesSha256Data() and aesSha256File().
- Added PKCS7 padding option to aesPadBuffer().
- Added AES_PKCS7_MASK to “Flags” argument in aesAttach() to set file padding to PKCS7.

Version 4.0: November 16, 2016

- Added aesDecryptBuffer() that decrypts buffer of (encrypted) bytes.
- Added aesEncryptBuffer() that encrypts buffer of bytes.
- Added aesRemovePad() that removes PKCS7 padding.
- Added aesSaltPass() that generates ("salts") additional password characters.
- Added example program HashDigest that computes SHA 256 hash digest.

Version 4.1: June 23, 2017

- Fixed problem in aesMakeUserKey() using AES_SHA256_METHOD.
- Added AES_MIXED_METHOD method to aesMakeUserKey().
- Added aesSetInteger() and AES_SET_SEED that seeds the random number generator.
- Added aesShredFile() that shreds (overwrites with zeros then deletes) a file.

Version 4.2: July 5, 2018

- Added cryptograhically secure pseudo-random number generator aesSecureRandom().
- Added AES_GET_SECURE_SIZE to aesGetInteger().

Version 5.0: July 8, 2020

- Fixed problem in which AES_SET_SEED did not always reset the seed.
- Replaced deprecated function strncpy().
- Fixed internal problem with long (over 42 characters) pass phrases.
- Added function aesEncodeBase64 that Base64 encodes a data buffer.
- Added function aesDecodeBase64 that decodes a Base64 encoded data buffer.

Version 6.0: February 14, 2022

- Increased internal IO_BUFFER_SIZE to 256K
- Fixed problem handling large files in aesEncryptWrite() & aesReadDecrypt()
- Replaced deprecated functions strcpy().and strlen()
- Added Diffie-Hellman key exchange function aesMakeKeyPair()
- Added Diffie-Hellman key exchange function aesMakeSharedKey()
- Added Visual Studio C/C++ 2022 project files.