MarshallSoft AES

(Advanced Encryption Standard)

Library for Visual FoxPro

Programmer's Manual

(AES4FP)

Version 4.1

June 28, 2017.

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

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

Email: info@marshallsoft.com
Web: www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

The **MarshallSoft Advanced Encryption Standard Library for Visual FoxPro (AES4FP)** is a toolkit that allows software developers to easily implement strong encryption and decryption into a Windows Visual FoxPro 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 Visual FoxPro** provides information needed to compile and run programs in a Visual FoxPro programming environment.

The **MarshallSoft Advanced Encryption Standard DLL's** will work under all versions of Windows through Windows 10. A Win32 DLL is provided (a 64-bit DLL is available).

AES4FP includes several Visual FoxPro example programs which demonstrate AES encryption and decryption.

The **MarshallSoft Advanced Encryption Standard Library for Visual FoxPro** component library supports and has been tested with all versions of 32-bit Visual FoxPro.

The **MarshallSoft AES** DLLs (AES32.DLL and AES64.DLL) can also be used from any language (C/C++, C#, Borland/Embarcadero Delphi, Visual Basic, COBOL, Xbase++, Visual dBase, Microsoft Office, etc.) capable of calling the Windows API.

For the latest version of the **MarshallSoft AES** software, see www.marshallsoft.com/aes4fp.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

Some of the many features of the **MarshallSoft Advanced Encryption Library (AES)** component library are as follows:

- Supports both 32-bit and 64-bit Windows.
- Implements the 256-bit "**Advanced Encryption Standard**" (Rijndael)
- Supports **ECB** (Electronic Cookbook) mode.
- Supports **CBC** (Cipher Block Chaining) mode.
- Supports SHA-256 cryptographic hash algorithm.
- Supports PKCS7 padding.
- **Free** technical support and updates for one year.
- License covers all programming languages.
- Royalty free distribution with a compiled application.
- Evaluation versions are fully functional. (30 day trial). No unlock code is required.
- Can be used from GUI mode or console mode programs.
- Is fully thread safe.
- Supports 32-bit and 64-bit Windows through Windows 10.
- Implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Will run on machines with or without .NET installed.
- Works with all 32-bit versions of Microsoft Visual FoxPro.
- Does not depend on support libraries. Makes calls to core Windows API functions only.
- Can also be used with any program (in any language) capable of calling Windows API functions such as C/C++, C#, Visual Basic, Delphi, Xbase++, dBASE or COBOL.
- 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 Visual FoxPro example programs with full source code is included. Refer to Section 4 for more details on each of the example programs.

```
1. Aesver     Displays AES4FP version
2. TestAES    Performs AES encryption / decryption tests
3. Crypto     Encrypts and/or decrypts a file
4. Encrypt    Encrypts a file
5. Decrypt    Decrypts a file
6. Password   Manages passwords kept encrypted on disk.
7. HashDigest Computes the SHA 256 hash digest
```

Registration includes one year of free technical support and updates.

1.2 Documentation Set

The complete set of documentation consists of three manuals in Adobe PDF format. This is the first manual (AES_4FP) in the set.

- AES4FP Programmer's Manual (AES_4FP.PDF)
- AES User's Manual (AES_USR.PDF)
- AES Reference Manual (AES_REF.PDF)

The AES_4FP Programmer's Manual (AES_4FP.PDF) is the language specific (Visual FoxPro) manual. All language dependent programming issues such as compiling, compilers and example programs are discussed in this manual.

The AES User's Manual (AES_USR.PDF) discusses email processing as well as language independent programming issues. Purchasing and license details are also provided.

The AES Reference Manual (AES_REF.PDF) contains details on each individual AES function. The manual also contains a list of AES error codes.

Online documentation can be accessed on the **MarshallSoft AES Engine for Visual FoxPro** product page at:

http://www.marshallsoft.com/aes4fp.htm

1.3 Example Program

The following example segment demonstrates the use of some of the **MarshallSoft AES for Visual FoxPro** component library functions:

```
Function EncryptFile(KeyBuffer, InFile, OutFile)
Local Code
Local Control
Local Vector
' vector not used in ECM mode
Vector = Chr(0)
Control = "*"
' initialize for encryption in ECB mode
Code = aesInitAES(@KeyBuffer,@Vector,AES_ECB_MODE,AES_ENCRYPT,@Control)
If Code < 0
  return Code
  exit
endif
' encrypt the file (InFile -> OutFile)
Code = aesEncryptFile(@Control, @InFile, @OutFile)
return Code
```

1.4 Installation

(1) Before installation of AES4FP, a Visual FoxPro compiler (any version) should already be installed on your system and tested.

(2) Unzip AES4FP40.ZIP (evaluation version) or AESxxxxx.ZIP (purchased version where xxxxx is your Customer ID) using any Windows unzip program.

(3) Run the installation program SETUP.EXE that will install all AES4FP files and copy the AES32.DLL to your Windows directory.

(4) You're ready to compile and run! For a quick start, load project file AESVER.PRG

1.5 Uninstalling

Uninstalling AES4FP is very easy.

First, run UINSTALL.BAT, which will delete AES32.DLL from your Windows directory, typically C:\WINDOWS.

Second, delete the **AES** project directory created when installing AES4FP.

1.6 Pricing

A developer license for the MarshallSoft AES Library can be purchased for $115 USD ($195 with ANSI C source code to the DLL.) Purchasing details can be found in the AES User's Manual, Section 1.4, "How to Purchase", (AES_USR.PDF).
Also see INVOICE.TXT or

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

Registration includes one year of free updates and technical support. Registered DLLs never expire.

1.7 Updates

When a developer license is purchased, the developer will be sent a registered DLL plus a license file (AESxxxxx.LIC, where xxxxx is the Customer ID). 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 and receive technical support. The license can be updated for:

- $30 if the update is ordered within one year of the original purchase (or previous update).
- $55 is the update is ordered between 1 and 3 years of the original purchase (or previous update).
- $75 if the update is ordered after three years of the original purchase (or previous update).

Note that the registered DLL, AES32. DLL does NOT expire. The update price also includes **technical support** for an additional year. Refer to the file UPDATES.TXT located in the /AES4FP/DOCS directory for more information.

2 Library Overview

The **MarshallSoft Advanced Encryption Library (AES)** component library has been tested on multiple computers running Windows XP through Windows 10.

The AES4FP library has been tested and works with all versions of 32-bit Visual FoxPro.

The SETUP installation program will copy the AES DLL to the Windows directory and copies the AES4FP files to the directory specified (default \AES4FP). Three sub-directories are created, as follows:

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

2.1 Dynamic Link Libraries

The **MarshallSoft AES** component library includes a Win32 dynamic link library (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".

2.2 Keycode

AES32.DLL has a keycode encoded within it. Your keycode is a 9 or 10-digit decimal number (unless it is 0), and will be found in the file KEYCODE.FOX. The keycode for the evaluation version is 0. You will receive a new keycode and a new AES32.DLL after purchasing or updating a developer license. The KEYCODE is passed to **aesAttach**.

If you get an error message (value -74) when calling **aesAttach**, it means that the keycode in your application does not match the keycode in the DLL. After registering, it is best to remove the evaluation version of the AES32.DLL from the Windows search path or delete it.

2.3 Win32/Win64 STDCALL and DECLSPEC

AES is written in ANSI C and is compiled using the _stdcall and _declspec keywords. This means that AES4FP uses the same calling conventions and file naming conventions as the Win32/Win64 API. In particular, function names are NOT decorated. Neither leading underscores nor trailing "@size" strings are added to the names of functions.

Any Windows application program may call the AES32 library provided that the proper declaration file is used.

2.4 Error Display

The error message text associated with **AES** error codes can be displayed by calling **aesErrorText**. Each sample program contains examples of error processing.

Also see the file seeErrors.txt for a list of all Winsock and **AES** error codes.

2.5 INCLUDE Files

All example programs contain two INCLUDE files; `KEYCODE.FOX` and `AES32CON.FOX`. The file `AES32CON.FOX` contains all the necessary constants for AES4FP, while the file `KEYCODE.FOX` contains your keycode, as discussed in Section 2.2.

Since function declarations cannot be in an INCLUDE file (at least through VFP version 5.0), they are listed in each program following the two INCLUDE files (`KEYCODE.FOX` and `AES32CON.FOX`). The complete list of function declarations is also in the file `AES32FUN.FOX`

Due to the behavior of Visual FoxPro regarding `INCLUDE` files, it is strongly recommended that the `INCLUDE` files `KEYCODE.FOX` and `AES32CON.FOX` be replaced with their contents in application programs (i.e., copy and paste contents) of the INCLUDE file.

2.6 FoxPro Forms

MarshallSoft AES functions can be called from any Visual FoxPro code module, such as programs, classes, and forms.

2.7 Adding AES to a VFP Program

1 - Add the AES constants (found in AES32CON.FOX) that will be used in the developer's application.

2 - Add the AES function declarations (found in AES32FUN.FOX) that will be called from the developer's application.

Refer to the example programs.

2.8 Dynamic Strings

The Visual FoxPro language uses a technique known as "garbage collection" to manage string space at runtime, and may be called internally at any time by the FoxPro runtime, asynchronous to what you may be doing in your code.

When passing a string buffer to a DLL function into which text will be copied, it is strongly recommended that the local string be allocated immediately before use. For example:

```
Code = aesVerifyControl(@Control)
if Code < 0
  * allocate buffer just before call
  Temp = SPACE(128)
  * put text in Temp
  Code = aesErrorText(Code,@Temp,128)
  ? Left(Temp,Code)
endif
```

This technique is not necessary for passing a string to a DLL function, only when passing a buffer to a DLL into which data is to be placed by the DLL function.

2.9 Null Terminated Strings

A string is "null terminated" if the last character in the string is a Chr(0) character.

Strings passed to AES functions should be null terminated unless the length of the string (or data) is also being passed or it is of known length. For example, in the following example only the PassPhrase string needs to be null terminated since the KeyBuffer string is of known length (always 32 bytes).

```
PassPhrase = PassPhrase + Chr(0)
KeyBuffer = Space(AES_KEY_SIZE)
Code = aesMakeUserKey(@PassPhrase, @KeyBuffer, 0)
```

All strings returned from AES functions are null terminated unless they are of known length. These strings may be converted for FoxPro in one of two ways: (1) if the length of the string is known, use the FoxPro `LEFT` function: For example,

```
Buffer = Space(128)
Code = aesErrorText(ErrCode, @Buffer, 127)
If Code > 0 Then
  ? Left(Buffer, Code)
End If
```

If the length of the null terminated string is not known, use the FoxPro `AT` function to find the position of Chr(0).

```
Pos = AT(Chr(0), Buffer)
if Pos > 0 Then
  ? Left(Buffer, Code)
End If
```

3 Compiler Issues

The **MarshallSoft AES Library for Visual FoxPro** component library works with all versions of 32-bit Visual FoxPro.

3.1 Compiling Programs

The example programs are compiled from the Visual FoxPro development environment. However, all AES functions can also be called from any Visual FoxPro code module such as VFP forms.

3.2 Compiling to an Executable

FoxPro programs end in ".PRG". They can be compiled to an executable using the FoxPro BUILD command.

For example, to create AESVER.EXE from AESVER.PRG in the C:\TEMP directory, type the following in the FoxPro command window:

```
BUILD PROJECT C:\TEMP\AESVER FROM C:\TEMP\AESVER
BUILD EXE C:\TEMP\AESVER FROM C:\TEMP\AESVER
```

FoxPro executables require VFP500.DLL and VFP5ENU.DLL (ENglish User), and may have to be copied from the VFP CDROM. If you are using an earlier or later version of VFP than version 5.0, substitute the appropriate DLL's for the above.

The FoxPro output display window will disappear as soon as your executable completes. In order to allow the user to control when the display window disappears, add the following code to your application, just before the final return.

```
?  " Type any key to exit..."
X = InKey(0)
```

4 Example Programs

All example programs are written for 32-bit FoxPro. Each has been tested and shows how to correctly use AES functions. It suggested that the developer compile and run the example programs before developing an application using AES4FP.

Note that all AES functions can also be called from all Visual FoxPro code modules.

4.1 aesver

The Aesver example program displays the **AES** library version number and registration string. . Its purpose is to display the **AES** version, build, and registration string as well as to verify that AES32.DLL is being found and loaded by Windows.

From the VFP command line, type

```
Do \aes4fp\apps\aesver.prg
```

4.2 TestAES

The **TestAES** example program demonstrates how to encrypt and decrypt messages.

From the VFP command line, type

```
Do \aes4fp\apps\TestAES.prg
```

4.3 Crypto

The **Crypto** example form demonstrates how to encrypt a file and to decrypt a (previously encrypted) file.

From the VFP menu bar, choose "File" then "Open" to load the form Crypto.scx

4.4 Encrypt

The **Encrypt** example program demonstrates how to encrypt a file.

From the VFP command line, type

```
Do \aes4fp\apps\Encrypt.prg
```

4.5 Decrypt

The **Decrypt** example program demonstrates how to decrypt a (previously encrypted) file.

From the VFP command line, type

```
Do \aes4fp\apps\Decrypt.prg
```

4.6 Password

Password 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.

From the VFP command line, type

```
Do \aes4fp\apps\password.prg
```

4.7 HashDigest

HashDigest computes the SHA 256 hash digest of a data buffer.

From the VFP command line, type

```
Do \aes4fp\apps\HashDigest.prg
```

5 Revision History

The MarshallSoft AES Engine DLLs (AES32.DLL and AES64.DLL) are written in ANSI C. All programming language versions of AES (C/C++, .NET, Visual Basic, VB .NET, PowerBASIC, Visual FoxPro, Delphi, dBase, Xbase++, COBOL, and FORTRAN) use the same identical DLLs.

Version 1.0: April 9, 2013.

- Initial release of the Visual FoxPro version.

Version 2.0: June 25, 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 PASSWORD example program.

Version 3.0: May 15, 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 23, 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 28, 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.