Amayeta SWF Encrypt 6.0 DLL Edition

This document contains information on how to successfully use the SWF Encrypt 6.0 DLL in your C++, C# and Delphi Applications.

Please note that this guide contains information specific to the SWF Encrypt 6.0 DLL Edition (SWFEncryptG.dll).

Note for Trial Users

The Evaluation DLL will add a Watermark to Encrypted SWFs. This watermark is not present in the commercially licensed version.

If you are encrypting Flash9/10 ActionScript 3.0 SWF Files, the Trial DLL will only Encrypt at 50% Encryption Intensity.

Note for Registered Users

If you are using the Full Registered version of the DLL you must call a *password* function in the DLL to validate your license and enable the DLL. The procedure for doing this is available at the end of each chapter under the heading “Full Version Serial Validation”.

Please contact sales@amayeta.com for Purchase Enquiries or visit http://www.amayeta.com

C++ Usage

In order to use the SWF Encrypt DLL you'll need to start by declaring two function pointers:

```
// function pointer types
typedef  void (__stdcall *encfilefuncPtr)(const char* srcswf,const
char* destswf);
typedef int (__stdcall *fileencedfuncPtr)(const char* destswf);
```

To load and set up the library you can use the following code (note: this assumes the DLL will be in the same directory as the executable):

```
HINSTANCE hinstDLL = LoadLibrary ("SWFEncryptG.dll");


// declare pointer to dll functions
encfilefuncPtr encfile;
fileencedfuncPtr fileenced;


// get the function addresses in the dll
encfile = (encfilefuncPtr)GetProcAddress (hinstDLL,"encfile");
fileenced = (fileencedfuncPtr)GetProcAddress
(hinstDLL,"fileenced");
```

As long as the *hInstDLL*, *encfile* and *fileenced* pointers are not equal to null, the DLL functions should be ready to use.

In the following code we are going to encode a SWF file in “c:\test\test_in.swf” to “c:\test\test_out.swf”. The function *encfile* takes two pointers to char type, the first parameter is the SWF source path and the second is the destination SWF name.

```
char *filename  = "C:\\test\\test_in.swf";

char *filename2  = "C:\\test\\test_out.swf";

encfile(filename,filename2);
```

Now to test if the file is encoded, we can use the *fileenced* function. This function takes one parameter, a pointer to char containing the location of the SWF, and returns a number.

```
int ifileEncResult = fileenced(filename2);
```

fileenced returns 1 if the SWF has been encrypted correctly and 0 if it could not be encrypted. You might want to declare this as a constant in your application:

```
// constant true for fileenced function

const FILEENC_TRUE = 1;
```

Once you have finished using the SWF Encrypt DLL, you just need to free it:

```
FreeLibrary(hinstDLL);
```

Advanced Usage

In order to use the additional parameters you’ll need to start by declaring a different *encfile* function pointer:

```
// function pointer types
typedef  void (__stdcall *goldencfilefuncPtr)(const char*
srcswf,const char* destswf,const int minlevel,const int enclevel);
```

Note that the encryption function is different and contains more parameters.

To load and set up the library you can use the following code (note: this assumes the DLL will be in the same directory as the executable):

```
HINSTANCE hinstDLL = LoadLibrary ("SWFEncryptG.dll");


// declare pointer to dll functions
goldencfilefuncPtr encfile;
fileencedfuncPtr fileenced;


// get the function addresses in the dll
encfile = (goldencfilefuncPtr)GetProcAddress (hinstDLL,"encfile");
fileenced = (fileencedfuncPtr)GetProcAddress
(hinstDLL,"fileenced");
```

As long as the *hInstDLL*, *encfile* and *fileenced* pointers are not equal to null, the DLL functions should be ready to use.

In the following code we are going to encode a SWF file in “c:\test\test_in.swf” to “c:\test\test_out.swf” . The function *encfile* takes two pointers to char type and 2 integers, the first parameter is the SWF source path and the second is the destination SWF name. The first integer specifies the minimum length of unencrypted bytes in the movie (20 in this example), the second specifies the encryption strength (range 1 to 7, 3 is used in this example).

```
char *filename  = "C:\\test\\test_in.swf";
char *filename2  = "C:\\test\\test_out.swf";
```

```
encfile(filename,filename2,20,3);
```

You can now use the *fileenced* function to check if the SWF is encrypted if you wish, usage of this function is previously detailed in the Silver documentation.

Once you have finished with the DLL, you must also free it the same way as specified previously.

The SWF Encrypt 6.0 DLL includes new functions specific to Flash 9/10 and ActionScript 3.0 Encryption. These are outlined below.

Enccodes Function

This function enables the new “Code Wrapping” Feature in SWF Encrypt 6.0. This function only applies to Flash 9/10/ActionScript 3.0 Files. This function should be disabled for maximum post-encryption compatibility, although encryption will then be weaker.

```
// get the function addresses in the dll

enccodesfuncPtr enccodes;

enccodes = (enccodesfuncPtr)GetProcAddress (hinstDLL, "enccodes");

// Execute function

enccodes (true);
```

Encnames Function

This function enables the new “Encrypt Public System Classes” Feature in SWF Encrypt 6.0. This function only applies to Flash 9/10/ActionScript 3.0 Files. This function should be disabled for maximum post-encryption compatibility, although encryption will then be weaker

```
// get the function addresses in the dll

encnamesfuncPtr encnames;

encnames = (encnamesfuncPtr)GetProcAddress (hinstDLL,
"encnamescodes");
```

```
// Execute function

encnames (true);
```

TestAS3 Function

This function is used to test if a specific SWF is Flash 9/10 and uses ActionScript 3.0:

```
// get the function addresses in the dll

testas3funcPtr testas3;

testas3 = (testas3funcPtr)GetProcAddress (hinstDLL, "testas3");

// Execute function

int result = testas3 (“C:\\test.swf”);
```

Full Version Serial Validation

If you are using the full commercial version of the DLL you must call the *password* function in the DLL to validate your license otherwise the *encfile* function will not encrypt your SWF’s. To access this function you must first declare a function pointer to it.

```
typedef void (__stdcall *passwordfuncPtr) (const char*password);
```

After you have called LoadLibrary to load the DLL, you must then get a pointer to the function in the DLL so you can call it later.

```
passwordfuncPtr password;

password =(passwordfuncPtr)GetProcAddress(hinstDLL,"password");
```

Now you can call the *password* function with your password. E.g.

```
char *passwordchar = "mypassword";

password(passwordchar);
```

Once you have called this function you can use the *encfile* function as shown before and it will produce encrypted SWFs if a correct password has been entered.

C# Usage

To use the SWF Encrypt DLL you must declare the two external functions in your C# application. (note: this assumes the DLL will be in the same directory as the executable):

```
[DllImport("SWFEncryptG.dll",CallingConvention=CallingConvention.StdCall)]

static extern void encfile(string srcswf,string destswf);


[DllImport("SWFEncryptG.dll",CallingConvention=CallingConvention.StdCall)]

static extern int fileenced(string srcswf);
```

Now you can use the functions in your C# application. In the following code we are going to encrypt the file “c:\test\test.swf” to “c:\test\test2.swf”

```
encfile("c:\\test\\test.swf","c:\\test\\test2.swf");
```

We can now use the *fileenced* function to check if the “c:\test\test2.swf” file is actually encrypted.

```
int enctest = fileenced("c:\\test\\test2.swf");
```

To check if the SWF File actually is encrypted, we must compare the result to 1. You can declare a constant that holds this value.

```
const int FILEENC_TRUE = 1;

if (enctest==FILEENC_TRUE)

{
      Console.WriteLine("SWF File is encrypted");

}

else

{

      Console.WriteLine("SWF file isn't encrypted");

}
```

Advanced Usage

In order to use the additional parameters you must declare a different *encfile* external function in your C# application. (Note: this assumes the DLL will be in the same directory as the executable):

```
[DllImport("SWFEncryptG.dll",CallingConvention=CallingConvention.Std
Call)]

static extern void encfile(string srcswf,string destswf,int
minlevel,int inclevel);
```

Now you can use the function in your C# application. In the following code we are going to encrypt the file “c:\test\test.swf” to “c:\test\test2.swf” using 20 bytes minimum length unencrypted bytes and encryption strength of 3 (acceptable values for encryption strength are between 1 to 7).

```
encfile("c:\\test\\test.swf","c:\\test\\test2.swf" ,20,3);
```

Full Version Serial Validation

If you are using the full commercial version of the DLL you must call the *password* function in the DLL to validate your license otherwise the *encfile* function will not encrypt your SWFs. To access this function you must first declare an additional external function.

```
[DllImport("SWFEncryptG.dll",CallingConvention=CallingConvention.Std
Call)]
```

```
static extern void password(string password);
```

Now you have the external function declared and set up, you must call the *password* function before using the other functions.

```
password("mypassword");
```

Delphi Usage

To use the SWF Encrypt DLL in Delphi, you must declare a handle and the two functions. In your Var section, we create two strings:

```
Var

   Hlib: Thandle;

   encfile: procedure (srcswf: PChar;destswf: PChar); stdcall;

   fileenced : function (srcswf: PChar):boolean; stdcall;

   filein, fileout : string;
```

Now in the function, we must try to load the DLL using the following code. (Note: this assumes the DLL will be in the same directory as the executable):

```
Hlib := LoadLibrary('SWFEncryptG.dll');

   if Hlib >= 32 then { success }

   begin

      // dll has been loaded so we can start using it now

   end;
```

If the DLL has loaded, it must then find the functions in the DLL so they can be used:

```
encfile := GetProcAddress(Hlib, 'encfile');

fileenced := GetProcAddress(Hlib, 'fileenced');
```

We can now use *encfile* function to encrypt a SWF file. In the following code we are going to encrypt “C:\test\test_in.swf” to “c:\test\test_out.swf”. We need to cast the strings to pointers to character as required by the DLL by using the PChar cast:

```
filein := 'C:\test\test.swf';

fileout := 'C:\test\test2.swf';

encfile(Pchar(filein),Pchar(fileout));
```

If we want to check if the destination SWF file has been encrypted, we can use the *fileenced* function. This function returns true or false and requires a pointer to char as a parameter:

```
if (fileenced(Pchar(fileout))) then begin

      MessageDlg('File is encoded', mtError, [mbOk], 0);

End;
```

Once we have finished with the DLL, we need to tell Windows that we don't need it anymore:

```
FreeLibrary(Hlib);
```

Advanced Usage

In order to use the additional parameters you must declare a different *encfile* function:

```
   encfile: procedure
(src:pchar;dst:pchar;minlen:integer=20;inc_level:integer=3);stdcall;
```

After calling LoadLibrary to load the DLL you must get the function so it can be used:

```
encfile := GetProcAddress(Hlib, 'encfile');
```

We can now use *encfile* function to encrypt a SWF file. In the following code we are going to encrypt “c:\test\test_in.swf” to “C:\test\test_out.swf” using 20 as the minimum bytes unencrypted and 4 as the encryption strength (Strength can range from 1 to 7) . We need to cast the strings to pointers to character as required by the DLL by using the PChar cast:

```
filein := 'C:\test\test.swf';

fileout := 'C:\test\test2.swf';

encfile(Pchar(filein),Pchar(fileout),20,4);
```

Full version Password Validation

If you are using the full version of the DLL you must call the *password* function in the DLL to validate your license otherwise the *encfile* function will not encrypt your SWFs. To access this function you must first declare an additional external function in your var section.

```
password : procedure (pass: PChar); stdcall;

pass : string;
```

After calling LoadLibrary to load the DLL you must get the function from the DLL so it can be used.

```
password := GetProcAddress(Hlib, 'password');
```

You can now use the *password* function to validate your license key. Once you have done this you can then use the other functions including *encfile*.

```
pass := 'mypassword';

password(Pchar(pass));
```