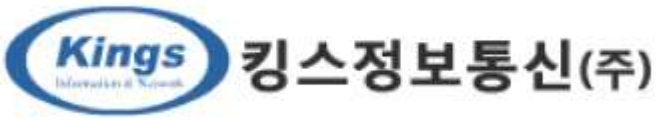
KESS_Crypto 연동 가이드



Kings Information & Network Co., Ltd.

Ver. 1.2

목차

1 개요

KESS_Crypto.dll 연동 문서입니다.

1.1. 개발 환경

- Visual Studio 2008 SP1

1.2. 동작 환경

- Windows XP SP2, Windows Vista, Windows 7, Windows 8, Windows 10

1.3. 구성 요소

- Application(API/MFC) 연동은 KESS_Crypto.dll/KESS_Crypto_x64.dll 모듈을 사용합니다.
- COM 연동은 KESS_CryptoCOM.dll 모듈을 사용합니다.
  - ■ KESS_CryptoCOM.tlb은 COM Dll의 import 파일로 필요 시 추가하여 사용합니다.
- Server(JNI) 연동은 KESS_Crypto.dll/KESS_Crypto_x64.dll 모듈을 사용합니다.
- 함수 정보는 KESS_CryptoTP.h 헤더파일에서 확인할 수 있습니다.
- kcu86.dll/kcu64.dll 모듈은 KESS_Crypto.dll/KESS_Crypto_x64.dll/KESS_CryptoCOM.dll 모듈과 같은 경로에 위치해야 합니다.

파일 이름	설명
KESS_Crypto.dll	암복호화 연동 DLL(x86)
KESS_Crypto_x64.dll	암복호화 연동 DLL(x64)
KESS_CryptoTP.h	암복호화 헤더
KESS_CryptoCOM.dll	암복호화 연동 COM DLL(x86)
KESS_CryptoCOM.tlb	암복호화 연동 COM Import file
kcu86.dll	파일 암복호화 모듈(x86)
kcu64.dll	파일 암복호화 모듈(x64)

1.4. 특이사항

- 암호화 함수(API/MFC/COM/JNI)에서 파라미터로 전달되는 경로에 대상 문서가 읽기전용의 속성을 가진 경우, 읽기전용 속성이 제거되고 문서가 암호화됩니다.

2 인터페이스(API/MFC)

2.1 초기화 KCT_Init

- 라이선스 인증 및 암호화 모듈 로드를 수행하며, 초기화 함수는 반드시 성공해야 합니다.
- 실패할 경우 다른 함수의 사용이 불가능합니다.
- 프로레스 당 최초 1회 호출 후 중복으로 호출되지 않도록 해야 합니다.

```
INT WINAPI KCT_Init( CONST IN PWCHAR pFilePath, CONST IN PWCHAR pLicense )
```

인자 값

- pFilePath : KESS_Crypto.dll 경로(파일명 미포함)
  - ex> C:₩Windows₩System32 혹은 C:₩Windows₩Syswow64
- pLicense : 라이선스 키 값(4~32자로 구성된 문자열)
  - ex> kdnavien

반환 값

- 성공 : RESULT_SUCCESS 0x00000000L
  실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

2.2 암호화 KCT_Encrypt

- 파일 암호화를 수행합니다.

```
INT WINAPI KCT_Encrypt( CONST IN PWCHAR pFilePath, INT iOption, CONST IN
PWCHAR pDstFilePath, INT iDstFilePathLength, pfnCryptoCallback IN
pCallback );
```

인자 값

- pFilePath : 대상 파일 절대 경로
- iOption :
  - 0: 대상 파일을 암호화할 경우
    2 :암호화된 파일을 지정한 경로에 생성할 경우
- pDstFilePath : 암호화된 파일 절대 경로, 사용하지 않는다면 NULL
- iDstFilePathLength : 파일 절대 경로의 길이, 사용하지 않는다면 0
- pCallback : 콜백 함수 포인터
  사용하지 않는다면 NULL, 사용할 경우 콜백 함수 포인터를 전달
  typedef void (CALLBACK *pfnCryptoCallback)(INT iProgress);

반환 값

- 성공 : RESULT_SUCCESS 0x00000000L
  실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

2.3 복호화 KCT_Decrypt

- 파일 복호화를 수행합니다.

```
INT WINAPI KCT_Decrypt( CONST IN PWCHAR pFilePath, INT iOption, CONST IN
PWCHAR pDstFilePath, INT iDstFilePathLength, pfnCryptoCallback IN
pCallback )
```

인자 값

- pFilePath : 대상 파일 절대 경로
- iOption :
  - 0 : 대상 파일을 복호화할 경우
    1 : 복호화된 파일을 대상 파일와 동일한 경로에 생성할 경우
    ex> " k_new_[기존파일 명].kenc" 확장자로 생성
    2 : 복호화된 파일을 지정한 경로에 생성할 경우
- pDstFilePath : 복호화된 파일 절대 경로, 사용하지 않는다면 NULL
- iDstFilePathLength : 복호화된 파일 절대 경로의 길이, 사용하지 않는다면 0
- pCallback : 콜백 함수 포인터
  사용하지 않는다면 NULL, 사용할 경우 콜백 함수 포인터를 전달
  typedef void (CALLBACK *pfnCryptoCallback)(INT iProgress);

반환 값

- 성공 : RESULT_SUCCESS 0x00000000L
  실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

2.4 암호화 여부 확인 KCT_IsEncrypt

- 파일의 암호화 여부를 확인합니다.

```
INT WINAPI KCT_IsEncrypt( CONST IN PWCHAR pFilePath )
```

인자 값

- pFilePath : 대상 파일 절대 경로

반환 값

- 암호화된 파일 : RESULT_SUCCESS 0x00000000L
  암호화되지 않은 파일 : FAIL_ISENC_NOT 0xC0000300L
  실패 : 위 2가지 값이 아닌 값일 경우(KESS_CryptoTP.h 참조)

2.5 읽기전용 처리 설정 KCT_SetReadOnly

- 파일 속성에 읽기전용이 있을 때 암복호화에 대한 처리를 구분합니다.

```
BOOL WINAPI KCT_SetReadOnly( BOOL bRemoveReadOnlyProperty )
```

인자 값

- bRemoveReadOnlyProperty :
  TRUE일 경우 파일 속성에 읽기전용 있으면 해제하고 암복호화가 처리됩니다.
  FALSE일 경우 파일 속성에 읽기전용이 있으면 암복호화 시 에러가 발생합니다.

반환 값

- 함수가 정상적으로 동작한다면 TRUE, 비정상이라면 FALSE를 반환합니다.

2.6 Visual Studio 2008, C++ 예제

```
#include “KESS_CryptoTP.h”

// KESS_Crypto.dll or KESS_Crypto_x64.dll
HMODULE hDLL = LoadLibrary( /*DLL Path*/ );

if( hDLL )
{
KCT_Init pInit = (KCT_Init)GetProcAddress( hDLL, "KCT_Init" );

if( pInit )
{
	DWORD dwResult = pInit( _T( “C:\Windows\System32” ),
_T( “F3C...” ) );

	if( K_CI_SUCCESS( dwResult ) )
	{
		dwResult = pEncrypt ( _T( “C:\Plain.docx” ), 0, NULL );

		// Check Return Value
		...

		pDecrypt ( _T( “C:\Enc.docx” ), NULL, 0, 0, NULL );

		// Check Return Value
		...

		dwResult = pIsEncrypt ( _T( “C:\Plain.docx” ) );

		// Check Return Value
		if( dwResult == RESULT_SUCCESS )
		{
			// File was encrypted
		}
		else if( dwResult == FAIL_ISENC_NOT )
		{
			// File was not encrypted
		}
		else
		{
			// Fail
		}
	}
	else
	{
		// Fail
	}

}
}
```

2.7 Visual Studio 2008, C# 예제

프로토 타입 선언

```
public delegate void cb(int x);

[DllImport("KESS_Crypto.dll", CharSet = CharSet.Unicode)]
public static extern int KCT_Init(string pFilepath, string pLicense);
//암호화
[DllImport("KESS_Crypto.dll", CharSet = CharSet.Unicode)]
public static extern int KCT_Encrypt(string pFilepath, int Option, string
pDstFilePath, int iDstFilePathLength, cb pCallback);
//복호화
[DllImport("KESS_Crypto.dll", CharSet = CharSet.Unicode)]
public static extern int KCT_Decrypt(string pFilepath, int Option, string
pDstFilePath, int iDstFilePathLength, cb pCallback);

//암복호화 확인
[DllImport("KESS_Crypto.dll", CharSet = CharSet.Unicode)]
public static extern int KCT_IsEncrypt(string pFilepath);
```

초기화

```
string output;

int result = KCT_Init(“kcu 전체 경로”, “연동 라이선스 값”);
if (result == 0)
{
	MessageBox.Show("초기화 성공");
}
else
{
	output = "초기화 오류 : " + result.ToString("X");
	MessageBox.Show(output);
}
```

암호화

```
int result = KCT_Encrypt(“파일 전체 경로”, 0, null, 0, null);
if (result == 0)
{
	MessageBox.Show("암호화 성공");
}
else
{
	output = "암호화 오류 : " + result.ToString("X");
	MessageBox.Show(output);
}
```

복호화

```
int result = KCT_Decrypt(“파일 전체 경로”, 0, null, 0, null);
if (result == 0)
{
	MessageBox.Show("복호화 성공");
}
else
{
	output = "복호화 오류 : " + result.ToString("X");
	MessageBox.Show(output);
```

```
}
```

암호화 여부 확인

```
int result = KCT_IsEncrypt(“파일 전체 경로”);
if (result == 0)
{
	MessageBox.Show("암호화 파일");
}
else
{
	MessageBox.Show("암호화 되지 않은 파일");
}
```

3 인터페이스(COM)

3.1 초기화 Init

- 라이선스 인증 및 암호화 모듈 로드를 수행하며, 초기화 함수는 반드시 성공해야 합니다.
- 실패할 경우 다른 함수의 사용이 불가능합니다.
- 프로레스 당 최초 1회 호출 후 중복으로 호출되지 않도록 해야 합니다.

```
virtual /* [helpstring] */ HRESULT STDMETHODCALLTYPE Init(
          /* [out] */ LONG *pResult,
          /* [in] */ BSTR strInitPath,
          /* [in] */ BSTR strLicense) = 0;
```

인자 값

- pResult : 함수 결과 값
- pFilePath : KESS_Crypto.dll 경로(파일명 미포함)
  - ex> C:₩Windows₩System32 혹은 C:₩Windows₩Syswow64
- pLicense : 라이선스 키 값(4~32자로 구성된 문자열)

  ex> kdnavien

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.

  함수 결과 값은 pResult로 반환한다.

  성공 : RESULT_SUCCESS 0x00000000L

  실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

3.2 암호화 Encrypt

- 파일 암호화를 수행합니다.

```
virtual /* [helpstring] */ HRESULT STDMETHODCALLTYPE Encrypt(
          /* [out] */ LONG *pResult,
          /* [in] */ BSTR strFilePath,
          /* [in] */ LONG lOption,
/* [in] */ BSTR pDstFilePath,
/* [in] */ LONG iDstFilePathLength) = 0;
```

인자 값

- pResult : 함수 결과 값
- pFilePath : 대상 파일 절대 경로
- iOption :
  - 0: 대상 파일을 암호화할 경우
    2 :암호화된 파일을 지정한 경로에 생성할 경우
- pDstFilePath : 암호화된 파일 절대 경로, 사용하지 않는다면 (BSTR)0
- iDstFilePathLength : 파일 절대 경로의 길이, 사용하지 않는다면 0

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.
  함수 결과 값은 pResult로 반환한다.
  - 성공 : RESULT_SUCCESS 0x00000000L
    실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

3.3 복호화 Decrypt

- 파일 복호화를 수행합니다.

```
virtual /* [helpstring] */ HRESULT STDMETHODCALLTYPE Decrypt(
          /* [out] */ LONG *pResult,
          /* [in] */ BSTR strFilePath,
          /* [in] */ LONG lOption
/* [in] */ BSTR pDstFilePath,
/* [in] */ LONG iDstFilePathLength) = 0;
```

인자 값

- pResult : 함수 결과 값
- pFilePath : 대상 파일 절대 경로
- iOption :
  - 0 : 대상 파일을 복호화할 경우
    1 : 복호화된 파일을 대상 파일와 동일한 경로에 생성할 경우
    ex> " k_new_[기존파일 명].kenc" 확장자로 생성
    2 : 복호화된 파일을 지정한 경로에 생성할 경우
- pDstFilePath : 암호화된 파일 절대 경로, 사용하지 않는다면 (BSTR)0
- iDstFilePathLength : 파일 절대 경로의 길이, 사용하지 않는다면 0

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.
  함수 결과 값은 pResult로 반환한다.
  - 성공 : RESULT_SUCCESS 0x00000000L
    실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

3.4 암호화 여부 확인 IsEncrypt

- 파일의 암호화 여부를 확인합니다.

```
virtual /* [helpstring] */ HRESULT STDMETHODCALLTYPE IsEncrypt(
          /* [out] */ LONG *pResult,
          /* [in] */ BSTR strFilePath) = 0;
```

인자 값

- pResult : 함수 결과 값
- pFilePath : 대상 파일 절대 경로

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.
  함수 결과 값은 pResult로 반환한다.
  - ■ 암호화된 파일 : RESULT_SUCCESS 0x00000000L
    암호화되지 않은 파일 : FAIL_ISENC_NOT 0xC0000300L
    실패 : 위 2가지 값이 아닌 값일 경우(KESS_CryptoTP.h 참조)

3.5 Visual Studio 2008, C++ 예제

```
#import "KESS_CryptoCOM.tlb" no_namespace

HRESULT             hResult             = 0;
CLSID        ClsID        = {0, };
ICrypto*     pICrypto     = NULL;

hResult = CLSIDFromProgID( OLESTR("KESS_CryptoCOM.Crypto"), &ClsID );

if( SUCCEEDED( hResult ) )
{
      hResult = CoCreateInstance( ClsID, NULL, CLSCTX_INPROC_SERVER,
__uuidof(ICrypto), (LPVOID*)&pICrypto );

      if( SUCCEEDED( hResult ) )
      {
            LONG   lResult      = 0;
            _bstr_t strInitPath( _T( "C:\Windows\System32" ) );
            _bstr_t strLicense( _T( "F3C..." ) );

            hResult = pICrypto->Init(&lResult, strInitPath, strLicense );

            if( hResult == S_OK && SUCCEEDED( lResult ) )
            {
                  _bstr_t strFilePath( _T( "C:\Plain.docx" ) );

                  hResult=pICrypto->Encrypt(&lResult,strFilePath,
0,(BSTR)0, 0 );

                  // Check Return Value
                  ...

                  hResult=pICrypto->Decrypt(&lResult,strFilePath,
0,(BSTR)0, 0);

                  // Check Return Value
                  ...

                  hResult=pICrypto->IsEncrypt( &lResult, strFilePath );

                  // Check Return Value

                 if( dwResult == RESULT_SUCCESS )
                {
                  // File was encrypted
                }
                else if( dwResult == FAIL_ISENC_NOT )
                {
                  // File was not encrypted
                }
                else
                {
                  // Fail
                }
            }
      }

      pICrypto->Release();
```

4 인터페이스(JNI)

4.1 초기화 Init

- 라이선스 인증 및 암호화 모듈 로드를 수행하며, 초기화 함수는 반드시 성공해야 합니다.
- 실패할 경우 다른 함수의 사용이 불가능합니다.
- 프로레스 당 최초 1회 호출 후 중복으로 호출되지 않도록 해야 합니다.

```
extern "C" JNIEXPORT jint JNICALL Java_KESS_Crypto_Init( JNIEnv* pEnv,
jobject pObj, jstring pInitPath, jstring pLicense )
```

인자 값

- pInitPath :
  - ex> C:₩Windows₩System32 혹은 C:₩Windows₩Syswow64
- pLicense : 라이선스 키 값(4~32자로 구성된 문자열)

  ex> kdnavien

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.

  성공 : RESULT_SUCCESS 0x00000000L

  실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

4.2 암호화 Encrypt

- 파일 암호화를 수행합니다.

```
extern "C" JNIEXPORT jint JNICALL Java_KESS_Crypto_Encrypt( JNIEnv* pEnv,
jobject pObj, jstring pFilePath, jint iOption, jstring pDstFilePath, jint
iDstFilePathLength )
```

인자 값

- pFilePath : 대상 파일 절대 경로
- iOption :
  - 0: 대상 파일을 암호화할 경우
    2 :암호화된 파일을 지정한 경로에 생성할 경우
- pDstFilePath : 암호화된 파일 절대 경로, 사용하지 않는다면 null
- iDstFilePathLength : 파일 절대 경로의 길이, 사용하지 않는다면 0

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.
  - 성공 : RESULT_SUCCESS 0x00000000L
    실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

4.3 복호화 Decrypt

- 파일 복호화를 수행합니다.

```
extern "C" JNIEXPORT jint JNICALL Java_KESS_Crypto_Decrypt( JNIEnv* pEnv, jobject pObj, jstring pFilePath, jint iOption, jstring pDstFilePath, jint iDstFilePathLength )
```

인자 값

- pFilePath : 대상 파일 절대 경로
- iOption :
  - 0 : 대상 파일을 복호화할 경우
- pDstFilePath : 암호화된 파일 절대 경로, 사용하지 않는다면 null
- iDstFilePathLength : 파일 절대 경로의 길이, 사용하지 않는다면 0

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.
  - 성공 : RESULT_SUCCESS 0x00000000L
    실패 : 성공이 아닌 값일 경우(KESS_CryptoTP.h 참조)

4.4 암호화 여부 확인 IsEncrypt

- 파일의 암호화 여부를 확인합니다.

```
extern "C" JNIEXPORT jint JNICALL Java_KESS_Crypto_IsEncrypt( JNIEnv*
pEnv, jobject pObj, jstring pFilePath )
```

인자 값

- pFilePath : 대상 파일 절대 경로

반환 값

- 함수 인자가 정상이라면 S_OK, 아니면 S_FALSE를 반환한다.
  - 암호화된 파일 : RESULT_SUCCESS 0x00000000L
    암호화되지 않은 파일 : FAIL_ISENC_NOT 0xC0000300L
    실패 : 위 2가지 값이 아닌 값일 경우(KESS_CryptoTP.h 참조)

4.5 읽기전용 처리 설정 SetReadOnly

- 파일 속성에 읽기전용이 있을 때 암복호화에 대한 처리를 구분합니다.

```
extern "C" JNIEXPORT jboolean JNICALL
Java_KESS_Crypto_SetReadOnly( JNIEnv* pEnv, jobject pObj, jboolean
bRemoveReadOnlyProperty )
```

인자 값

- bRemoveReadOnlyProperty :
  TRUE일 경우 파일 속성에 읽기전용 있으면 해제하고 암복호화가 처리됩니다.
  FALSE일 경우 파일 속성에 읽기전용이 있으면 암복호화 시 에러가 발생합니다.

반환 값

- 함수가 정상적으로 동작한다면 TRUE, 비정상이라면 FALSE를 반환합니다.

4.6 예제

```
package KESS;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;

import javax.servlet.http.HttpServletRequest;


public class Crypto {

	/**
	 * 초기화 Init
	 * -   라이선스 인증 및 암호화 모듈 로드를 수행하며, KCT_Init() 함수는 반드시 성공해야 합니다.
	 * -   실패할 경우 다른 함수의 사용이 불가능합니다.
	 * @param pFilePath KESS_Crypto.dll 경로
	 * @param pLicense 라이선스 키 값(4~32자로 구성된 문자열)
	 * @return
	 */
	public native int Init ( String pFilePath, String pLicense);

	/**
	 * 암호화 Encrypt
	 * - 파일 암호화를 수행합니다.
	 * @param pFilePath 대상 파일 절대 경로
	 * @param iOption   0 : 대상 파일을 암호화할 경우
	                                    1 : 암호화된 파일을 대상 파일와 동일한 경로에 생성할 경우 "[기존파일 명].kenc" 확장자로 생성
	                                    2 : 암호화된 파일을 지정한 경로에 생성할 경우
	 * @param pDstFilePath 암호화된 파일 절대 경로( ex > c:\kings\enctemp\[기존파일 명].kenc ), 사용하지 않는다면 NULL
	 * @param iDstFilePathLength 파일 절대 경로의 길이 ( ex> pDstFilePath.length() ), 사용하지 않는다면 0
	 * @return
	 */
	public native int Encrypt( String pFilePath, int iOption, String pDstFilePath, int iDstFilePathLength);

	/**
	 * 복호화 Decrypt
	 * - 파일 복호화를 수행합니다.
	 * @param pFilePath 대상 파일 절대 경로
	 * @param iOption   0 : 대상 파일을 복호화할 경우
	                                    1 : 복호화된 파일을 대상 파일와 동일한 경로에 생성할 경우" k_new_[기존파일 명].kenc" 확장자로 생성
	                                    2 : 복호화된 파일을 지정한 경로에 생성할 경우
	 * @param pDstFilePath 복호화된 파일 절대 경로( ex > c:\kings\enctemp\k_new_[기존파일 명].kenc ), 사용하지 않는다면 NULL
	 * @param iDstFilePathLength 복호화된 파일 절대 경로의 길이 ( ex> pDstFilePath.length() ), 사용하지 않는다면 0
	 * @return
	 */
	public native int Decrypt( String pFilePath, int iOption, String pDstFilePath, int iDstFilePathLength );
```

```
	/**
	 * 암호화 여부 확인 KCT_IsEncrypt
	 * -   파일의 암호화 여부를 확인합니다.
	 * @param pFilePath 대상 파일 절대 경로
	 * @return 암호화된 파일 : RESULT_SUCCESS 0x00000000L, 암호화되지 않은 파
일 : FAIL_ISENC_NOT 0xC0000300L

	 */
	public native int IsEncrypt( String pFilePath );


	/*
	 * 예제 소스
	 */
	public Crypto()
	{

	}

	/**
	 * 클래스 초기화
	 * @param request
	 * @param license 라이선스 키 값(4~32자로 구성된 문자열)
	 * @return 결과값 (0:성공)
	 */
	public String KessInit(HttpServletRequest request, String license)
	{
		String classPath =
request.getSession().getServletContext().getRealPath("/");
		return KessInit(classPath, license);
	}

	/**
	 * 클래스 초기화
	 * @param classPath classPath =
request.getSession().getServletContext().getRealPath("/")
	 * @param license 라이선스 키 값(4~32자로 구성된 문자열)
	 * @return 결과값 (0:성공)
	 */
	public String KessInit(String classPath, String license)
	{
		//DLL 경로 설정
		String DllFilePath = classPath+"WEB-INF\\classes\\KESS";

		//
		String bits = System.getProperty("sun.arch.data.model");

		if(bits.equals("64"))
System.load(DllFilePath+"\\KESS_Crypto_x64.dll");
		else if(bits.equals("32"))
System.load(DllFilePath+"\\KESS_Crypto.dll");

		//DLL 초기화
		int iRet = Init(DllFilePath, license);

		Integer a = new Integer(iRet);
		String isencResult =
String.format("%x",fromArray(toArray(iRet)));
		System.out.println( "init Ret : " + iRet + ", convert : " +
isencResult);

		return isencResult;
	}

	/**
```

```
	 * 파일 암호화를 수행합니다.
	 * @param pFilePath 대상 파일 절대 경로 (ex : D:\\temp\\test.txt)
	 * @return 결과값 (0:성공)
	 */
	public String KessEncrypt(String pFilePath)
	{
		int iOption = 0; //0:  대상 파일을 암호화할 경우
		String pDstFilePath = null; //암호화된 파일 절대 경로, 사용하지 않는다면 NULL

		return KessEncrypt(pFilePath, iOption, pDstFilePath);
	}

	/**
	 * 파일 암호화를 수행합니다.
	 * @param pFilePath 대상 파일 절대 경로 (ex : D:\\temp\\test.txt)
	 * @param iOption //0:  대상 파일을 암호화할 경우
	 * @param pDstFilePath  //암호화된 파일 절대 경로, 사용하지 않는다면 NULL
	 * @return 결과값 (0:성공)
	 */
	public String KessEncrypt(String pFilePath, int iOption, String pDstFilePath)
	{
		System.out.println("Start Encrypt");

		//파일 암호화
		int iDstFilePathLength = (pDstFilePath == null)?0:pDstFilePath.length(); //파일 절대 경로의 길이, 사용하지 않는다면 0

		System.out.println("doenc param - pFilePath : "+ pFilePath + ",iOption :" + iOption+ ",pDstFilePath :" + pDstFilePath+ ",iDstFilePathLength :" + iDstFilePathLength);
		int iRet = Encrypt(pFilePath, iOption, pDstFilePath, iDstFilePathLength);

		String isencResult = String.format("%x",fromArray(toArray(iRet)));
		System.out.println( "doenc Ret : " + iRet + ", convert : " + isencResult );

		if(isencResult.equals("0")){
			System.out.println("[암호화 성공]");
		}else{
			System.out.println("[암호화 실패]");
		}
		System.out.println("End Encrypt");

		return isencResult;
	}

	/**
	 * 파일 복호화를 수행합니다.
	 * @param pFilePath 대상 파일 절대 경로 (ex : D:\\temp\\test.kenc)
	 * @return 결과값 (0:성공)
	 */
	public String KessDecrypt(String pFilePath)
	{
		//파일 복호화
		int iOption = 0; //0:  대상 파일을 복호화할 경우 , 2 :복호화된 파일을 지정한 경로에 생성할 경우
		String pDstFilePath = null; //복호화된 파일 절대 경로, 사용하지 않는다면 NULL
```

```
		return KessDecrypt(pFilePath, iOption, pDstFilePath);
	}

	/**
	 * 파일 복호화를 수행합니다.
	 * @param pFilePath 대상 파일 절대 경로 (ex : D:\\temp\\test.kenc)
	 * @param iOption //0:  대상 파일을 복호화할 경우 , 2 :복호화된 파일을 지정한 경로에 생성할 경우
	 * @param pDstFilePath  //복호화된 파일 절대 경로, 사용하지 않는다면 NULL
	 * @return 결과값 (0:성공)
	 */
	public String KessDecrypt(String pFilePath, int iOption, String pDstFilePath)
	{
		System.out.println("Start Decrypt");

		//파일 복호화
		int iDstFilePathLength = (pDstFilePath == null)?0:pDstFilePath.length(); //복호화된 파일 절대 경로의 길이, 사용하지 않는다면 0

		System.out.println("dodec param - pFilePath : "+ pFilePath + ",iOption :" + iOption+ ",pDstFilePath :" + pDstFilePath+ ",iDstFilePathLength :" + iDstFilePathLength);
		int iRet = Decrypt(pFilePath, iOption,pDstFilePath, iDstFilePathLength);
		String isencResult = String.format("%x",fromArray(toArray(iRet)));
		System.out.println("dodec Ret : "+ iRet + ",convert :" + isencResult);


		if(isencResult.equals("0")){
			System.out.println("[복호화 성공]");
		}else{
			System.out.println("[복호화 실패]");
		}
		System.out.println("End Decrypt");

		return isencResult;
	}

	/**
	 * 파일의 암호화 여부를 확인합니다.
	 * @param pFilePath 대상 파일 절대 경로 (ex : D:\\temp\\test.kenc)
	 * @return 결과값 (0:암호화된파일, C0000300:암호화되지 않은 파일)
	 */
	public String KessIsEncrypt(String pFilePath)
	{
		System.out.println("Start IsEncrypt");

		//암호화 여부 체크
		int iRet = IsEncrypt(pFilePath);
		String isencResult = String.format("%x",fromArray(toArray(iRet)));
		System.out.println( "isenc Ret : " + iRet + ", convert : " + isencResult );

		if(isencResult.equals("0")){
			System.out.println("[암호화 여부] : 암호화된 파일");
		}else if(isencResult.equals("C0000300")){
			System.out.println("[암호화 여부] : 암호화 된 파일이 아님");
		}else{
```

```
                System.out.println("[암호화 여부 체크 에러]");
            }
            System.out.println("End IsEncrypt");

            return isencResult;
        }


        public static  int fromArray(byte[] payload){
            ByteBuffer buffer = ByteBuffer.wrap(payload);
            buffer.order(ByteOrder.BIG_ENDIAN);
            return buffer.getInt();
        }

        public static byte[] toArray(int value){
            ByteBuffer buffer = ByteBuffer.allocate(4);
            buffer.order(ByteOrder.BIG_ENDIAN);
            buffer.putInt(value);
            buffer.flip();
            return buffer.array();
        }

        public void main(){

            //암복호화 클래스 선언
            Crypto t = new Crypto();

            //초기화
            t.KessInit("C:\\Kings\\kess\\dll\\", "KINGS");

            String testFilePath = "c:\\kings\\test.doc";

            //암호화 실행
            String isencResult = t.KessEncrypt(testFilePath);

            //암호화 성공
            if(isencResult.equals("0"))
            {
                //암호화 여부 체크 - 0:암호화
                isencResult = t.KessIsEncrypt(testFilePath);

                //암호화 파일
                if(isencResult.equals("0"))
                {
                    //복호화 실행
                    isencResult = t.KessDecrypt(testFilePath);
                }
            }

        }

}

/*
//-------------------------------------------------------------------
//                                          Error Code
//-------------------------------------------------------------------
#define RESULT_SUCCESS                                                  0
#define RESULT_FAIL                                                     FFFFFFFF

#define FAIL_NEED_CALL_INIT                                             C0000099

//--   Init               100
#define FAIL_INIT_KC_INIT                                       C0000100
#define FAIL_INIT_SIZE                                                  C0000101
#define FAIL_INIT_PATH                                                  C0000102
#define FAIL_INIT_EXIST                                                 C0000103
```

```
#define FAIL_INIT_LOAD                                          C0000104
#define FAIL_INIT_TEMPDIR                                       C0000105
#define FAIL_INIT_LICENSE                                       C0000106
#define FAIL_INIT_JNI                                           C0000107
#define FAIL_INIT_THIRD_PARTY_LICENSE                   C0000108

//--   IsEnc               300
#define FAIL_ISENC_NOT                                          C0000300
#define FAIL_ISENC_PATH                                         C0000301
#define FAIL_ISENC_SEQSIZE                                      C0000302
#define FAIL_ISENC_FILESIZE                                     C0000303
#define FAIL_ISENC_OPEN                                         C0000304
#define FAIL_ISENC_MEMCMP                                       C0000305
#define FAIL_ISENC_WRITEOPEN                                    C0000306
#define FAIL_ISENC_JNI                                          C0000307

//--   DoEnc               400
#define FAIL_DOENC_PATH                                         C0000400
#define FAIL_DOENC_SIZE                                         C0000401
#define FAIL_DOENC_ISENC                                        C0000402
#define FAIL_DOENC_DOHASH                                       C0000403
#define FAIL_DOENC_CREATE_DIRECTORY                             C0000404
#define FAIL_DOENC_CREATE_FILE                                  C0000405
#define FAIL_DOENC_KC_SK_KEYGEN                                 C0000406
#define FAIL_DOENC_OPEN_FILE                                    C0000407
#define FAIL_DOENC_READ_FILE                                    C0000408
#define FAIL_DOENC_COPY_FILE                                    C0000409
#define FAIL_DOENC_WRITE_FILE                                   C0000410
#define FAIL_DOENC_ALREADY_FILE_ENCRYPTED               C0000411
#define FAIL_DOENC_JNI                                          C0000412

//--   DoDec               500
#define FAIL_DODEC_PATH                                         C0000501
#define FAIL_DODEC_SIZE                                         C0000502
#define FAIL_DODEC_KC_SK_DEC                                    C0000503
#define FAIL_DODEC_CREATE_DIRECTORY                             C0000504
#define FAIL_DODEC_CREATE_FILE                                  C0000505
#define FAIL_DODEC_ISENC                                        C0000506
#define FAIL_DODEC_GETENCINFO                                   C0000507
#define FAIL_DODEC_HASH_VERIFY                                  C0000508
#define FAIL_DODEC_DOHASH                                       C0000509
#define FAIL_DODEC_OPEN_FILE                                    C0000510
#define FAIL_DODEC_READ_FILE                                    C0000511
#define FAIL_DODEC_LICENSE_VERIFY                       C0000512
#define FAIL_DODEC_TEMPPATH                                     C0000513
#define FAIL_DODEC_FILESIZE                                     C0000514
#define FAIL_DODEC_COPY_FILE                                    C0000515
#define FAIL_DODEC_WRONG_OPTION                                 C0000516
#define FAIL_DODEC_WRITE_FILE                                   C0000517
#define FAIL_DODEC_JNI                                          C0000518
*/
```

5 에러코드

5.1 에러코드

5.1.1 공통

- RESULT_SUCCESS : 성공.
- RESULT_FAIL : 실패.
- FAIL_NEED_CALL_INIT : 초기화 함수 호출없이 다른 함수를 사용한 경우.

```
#define RESULT_SUCCESS
       0x00000000L
#define RESULT_FAIL                                                    0xFFFFFFFF
#define FAIL_NEED_CALL_INIT
       0xC0000099L
```

5.1.2 초기화

- FAIL_INIT_KC_INIT : kcu86.dll/kcu64.dll 초기화 호출 반환 값이 실패.
- FAIL_INIT_SIZE : 예약된 값.
- FAIL_INIT_PATH : 올바르지 않은 경로 파라미터.
- FAIL_INIT_EXIST : 이미 Init 함수가 호출됨.
- FAIL_INIT_LOAD : kcu86.dll/kcu64.dll 모듈 로드 실패
- FAIL_INIT_TEMPDIR : 임시경로 관련 에러.
- FAIL_INIT_LICENSE : 올바르지 않은 라이선스 파라미터.
- FAIL_INIT_JNI : 파라미터 변환 실패.
- FAIL_INIT_THIRD_PARTY_LICENSE : 타사 연동 라이선스 체크 실패.

```
#define FAIL_INIT_KC_INIT                                              0xC0000100L
#define FAIL_INIT_SIZE
       0xC0000101L
#define FAIL_INIT_PATH
       0xC0000102L
#define FAIL_INIT_EXIST                                                0xC0000103L
#define FAIL_INIT_LOAD
       0xC0000104L
#define FAIL_INIT_TEMPDIR                                              0xC0000105L
#define FAIL_INIT_LICENSE                                              0xC0000106L
#define FAIL_INIT_JNI
       0xC0000107L
#define FAIL_INIT_THIRD_PARTY_LICENSE                                  0xC0000108L
```

5.1.3 암호화

- FAIL_DOENC_PATH : 올바르지 않은 경로 파라미터.
- FAIL_DOENC_SIZE : 암호화 헤더에서 평문 파일 크기를 찾을 수 없음.
- FAIL_DOENC_ISENC : 대상 파일이 이미 암호화되어 있음.
- FAIL_DOENC_DOHASH: 해시 생성 실패.
- FAIL_DOENC_CREATE_DIRECTORY : 임시파일 경로 생성 실패.
- FAIL_DOENC_CREATE_FILE : 임시파일 생성 실패.
- FAIL_DOENC_KC_SK_KEYGEN : 암호화 키 생성 실패.
- FAIL_DOENC_OPEN_FILE : 평문파일 열기 실패.
- FAIL_DOENC_READ_FILE : 평문파일 읽기 실패.

- FAIL_DOENC_COPY_FILE : 임시파일을 대상 경로로 복사 실패.
- FAIL_DOENC_WRITE_FILE : 임시파일 쓰기 실패.
- FAIL_DOENC_ALREADY_FILE_ENCRYPTED : 대상 파일이 이미 암호화되어 있음.
- FAIL_DOENC_JNI : 파라미터 변환 실패.

```
#define FAIL_DOENC_PATH                                    0xC0000400L
#define FAIL_DOENC_SIZE                                    0xC0000401L
#define FAIL_DOENC_ISENC                                   0xC0000402L
#define FAIL_DOENC_DOHASH                                  0xC0000403L
#define FAIL_DOENC_CREATE_DIRECTORY
       0xC0000404L
#define FAIL_DOENC_CREATE_FILE                             0xC0000405L
#define FAIL_DOENC_KC_SK_KEYGEN                            0xC0000406L
#define FAIL_DOENC_OPEN_FILE
       0xC0000407L
#define FAIL_DOENC_READ_FILE
       0xC0000408L
#define FAIL_DOENC_COPY_FILE
       0xC0000409L
#define FAIL_DOENC_WRITE_FILE
       0xC0000410L
#define FAIL_DOENC_ALREADY_FILE_ENCRYPTED
       0xC0000411L
#define FAIL_DOENC_JNI
       0xC0000412L
```

5.1.4 복호화

- FAIL_DODEC_PATH : 올바르지 않은 경로 파라미터.
- FAIL_DODEC_SIZE : 예약된 값.
- FAIL_DODEC_KC_SK_DEC : 복호화 실패.
- FAIL_DODEC_CREATE_DIRECTORY : 임시파일 경로 생성 실패.
- FAIL_DODEC_CREATE_FILE : 임시파일 생성 실패
- FAIL_DODEC_ISENC : 대상 파일이 이미 복호화되어 있음.
- FAIL_DODEC_GETENCINFO : 암호화 헤더 정보를 찾을 수 없음.
- FAIL_DODEC_HASH_VERIFY : 복호화된 평문파일의 유효성 검증 실패.
- FAIL_DODEC_DOHASH : 해시 생성 실패.
- FAIL_DODEC_OPEN_FILE : 대상파일 열기 실패.
- FAIL_DODEC_READ_FILE : 대상파일 읽기 실패.
- FAIL_DODEC_LICENSE_VERIFY : 라이선스 정보 검증 실패.
- FAIL_DODEC_TEMPPATH : 임시경로 관련 에러.
- FAIL_DODEC_FILESIZE : 대상파일 크기 제한.
- FAIL_DODEC_COPY_FILE : 임시파일을 대상 경로로 복사 실패.
- FAIL_DODEC_WRONG_OPTION : 올바르지 않은 iOption 파라미터.
- FAIL_DODEC_WRITE_FILE : 임시파일 쓰기 실패.
- FAIL_DODEC_JNI : 파라미터 변환 실패.

```
#define FAIL_DODEC_PATH                                    0xC0000501L
#define FAIL_DODEC_SIZE                                    0xC0000502L
#define FAIL_DODEC_KC_SK_DEC
       0xC0000503L
#define FAIL_DODEC_CREATE_DIRECTORY
       0xC0000504L
#define FAIL_DODEC_CREATE_FILE                             0xC0000505L
#define FAIL_DODEC_ISENC                                   0xC0000506L
```

```
#define FAIL_DODEC_GETENCINFO
       0xC0000507L
#define FAIL_DODEC_HASH_VERIFY                                  0xC0000508L
#define FAIL_DODEC_DOHASH                                       0xC0000509L
#define FAIL_DODEC_OPEN_FILE
       0xC0000510L
#define FAIL_DODEC_READ_FILE
       0xC0000511L
#define FAIL_DODEC_LICENSE_VERIFY                               0xC0000512L
#define FAIL_DODEC_TEMPPATH
       0xC0000513L
#define FAIL_DODEC_FILESIZE
       0xC0000514L
#define FAIL_DODEC_COPY_FILE
       0xC0000515L
#define FAIL_DODEC_WRONG_OPTION                                 0xC0000516L
#define FAIL_DODEC_WRITE_FILE
       0xC0000517L
#define FAIL_DODEC_JNI
       0xC0000518L
```

5.1.5 암호화 여부 확인

- FAIL_ISENC_NOT : 대상 파일이 암호화되어 있지 않음.
- FAIL_ISENC_PATH : 올바르지 않은 경로 파라미터.
- FAIL_ISENC_SEQSIZE : 예약된 값.
- FAIL_ISENC_FILESIZE : 대상파일 크기 제한.
- FAIL_ISENC_OPEN : 대상파일 열기 실패.
- FAIL_ISENC_MEMCMP : 대상파일 읽기 실패.
- FAIL_ISENC_WRITEOPEN : 예약된 값.
- FAIL_ISENC_JNI : 파라미터 변환 실패.

```
#define FAIL_ISENC_NOT
       0xC0000300L
#define FAIL_ISENC_PATH                                         0xC0000301L
#define FAIL_ISENC_SEQSIZE                                      0xC0000302L
#define FAIL_ISENC_FILESIZE
       0xC0000303L
#define FAIL_ISENC_OPEN                                         0xC0000304L
#define FAIL_ISENC_MEMCMP                                       0xC0000305L
#define FAIL_ISENC_WRITEOPEN
       0xC0000306L
#define FAIL_ISENC_JNI
       0xC0000307L
```