개발형 OS 문서 저작 도구 DRM Interface

Ver 1.0

㈜ 파 수

# Software License Agreement

양식의 맨 위

양식의 맨 아래

Copyright 2020 Fasoo

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

# Introduction

본 문서는 파수닷컴의 개방형 OS의 문서 저작 도구들이 파수 DRM을 이용하여 보안문서를 지원할 수 있도록 하기 위한 인터페이스(이하 DRM Interface) 사용 방법 및 정의 문서이다.

DRM Interface를 통하여 응용 프로그램에서 암호화된 문서 파일을 사용할 수 있다.

DRM Interface는 기존에 파일 입/출력에 사용되는 함수를 (runtime 라이브러리의 경우 open(), read(), write(), close() 등) 대체하는 함수인 ADKOpenContent(), ADKReadContent(), ADKWriteContent(), ADKCloseContent() 함수와 그 밖에 함수들로 구성되어 있다.

문서 저작도구 개발자는, 일반 문서의 경우 기존의 함수를 사용하고 보안 문서의 경우에만 ADK 함수를 사용하거나, 보안 문서 여부에 상관 없이 모든 문서 파일에 대해 ADK 함수를 사용하는 방식 중 선택을 할 수 있다.

DRM Interface는 공개 SW의 인터페이스로서, 원할한 문서 저작도구의 DRM 지원을 위해 인터페이스가 수정/추가/삭제 될 수 있다.

# Definition

**일반 문서(원본 문서)**

응용프로그램에서 정보를 저장하는데 사용하는 파일 포맷을 의미한다.

예를 들어 아래한글은 확장자가 HWP인 문서를 사용한다. 응용프로그램에서 생성된 일반 문서는 그 응용프로그램이 설치된 PC에서는 어디서나 문서를 열어서 그 내용을 확인할 수 있다.

**보안 문서**

㈜파수 DRM 암호화 기술을 적용하여 일반 문서를 암호화한 문서를 의미한다. 보안 문서는 암호화되어 있어서 DRM Client가 설치되어 있지 않은 PC에서는 응용프로그램이 설치되어 있어도 문서를 열수 없고, DRM Client가 있더라도 적절한 사용자 인증을 거친 사용자만이 보안 문서를 열어서 사용할 수 있다.

**DRM 지원 문서 저작 도구**

보안 문서를 사용할 수 있는 개방형 OS상에서 동작하는 문서 저작 도구를 의미한다. 일반 문서를 지원하는 문서 저작 도구에 DRM Interface를 적용하여 DRM 지원 가능 문서 저작 도구를 만들 수 있다.

**보안 템플릿 문서**

불러들인 보안문서를 다른 이름으로 저장하는 경우 생성될 보안문서의 기준이 되는 보안 문서를 의미한다.

보안 응용프로그램에서 보안문서를 편집한 후 저장하거나, 불러들인 보안 문서를 다른이름으로 저장하는 경우 보안문서로 저장이 되어야 한다. 이 때 저장되는 보안문서는 응용프로그램에서 임의대로 문서의 권한을 지정할 수 없고, 반드시 기존에 존재하는 보안문서중 하나를 지정하여 그 문서와 같은 권한을 갖도록 해야한다.

**라이센스**

보안 문서를 사용하기 위해서는 DRM 서버에서 문서를 사용하기 위한 라이선스를 획득해야 한다. 라이센스는 보안문서를 여는 순간에 DRM 서버와 통신을 통해 실시간으로 얻어진다.

**권한(Rights)/목적(Purpose)**

DRM Interface에서 사용되는 문서의 사용권한이 아래 표에 나타나 있다.

Purpose Code는 DRM 지원 문서 저작 도구에서문서를 사용하기 전에 적절한 권한이 있는지 확인하는데 사용되는 코드이다. ADK\_PURPOSE\_VIEW 는 보안 응용프로그램이 “열람을 하려고 한다”는 것을 의미한다.

|  |  |  |
| --- | --- | --- |
| **권한** | **Purpose Code** | **Description** |
| VIEW | ADK\_PURPOSE\_VIEW | 보안문서의 내용을 볼 수 있다. 보안 응용프로그램은 보안문서에 기본적으로 VIEW 권한이 있어야 사용할 수 있다. |
| SAVE | ADK\_PURPOSE\_SAVE | 보안문서를 일반문서로 저장할 수 있다. 보안 응용 프로그램은, 이 목적에 대한 권한이 있다고 해서 보안문서를 일반문서로 저장해서는 안된다. (암호화 해제) |
| EDIT | ADK\_PURPOSE\_EDIT | 보안문서의 내용을 편집할 수 있다.  통상적으로 EDIT권한이 있는 문서는 SECURE\_SAVE권한도 가지고 있다. |
| EXTRACT | ADK\_PURPOSE\_EXTRACT | 보안문서의 내용을 복사할 수 있다. 보안문서에 EXTRACT 권한이 없는 경우, 문서 내용에 대한 복사,붙여넣기 동작이 이루어지지 않도록 해야한다. |
| PRINT | ADK\_PURPOSE\_PRINT | 보안문서를 인쇄할 수 있다. |
| PRINT\_SCREEN | ADK\_PURPOSE\_PRINT\_SCREEN | 보안문서의 화면 캡처가 가능하다.  보안문서에 PRINT\_SCREEN 권한이 없는 경우, 보안 응용프로그램 캡처방지 기능을 활성화하여야 한다. |
| MACRO | ADK\_PURPOSE\_MACRO | 보안 문서가 매크로를 이용 할 수 있는지 권한 여부를 판단한다. |
| SECURE\_SAVE | ADK\_PURPOSE\_SECURE\_SAVE | 보안문서의 내용이 수정되었을 때 수정된 내용을 다시 저장할 수 있다.  통상적으로 SECURE\_SAVE권한이 있는문서는 EDIT권한도 가지고 있다.  이 권한이 있다고 해서 보안문서를 일반문서로 저장해서는 안된다.  (암호화 파일의 저장 권한) |

# ADK 구성

|  |  |
| --- | --- |
| f\_extadk.h | header file |
| DRM Interface를 적용할 module source compile 경로상에 위치  Document Interface와 System Interface를 지원한다. |
| [libfasooextadk.so](https://github.com/Fasoo-OpenProject/OpenOS-DRM/blob/master/ExtADK/libfasooextadk.so) | DRM Interface 모듈 |
| ㈜파수에서 제공하는 DRM Interface API 모듈 및 문서 저작 도구에서 제공하는 이벤트를 전달할 Interface 모듈 |
| 개방형 OS 문서 저작 도구 DRM Interface.docx | DRM Interface 설명서 |

# ADK 적용

1. 적용 빌드
   1. TMAXOS 3.0
   2. TMAX Office
2. Interface 적용 방식
   1. DRM Interface 랩핑 SO 등록/로드, 상호 초기화 방법
      1. DRM이 동작 할 수 있는 모듈 (Add-in)은 /usr/local/fasoo에 존재
      2. 문서 저작 도구 기동 시 상기 2-A-I 모듈 로드
      3. 모듈 로드 후, Initialize 함수 호출  
         인자로 TMAX Office에서 DRM에게 지원 하는 askCommand Function Point와 DRM이 지원하는 기능 Function List (DRM Document Interface List)를 전달받을 구조체 포인트 전달 (정보의 요청, 특정 옵션 요청 등…)
   2. SO 정보 기록
      1. 문서 저작 도구의 Config 파일은 “/usr/local/fasoo”에 기록 한다.

# Function Description

* 1. 공통 Interface  
     DRM Interface의 동작 가능 여부 판단과 Entry Point 등을 위한 공통 함수   
     이 함수는 SO에서 노출 되어 사용되는 유일한 함수이다.

|  |
| --- |
| Interface No. : INT-COM-1 |
| int DRMInterfaceInitialize (void \*\*DocumentInterface, void \*\*SystemInterface, void \*pReserved1 void \*pReserved2); |
| 인자 :  pDocumentInterface : [out] 아래 B. Document Interface를 받을 주소  PSystemInterface : [out] 아래 C. System Interface를 받을 주소  pReserved1, pReserved2 : 예약된 파라메터  반환 : 성공시 0보다 큰 값, 실패시 0 (설명 참고) |
| 설명 이 함수를 호출하여 DRM Interface의 초기화 함수이다.  이 함수를 통해서 Document Interface와 SystemInterface를 반환 받을 수 있다.  이 함수는 DRM 시스템 내부적으로 사용하는 기능들이 준비 되었는지 검사한다. 이 함수가 0을 리턴하면, 파일의 암/복호화(Open/Read/Write/Close)를 위한 함수들을 사용해서는 안된다.  3,4 번째 파라메터는 예약 인자로 현재 사용 되지 않는다.  값을 입력해도 내부에서는 사용 되지 않는 인자이다.  (치명적인 문제가 생기는 것은 아니지만 모든 함수가 실패한다.) |

* 1. **Document Interface (순수 문서 저작 도구의 UI와 관련된 Interface)**DRM이 제공하는 기능 Function Point 리스트 (Interface)  
     DRM의 비즈니스 로직(문서 저작 툴의 기능/UI)와 관련한 Interface  
       
     Document Inerface 전체 함수

|  |
| --- |
| typedef unsigned long(\*\_LPGETVERSION)();  typedef unsigned int(\*\_LPGETAUTHENTICATE)  (  char\* szFilePath;  );  typedef unsigned long(\*\_LPSETNOTIFYMESSAGE)  (  void \*pContentInfo;  char \*szFilePath;  int iId;  unsinged int uiParam1;  unsigned int uiParam2;  );  typedef unsigned int(\*\_LPSETMENU)  (  void \*pContentInfo;  unsigned int uiMenuID  );;  typedef struct FasooDocumentInterfaceManager  {  \_LPGETVERSION pfngetVersion;  \_LPGETAUTHENTICATE pfngetAuthenticate;  \_LPSETNOTIFYMESSAGE pfnsetNotifyMessage;  \_LPSETMENU pfnsetMenu;  } DIM;  typedef DIM \*LPDIM; |
|  |

각 인터페이스 함수 상세 설명

|  |
| --- |
| Interface No. : INT-DOC-1 |
| unsigned long getVersion(); |
| 인자 : 없음  반환 : 버전 정보 (설명 참조) |
| 설명 DRM모듈에서 사용하는 IDRM\_Open\_Application 인터페이스의 버전을 얻어온다. 이는 향후 인터페이스가 변경이 되어도 버전정보를 통한 하위호환을 위해서 제공된다. 현재 인터페이스의 버전은 1.0.0.0이다. Unsinged long 값으로는 0x01000000이다. (각 자리수마다 2byte를 사용함) 각 문서 저작 도구는 DRM모듈을 로드할 때 GetVersion()을 확인하며 버전에 맞도록 동작한다. |

|  |
| --- |
| Interface No. : INT-DOC-2 |
| unsigned int getAuthenticate(char \*szFilePath) |
| 인자 szFilePath : [in] 마지막 ‘\0’로 끝나는 문서 파일의 FullPath  반환 : 인자로 전달받은 파일의 권한 정보 모음 - 설명 참고 |
| 설명 **이 함수를 호출 할 때, szFilePath에 해당하는 파일은 최소한 Open/Read/Close가 가능 해야 한다.**  szFilePath의 문서의 권한 정보를 확인 하고자 할 때, 호출 한다.  일반적으로 문서의 열람/저장/인쇄 전에 호출 된다.  getAuthenticate 함수가 실행되면 (v1.0.0.0 기준) 아래와 같은 권한 정보를 반환하며, 각 권한은 비트별로 맵핑되어 있다. 전제 권한을 모두 반환 하지는 않는다. **(UI 상으로 필요한 권한에 한해서만 값을 전달 한다.)**   |  |  | | --- | --- | | VIEW (0번 비트) | 문서의 읽기 권한, 이 권한이 없으면 Open 자체가 불가능 하다. | | EDIT (1번 비트) | 문서의 수정/저장 권한, 이 권한이 없으면 편집/저장이 불가능 하다.  이 권한이 있으면 클립보드 붙여 넣기, 개체 삽입 등 문서가 편집 되는 행위는 허용 하지만,  복사하기, 내보내기는 다른 권한(Extract)을 확인 해야 한다. | | PRINT (2번 비트) | 문서의 인쇄 권한, 이 권한이 없으면 이 문서는 인쇄/인쇄 미리보기가 되지 않는다. | | EXTRACT (3번 비트) | 문서의 내용을 다른 프로그램으로 추출하는 기능, 즉 복사, 잘라내기, 내보내기 관련된 권한이다. 주로 EDIT 권한과 조합되어 사용되는데 EDIT 권한이 없을 경우에는 복사기능만 활성화 된다. (만약 Edit 권한이 없고, Extract 권한이 있다면, 잘라내기/붙이기 기능은 비활성화 되고, 복사하기 기능은 활성화 된다. ) | | SAVE  (4번 비트) | 보안 문서를 일반 문서로 저장 할 수 있는 권한으로, 암호화 해제 권한을 의미한다. | | PRINT\_SCREEN  (5번 비트) | 보안문서의 화면 캡처를 허용 할지 여부의 권한이다.  이 권한이 없으면 캡쳐 프로그램이 영역을 캡쳐하면, 캡쳐가 되지 않게 차단한다. | | MACRO  (6번 비트) | 보안 문서가 매크로 기능을 수행해도 되는지의 권한 | | SECURE\_SAVE  (7번 비트) | 보안 문서의 저장 권한 |   위 권한들은 비트의 조합으로 여러 권한을 가질 수 있다.  반환값   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 8~32 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | Reserved | SECURE\_SAVE | MACRO | PRINT\_SCREEN | Save | EXTRACT | PRINT | EDIT | VIEW | |

|  |
| --- |
| Interface No. : INT-DOC-3 |
| unsigned long setNotifyMessage(); |
| 인자 :  pContentInfo : [in] 이 메시지가 발생한 문서의 ContentInfo szFilePath : [in] 마지막 ‘\0’로 끝나는 문서 파일의 FullPath  iId : [in] 오피스(문서 저작 도구)에서 발생하는 이벤트 ID  uiParam1 : [in] 각 이벤트 별 사용 되는 인자 (설명 참고)  uiParam2 : [in] 각 이벤트 별 사용 되는 인자 (설명 참고)  반환 : 각 이벤트 별 사용 되는 반환 값 (설명 참고) |
| 설명 오피스(문서 저작 도구)에서 발생된 이벤트를 DRM 모듈에게 전달 하기 위해, 오피스로부터 호출 된다.  iId (Notify ID)   |  |  |  |  | | --- | --- | --- | --- | | ID | 값 | 설명 | DRM 인터페이스 버전 | | DocumentBeforeOpen | 0x0101 | 문서가 열리기 직전 | 0x01000000 | | DocumentAfterOpen | 0x0102 | 문서가 열린 후 | 0x01000000 | | DocumentBeforeSave | 0x0103 | 문서가 저장 되기 직전 | 0x01000000 | | DocumentAfterSave | 0x0104 | 문서가 저장 된 후 | 0x01000000 | | DocumentBeforeInsert | 0x0105 | 문서에 다른 Content(문서/이미지)가 삽입 되기 전 | 0x01000000 | | DocumentPreview | 0x0106 | 문서 미리보기 (인쇄 등.) | 0x01000000 | | AutoSaveBefore | 0x0107 | 자동저장 파일 저장하기 전 | 0x01000000 | | TempFileCreate | 0x0201 | 임시파일이 생성 | 0x01000000 | | TempFileDelete | 0x0202 | 임시파일이 삭제 | 0x01000000 | | InitApp | 0x0300 | 프로그램 실행시 발생 | 0x01000000 | | BeforeMacroFunction | 0x0400 | 매크로 기능 수행 전 | 0x01000000 | | BeforeMenuEvent | 0x0500 | 사용자가 버튼/단축키 가 눌려지고 동작하기 전 | 0x01000000 |  * DRM 인터페이스 버전은 getVersion 값   각 이벤트별로 uiParam1, uiParam2는 아래의 의미로 사용 형변환을 이용하여 인자를 사용 한다.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | ID | WPARAM | | LPARAM | | | Type | 설명 | type | 설명 | | DocumentBeforeOpen | Char \* | 열릴 문서의 path | Int | path size | | DocumentAfterOpen | Char \* | 열린 문서의 path | Int | path size | | DocumentBeforeSave | Char \* | 저장될 문서의 path | Int | path size | | DocumentAfterSave | Char \* | 저장된 문서의 path | Int | path size | | DocumentBeforeInsert | Char \* | 추가될 object(그림, 문서)의 path | Int | path size | | DocumentPreview | Char \* | 미리보기 할 문서의 path | Int | path size | | AutoSaveBefore | Char \* | 저장될 AutoSave 파일Path | Char \* | 원본 문서의 Path | | TempFileCreate | Char \* | 원본 문서의 path | Char \* | 생성된 임시파일의 path | | TempFileDelete | Char \* | 원본 문서의 path | Char \* | 삭제된 임시파일의 path | | InitApp | void | Reserved | Void | Reserved | | BeforeMacroFunction | void | Reserved | Void | Reserved | | BeforeMenuEvent | Int | 실행되는 메뉴/단축 명령 ID | Void | Reserved |   각 이벤트 별 리턴값의 의미   |  |  |  |  | | --- | --- | --- | --- | | ID | 값 | 설명 | DRM 인터페이스 버전 | | DocumentBeforeOpen | 0,1 | 0 : DRM 시스템에서 불가  1: DRM 시스템에서 동작 가능 | 0x01000000 | | DocumentAfterOpen | 0 | Not Used | 0x01000000 | | DocumentBeforeSave | 0,1 | 0 : DRM 시스템에서 불가  1: DRM 시스템에서 동작 가능 | 0x01000000 | | DocumentAfterSave | 0 | Not Used | 0x01000000 | | DocumentBeforeInsert | 0,1 | 0 : DRM 시스템에서 불가  1: DRM 시스템에서 동작 가능 | 0x01000000 | | DocumentPreview | 0,1 | 0 : DRM 시스템에서 불가  1: DRM 시스템에서 동작 가능 | 0x01000000 | | AutoSaveBefore | 0,1 | 0 : DRM 시스템에서 불가  1: DRM 시스템에서 동작 가능 | 0x01000000 | | TempFileCreate | 0 | Not Used |  | | TempFileDelete | 0 | Not Used |  | | InitApp | 0 | Not Used | 0x01000000 | | BeforeMacroFunction | 0,1 | 0 : DRM 시스템에서 불가  1: DRM 시스템에서 동작 가능 | 0x01000000 | | BeforeMenuEvent | 0,1 | 0: DRM 시스템에서 불가  1: DRM 시스템에서 동작 가능 | 0x01000000 | |

|  |
| --- |
| Interface No. : INT-DOC-4 |
| unsigned int setMenu(void \*pContentInfo, unsigned int uiMenuID); |
| 인자 uiMenuID: [in] Menu ID  반환 : uiMenuID에 해당하는 메뉴의 상태 정보   0 : Disable  1 : Enable |
| 설명  DRM 시스템의 권한에 따라 Menu의 En/Disable 정보를 리턴한다.  오피스에서 암호화 문서를 열람하면, 권한에 따라서 오피스 메뉴를 Disable 처리 해야 한다.  오피스에서 setMenu()를 호출 할 때 Menu ID를 주면, 해당 메뉴의 기능과 권한을 비교 판단하여 Disable 해야 하면 0, Enable 해야 하면 1을 리턴한다 |

* 1. **System Interface (파일의 암/복호화 Interface)**  
     파일의 암/복호화 및 DRM 시스템과 시스템의 I/O처리시 필요한 API Interface  
       
     System Interface 전체 함수

|  |
| --- |
| typedef int(\*\_LPADKLISLICENSEVALIDBYPATH)  (  char \*szFilePath, int iPurpose  );  typedef int(\*\_LPADKISSECURECONTENTBYPATH)  (  char \*szFilePath  );  typedef int(\*\_LPADKGETDOMAINIDBYPATH)  (  char \*szFilePath, char \*szDomainID, int iBufferSize  );  typedef void\* (\*\_LPADKOPENCONTENT)  (  char \*szFilePath, int iWritable, int iTruncate  );  typedef int(\*\_LPADKISSECURECONTENT)  (  void \*pContentInfo  );  typedef int(\*\_LPADKISLICENSEVALID)  (  void \*pContentInfo, unsigned short wPurpose  );  typedef int(\*\_LPADKSETCONTENTPOINTER)  (  void \*pContentInfo, long IOffset, int iOrigin  );  typedef int(\*\_LPADKREADCONTENT)  (  void \*pContentInfo, void\* pBuffer, unsigned long dwNumberOfBytesToRead, unsigned long\* lpdwNumberOfBytesToRead  );  typedef int(\*\_LPADKWRITECONTENT)  (  void \*pContentInfo, void\* pBuffer, unsigned long dwNumberOfBytesToWrite, unsigned long\* lpdwNumberOfBytesToWrite  );  typedef int(\*\_LPADKSETENDOFCONTENT)  (  void \*pContentInfo  );  typedef int(\*\_LPADKCLOSECONTENT)  (  void \*pContentInfo  );  typedef unsigned long(\*\_LPADKGETLASTERROR)();  typedef int(\*\_LPADKSETLASTERROR)  (  unsigned long dwError  );  typedef unsigned long(\*\_LPADKGETDOMAINID)  (  void \*pContentInfo, char \*szDomainID, unsigned int cchDomainId  );  typedef int(\*\_LPADKGETCONTENTTYPE)  (  void \*pContentInfo  );  typedef int(\*\_LPADKPACKCONTENT)  (  void \*pContentInfoTarget, void \*pContentInfoTemplate  );  typedef int(\*\_LPADKAUTOPACKCONTENT)  (  char \*szFilePath  );  typedef int(\*\_LPADKSETUSAGE)  (  void \*pContentInfoTarget, unsigned short wPurpose  );  typedef int(\*\_LPADKISLOGONSTATUS)  (  char \*szDomainID  );  typedef (\*\_LPADKAUTOPACKCONTENT)  (  char \*szFilePath  }  typedef struct FasooSystemInterfaceManager  {  \_LPADKLISLICENSEVALIDBYPATH pfnADKIsLicenseVaildByPath;  \_LPADKISSECURECONTENTBYPATH pfnADKIsSecureContentByPath;  \_LPADKGETDOMAINIDBYPATH pfnADKGetDomainIdByPath;  \_LPADKOPENCONTENT pfnADKOpenContent;  \_LPADKISSECURECONTENT pfnADKIsSecureContent;  \_LPADKISLICENSEVALID pfnADKIsLicenseValid;  \_LPADKSETCONTENTPOINTER pfnADKSetContentPointer;  \_LPADKREADCONTENT pfnADKReadContent;  \_LPADKWRITECONTENT pfnADKWriteContent;  \_LPADKSETENDOFCONTENT pfnADKSetEndOfContent;  \_LPADKCLOSECONTENT pfnADKCloseContent;  \_LPADKGETLASTERROR pfnADKGetLastError;  \_LPADKSETLASTERROR pfnADKSetLastError;  \_LPADKGETDOMAINID pfnADKGetDomainId;  \_LPADKGETCONTENTTYPE pfnADKGetContentType;  \_LPADKPACKCONTENT pfnADKPackContent;  \_LPADKAUTOPACKCONTENT pfnADKAutoPackContent;  \_LPADKSETUSAGE pfnADKSetUsage;  \_LPADKISLOGONSTATUS pfnADKIsLogonStatus;  } SIM;  typedef SIM \*LPSIM; |

각 인터페이스 함수 상세 설명

|  |
| --- |
| Interface No. : INT-SYS-1 |
| void ADKIsLicenseValidByPath(char \*szFilePath, int iPurpose); |
| 인자 :   szFilePath : [in] 파일의 Full Path  iPurpose : [in] 이 파일의 확인하고자 하는 권한 (문서를 사용하는 목적)  반환 : 없음. |
| 설명 열람 또는 편집 등, szFilePath의 파일이 iPurpose 권한을 가지고 있는지 확인하기 위한 API 이때, ADKGetLastError()을 확인하여 E\_ADK\_CONTENT\_INVALID\_LICENSE 일 경우에만 지정한 권한이 없음을 보장한다. 나머지는 DRM Client 시스템이 정상적으로 동작하지 않은 상태이다.. |

|  |
| --- |
| Interface No. : INT-SYS-2 |
| int ADKIsSecureContentByPath(char \*szFilePath); |
| 인자 :  szFilePath : [in] 파일의 Full Path  반환 :  0 : 암호화 되지 않은 일반 파일 1 : 암호화 파일 |
| 설명 ADKOpenContent 함수를 실행 하지 않고 사용할 수 있는 함수이다.  반환 값이 0이면서 ADKGetLastError()의 반환 값이 E E\_ADK\_NOT\_SECURE\_CONTENT 의 경우에만 보안 문서가 아님을 보장 한다. 만약 ADKGetLastError()의 반환 값이 다른 값인 경우, DRM Client 시스템이 정상적이지 않은 상태이다. |

|  |
| --- |
| Interface No. : INT-SYS-3 |
| int ADKGetDomainIdByPath (char \*szFilePath, char \*szDomainID, int iBufferSize); |
| 인자 :  szFilePath : [in] 파일의 Full Path  szDomainID : [out] 출력 스트링 버퍼  iBufferSize : [in] 출력 스트링 버터 사이즈 (글자수)  반환 :  0 : 실패  0 보다 큼 : 성공 (DomainID의 글자 수) |
| 설명  일반적으로 이 함수를 사용 할 일은 없다.  예를 들면,  보안 문서를 열때,  문서에 해당하는 SSO가 로그온 되어 있지 않은 경우, 예외 처리 등을 하기 위해 사용한다.  이 함수는 ADKOpenContent() 함수를 호출 하지 않고 사용 할 수 있다.  szDomainID를 NULL로 설정하고, 함수를 호출 하면 필요한 버퍼 크기를 리턴 한다.  (이 경우, ADKGetLastError()의 값은 E\_ADK\_INSUFFICIENT\_BUFFER를 반환한다.) |

|  |
| --- |
| Interface No. : INT-SYS-4 |
| Void \* ADKOpenContent(char \*szFilePath, int iWritable, int iTruncate); |
| 인자 :  szFilePath : [in] Open 하려는 파일의 Full Path  iWritable : [in] 쓰기 모드 사용 여부 (0:읽기전용, 0 이상 쓰기가능)  ITruncate : [in] 문서 내용 비우기 여부 (0:비우지 않음, 0 이상 비우기 모드)    반환 :  0(null) : 실패 0 이외 : 성공 (이하 ContentInfo로 표시한다.) |
| 설명 **Windows API의 CreateFile 또는 표준 C의 fopen의 역할을 하는 함수이다.**  szFilePath의 보안 문서를 읽기만 할 경우, iWritable를 0으로 설정한다.  iWritable > 0 이면, 쓰기 모드로 동작한다.  만약 보안 문서의 내용을 새 이름으로 저장 할 경우라면, iWritable와 iTruncate 모두 0으로 설정해야 한다. iTruncate == 0 이면, 사용하지 않을 보안문서의 라이선스를 전송 받게 된다.  반환 값이 0(null)인 경우, ADKGetLastError()의 리턴 값으로 실패 이유를 확인 할 수 있다.  반환 된 void \* 값은 ADKCloseContent()로 닫아야 한다. |

|  |
| --- |
| Interface No. : INT-SYS-5 |
| Int ADKIsSecureContent(void \*pContentInfo); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  반환 :  0 : 일반문서 0 보다 큼 : 암호화 문서 |
| 설명 암호화 분서일 경우, 0 보다 큰 값이 반환되고  암호화 되지 않은 일반 문서인 경우, 0이 반환 된다. |

|  |
| --- |
| Interface No. : INT-SYS-6 |
| Int ADKIsLicenseValid(void \*pContentInfo, word wPurpose); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  wPurpose : [in] 사용자가 pContentInfo 문서에 가지고 있는지 확인하고 싶은 권한  반환 :  0 : 권한 없음. 0 보다 큼 : wPurpose 권한을 가지고 있음. |
| 설명 pContentInfo에 해당하는 문서에, 사용자가 가지고 있는 권한을 검사하기 위한 함수  예를 들어 PContentInfo 문서에 인쇄 권한을 가지고 있는지 확인하는 등에 사용  0이 리턴 되었을 경우,  ADKGetLastError() 를 확인하여 E\_ADK\_CONTENT\_INVALID\_LICENSE 일 경우에만 지정한 목적에 대한 권한이 없음을 보장하고, 그 외엔 함수 동작이 실패한 것이다. (Error Code 참고) |

|  |
| --- |
| Interface No. : INT-SYS-7 |
| longlong ADKGetContentSize(void \*pContentInfo, word wPurpose); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  반환 : 컨텐트의 크기 실패시 INVALID\_CONTENT\_SIZE 리턴 (Error Code 참고) |
| 설명 **Windows API의 GetFileSize 또는 표준 C의 fseek(….SEEK\_END); ftell();의 역할을 한다.**  보안 문서의 컨텐트 크기를 반환한다. 🡺 암호화 되기 이전의 데이터 크기를 알려 준다.  일반 문서의 경우, 파일 크기를 반환 한다.  보안 문서는 암호화된 일반 문서의 컨텐트 이외에, 메타데이터(보안 헤더)가 추가 되기 때문에 일반 문서 일 때의 파일 크기 보다 커지게 된다.  문서 저작 도구가 암호화 파일의 크기가 아닌 실제 컨텐트의 크기가 필요 할 때에 사용 한다.  DRM 환경은 4GB 이상의 파일 크기를 지원하지 않는다. |

|  |
| --- |
| Interface No. : INT-SYS-8 |
| int ADKSetContentPointer(void \*pContentInfo, long lOffset, int iOrigin); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  lOffset : [in] 이동할 거리  iOrigin : [in] SEEK\_SET, SEEK CUR, SEEK END 중 하나  반환 : Content의 새 위치 실패시 INVALID\_SET\_CONTENT\_POINTER 리턴 (Error Code 참고) |
| 설명 **Windows API의 SetFilePointer() 또는 표준 C의 fseek()의 역할을 한다.**  현재 Content의 읽기/쓰기를 할 위치.  iOrigin을 SEEK CUR, IOffset을 0으로 설정하면, 현재 Content 포인터를 알 수 있다. |

|  |
| --- |
| Interface No. : INT-SYS-9 |
| int ADKReadContent(void \*pContentInfo, void \*pBuffer, long lNumberOfBytesToRead, long lNumberOfBytesRead); |
| 인자 :  pContentInfo : ADKOpenContent의 반환값, ContentInfo  pBuffer : 읽은 데이터를 저장할 메모리  lNumberOfBytesToRead : 읽으려는 데이터의 바이트 수  lNumberOfBytesRead : 실제로 읽어 들인 바이트 수  반환 :  0 : 실패  0 이상 : 성공 |
| 설명 **Windows API의 ReadFile() 또는 표준 C의 fread()의 역할을 한다.**  이 함수는 보안 문서의 내용을 복호화 하므로 사용자 권한에 따른 보안에 주의 해야 한다.  컨텐트의 포인터는 이 함수로 읽은 받이트 만큼 뒤쪽으로 이동한다. |

|  |
| --- |
| Interface No. : INT-SYS-10 |
| int ADKWriteContent(void \*pContentInfo, void \*pBuffer, long lNumberOfBytesToWrite, long \*pNumberOfBytesWritten); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  pBuffer : [out] 쓰여질 데이터를 저장할 메모리  lNumberOfBytesToRead : [in] 쓰려는 데이터의 바이트 수  pNumberOfBytesRead : [out] 실제로 쓰여진 바이트 수  반환 :  0 : 실패  0 이상 : 성공 |
| 설명 **Windows API의 WriteFile() 또는 표준 C의 fwrite()의 역할을 한다.**  읽기 모드로 오픈한 컨텐트 핸들로는 본 함수를 사용할 수 없다.  보안 문서에 SECURE\_SAVE 권한이 없다면 본 함수를 사용할 수 없다.  기록하는 내용이 보안문서의 내용이라면, ADKIsSecureContent() 함수를 이용해 보안문서 여부를 판단하고, 일반 문서일 경우 ADKPackContent() 함수를 이용해 보안문서로 만든 후 기록해야 한다.  보안 응용 프로그램에서 임의로 보안문서의 내용을 일반파일일로 저장하는 것은 보안 문제가 될 수 있으므로 주의한다.  컨텐트의 포인터는 이 함수로 기록한 바이트만큼 뒤쪽으로 이동한다. |

|  |
| --- |
| Interface No. : INT-SYS-11 |
| int ADKSetEndOfContent(void \*pContentInfo); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  반환 :  0 : 실패  0 이상 : 성공 |
| 설명 **Windows API의 SetEndOfFile()와 같은 역할을 한다.**    읽기 모드로 오픈한 컨텐트 핸들로는 본 함수를 사용할 수 없다.  보안 문서에 SECURE\_SAVE 권한이 없다면 본 함수를 사용할 수 없다.    컨텐트의 끝을 지정하는 함수로, 현재 컨텐트 포인트 (ADKSetContentPointer로 지정한)를 파일의 끝으로 설정한다. 이 함수가 성공하면 Content Pointer 이후의 내용은 지워진다. |

|  |
| --- |
| Interface No. : INT-SYS-12 |
| int ADKCloseContent (void \*pContentInfo); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  반환 :  0 : 실패  0 이상 : 성공 |
| 설명 **Windows API의 CloseHandle() 또는 표준 C의 fclose와 같은 역할을 한다.**  ADKOpenContent후 파일 사용이 완료되면 반드시 ADKCloseContent를 호출 해야 한다. |

|  |
| --- |
| Interface No. : INT-SYS-13 |
| Long ADKGetLastError (); |
| 인자 : 없음.  반환 :  0 : 가장 최근에 발생한 오류 코드 |
| 설명 **Windows API의 GetLastError()과 같은 역할을 한다.** |

|  |
| --- |
| Interface No. : INT-SYS-14 |
| Void ADKSetLastError(long lError); |
| 인자 : [in] 임의로 설정할 에러 값  반환 : 없음. |
| 설명 Windows API의 SetLastError()과 같은 역할을 한다.  다른 ADK 함수를 호출하기 전에 이 함수를 사용하여 임의의 값으로 Error 를 설정하면, 이후 호출되는 ADKGetLastError() 의 리턴값이 그 이후에 변경된 값이라는 것을 보장 할 수 있다. |

|  |
| --- |
| Interface No. : INT-SYS-15 |
| int ADKGetDomainId(void \*pContentInfo, char \* szDomainID, int iBufferSize); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo  szDomainID : [out] 출력 스트링 버퍼  iBufferSize : [in] 출력 스트링 버터 사이즈 (글자수)  반환 :  0 : 실패  0 보다 큼 : 성공 (DomainID의 글자 수) |
| 설명  일반적으로 이 함수를 사용 할 일은 없다.  보안 문서의 출처에 따라 다른 처리를 해야 할, 특별한 경우에만 사용 한다.  szDomainID를 NULL로 설정하고, 함수를 호출 하면 필요한 버퍼 크기를 리턴 한다.  (이 경우, ADKGetLastError()의 값은 E\_ADK\_INSUFFICIENT\_BUFFER를 반환한다.) |

|  |
| --- |
| Interface No. : INT-SYS-16 |
| int ADKGetContentType(void \*pContentInfo); |
| 인자 :  pContentInfo : [in] ADKOpenContent의 반환값, ContentInfo    반환 :  0 : 실패  0 보다 큼 : 성공 (ADK\_TYPE\_\*로 정의된 값) |
| 설명  일반적으로 이 함수를 사용 할 일은 없다.  보안 문서의 종류에 따라 다른 처리를 해야 할, 특별한 경우에만 사용 한다. |

|  |
| --- |
| Interface No. : INT-SYS-17 |
| int ADKGetContentType(void \*pContentInfoTarget, void \*pContentInfoTemplate); |
| 인자 :  pContentInfoTarget : [in] 일반 문서(암호화 되지 않은) 문서의 ContentInfo  pContentInfoTemplate: [in] 암호화 할 컨텐츠 핸들 템플릿    반환 :  0 : 실패  0 보다 큼 : 성공 |
| 설명  본 함수는 일반 파일을 암호화 하여 저장이 필요 한 경우 사용한다.  예를 들어, 암호화 문서를 열람/편집 하여 저장 할 때, 문서 저작 도구에 따라서 새 파일을 만들어서, 기존 파일에 덮어 쓰는 경우가 있다.  이 경우, 처음 열람한 암호화 파일의 ContentInfo를 템플릿으로 사용하여, 새로 만들어지는 문서를 암호화 한다. |

|  |
| --- |
| Interface No. : INT-SYS-18 |
| int ADKSetUsage (void \*pContentInfoTarget, word wPurpose); |
| 인자 :  pContentInfoTarget : [in] ADKOpenContent의 반환값, ContentInfo  wPurpose : 사용 내역    반환 :  0 : 실패  0 보다 큼 : 성공 |
| 설명  본 함수는 DRM 문서가 특정 권한에 맞게 사용 되었다는 사용 내역을 전송 할 때 사용 한다.  예를 들어, 문서를 인쇄 할 때, 인쇄를 했다는 사용 내역을 전송할 필요가 있을 때 사용 한다. |

|  |
| --- |
| Interface No. : INT-SYS-19 |
| int ADKIsLogonStatus (char \*szDomainID); |
| 인자 :  szDomainID : [in] 로그인 확인을 위한 DomainID    반환 :  0 : 실패  0 보다 큼 : 성공 |
| 설명  일반적으로 사용하지 않은 함수이다.  출처의 로그인이 되어 있는지 확인 하기 위한 함수이다. |

|  |
| --- |
| Interface No. : INT-SYS-20 |
| Int ADKAutoPackContent(char \*szFilePath); |
| 인자 : [in] szFilePath 암호화 할 파일의 Full Path 정보  반환 : 성공 : 1  실패 : 0. |
| 설명 일반 문서를 DRM 설정에 따라 암호화 한다.  이 함수는 새 문서 만들기 또는, 문서를 열람하기 전부터 암호화 되지 않은 문서에 대해서 수행 해야 한다.  파일로 저장후, 이 API를 호출 하면 내부에서 DRM 설정이 일반 파일을 암호화 해야 하는지, 아닌지 판단하여 DRM의 설정에 따라 암호화 한다.  상세 에러 코드는 ADKGetLastError()에서 반환값을 참고 한다. |

|  |
| --- |
| Interface No. : INT-SYS-21 |
| Int ADKPackContent(void \*pContentInfoTarget, void \*pContentInfoTemplate); |
| 인자 : [in] pContentInfoTarget : 암호화 할 파일의 ContentInfo  [in] pContentInfoTemplate : 암호화 할 파일이 파생된 원본 암호화 파일의 ContentInfo  반환 : 성공 : 1  실패 : 0. |
| 설명 암호화 문서를 사용 중, 다른 이름으로 저장 또는 내보내기 등, 다른 파일로 저장할 때 사용하는 암호화 함수 이다.  ADKAutoPackContent와의 차이점은 현재 문서가 일반 문서일 때는 ADKAutoPack함수를 이용하고,  현재 사용하고 있는 문서가 암호화 문서일 때는 ADKPackContent API를 사용해야 한다.  상세 에러 코드는 ADKGetLastError()에서 반환값을 참고 한다. |

# Error Code

|  |  |  |
| --- | --- | --- |
| Error Code | Value | Description |
| E\_ADK\_OK | 0x0 | 정상 |
| E\_ADK\_LOAD\_NXL\_FAIL | 0x1 | F\_nxl.dll 로드 실패 |
| E\_ADK\_NXL\_INIT\_FAIL | 0x2 | F\_nxl.dll 초기화 실패 |
| E\_ADK\_WIN32\_CREATEFILE\_FAIL | 0x3 | 파일 오픈 실패 |
| E\_ADK\_INVALID\_ARGS | 0x4 | 부적절한 파라미터 |
| E\_ADK\_ALREADY\_PRINTING | 0x5 | 또다른 인쇄를 하려면 기다려야 함 |
| E\_ADK\_PRINTING\_NOT\_STARTED | 0x6 | 인쇄가 시작되지 않음 |
| E\_ADK\_MEMORY\_ERROR | 0x7 | 메모리 할당 오류 |
| E\_ADK\_INSUFFICIENT\_BUFFER | 0x8 | 충분하지 않은 크기의 버퍼 |
| E\_ADK\_CONVERSION\_FAIL | 0x9 | 형 변환 오류 |
| E\_ADK\_DRM\_CLIENT\_NOT\_FOUND | 0xD | DRM Client 확인되지 않음 |
| E\_ADK\_NOT\_SECURE\_FILE | 0xFF | 일반 문서 파일 |
| E\_ADK\_CERT\_NOT\_ALLOWED\_FILE | 0x101 | Fac 파일로 인증되지 않은 모듈 |
| E\_ADK\_FAC\_LOAD\_FAIL | 0x201 | 모듈에 맞는 fac 파일 로드 실패 |
| E\_ADK\_CONTENT\_OPEN\_FAIL | 0x301 | 컨텐트 오픈 실패 |
| E\_ADK\_CONTENT\_READ\_FAIL | 0x302 | 컨텐트 읽기 실패 |
| E\_ADK\_CONTENT\_WRITE\_FAIL | 0x303 | 컨텐트 기록 실패 |
| E\_ADK\_CONTENT\_INVALID\_LICENSE | 0x304 | 허가되지 않은 동작 |
| E\_ADK\_CONTENT\_NOT\_ENOUGH\_INFO | 0x305 | 보안문서에 대한 정보를 얻을 수 없음 |
| E\_ADK\_CONTENT\_PACK\_FAIL | 0x306 | 템플릿 이용한 보안파일 생성 실패 |
| E\_ADK\_CONTENT\_SET\_POINTER\_FAIL | 0x307 | SetContentPointer 실패 |
| E\_ADK\_CONTENT\_SET\_END\_FAIL | 0x308 | SetEndOfContent 실패 |
| E\_ADK\_CONTENT\_DIFFER\_FROM\_START | 0x309 | ADKStartPrint 에서와 다른 컨텐트 |
| E\_ADK\_CONTENT\_INIT\_RIGHTS\_FAIL | 0x30A | 잘못된 컨텐트 ID 인자 |
| E\_ADK\_CONTENT\_LICENSE\_DEL\_FAIL | 0x30B | 라이선스 파일 삭제 실패 |
| E\_ADK\_INTERNAL\_GET\_CLSID\_FAIL | 0x401 | 보안문서에 대한 정보를 얻을 수 없음 |
| E\_ADK\_INTERNAL\_ENCRYPTION | 0x402 | 암호화 오류 |
| E\_ADK\_INTERNAL\_FNC\_POINTER\_FAIL | 0x404 | 내부 함수 포인터 획득 실패 |
| E\_ADK\_GENERAL | 0xABCD | 정의 되지 않은 오류 |

# 상수 정의

|  |  |
| --- | --- |
| INVALID\_SET\_CONTENT\_POINTER | 0xFFFFFFFF |
| SCP\_BEGIN | 0 |
| SCP\_CURRENT | 1 |
| SCP\_END | 2 |
|  |  |
| ADK\_CONTENT\_FSD | 1 |
| ADK\_CONTENT\_FSE | 2 |
| ADK\_CONTENT\_FSN | 3 |
| ADK\_CONTENT\_NX | 4 |
|  |  |
| ADK\_PURPOSE\_VIEW | 2 |
| ADK\_PURPOSE\_SAVE | 3 |
| ADK\_PURPOSE\_EDIT | 4 |
| ADK\_PURPOSE\_EXTRACT | 5 |
| ADK\_PURPOSE\_PRINT | 6 |
| ADK\_PURPOSE\_PRINT\_SCREEN | 8 |
| ADK\_PURPOSE\_TRANSFER | 10 |
| ADK\_PURPOSE\_SECURE\_SAVE | 12 |
| ADK\_PURPOSE\_SECURE\_PRINT | 13 |
|  |  |
| NOT\_ALLOWED\_VIRTUAL\_PRINTER | 0 |
| ALLOWED\_VIRTUAL\_PRINTER | 1 |
| REAL\_PRINTER | 2 |

# 연동 방법과 샘플

인터페이스 연동 방법에 대해 설명과 샘플 코드이다.

(연동 방법의 설명 등 업데이트가 될 수 있으니, 최신 버전은 [GitHub](https://github.com/Fasoo-OpenProject/OpenOS-DRM) 에서 참고)

- 준비 과정

1. Interface 코드 다운로드
   1. [Github](https://github.com/Fasoo-OpenProject/OpenOS-DRM)에서 소스 코드 및 Interface 정의서등 다운로드

|  |
| --- |
| #include "f\_extadk.h"  #define EXTADKPATH "/usr/local/fasoo/libfasooextadk.so" |

1. Extadk.h 소스 포함

- 초기화 코드

1. Interface 초기화 코드

|  |
| --- |
| void \*dlHandle = NULL;  dlHandle = dlopen(EXTADK\_PATH, RTLD\_LAZY);  if (!dlHandle)  {  fputs(dlerror(), stderr);  printf("dlopen error '%s'\n", dlerror());  return -1;  }  else  {  printf("dlopen success.\n");  } |

* 1. dlopen()API를 이용하여 .so 모듈 로드
  2. Interface 모듈 초기화 API 호출 준비  
     초기화 API 함수 포인터 획득

|  |
| --- |
| typedef int(\*\_LPDRMInterfaceInitialize)(void \*\*ppDocumentInterface, void \*\*ppSystemInterface, void \*pReserved, void\* pReserved);  \_LPDRMInterfaceInitialize pfnDRMInterfaceInitialize = (\_LPDRMInterfaceInitialize)dlsym(hDll, "DRMInterfaceInitialize");  if (!pfnDRMInterfaceInitialize)  {  \_TRACEA(0, "[FasooDrmInitialize] ExtADK dlsym failed. '%s'", dlerror());  } |

|  |
| --- |
| // 현재는 사용 되지 않는다, 추후 다른 개방형 OS지원시 이용 예정이다.  // 초기화 함수에 구조체를 인자로 넣어도 현재 버전은 사용 하지 않는다.  typedef struct FVerInfo  {  char szname[256];  unsigned long ulVersion;  } FVI;  typedef FVI\* LPFVI;  LPSIM g\_lpSIM = NULL;  LPDIM g\_lpDIM = NULL;  FVI g\_fOSVI;  FVI g\_fOFFICEVI;  if (pfnDRMInterfaceInitialize((void\*\*)&g\_lpDIM, (void\*\*)&g\_lpSIM, (void\*)&g\_fOSVI, (void\*)&g\_fOFFICEVI) == 0)  {  \_TRACEA(0, "[FasooDrmInitialize] DRMInterfaceInitialize failed.");  return false;  } |

* 1. 초기화 API 호출  
     초기화 API 호출하여 Interface Pointer 획득

- API의 사용

1. 문서의 열람

|  |
| --- |
| LPCONTENTINFO lpContentInfo =  (LPCONTENTINFO)g\_lpSIM->pfnADKOpenContent(szFilePath, argv[3], bWritable, bTruncate);  if (lpContentInfo != NULL)  printf("OpenContent success.\n");  else  printf("OpenContent failed.\n"); |

1. 암호화 문서에 열람 권한이 있는지 확인

|  |
| --- |
| if (g\_lpSIM->pfnADKIsLicenseVaildByPath(szFilePath, kDRMPurposeView) > 0)  {  printf("'%s' VIEW license valid.\n", szFilePath);  }  else  {  printf("'%s' VIEW license invalid.\n", szFilePath);  } |

1. 암호화 문서 읽기

|  |
| --- |
| const char \*FileMode = "r";  LPCONTENTINFO lpContentInfo =  (LPCONTENTINFO)g\_lpSIM->pfnADKOpenContent(szFilePath, (char\*)FileMode, false, false);  if (lpContentInfo != NULL)  {  char szBuffer[1024] = { 0 };  unsigned long lBytesReaden;  if (g\_lpSIM->pfnADKReadContent(lpContentInfo, &szBuffer, 50, &lBytesReaden) > 0)  {  printf("ReadContent success. '%s'", szBuffer);  }  else  {  printf("ReadContent failed.\n");  }  } |

1. 암호화 문서 해제

|  |
| --- |
| if (g\_lpSIM->pfnADKUnpackByContentPath(szFilePath, NULL))  {  printf("UnpackbyContentInfo success.\n");  }  else  {  printf("UnpackbyContentInfo failed.\n");  } |