Easy Transparent Encryption File System

(Minifilter Based)

ETEFS_Mini - Version 4.2

SDK Reference

0 Copyright notice

Please read the “license.doc” to get enough copyright information first. If you do not agree with this license term, please don’t use any software provided by ETEFS.COM.

1 Introduction

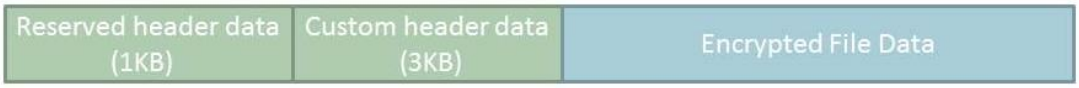
Summary

This document is writing for developer who uses ETEFS_Mini SDK to create their own data security related products. It will illustrate the architecture of ETEFS_Mini and describe the API functions exported by ETEFS_Mini in detail.

Concept

- ✓ **Header Data**

The header data is a data stream that bound to the normal data body for an encrypted file by ETEFS_Mini. It is consist of two parts. One part is reserved by ETEFS_Mini and another part is free for developer. The flowing diagram shows the layout of an encrypted file.



2 Features

The full feature list includes 4 functionality modules. They are “encryption application support”, “enhancement support for encryption”, “file event monitor” and “file access control”. We will give a table in detail for each functionality module.

✓ **Encryption application support**

Application Name	Application Version	Comment
Notepad	XP(SP3) Win7/Win8/Win8.1/Win10(X86/X64)	OS accessory
Wordpad	XP(SP3) Win7/Win8/Win8.1/Win10(X86/X64)	OS accessory
Mspaint	XP(SP3) Win7/Win8/Win8.1/Win10(X86/X64)	OS accessory
Office Word	Office 2003、2007、2010、2012、2013、2016	
Office Excel	Office 2003、2007、2010、2012、2013、2016	
Office PowerPoint	Office 2003、2007、2010、2012、2013、2016	
Foxit PDF	Version 4.0	
Adobe Acrobat	Version 9.0	
Adobe Reader	V8.0/9.0/10.0/11.0	
AutoCAD	AutoCAD R14、2004、2008	

These policies for these applications are provided by ETEFS_Mini itself. ETEFS_Mini can support any type of application by giving a correct policy.

✓ **Enhancement support for encryption**

The flowing table shows all Enhancement polies.

Name	Description	Supported application
Random FEK	Allocate a random file encryption key for each file	All applications
Binding custom data	Binding custom data into an encrypted file	All applications
Encryption manually	Overrides the default on-access encryption option	All applications
Faked exe checking	Prevent the faked exe to read the plain data of an encrypted file	All applications
Save-as encryption	Encrypt the new file created from save-as operation	NOTEPAD、WORDPAD、OFFICE

✓ **File event monitor**

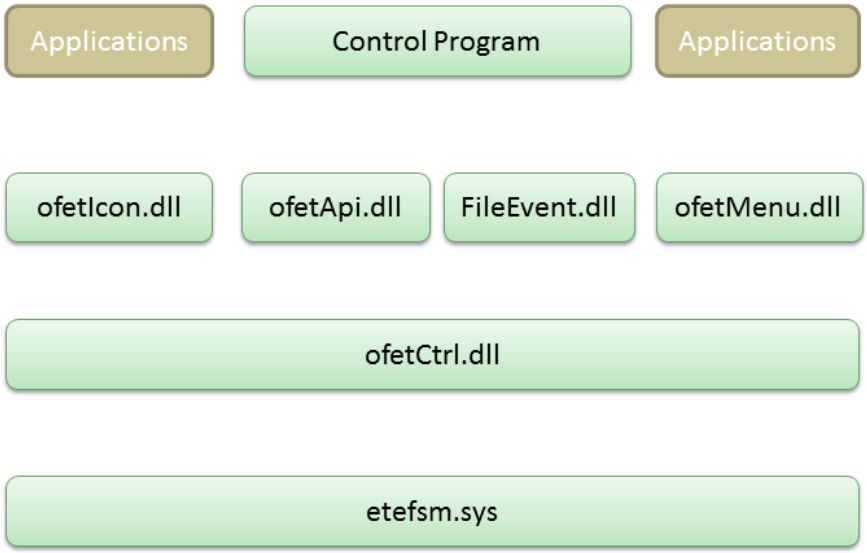
Event Name	Report Time	Supported application
QUERY_PRIVILEGE	When a file is opened, ETEFS_Mini will send this event to query access privilege for the file.	NOTEPAD、WORDPAD、OFFICE
FILE_PRINT	When a user print this file	NOTEPAD、WORDPAD、OFFICE
OPEN_FROM_API	The first call of ZwCreateFile to access this file	NOTEPAD、WORDPAD、OFFICE
CLOSE_FROM_WND	When the window closed	NOTEPAD、WORDPAD、OFFICE
FILE_SAVE_AS	When a user save-as a file	NOTEPAD、WORDPAD、OFFICE
FAKED_DETECTED	When a faked exe is detected	NOTEPAD、WORDPAD、OFFICE

✓ **File access control**

Event Name	Report Time	Supported application
Read-only	Disable writing data to file	NOTEPAD、WORDPAD、OFFICE
Disable Print	Disable printing file	NOTEPAD、WORDPAD、OFFICE
Disable Open	Disable open file	NOTEPAD、WORDPAD、OFFICE
Disable copy data to clipboard	Disable coping data to clipboard	NOTEPAD、WORDPAD、OFFICE
Disable SAVE-AS	Disable SAVE-AS	NOTEPAD、WORDPAD、OFFICE
Disable Drag-and-Drop	Disable drag data to other window	NOTEPAD、WORDPAD、OFFICE

3 Architecture

The flowing diagram is the architecture of ETEFS_Mini.



This table illustrates the function for each module.

NO	Name	Description
1	ofetIcon.dll	A shell plugin that overlays a small lock icon on the main icon for an encrypted file
2	ofetMenu.dll	A shell plugin that provides some menu items to manage the state of an encrypted file
3	ofetApi.dll	An interface DLL which is used to send, cancel policy and enable the enhancement features
4	FileEvent.dll	Push the event log to control program
5	ofetCtrl.dll	Capture file operation logs and do file access privilege control
6	Etefsm.sys	Implementing the file transparent encryption feature with a minifilter driver

4 ofetApi Module

4.1 Start/Stop Filtering

The developers of ETEFS_Mini can use SetStartFiltering and GetStartFiltering to start, stop filtering and query the state. Before sending any process encryption policy you must use SetStartFiltering to set the filtering engine into running state. The proto type and parameters are shown below.

SetStartFiltering

ITEM	CONTENT
Declaration	DWORD SetStartFiltering (BOOL bStart);
Function	Set the working state of ETEFS_Mini engine
Parameter	bStart – TRUE. Set ETEFS_Mini engine into running state FALSE. Set ETEFS_Mini engine into stopped state
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	You must use SetStartFiltering to set the filtering engine into running state; otherwise ETEFS_Mini will not accept any process encryption policy.

GetStartFiltering

ITEM	CONTENT
Declaration	DWORD GetStartFiltering (BOOL* bStart);
Function	Query the running state of filtering engine
Parameter	bStart – an output parameter to receive the state value. If running, set to TRUE.
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	N/A

4.2 Set/Get cipher configuration

SetDriverCryptConfig

ITEM	CONTENT
Declaration	DWORD SetDriverCryptConfig(DWORD CihperID, char* szKeyBuffer, int nKeyLength);
Function	Set cipher configuration for ETEFS_Mini
Parameter	CihperID–CIPHER_ID_XTEA. Tell ETEFS_Mini using XTEA as cipher. CIHPER_ID_AES. Tell ETEFS_Mini using AES as cipher. szKeyBuffer – pointer to the key buffer nKeyLength– if using XTEA, nKeyLength must be equal to 16 if using AES, nKeyLength must be equal to 32
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	N/A

GetCurrentCipherID

ITEM	CONTENT
Declaration	DWORD GetCurrentCipherID (DWORD* CihperID);
Function	Get cipher configuration currently used by ETEFS_Mini
Parameter	CihperID– An output parameter to receive the Cipher ID
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	N/A

4.3 Set/Get random key mode

SetRandomKeyMode

ITEM	CONTENT
Declaration	DWORD SetRandomKeyMode(DWORD Value);
Function	Get the encryption configuration from the encrypted file
Parameter	Value– Set TRUE to enable the random file encryption key feature.
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

GetRandomKeyMode

ITEM	CONTENT
Declaration	DWORD GetRandomKeyMode(DWORD* Value);
Function	Get the encryption configuration from the encrypted file
Parameter	Value– If random key is enables, this parameter is set to TRUE.
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

4.4 Get file encryption configuration

GetFileCryptConfig

ITEM	CONTENT
Declaration	DWORD GetFileCryptConfig(WCHAR* wszFileName, PFILE_CRYPT_CONFIG cryptConfig);
Function	Get the encryption configuration from the encrypted file
Parameter	wszFileName- Full file path name cryptConfig – Receive the encryption configuration.

Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	typedef struct _FILE_CRYPT_CONFIG { ULONG dwCipherID; ULONG bRandomKeyEnabled; CHAR szFileKey[128]; }FILE_CRYPT_CONFIG,*PFILE_CRYPT_CONFIG; Unless the file encryption key is correctly set, the szFileKey field will not receive the corresponding file key.

IsEncryptedFile

ITEM	CONTENT
Declaration	DWORD IsEncryptedFile(WCHAR* wszFileName);
Function	To check whether the file is encrypted or not
Parameter	wszFileName- Full file path name
Return Value	If encrypted, returns ERR_SUCCESS an appropriate error code on failure.
Comment	N/A

4.5 Encrypting and decrypting file

efs_EncryptFile

ITEM	CONTENT
Declaration	BOOL efs_EncryptFile(WCHAR* wszFileName);
Function	Encrypting a file
Parameter	wszFileName- Full file path name
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

efs_EncryptFileAlone

ITEM	CONTENT
Declaration	DWORD efs_EncryptFileAlone(WCHAR* wszFileName, int CipherID, int bUseRandomKey, char* FileKey);
Function	Encrypting a file with a specified key.
Parameter	wszFileName- Full file path name CipherID– The cipher ID bUseRandomKey– whether allocates a random key or not FileKey – the file key.
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

efs_DecryptFile

ITEM	CONTENT
Declaration	DWORD efs_DecryptFile(WCHAR* wszFileName);
Function	Decrypting a file
Parameter	wszFileName- Full file path name
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	N/A

efs_DecryptFileAlone

ITEM	CONTENT
Declaration	DWORD efs_DecryptFileAlone(WCHAR* wszFileName, int CipherID, char* FileKey);
Function	Decrypting a file with a specified key.
Parameter	wszFileName- Full file path name CipherID– The cipher ID

	FileKey – the file key.
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	N/A

4.6 Reading and writing custom header data

SetCustomData

ITEM	**CONTENT**
Declaration	DWORD SetCustomData(WCHAR* wszFileName, void* CustomDataBuffer, ULONG BufferSize);
Function	Writing custom header data to an encrypted file.
Parameter	wszFileName- Full file path name CustomDataBuffer –pointer to buffer BufferSize –size of buffer, must be 1024 bytes
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	#define CUSTOM_DATA_SIZE 1024

SetCustomDataAlone

ITEM	**CONTENT**
Declaration	DWORD SetCustomDataAlone(WCHAR* wszFileName, PVOID CustomDataBuffer, ULONG BufferSize, int CipherID, char* FileKey, int KeyLength);
Function	Writing custom header data to an encrypted file with a specified key.
Parameter	wszFileName- Full file path name CustomDataBuffer –pointer to buffer BufferSize –size of buffer, must be 1024 bytes CipherID – Cipher ID FileKey – the file key.

	KeyLength – length o key.
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	#define CUSTOM_DATA_SIZE 1024

GetCustomData

ITEM	**CONTENT**
Declaration	DWORD GetCustomData(WCHAR* wszFileName, PVOID CustomDataBuffer, ULONG BufferSize);
Function	Reading custom header data from an encrypted file
Parameter	wszFileName- Full file path name CustomDataBuffer –pointer to buffer BufferSize –size of buffer, must be 1024 bytes
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	#define CUSTOM_DATA_SIZE 1024

GetCustomDataAlone

ITEM	**CONTENT**
Declaration	DWORD GetCustomDataAlone(WCHAR* wszFileName, PVOID CustomDataBuffer, ULONG BufferSize, int CipherID, char* FileKey, int KeyLength);
Function	Reading custom header data from an encrypted file with a specified key.
Parameter	wszFileName- Full file path name CustomDataBuffer –pointer to buffer BufferSize –size of buffer, must be 1024 bytes CipherID – Cipher ID FileKey – the file key. KeyLength – length o key.

Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	#define CUSTOM_DATA_SIZE 1024

4.7 Enable/Disable binding custom header data

SetCustomDataResident

ITEM	**CONTENT**
Declaration	DWORD SetCustomDataResident(DWORD dwValue);
Function	Enable or disable binding custom header data
Parameter	dwValue – 1 or TRUE to enable, 0 or false to disable
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

GetCustomDataResident

ITEM	**CONTENT**
Declaration	DWORD GetCustomDataResident(DWORD* dwValue);
Function	Get the configuration of binding custom header data
Parameter	dwValue – (output) 1 or TRUE equals enabled, 0 or false equals disabled
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

4.8 Enable/Disable manually encryption

SetManualEncryptFile

ITEM	**CONTENT**
Declaration	DWORD SetManualEncryptFile(DWORD dwValue);

Function	Enable or disable manually encryption
Parameter	dwValue – 1 or TRUE to enable, 0 or false to disable
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

GetManualEncryptFile

ITEM	**CONTENT**
Declaration	DWORD GetManualEncryptFile(DWORD* dwValue);
Function	Get the configuration of manually encryption
Parameter	dwValue – 1 or TRUE to enable, 0 or false to disable
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

4.9 White directory configuration

AddWhiteDir

ITEM	**CONTENT**
Declaration	DWORD AddWhiteDir(WCHAR* wszDir);
Function	Add a white directory to system. Encryption and decryption will be disabled if the target file located in a white dir.
Parameter	wszDir – Full path name
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

RemoveWhiteDir

ITEM	**CONTENT**
Declaration	DWORD RemoveWhiteDir(WCHAR* wszDir);

Function	Remove a white directory from system.
Parameter	wszDir – Full path name
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

4.10 Loading/Unloading process encryption policy

AddPolicy

ITEM	**CONTENT**
Declaration	DWORD AddPolicy(WCHAR* wszExeName, WCHAR* wszExtNameList, ULONG nFlags);
Function	Adding one process encryption policy.
Parameter	wszExeName– process name. example “NOTEPAD.EXE”. wszExtNameList– extension name list. “.TXT|.DOC|” , must be end with “|”. nFlags –PROC_FLAG_ENCRYPT_ON_OPEN
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	#define PROC_FLAG_ENCRYPT_ON_OPEN 0x00000002

CancelPolicy

ITEM	**CONTENT**
Declaration	DWORD CancelPolicy (WCHAR* wszExeName);
Function	Deleteing one process encryption policy.
Parameter	wszExeName– process name. example “NOTEPAD.EXE”.
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

4.11 Built in process encryption polices

Please reference these functions name with “load or unload” prefix in the ofetapi.h file.

5 ofetCtrl Module

5.1 Start/Stop Control Module

StartControlModule

ITEM	CONTENT
Declaration	DWORD StartControlModule ( );
Function	Start the control module and efsKrnl
Parameter	NONE
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

StopControlModule

ITEM	CONTENT
Declaration	DWORD StopControlModule( );
Function	Stop the control module and efsKrnl
Parameter	NONE
Return Value	If succeed, returns ERR_SUCCESS an appropriate error code on failure.
Comment	

5.2 Start/Stop event report

SetEventReportValue

ITEM	CONTENT
Declaration	BOOL SetEventReportValue(BOOL bValue);
Function	Start or stop event reporting
Parameter	bValue – 1. Start event reporting 0. Stop event reporting
Return Value	If succeed, returns TURE, FALSE on failure.
Comment	

GetEventReportValue

ITEM	CONTENT
Declaration	BOOL GetEventReportValue(BOOL bValue);
Function	Query the state of event reporting module
Parameter	bValue – 1. Event reporting has started 0. Event reporting has stopped
Return Value	If succeed, returns TURE, FALSE on failure.
Comment	

5.3 Start/Stop access control module

SetPrivilegeControlValue

ITEM	CONTENT
Declaration	BOOL SetPrivilegeControlValue(BOOL bValue);
Function	Start or stop access control module
Parameter	bValue – 1. Start access control module 0. Stop access control module
Return Value	If succeed, returns TURE, FALSE on failure.
Comment	

GetPrivilegeControlValue

ITEM	CONTENT
Declaration	BOOL GetPrivilegeControlValue(BOOL* bValue);
Function	Query the state of access control module
Parameter	bValue – 1. Access control module has started 0. Access control module has stopped
Return Value	If succeed, returns TURE, FALSE on failure.
Comment	

5.4 Start/Stop faked exe checking

SetFakedExeCheckValue

ITEM	CONTENT
Declaration	BOOL SetFakedExeCheckValue (BOOL bValue);
Function	Start or stop faked exe checking
Parameter	bValue – 1. Start faked exe checking 0. Stop faked exe checking
Return Value	If succeed, returns TURE, FALSE on failure.
Comment	

GetFakedExeCheckValue

ITEM	CONTENT
Declaration	BOOL GetFakedExeCheckValue (BOOL* bValue);
Function	Query the state of faked exe checking
Parameter	bValue – 1. faked exe checking has started 0. faked exe checking has stopped
Return Value	If succeed, returns TURE, FALSE on failure.
Comment	

6 Compiling and deployment

To compile ETEFS_Mini requires WDK 7600.16385.0 and Visual Studio 2010 or above. Open ETEFS_Mini.sln and compiles the UsingSDK project. Copy the Usingsdk.exe into program directory of ETEFS_Mini demo install package, typically it is located in “C:\Program Files\ETEFS_Mini ”.