Introduction

File system filter driver

A file system filter driver intercepts requests targeted at a file system or another file system filter driver. By intercepting the request before it reaches its intended target, the filter driver can extend or replace functionality provided by the original target of the request. It is developed primarily to allow the addition of new functionality beyond what is currently available.

File system monitor filter

File system monitor filter can monitor the file system activities on the fly. With file system monitor filter you can monitor the file activities on file system level, capture file open/create/replace, read/write, query/set file attribute/size/time security information, rename/delete, directory browsing and file close request. You can develop the software for the following purposes:

- Continuous data protection (CDP).
- Auditing.
- Access log.
- Journaling.

File system control filter

File system control filter can control the file activities, which you can intercept the file system call, modify its content before or after the request goes down to the file system, allow/deny/cancel its execution based on the filter rule. You can fully control file open/create/replace, read/write, query/set file attribute/size/time security information, rename/delete, directory browsing these Io requests. With file system control filter you can developer these kinds of software:

- Data protection.
- Security.

The rules to use of file system control filter

To use the file system control filter, you need to follow the following rules, or might cause the system deadlock.

1. *Avoid the re-entrance issue, don't generate any new I/O request which will cause the request comes to the control filter handler again.*
2. *Avoid using any file operations in buffered mode, open any file in the control filter handler with FILE_FLAG_NO_BUFFERING flag set.*
3. *Avoid asynchronous procedure calls.*
4. *Avoid any GUI (user interface) operations.*

File system encryption filter

File system encryption filter provides a comprehensive solution for transparent file level encryption. It allows developers to create transparent encryption products which it can encrypt or decrypt file on-the-fly. Our encryption engine uses a strong cryptographic algorithm called Rijndael (256-bit key), it is a high security algorithm created by Joan Daemen and Vincent Rijmen (Belgium). Rijndael is the new Advanced Encryption Standard (AES) chosen by the National Institute of Standards and Technology (NIST).

Supported Platforms

- Windows 2012 Server R2 (32bit, 64bit)
- Windows 2008 Server R2 (32bit, 64bit)
- Windows 7 (32bit,64bit)
- Windows 2008 Server (32bit, 64bit)
- Windows Vista (32bit,64bit)
- Windows 2003 Server(32bit,64bit)
- Windows XP(32bit,64bit)

Symbol Reference

Structures, Enums

Typedef enum FilterType

```
{
    FILE_SYSTEM_MONITOR                    = 0,
    FILE_SYSTEM_CONTROL                    = 1,
    FILE_SYSTEM_ENCRYPTION                 = 2,
    FILE_SYSTEM_CONTROL_ENCRYPTION         = 3,
    FILE_SYSTEM_MONITOR_ENCRYPTION         = 4,
};
```

Comments

FILE_SYSTEM_MONITOR filter is used to get file system notification after it was completed.

FILE_SYSTEM_CONTROL filter is used to control the I/O request before it is passed down to the file system or after it is completed.

FILE_SYSTEM_ENCRYPTION filter is used to encrypt or decrypt files on-the-fly.

FILE_SYSTEM_CONTROL_ENCRYPTION filter has both control filter and encryption filter features.

FILE_SYSTEM_MONITOR_ENCRYPTION filter has both monitor filter and encryption filter features.

```
typedef enum  MessageType
{
        PRE_CREATE                                          = 0x00000001,
        POST_CREATE                                         = 0x00000002,
        PRE_FASTIO_READ                                     = 0x00000004,
        POST_FASTIO_READ                                    = 0x00000008,
        PRE_CACHE_READ                                      = 0x00000010,
        POST_CACHE_READ                                     = 0x00000020,
        PRE_NOCACHE_READ                                    = 0x00000040,
        POST_NOCACHE_READ                                   = 0x00000080,
        PRE_PAGING_IO_READ                                  = 0x00000100,
        POST_PAGING_IO_READ                                 = 0x00000200,
        PRE_FASTIO_WRITE                                    = 0x00000400,
        POST_FASTIO_WRITE                                   = 0x00000800,
        PRE_CACHE_WRITE                                     = 0x00001000,
        POST_CACHE_WRITE                                    = 0x00002000,
        PRE_NOCACHE_WRITE                                   = 0x00004000,
        POST_NOCACHE_WRITE                                  = 0x00008000,
        PRE_PAGING_IO_WRITE                                 = 0x00010000,
        POST_PAGING_IO_WRITE                                = 0x00020000,
        PRE_QUERY_INFORMATION                               = 0x00040000,
        POST_QUERY_INFORMATION                              = 0x00080000,
        PRE_SET_INFORMATION                                 = 0x00100000,
        POST_SET_INFORMATION                                = 0x00200000,
        PRE_DIRECTORY                                       = 0x00400000,
        POST_DIRECTORY                                      = 0x00800000,
        PRE_QUERY_SECURITY                                  = 0x01000000,
        POST_QUERY_SECURITY                                 = 0x02000000,
        PRE_SET_SECURITY                                    = 0x04000000,
        POST_SET_SECURITY                                   = 0x08000000,
        PRE_CLEANUP                                         = 0x10000000,
        POST_CLEANUP                                        = 0x20000000,
        PRE_CLOSE                                           = 0x40000000,
        POST_CLOSE                                          = 0x80000000,

};
```

Members

PRE_CREATE

PRE_CREATE request is the create I/O request before it goes down to the file system.

POST_CREATE

POST_CREATE request is the create I/O request after it is completed by file system.

PRE_FASTIO_READ

PRE_FASTIO_READ is the read I/O request before it goes to the Cache Manager.

POST_FASTIO_READ

POST_FASTIO_READ is the read I/O request after it comes back from the Cache Manager.If the data is not in the Cache Manager,it will return false,and the I/O Manager will reissue a new request to the file system.

PRE_CACHE_READ

PRE_CACHE_READ is the read I/O request with data cache before it goes to the Cache Manager.

POST_CACHE_READ

POST_CACHE_READ is the read I/O request after it come back from Cache Manager.If the data is not in the Cache Manager, it will trigger a paging I/O read request and load the data from the storage to the Cache Manager.Normally you will see the paging I/O read request follows the cache read request.

PRE_NONCACHE_READ

PRE_NONCACHE_READ is the read I/O request without data cache before it goes to the file system.

POST_NONCACHE_READ

POST_NONCACHE_READ is the read I/O request after it comes back from the file system.The data won’t cache in the Cache Manager. You will see the noncache read request if you open a file and specify FILE_NO_INTERMEDIATE_BUFFERING.

PRE_PAGING_IO_READ

PRE_PAGING_IO_READ is the read I/O request before it goes to the file system.It is initiated by the virtual memory system in order to satisfy the needs of the demand paging system.

POST_PAGING_IO_READ

POST_PAGING_IO_READ is the read I/O request after it come back from file system.For memory mapping file open you will see this request without the cache read request, for example open file with notepad application.

PRE_FASTIO_WRITE

PRE_FASTIO_WRITE is the write I/O request before it writes to the Cache Manager.

POST_FASTIO_WRITE

POST_FASTIO_WRITE is the write I/O request after it wrote to the Cache Manager. Normally you will see the paging I/O write request follows the fast I/O write request.

PRE_CACHE_WRITE

PRE_CACHE_WRITE is the write I/O request with data cache before it writes to the Cache Manager.

POST_CACHE_WRITE

POST_CACHE_WRITE is the write I/O request after it wrote to the Cache Manager. Normally you will see the paging I/O write request follows the cache write request.

PRE_NONCACHE_WRITE

PRE_NONCACHE_WRITE is the write I/O request without data cache before it wrote to the storage by the file system.

POST_NONCACHE_WRITE

POST_NONCACHE_WRITE is the write I/O request after it comes back from the file system.The data won’t cache in the Cache Manager. You will see the noncache write request if you open a file and specify FILE_NO_INTERMEDIATE_BUFFERING.

PRE_PAGING_IO_WRITE

PRE_PAGING_IO_WRITE is the write I/O request on behalf of the Virtual Manager system before it writes to the storage by the file system.

POST_PAGING_IO_WRITE

`POST_PAGING_IO_WRITE is the write I/O request after it come back from file system.`

PRE_QUERY_INFORMATION

`PRE_QUERY_INFORMATION is the I/O request which retrives information for a given file before it goese down to the file system. The file information class tells the type of the information will be returned.`

POST_QUERY_INFORMATION

`POST_QUERY_INFORMATION is the I/O request which retrives information for a given file after it comes back from the file system. The file information class tells the type of the information will be returned.`

PRE_SET_INFORMATION

`PRE_SET_INFORMATION is the I/O request which set information for a given file before it goese down to the file system. The file information class tells the type of the information will be set.`

POST_SET_INFORMATION

`POST_SET_INFORMATION is the I/O request which set information for a given file after it comes back from the file system. The file information class tells the type of the information will be set.`

PRE_DIRECTORY

`PRE_DIRECTORY is the folder browsing I/O request before it goese down to the file system. It retrive various kinds of information about files in the given directory. The information class tells the type of information will be returned.`

POST_DIRECTORY

`POST_DIRECTORY is the folder browsing I/O request after it comes back from the file system. It retrive various kinds of information about files in the given directory. The information class tells the type of information will be returned.`

PRE_QUERY_SECURITY

PRE_QUERY_SECURITY is the query security request before it goes down to the file system. It will retrive the security descriptor for a given file. The security information tells the the type of the security descriptor.

POST_QUERY_SECURITY

POST_QUERY_SECURITY is the query security request after it comes back from the file system. It will retrive the security descriptor for a given file. The security information tells the the type of the security descriptor.

PRE_SET_SECURITY

PRE_SET_SECURITY is the set security request before it goes down to the file system. It will set the security state for a given file. The security information tells the the type of the security descriptor.

POST_SET_SECURITY

POST_SET_SECURITY is the set security request after it comes back from the file system. It will set the security state for a given file. The security information tells the the type of the security descriptor.

PRE_CLEANUP

PRE_CLEANUP is the cleanup request before it goes down to the file system. It indicates that the handle reference count on a file object has reached zero. In other words, all handles to the file object have been closed. Often it is sent when a user-mode application has called the Microsoft Win32 CloseHandle function on the last outstanding handle to a file object.

POST_CLEANUP

POST_QUERY_SECURITY is the cleanup request after it comes back from the file system.

PRE_CLOSE

PRE_CLOSE is the close request before it goese down to the file system. It indicates that the reference count on a file object has reached zero, usually because a file system driver or other kernel-mode component has called ObDereferenceObject on the file object. This request normally follows a cleanup request. However, this does not necessarily mean that the close request will be received immediately after the cleanup request.

POST_CLOSE

POST_CLOSE is the close request after it comes back from the file system.

Comments

Register the I/O request with the combination of the request type you want to monitor. For file system monitor filter, only post requests are affected.

typedef enum AccessFlag

```
{
  EXCLUDE_FILTER_RULE                        = 0X00000000,
  EXCLUDE_FILE_ACCESS                        = 0x00000001,
  REPARSE_FILE_OPEN                          = 0x00000002,
  HIDE_FILES_IN_DIRECTORY_BROWSING           = 0x00000004,
  FILE_ENCRYPTION_RULE                       = 0x00000008,
  ALLOW_OPEN_WTIH_ACCESS_SYSTEM_SECURITY     = 0x00000010,
  ALLOW_OPEN_WITH_READ_ACCESS                = 0x00000020,
  ALLOW_OPEN_WITH_WRITE_ACCESS               = 0x00000040,
  ALLOW_OPEN_WITH_CREATE_OR_OVERWRITE_ACCESS = 0x00000080,
  ALLOW_OPEN_WITH_DELETE_ACCESS              = 0x00000100,
  ALLOW_READ_ACCESS                          = 0x00000200,
  ALLOW_WRITE_ACCESS                         = 0x00000400,
  ALLOW_QUERY_INFORMATION_ACCESS             = 0x00000800,
  ALLOW_SET_INFORMATION                      = 0x00001000,
  ALLOW_FILE_RENAME                          = 0x00002000,
  ALLOW_FILE_DELETE                          = 0x00004000,
  ALLOW_FILE_SIZE_CHANGE                     = 0x00008000,
  ALLOW_QUERY_SECURITY_ACCESS                = 0x00010000,
  ALLOW_SET_SECURITY_ACCESS                  = 0x00020000,
  ALLOW_DIRECTORY_LIST_ACCESS                = 0x00040000,
  ALLOW_MAX_RIGHT_ACCESS                     = 0xfffffff0,

};
```

Members

EXCLUDE_FILTER_RULE

EXCLUDE_FILTER_RULE is the rule which bypass the files matched the FilterMask. It can`t combine to use with the other access flags. If a file matchs the exclude filter rule,the filter will bypass this file,you won`t get any Io request notification or control. If a file matches both the exclude filter rule and monitor rule, the exclude filter rule will be applied.

EXCLUDE_FILE_ACCESS

EXCLUDE_FILE_ACCESS is the flag indicates the filter will deny the access to the files which match the FilterMask.

REPARSE_FILE_OPEN

REPARSE_FILE_OPEN is the rule which reparses the file matched the FilterMask open to the other files which match the ReparseMask.

Example:

```
AddFilterRule(REPARSE_FILE_OPEN,L"c:\\test\\*",L"d:\\reparse\\*");
```

All the open request to the files in the folder c:\test will reparse to the files in the folder d:\reparse.

HIDE_FILES_IN_DIRECTORY_BROWSING

HIDE_FILES_IN_DIRECTORY_BROWSING is the flag let you hide the files in the managed folder when it matches the mask.

Example:

```
AddFilterRule(ALLOW_MAX_RIGHT_ACCESS|HIDE_FILES_IN_DIRECTORY_BROWSING,L"c:\\test\\*",L"*.txt");
```

When you browse the folder c:\test, all the files with extension “.txt” will be hidden.

ENCRYPTION_FILTER_RULE

ENCRYPTION_FILTER_RULE is the flag indicates the filter will encrypt the new created files which match the FilterMask.If the other flag were set, this flag is automatically enabled.

ALLOW_OPEN_WTIH_ACCESS_SYSTEM_SECURITY

ALLOW_OPEN_WTIH_ACCESS_SYSTEM_SECURITY is the flag indicates if you can open the file with the desired access with the ACCESS_SYSTEM_SECURITY set.

ALLOW_OPEN_WITH_READ_ACCESS

ALLOW_OPEN_WITH_READ_ACCESS is the flag indicates if you can open the file with read access.

ALLOW_OPEN_WITH_WRITE_ACCESS

ALLOW_OPEN_WITH_WRITE_ACCESS is the flag indicates if you can open the file with write access.

ALLOW_OPEN_WITH_CREATE_OR_OVERWRITE_ACCESS

ALLOW_OPEN_WITH_CREATE_OR_OVERWRITE_ACCESS is the flag indicates if you can open with create a new file or overwrite the exist file.

ALLOW_OPEN_WITH_DELETE_ACCESS

ALLOW_OPEN_WITH_DELETE_ACCESS is the flag indicates if you can open the file for deletion or rename access.

ALLOW_READ_ACCESS

ALLOW_READ_ACCESS is the flag indicates if you have the permission to read the file.

ALLOW_WRITE_ACCESS

ALLOW_WRITE_ACCESS is the flag indicates if you have the permission to write the file.

ALLOW_QUERY_INFORMATION_ACCESS

ALLOW_QUERY_INFORMATION_ACCESS is the flag indicates if you have the permission to query the file information.

ALLOW_SET_INFORMATION

ALLOW_SET_INFORMATION is the flag indicates if you have the permission to set the file information.

ALLOW_FILE_RENAME

ALLOW_FILE_RENAME is the flag indicates if you have the permission to rename the file. If the flag ALLOW_SET_INFORMATION is unset, the rename is blocked automatically.

ALLOW_FILE_DELETE

ALLOW_FILE_DELETE is the flag indicates if you have the permission to delete the file. If the flag ALLOW_SET_INFORMATION is unset, the deletion is blocked automatically.

ALLOW_FILE_SIZE_CHANGE

ALLOW_FILE_SIZE_CHANGE is the flag indicates if you have the permission to change the file size. If the flag ALLOW_SET_INFORMATION is unset, the file size chage is blocked automatically.

ALLOW_QUERY_SECURITY_ACCESS

ALLOW_QUERY_SECURITY_ACCESS is the flag indicates if you have the permission to query the file security.

ALLOW_SET_SECURITY_ACCESS

ALLOW_SET_SECURITY_ACCESS is the flag indicates if you have the permission to set the file security.

ALLOW_DIRECTORY_LIST_ACCESS

ALLOW_DIRECTORY_LIST_ACCESS is the flag indicates if you have the permission to browse the directory.

ALLOW_MAX_RIGHT_ACCESS

ALLOW_MAX_RIGHT_ACCESS indicates if you have the maximum access right to the file.

Comments

A accessFlag is associated to a filter rule, used to control the access to the files matched the FilterMask.

Typedef enum FilterStatus

```
{
     FILTER_MESSAGE_IS_DIRTY              = 0x00000001,
     FILTER_COMPLETE_PRE_OPERATION        = 0x00000002,
     FILTER_DATA_BUFFER_IS_UPDATED        = 0x00000004,
};
```

Members

FILTER_MESSAGE_IS_DIRTY

FILTER_MESSAGE_IS_DIRTY is the flag indicates the reply message was modified and needs to be processed in filter driver. Set this flag if you change the reply message.

FILTER_COMPLETE_PRE_OPERATION

FILTER_COMPLETE_PRE_OPERATION is the flag indicates the filter needs to complete this pre I/O request.Only set this flag with pre operation request when you don`t want the request goes down to the file system.

FILTER_DATA_BUFFER_IS_UPDATED

FILTER_DATA_BUFFER_IS_UPDATED is the flag indicates the data buffer of the reply message was updated.The filter will process this data buffer.

Comments

FitlerStatus is the status code which returns to the filter driver,it is for control filter only.It instructs the filter what process needs to be done.

typedef struct _MESSAGE_SEND_DATA

```
{
     ULONG                   MessageId;
     PVOID                   FileObject;
     PVOID                   FsContext;
     ULONG                   MessageType;
```

```
    ULONG               ProcessId;
    ULONG               ThreadId;
    LONGLONG            Offset;
    ULONG               Length;
    LONGLONG            FileSize;
    LONGLONG            TransactionTime;
    LONGLONG            CreationTime;
    LONGLONG            LastAccessTime;
    LONGLONG            LastWriteTime;
    ULONG               FileAttributes;
    ULONG               DesiredAccess;
    ULONG               Disposition;
    ULONG               ShareAccess;
    ULONG               CreateOptions;
    ULONG               CreateStatus;
    ULONG               InfoClass;
    ULONG               Status;
    ULONG               FileNameLength;
    WCHAR               FileName[MAX_FILE_NAME_LENGTH];
    ULONG               SidLength;
    UCHAR               Sid[MAX_SID_LENGTH];
    ULONG               DataBufferLength;
    UCHAR               DataBuffer[MAX_MESSAGE_SIZE];
    ULONG               VerificationNumber;

} MESSAGE_SEND_DATA, *PMESSAGE_SEND_DATA;
```

Members

MessageId
This is the sequential number of the transaction.

FileObject
The FileObject is the pointer to the file object,it is a unique number to every file open.

FsContext
The FsContext is the pointer to the file context,it is unique number to the same file.

MessageType
MessageType is the I/O request type for this transaction.

ProcessId
The ProcessId is the id of the process associated with the thread that originally requested the I/O operation.

ThreadId

The ThreadId is the id of thread which requested the I/O operation.

Offset

The Offset is the read or write offset.

Length

The Length is the length for read or write.

FileSize

The FileSize is the size of the file for this I/O request.

TransactionTime

The transaction time in UTC format of the request.

CreationTime

The creation time in UTC format of the file we are requesting.

LastAccessTime

The last access time in UTC format of the file we are requesting.

LastWriteTime

The last write time in UTC format of the file we are requesting.

FileAttributes

The file attributes of the file we are requesting.

DesiredAccess

The DesiredAccess is the request access to the file for the Create I/O request,which can be summarized as read,write,both or neither zero.For more information reference the Windows API CreateFile.

Disposition

The disposition is the action to take on a file that exist or does not exist. For more information reference the Windows API CreateFile.

SharedAccess

The SharedAccess is the requested sharing mode of the file which can be read,write,both,delete,all of these,or none. For more information reference the Windows API CreateFile.

CreateOptions

The CreateOptions specifies the options to be applied when creating or opening the file. For more information reference the Windows API CreateFile.

CreateStatus

The CreateStatus is the status after the Create I/O request completed.It could be the one of the following values:

```
FILE_SUPERSEDED = 0x00000000,
FILE_OPENED = 0x00000001,
FILE_CREATED = 0x00000002,
FILE_OVERWRITTEN = 0x00000003,
FILE_EXISTS = 0x00000004,
FILE_DOES_NOT_EXIST = 0x00000005,
```

InfoClass

The infoClss is the information class for query/set information I/O request, or directory browsing request. For query/set security request, it is the security information.For more information reference the windows Filter API FltQueryInformationFile, FltQueryDirectoryFile,FltQuerySecurityObject.

Status

The Status is the I/O status which returns from the file system,indicates if the I/O request succeeded.It is only meaningful to the post I/O requests.

FileNameLength

The file name length in byte of the file we are requesting.

FileName

The file name we are requesting.

SidLength

The length of the security identifier buffer in byte.

Sid

The buffer of the security identifier data.

DataBufferLength

The data buffer length for read, write, security, information,directory I/O requests.

DataBuffer

The The data buffer length for read, write, security, information, directory I/O requests.

VerificationNumber

The verification number to verify the data structure integerity.

Comments

The MESSAGE_SEND_DATA structure is used to transfer the data from kernel to the user mode application. It includes all the information needed for the user.

typedef struct _MESSAGE_REPLY_DATA

```
{
      ULONG           MessageId;
      ULONG           MessageType;
      ULONG           ReturnStatus;
      ULONG           FilterStatus;
      ULONG           DataBufferLength;
      UCHAR           DataBuffer[MAX_MESSAGE_SIZE];

} MESSAGE_REPLY_DATA, *PMESSAGE_REPLY_DATA;
```

Members

MessageId

This is the sequential number of the transaction.

MessageType

MessageType is the I/O request type for this transaction. Reference MessageType enum type.

ReturnStatus

The ReturnStatus is the I/O status which returns to filter driver, and filter will return this status to the user application for the request.

FilterStatus

The FitlerStatus is the status code which returns to the filter driver,it instructs the filter what process needs to be done. For more information reference the FilterStatus enum.

DataBufferLength

The data buffer length which returns to the filter driver.

DataBuffer

The data buffer which returns to the filter driver.

Comments

MESSAGE_REPLY_DATA is only for control filter, when it needs to change the data or status of the I/O request. To update the reply data buffer, you must understand the format of the buffer, incorrect data could cause your system unfunctional, even crash.

Types

typedef BOOL (__stdcall *Proto_Message_Callback)(

```
   IN     PMESSAGE_SEND_DATA  pSendMessage,
   IN OUT PMESSAGE_REPLY_DATA pReplyMessage)
```

Comments

This is the proto type of the message callback function. The function will be called when the registed I/O requests match the filter rule. The second parameter “pReplyMessage” is always NULL for the file system monitor filter.

typedef VOID (__stdcall *Proto_Disconnect_Callback)()

Comments

This is the proto type of disconnect function.The function will be called when the connection to the filter is disconnected.

Exported API

BOOL

InstallDriver()

Return Value

Return true if it succeeds, else return false.

Comments

Install the EaseFilter driver to the system.To install the driver you need the administrator permission.

BOOL

UnInstallDriver()

Return Value

Return true if it succeeds, else return false.

Comments

UnInstall the EaseFilter driver from the system. To UnInstall the driver you need the administrator permission.

BOOL

SetRegistrationKey(

IN WCHAR RegisterName,*
IN WCHAR RegisterKey)*

Parameters

RegisterName

Your register name.

RegisterKey

Your register key.

Return Value

Return true if it succeeds, else return false.

Comments

You have to set the registration key before you can start the filter.

BOOL

RegisterMessageCallback(

```
     ULONG ThreadCount,
     Proto_Message_Callback MessageCallback,
     Proto_Disconnect_Callback DisconnectCallback )
```

Parameters

ThreadCount

The number of threads used for connection to the filter.

MessageCallback

The message callback function for the registered I/O requests.

DisconnectCallback

The disconnect callback function when the connection is disconnected.

Return Value

Return true if it succeeds, else return false.

Comments

RegisterMessageCallback is the first API you need to call, it is the API start the filter and create the connection to the filter.

VOID

Disconnect()

Comments

Disconnect is the API when you want to stop filter and filter connection.

BOOL

GetLastErrorMessage(WCHAR Buffer, PULONG BufferLength)*

Parameters

Buffer

This the pointer of the buffer to receive the last error message.

BufferLength

The length of the buffer.

Return Value

Return true if it succeeds,else return false if the buffer length is not big enough to contain the message,and the BufferLength is set with the right size needed.

Comments

This API is called right after if the other API is failed. It will return the error message.

BOOL

ResetConfigData();

Return Value

Return true if it succeeds, else return false.

Comments

ResetConfigData is the API reset all the configuration of the filter, it will clear up all the setting includes the filter rules.

BOOL

SetFilterType(ULONG FilterType)

Parameters

FilterType

The type of the filter you want to set. There are FILE_SYSTEM_MONITOR filter and FILE_SYSTEM_CONTROL filter.

Return Value

Return true if it succeeds, else return false.

Comments

The default filter type is file system monitor filter.

BOOL

SetConnectionTimeout(ULONG TimeOutInSeconds)

Parameters

TimeOutInSeconds

The value of the filter wait time out.

Return Value

Return true if it succeeds, else return false.

Comments

This is the maixmum time for the filter driver wait for the response from user mode, the user mode application should return as fast as possible, or it will block the system requests.Set it bigger if your application needs to process with more time.

BOOL

AddFilterRule(

```
     IN   ULONG* AccessFlag,
     IN   WCHAR* FilterMask,
     IN   WCHAR* ReparseMask,
     IN   ULONG  KeyLength,
     IN   UCHAR* Key )
```

Parameters

AccessFlag

The AccessFlag of this filter rule.

FilterMask

The FilterMask set the monitor folder or files.The mask is dos format,it can include wild character '*'or '?'. For example:

C:\test\*txt
The filter only monitor the files end with 'txt' in the folder c:\test.

ReparseMask

Set the reparse folder mask when the AccessFlag is REPARSE_FILE_OPEN.It can include the wild character, but it must match the wild character in FilterMask.

For example:
FilterMask = c:\test\*txt
ReparseMask = d:\reparse\*doc

If you open file c:\test\MyTest.txt, it will reparse to the file d:\reparse\MyTest.doc

KeyLength

The length of the encryption key,if AccessFlag includes encryption filter rule,it has to set this length to 16,24 or 32.

Key

The encryption key for encryption filter rule if it was set.

Return Value

Return true if it succeeds, else return false.

Comments

AddFilterRule is the API to setup the filter rule,You can set up multiple filte rules, the FilterMask must be different, if the FilterMask is the same, it will overwrite the previous one.

BOOL

RemoveFilterRule(WCHAR FilterMask);*

Parameters

FilterMask

The FilterMask associated to the filter rule.

Return Value

Return true if it succeeds, else return false.

Comments

You can remove the filter rule which was set by AddFilterRule API.

BOOL

AddExcludedProcessId(ULONG ProcessId)

Parameters

ProcessId
The process Id you want to be excluded by filter.

Return Value
Return true if it succeeds, else return false.

Comments

This API let you can bypass the filter for specific processes, you can add multiple process Id.

BOOL

RemoveExcludeProcessId(ULONG ProcessId)

Parameters

ProcessId
The process Id you want to remove which set by AddExcludedProcessId API.

Return Value
Return true if it succeeds, else return false.

Comments

This API remove previous set excluded process Id from filter.

BOOL

RegisterIoRequest(ULONG RequestRegistration)

Parameters

RequestRegistration
The RequestRegistration is the bit combination of the request type.

Return Value

Return true if it succeeds, else return false.

Comments

Register the I/O requests which you want to monitor. For File_SYSTEM_MONITOR filter, only post I/O requests registration are affected, since it only can get notification after the request was completed by file system.

For FILE_SYSTEM_CONTROL filter you can register both pre and post reqeusts. If you want to deny, cancel or return with your own data instead of going down to the file system, you need to register the pre request.

For some post I/O requests, you can’t cancel or deny it, for example Create, Set information,Set security, Write requests.

BOOL

GetFileHandleInFilter(WCHAR FileName, ULONG DesiredAccess, Handle* FileHandle);*

Parameters

FileName

The full path of the file which you want to open.

DesiredAccess

The requested access to the file or device, which can be summarized as read, write, both or neither zero).

FileHandle

The pointer to the file handle which will receive the file handle after the file was opened.

Return Value

Return true if it succeeds, else return false.

Comments

Use this API to open the file,it will bypass the filter, avoid reentrant issue.It also will bypass the security check.Close the handle with CloseHandle win32 API.

BOOL

AESEncryptFile(

```
IN   WCHAR* FileName,
IN   ULONG  KeyLength,
IN   UCHAR* Key,
IN   ULONG  IVLength,
IN   UCHAR* IV,
IN   BOOL   AddIVTag )
```

Parameters

FileName

The file name to be encrypted.

KeyLength

The encryption key length,it has to be 16(128bits),24(192bits) or 32(256bits).

Key

The encryption key,it is an unsigned char array with KeyLength size.

IVLength

The initial vector length,if it is 0, the sysem will allocate an unique IV for the file.

IV

The initial vector,when IVLenght is 0, it sets to NULL.

AddIVTag

If it is true,it will add the IV to the encrypted file as reparse point tag,then the encryption filter driver can recognize this encrypted file.

Return Value

Return true if it succeeds, else return false.

Comments

AESEncryptfile is the API to encrypt file file with AES encryption cryptographic algorithm.

BOOL

AESDecryptFile(

```
IN   WCHAR* FileName,
IN   ULONG  KeyLength,
IN   UCHAR* Key,
IN   ULONG  IVLength,
IN   UCHAR* IV )
```

Parameters

FileName

The file name to be decrypted.

KeyLength

The encryption key length,it has to be 16(128bits),24(192bits) or 32(256bits).

Key

The encryption key,it is an unsigned char array with KeyLength size.

IVLength

The initial vector length,if the encrypted file already has IVTag,it will use the IV tag instead of the pass in IV, if the encrypted file doesn't set the IV tag,then the IVLength can't be 0, and IV can't be NULL.

IV

The initial vector,when the encrypted file doesn't set IV tag, the IV can't be NULL, or it can be NULL.

Return Value

Return true if it succeeds, else return false.

Comments

AESDecryptfile is the API to decrypt file file with AES encryption cryptographic algorithm.

BOOL

AddIVTag(

```
IN   WCHAR* FileName,
IN   ULONG  IVLength,
IN   UCHAR* IV )
```

Parameters

FileName

The file name was encrypted.

IVLength

The initial vector length.

IV

The initial vector.

Return Value

Return true if it succeeds, else return false.

Comments

AddIVTag is the API to add the IV tag to the encrypted file if it doesn't have the iv tag set, or it will return false.

BOOL

GetIVTag(

```
IN          WCHAR* FileName,
IN Out      ULONG* IVLength,
IN out      UCHAR* IV )
```

Parameters

FileName

The file name was encrypted.

IVLength

The pointer to the initial vector length,the iv length always is 16,it has to be 16,it will return 0 if the file is not encrypted.

IV

The pointer to the buffer to receive the initial vector.

Return Value

Return true if it succeeds, else return false.

Comments

GetIVTag is the API to get the IV tag from the encrypted file if it has the iv tag set, or IVLength will return 0.

BOOL

DeleteIVTag(

```
      IN         WCHAR* FileName )
```

Parameters

FileName

    The file name was encrypted.

Return Value

    Return true if it succeeds, else return false.

Comments

GetIVTag is the API to delete the IV tag from the encrypted file if it has the iv tag set, or it will return true.

How to use

The components

The EaseFilter file system filter SDK includes two components (EaseFlt.sys and FilterAPI.dll), The EaseFlt.sys and FilterAPI.dll are different for 32bit and 64bit windows system. EaseFlt.sys is the file system filter driver which implements all the functionalities in the file system level. FilterAPI.dll is a wrapper DLL which exports the API to the user mode applications.

To check the binary is 32 bit or 64 bit you can right click file and go to the property, then go to the “Details” tag and check the “file description” section .

Set up the filter

Install the filter driver with InstallDriver() method if the driver has not been installed yet. After filter driver was installed, the filter was loaded, if not you can load the filter with command “Fltmc load EaseFlt” in dos prompt. To remove the filter driver from the system, call UninstallDriver() method.

Start the filter

1. Activate the filter with API SetRegistrationKey(). You can request the trial license key with the link: http://www.easefilter.com/Order.htm   or email us info@easefilter.com
2. After register the callback function with API RegisterMessageCallback, filter is started.

   BOOL ret = RegisterMessageCallback(FilterConnectionThreadsCount, MessageCallback, DisconnectCallback);

3. Setup the filter configuration after filter was started. First select the filter type, then add filter rule and register the I/O request:

   BOOL ret = SetFilterType(FILE_SYSTEM_MONITOR);
   BOOL ret = AddFilterRule(L"C:\\MyMonitorFolder");*

   BOOL ret = RegisterIORequest(POST_CREATE|POST_CLEANUP);

We provide C++ example and C# example to demonstrate how to use the EaseFilter File System Monitor and Control Filter.

C++ Example

Copy the correct version (32bit or 64bit) EaseFlt.sys , FilterAPI.DLL ,FilterAPI.h and FilterAPI.lib to your folder. FilterAPI.h file includes all the functions and structures used for connecting to the filter driver. WinDataStructures.h file is part of the structures of windows API which is used in the example, for more structures please reference Microsoft MSDN website.

For monitor filter, it will only display the file system call messages which include process Id, Thread Id, file name, user name, file system I/O type , etc.

For Control filter, the filter will block and wait for the response if that I/O was registered, so it is better handle this request as soon as possible, or it will block the system call.

C# Example

Copy the correct version (32bit or 64bit) EaseFlt.sys , FilterAPI.DLL and ,EaseFilter.cs to your folder. EaseFilter.cs has the structures and APIs used for connecting to the filter driver.