

## 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 by which user and process name. 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 I/O requests. With file system control filter, you can develop these kinds of software:

- Data protection.
- File Screening
- Access Control.
- File 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.

# EaseFilter File System Filter SDK Manual

---

## 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, on-access, per-file 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

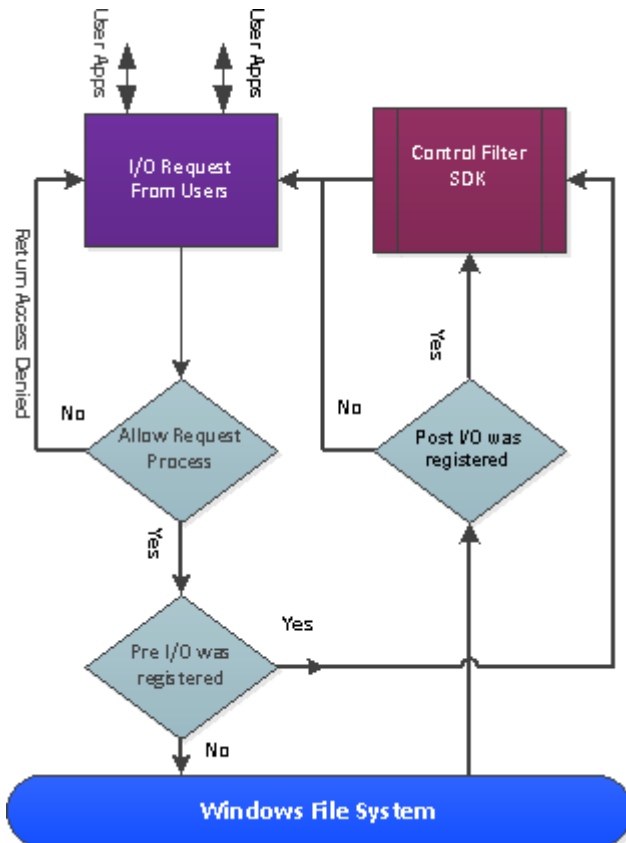
- Windows 2016
- Windows 10 (32bit, 64bit)
- Windows 8 (32bit, 64bit)
- Windows 2012 Server R2
- Windows 2008 Server R2 (32bit, 64bit)
- Windows 7 (32bit,64bit)
- Windows 2008 Server (32bit, 64bit)
- Windows Vista (32bit,64bit)
- Windows 2003 Server(32bit,64bit)
- Windows XP(32bit,64bit)

## Overview and Basic Concepts

### The EaseFilter file system filter driver

EaseFilter file system filter driver is a kernel component module which is sitting on the layer between the I/O manager and the file system. When a user application invokes a win32 API to a file, the filter driver can intercept this I/O, based on the policies was set with the filter rule, the I/O information can be sent to the user, or be modified/blocked the access based on the setting as below figure.

# EaseFilter File System Filter SDK Manual



## Filter Rule

A filter rule is the policy for the filter driver, it tells the filter driver how to manage the files. To manage the files by filter driver, you need to create at least one filter rule, you can have multiple filter rules with different policies to manage different files to meet your requirement.

A filter rule has only one unique include file mask, when a new filter rule was added to the filter driver, if a filter rule with the same include file filter mask exists, the new filter rule will replace the older one.

A filter rule has a AccessFlags to control if the file access can be proceeded, for Monitor filter driver, it always should be set with the maximum rights. If it is 0, it means this is a exclude filter rule, all the files match the include filter mask will be skipped by this filter rule. Exclude filter rules always are sitting on the top of the other filter rules.

A filter rule can register the file events (reference FileEventType), when it the file I/O events were triggered, the filter driver will send the file events message back to the user mode service.

A filter rule can have multiple exclude file filter masks, it is optional. When a file was opened, if the file name matches the exclude filter mask, the filter driver will skip this file I/O.

# EaseFilter File System Filter SDK Manual

---

A filter rule can have multiple include process names, process Ids, exclude process names, process Ids, include user names and exclude user names. These are optional setting.

If a filter rule has the flag `ENCRYPTION_FILTER_RULE` enabled, the encrypted data can be decrypted transparently in memory by filter driver when it was read, and the new data will be encrypted and be written to the disk when the encrypted file was written. If the flag `ALLOW_NEW_FILE_ENCRYPTION` is enabled, the new created file will be encrypted automatically.

If a filter rule has the flag `HIDE_FILES_IN_DIRECTORY_BROWSING` enabled, the filter rule can add multiple hidden file filter mask, when user browses the folder, if the file names match the hidden filter mask, they will be hidden from the user.

A filter rule only can have one unique include file mask, when a new filter rule was added to the filter driver, if a filter rule with the same include file mask already exists, then the new filter rule will replace the older one. A filter rule can have multiple exclude file masks, multiple include process names and exclude process names, multiple include process Ids and exclude process Ids, multiple include user names and exclude user names.

## How to manage the file I/O with the filter rule

When an file I/O occurs, the filter driver will check the filter rule policies with the below order to decide if it should manage this file I/O or not:

- a. If the exclude filter rules (`AccessFlags` is 0) exist, the filter driver will look for all the exclude filter rules first, if the files match the include file filter mask of the exclude filter rule, the filter driver will skip this file I/O, or go to next step.
- b. Check the next filter rule, If the file name matches the include file filter mask, then go to next step, or continue step b, till no more filter rules.
- c. Check if the exclude file filter mask list is empty, if it is not empty, then it will check if the file name matches the exclude file mask, if it matches, then go to step b, or go to next step.
- d. Check if the include process name and include process Id list are empty, if they are not empty, then check if the current process name or process Id is in the include list, if not then go to step b, or go to next step.
- e. Check if the exclude process name and exclude process Id list are empty, if they are not empty, then check if the current process name or process Id is in the list, if yes then go to step b, or go to next step.
- f. Check if the include user names list is empty, if it is not empty, then check if the current user name is in the list, if not then go to the step b, or go to next step.
- g. Check if the exclude user names list is empty, if it is not empty, then check if the current user name is in the list, if yes then go to the step b, or the filter driver will manage this file I/O with this filter rule.

If the files match multiple filter rules, the filter driver will choose the deepest one.

For example:

# EaseFilter File System Filter SDK Manual

---

There are filter rule1 with include file mask c:\test\\*;  
filter rule2 with include file mask c:\test\test2\\*;  
filter rule3 with include file mask c:\test\test2\test3\\*

the checking order will be rule3, rule2, then rule1. If you have a file with name c:\test\test2\test.txt, the filter driver will choose filter rule 2.

## Control filter driver filter rule

For control filter driver, you not only have the same filter rule setting as the monitor filter, but also have the following extra settings:

- a. Access control of the filter rule.
- b. Access control for specific processes.
- c. Access control for specific users.
- d. Reparse file open filter rule.
- e. Hidden file filter rule.

## How to manage the file I/O with the IO registration

### *Pre-IO and Post IO*

A file IO has two steps, pre-IO and post IO. A pre-IO is the I/O process in the filter driver layer, it didn't go down to the file system, a post IO is the I/O process in the filter driver, it already went down to the file system and came back to the filter driver.

You can register the I/O message types (reference Message Type ) you are interested, it is a global setting. When a file I/O occurs, if a file name matches a filter rule, and the file I/O was registered, the filter driver will send the I/O message to the user.

A monitor filter driver only can register the post IOs, it means it only can monitor the file I/O result, if a post IO was triggered, the filter driver will send the I/O message to the user and return back right away and won't block and wait.

A control filter driver can register all the IOs, it has the ability to control the file I/O behaviour, it can modify, allow, deny or cancel the file IO and data. When a file IO was triggered, the filter driver will send the I/O message to the user, it will block and wait for the result from the user, the filter driver will go to modify the file I/O and data or not based on the return result.

### *An I/O message*

A typical I/O message includes the file I/O type, file name, file information (file size, file attribute, file time), user name, process name and the I/O result for the post I/O.

## 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,  
    FILE_SYSTEM_EASE_FILTER_ALL    = 5,  
    FILE_SYSTEM_MONITOR_CONTROL    = 8,  
};
```

#### **Comments**

FILE\_SYSTEM\_MONITOR filter type is the file system filter driver which only intercept the file I/O notification after it was completed.

FILE\_SYSTEM\_CONTROL filter type is the file system filter driver which can control the file I/O request's behaviour before it goes down to the file system or after it is completed by the file system.

FILE\_SYSTEM\_ENCRYPTION filter type is the file system filter driver which can encrypt and decrypt the files on-the-fly.

FILE\_SYSTEM\_CONTROL\_ENCRYPTION filter type includes both control filter and encryption filter driver.

FILE\_SYSTEM\_MONITOR\_ENCRYPTION filter type includes both monitor filter and encryption filter driver.

FILE\_SYSTEM\_EASE\_FILTER\_ALL filter type includes monitor filter, control filter and encryption filter driver.

FILE\_SYSTEM\_MONITOR\_CONTROL filter type includes both monitor filter and control filter driver.

#### *typedef enum FileEventType*

```
{  
    FILE_WAS_CREATED           = 0x00000020,  
    FILE_WAS_WRITTEN           = 0x00000040,
```

# EaseFilter File System Filter SDK Manual

---

```
FILE_WAS_RENAMED           = 0x00000080,  
FILE_WAS_DELETED           = 0x00000100,  
FILE_SECURITY_CHANGED      = 0x00000200,  
FILE_INFO_CHANGED         = 0x00000400,  
FILE_WAS_READ             = 0x00000800,  
};
```

## Members

### **FILE\_WAS\_CREATED**

The new file was created event.

### **FILE\_WAS\_WRITTEN**

The file was written with data event.

### **FILE\_WAS\_RENAMED**

The file was renamed event.

### **FILE\_SECURITY\_CHANGED**

The file security was changed event.

### **FILE\_INFO\_CHANGED**

The file information was changed event.

### **FILE\_WAS\_READ**

The file data was read event.

## Comments

This is the registered event types of the filter rule, monitor the file events which only happens in the filter rule.

## *typedef enum MessageType*

```
{  
    PRE_CREATE           = 0x00000001,  
    POST_CREATE          = 0x00000002,  
    PRE_FASTIO_READ     = 0x00000004,  
    POST_FASTIO_READ    = 0x00000008,  
};
```

# EaseFilter File System Filter SDK Manual

---

```
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.



# EaseFilter File System Filter SDK Manual

---

## **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**

# EaseFilter File System Filter SDK Manual

---

`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 retrieves information for a given file before it goes down to the file system. The file information class tells the type of the information will be returned.

# EaseFilter File System Filter SDK Manual

---

## **POST\_QUERY\_INFORMATION**

POST\_QUERY\_INFORMATION is the I/O request which retrieves 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 goes 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 goes down to the file system. It retrieve 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 retrieve 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 retrieve 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 retrieve the

# EaseFilter File System Filter SDK Manual

---

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 goes 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.

# EaseFilter File System Filter SDK Manual

---

## 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_WITH_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_FILE_ACCESS_FROM_NETWORK = 0x00080000,
    ALLOW_NEW_FILE_ENCRYPTION    = 0x00100000,
    ALLOW_ALL_SAVE_AS            = 0x00200000,
    ALLOW_COPY_PROTECTED_FILES_OUT = 0x00400000,
    ALLOW_FILE_MEMORY_MAPPED     = 0x00800000,
    ALLOW_MAX_RIGHT_ACCESS       = 0xffffffff,
};
```

## 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 matches the exclude filter rule, the filter will bypass this file, you won't get any Io request notification or control. If a file

# EaseFilter File System Filter SDK Manual

---

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\_WITH\_ACCESS\_SYSTEM\_SECURITY**

# EaseFilter File System Filter SDK Manual

---

`ALLOW_OPEN_WITH_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**

# EaseFilter File System Filter SDK Manual

---

`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 change 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\_FILE\_ACCESS\_FROM\_NETWORK**

`ALLOW_FILE_ACCESS_FROM_NETWORK` is the flag indicates if you have the permission to access the files from the network server.

## **ALLOW\_NEW\_FILE\_ENCRYPTION**

`ALLOW_NEW_FILE_ENCRYPTION` is the flag indicates if you allow to encrypt the new created files by the filter driver.



# EaseFilter File System Filter SDK Manual

---

## **ALLOW\_ALL\_SAVE\_AS**

ALLOW\_ALL\_SAVE\_AS is the flag indicates if the process has the permission to create a new file.

## **ALLOW\_COPY\_PROTECTED\_FILES\_OUT**

ALLOW\_COPY\_PROTECTED\_FILES\_OUT is the flag indicates if allow the protected files being copy out of the protected folder, if allow\_all\_save\_as is true.

## **ALLOW\_FILE\_MEMORY\_MAPPED**

ALLOW\_FILE\_MEMORY\_MAPPED indicates if the file can be opened with memory mapped, a file execution must be opened with memory mapped.

## **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 AESFlags***

```
{
    Flags_Enabled_Expire_Time           = 0x00000010,
    Flags_Enabled_Check_ProcessName     = 0x00000020,
    Flags_Enabled_Check_UserName        = 0x00000040,
    Flags_Enabled_Check_AccessFlags     = 0x00000080,
    Flags_Enabled_Check_User_Permit     = 0x00000100,
    Flags_AES_Key_Was_Embedded          = 0x00000200,
    Flags_Enabled_Request_IV_And_Key    = 0x00000400,
    Flags_Enabled_Revoke_Access_Control = 0x00000800,
    Flags_Enabled_Check_Computer_Id     = 0x00001000,
    Flags_Enabled_Check_User_Password   = 0x00002000,
};
```

## **Members**

**Flags\_Enabled\_Expire\_Time**

# EaseFilter File System Filter SDK Manual

---

`Flags_Enabled_Expire_Time` is the flag which indicates that the filter driver will check if the encrypted file was expired before it can be opened.

## **Flags\_Enabled\_Check\_ProcessName**

`Flags_Enabled_Check_ProcessName` is the flag which indicates that the filter driver will check if the process has the permission to open the encrypted file.

## **Flags\_Enabled\_Check\_UserName**

`Flags_Enabled_Check_UserName` is the flag which indicates that the filter driver will check if the user has the permission to open the encrypted file.

## **Flags\_Enabled\_Check\_AccessFlags**

`Flags_Enabled_Check_AccessFlags` is the flag which indicates that the filter driver will check the access flags if the encrypted file can be opened.

## **Flags\_Enabled\_Check\_User\_Permit**

`Flags_Enabled_Check_User_Permit` is the flag which indicates that the filter driver will require the user permission before the encrypted file can be opened.

## **Flags\_AES\_KEY\_WAS\_EMBEDDED**

`Flags_AES_KEY_WAS_EMBEDDED` is the flag which indicates that the filter driver will use the embedded AES key to decrypt the file.

## **Flags\_Enabled\_Request\_IV\_And\_Key**

`Flags_Enabled_Request_IV_And_Key` is the flag which indicates that the filter driver will require encryption key and IV from user mode service to decrypt the file.

## **Flags\_Enabled\_Revoke\_Access\_Control**

`Flags_Enabled_Revoke_Access_Control` is the flag which indicates that the filter driver will require the permission from the control server and get the encryption key and IV to decrypt the file.

# EaseFilter File System Filter SDK Manual

---

## **Flags\_Enabled\_Check\_Computer\_Id**

Flags\_Enabled\_Check\_Computer\_Id is the flag which indicates that the filter driver will check if the computer can access the encrypted files.

## **Flags\_Enabled\_Check\_User\_Password**

Flags\_Enabled\_Check\_User\_Password is the flag which indicates that the filter driver will require the user password before the encrypted file can be opened.

## **Comments**

A AESFlags is a flag inside the metadata of the encrypted file.

## ***typedef struct \_EASETAG\_AES\_DATA***

```
{
    ULONG      EaseTagKey;
    ULONG      AESFlags;
    ULONG      IVLength;
    UCHAR      IV[16];
    ULONG      EncryptionKeyLength;
    UCHAR      EncryptionKey[32];
    LONGLONG   CreationTime;
    LONGLONG   ExpireTime;
    ULONG      AccessFlags;
    LONGLONG   FileSize;
    ULONG      LengthOfIncludeProcessNames;
    ULONG      OffsetOfIncludeProcessNames;
    ULONG      LengthOfExcludeProcessNames;
    ULONG      OffsetOfExcludeProcessNames;
    ULONG      LengthOfIncludeUserNames;
    ULONG      OffsetOfIncludeUserNames;
    ULONG      LengthOfExcludeUserNames;
    ULONG      OffsetOfExcludeUserNames;
    ULONG      LengthOfAccountName;
    ULONG      OffsetOfAccountName;
    ULONG      LengthOfComputerId;
    ULONG      OffsetOfComputerId;
    ULONG      LengthOfUserPassword;
    ULONG      OffsetOfUserPassword;
    //the data store here.
    //IncludeProcessNames;
    //ExcludeProcessNames;
    //IncludeUserNames;
```

# EaseFilter File System Filter SDK Manual

---

```
//ExcludeUserNames;  
//AccountName;  
//ComputerId;  
//UserPassword;  
ULONG      TotalSizeOfAESData;  
  
}
```

## Members

### EaseTagKey

The verification key of the AES data structure, the value is 0xccb76e80.

### AESFlags

The Flags of the encrypted file's meta data, indicates the filter driver what action needs to do for the encrypted file opens.

### IVLength

The length of the initialization vector, it should be 16.

### IV

The 16 bytes initialization vector, is an arbitrary number that can be used along with an encryption key for the file encryption.

### EncryptionKeyLength

The length of the encryption key, it should be 16,24 or 32.

### EncryptionKey

The encryption key for the file encryption.

### CreationTime

The date and time of the encrypted file was created.

### ExpireTime

The expiration time of the encrypted file.

### Accesslags

# EaseFilter File System Filter SDK Manual

---

The access flags of the encrypted file, control the access of the encrypted file if `Flags_Enabled_Check_AccessFlags` is enabled, reference the `AccessFlag` enumeration.

## **FileSize**

The file size of the encrypted file.

## **LengthOfIncludeProcessNames**

The length of the include process name string,if the flag `Flags_Enabled_Check_ProcessName` is enabled,only the included processes which were stored the include process name buffer can access the encrypted files.

## **OffsetOfIncludeProcessNames**

The offset of the include process name string buffer from the beginning of the structure.

## **LengthOfExcludeProcessNames**

The length of the exclude process name string,if the flag `Flags_Enabled_Check_ProcessName` is enabled,only the excluded processes which were stored the exclude process name buffer can't access the encrypted files.

## **OffsetOfExcludeProcessNames**

The offset of the exclude process name string buffer from the beginning of the structure.

## **LengthOfIncludeUserNames**

The length of the include user name string,if the flag `Flags_Enabled_Check_UserName` is enabled,only the included users which were stored the include user name buffer can access the encrypted files.

## **OffsetOfIncludeProcessNames**

The offset of the include user name string buffer from the beginning of the structure.

## **LengthOfExcludeUserNames**

# EaseFilter File System Filter SDK Manual

---

The length of the exclude user name string, if the flag `Flags_Enabled_Check_UserName` is enabled, only the excluded users which were stored in the exclude user name buffer can't access the encrypted files.

## **OffsetOfExcludeUserNames**

The offset of the exclude user name string buffer from the beginning of the structure.

## **LengthOfAccountName**

The length of the account name in the server which identifies the user.

## **OffsetOfAccountName**

The offset of the account name buffer from the beginning of the structure.

## **LengthOfComputerId**

The length of the computer Id string, if the flag `Flags_Enabled_Check_Computer_Id` is enabled, only the computers that match the computer Id string can access the encrypted file.

## **OffsetOfComputerId**

The offset of the computer Id string buffer from the beginning of the structure.

## **LengthOfUserPassword**

The length of the user password, if the flag `Flags_Enabled_Check_Password` is enabled, it requires the user to input the password. The filter driver can open the encrypted file only when the correct password was input.

## **OffsetOfUserPassword**

The offset of the user password buffer from the beginning of the structure.

## **IncludeProcessNames**

# EaseFilter File System Filter SDK Manual

---

The buffer of the include process name string buffer if the `LengthOfIncludeProcessNames` is greater than 0, the multiple process names are separated with character ``;'``.

## **ExcludeProcessNames**

The buffer of the exclude process name string buffer if the `LengthOfExcludeProcessNames` is greater than 0, the multiple process names are separated with character ``;'``.

## **IncludeUserNames**

The buffer of the include user name string buffer if the `LengthOfIncludeUserNames` is greater than 0, the multiple user names are separated with character ``;'``.

## **ExcludeUserNames**

The buffer of the exclude user name string buffer if the `LengthOfExcludeUserNames` is greater than 0, the multiple user names are separated with character ``;'``.

## **AccountName**

The buffer of the account name string buffer if the `LengthOfAccountName` is greater than 0.

## **ComputerIds**

The buffer of the computer Id string buffer if the `LengthOfComputerId` is greater than 0, the multiple computer Ids are separated with character ``;'``.

## **UserPassword**

The buffer of the password buffer if the `LengthOfUserPassword` is greater than 0.

## **TotalSizeOfAESData**

The total size of the structure `EASETAG_AES_DATA`.

## **Comments**

`EASETAG_AES_DATA` structure is the metadata of the encrypted file, includes the access control information if `Flags_Enabled_Revoke_Access_Control` is not enabled, or the access control information was stored in the remote server.

## *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**

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

## *typedef struct \_MESSAGE\_SEND\_DATA*

```
{  
    ULONG           MessageId;  
    PVOID           FileObject;  
    PVOID           FsContext;  
    ULONG           MessageType;  
    ULONG           ProcessId;  
    ULONG           ThreadId;  
    LONGLONG        Offset;
```



# EaseFilter File System Filter SDK Manual

---

```
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 a 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

# EaseFilter File System Filter SDK Manual

---

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.

# EaseFilter File System Filter SDK Manual

---

## **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 infoClass 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.

# EaseFilter File System Filter SDK Manual

---

## **DataBuffer**

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

## **VerificationNumber**

The verification number to verify the data structure integrity.

## **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;
    union
    {
        Struct{
            ULONG          DataBufferLength;
            UCHAR          DataBuffer[MAX_MESSAGE_SIZE];
        }Data;

        Struct{
            ULONG          AESDataLength;
            UCHAR          IV[16];
            ULONG          EncryptionKeyLength;
            UCHAR          EncryptionKey[1];
        }AESData;

        Struct{
            ULONG          UserNameLength;
            UCHAR          UserName[1];
        }UserInfo;

        Struct{
            ULONG          FileNameLength;
            UCHAR          FileName[1];
        }FileInfo;
    }ReplyData
}
```

# EaseFilter File System Filter SDK Manual

---

```
} 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 FilterStatus 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.

### AESDataLength

The total length of the AESData.

### IV

The 16 bytes initialization vector returns to filter driver.

### EncryptionKeyLength

The length of the encryption key.

### EncryptionKey

The encryption key returns to the filter driver.

### UserNameLength

The length of the user name.

### UserName

# EaseFilter File System Filter SDK Manual

---

The user name buffer.

## **FileNameLength**

The length of the file name.

## **FileName**

The file name buffer.

## **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 registered 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.

# EaseFilter File System Filter SDK Manual

---

**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**



# EaseFilter File System Filter SDK Manual

---

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.

# EaseFilter File System Filter SDK Manual

---

**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 maximum 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**

## ***AddNewFilterRule(***

*IN*    *ULONG*   *AccessFlag,*  
*IN*    *WCHAR\** *FilterMask*

**)**

### **Parameters**

#### **AccessFlag**

The AccessFlag of this filter rule.

#### **FilterMask**

The FilterMask set the target 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.

#### **Return Value**

Return true if it succeeds, else return false.

### **Comments**

# EaseFilter File System Filter SDK Manual

---

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

**BOOL**

## **RegisterEventTypeToFilterRule(**

```
    IN    WCHAR* FilterMask,  
    IN    ULONG  EventType  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

#### **EventType**

The event types were registered to the filter rule, were used to monitor the file events.

### **Comments**

If you want to monitor the file events for the filter rule, this is the API to register the event types.

**BOOL**

## **RegisterMonitorToFilterRule(**

```
    IN    WCHAR* FilterMask,  
    IN    ULONG  RegisterIO  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

#### **RegisterIO**

The IOs were registered to the filter rule, only post-IOs can be registered, it was used to monitor the file IOs, when it was triggered, filter driver will send the notification to the user.

### **Comments**

# EaseFilter File System Filter SDK Manual

---

If you want to get the notification of the file IOs for the filter rule, this is the API to register the IOs which you are interested.

**BOOL**

## ***RegisterControlToFilterRule(***

```
    IN    WCHAR*  FilterMask,  
    IN    ULONG   RegisterIO  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

#### **RegisterIO**

The IOs were registered to the filter rule, were used to control the file IOs, when it was triggered, filter driver will send the notification to the user, block and wait for the response, it can control the IOs data and status based on the return result.

### **Comments**

If you want to control the file requests, this is the API to register the IOs which you are interested.

**BOOL**

## ***AddEncryptionKeyToFilterRule(***

```
    IN    WCHAR*  FilterMask,  
    IN    ULONG   EncryptionKeyLength,  
    IN    UCHAR*  EncryptionKey  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

#### **EncryptionKeyLength**

The length of the encryption key.

#### **EncryptionKey**

The encryption key for the filter rule.

# EaseFilter File System Filter SDK Manual

---

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

If the encryption was enabled in the access flag in the API `AddNewFilterRule`, this is the API to add the encryption key for the filter rule.

**BOOL**

## ***AddReparseFileMaskToFilterRule(***

```
    IN    WCHAR* FilterMask,  
    IN    WCHAR* ReparseFilterMask  
)
```

## **Parameters**

### **FilterMask**

The FilterMask which was set in API `AddNewFilterRule`.

### **ReparseFilterMask**

The reparse folder mask, it can include the wild character, but it must match the wild character in FilterMask.

For example:

```
FilterMask = c:\test\*txt  
ReparseFilterMask = d:\reparse\*doc
```

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

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

If the `REPARSE_FILE_OPEN` was enabled in the access flag in the API `AddNewFilterRule`, this is the API to add the reparse filter mask for the filter rule.

**BOOL**

## ***AddHiddenFileMaskToFilterRule(***

```
    IN    WCHAR* FilterMask,
```

# EaseFilter File System Filter SDK Manual

---

```
    IN    WCHAR* HiddenFileFilterMask
)
```

## Parameters

### FilterMask

The FilterMask which was set in API AddNewFilterRule.

### HiddenFileFilterMask

The hidden file filter mask for the files to be hidden.

For example:

```
FilterMask = c:\hideFilesTest\*
```

```
HiddenFileFilterMask = *.doc
```

If you open folder c:\hideFilesTest, all the files with extension .doc won't show up in the folder.

## Return Value

Return true if it succeeds, else return false.

## Comments

If the `HIDE_FILES_IN_DIRECTORY_BROWSING` was enabled in the access flag in the API AddNewFilterRule, this is the API to add the hidden filter mask for the filter rule.

## BOOL

### **AddExcludeFileMaskToFilterRule(**

```
    IN    WCHAR* FilterMask,
    IN    WCHAR* ExcludeFileFilterMask
)
```

## Parameters

### FilterMask

The FilterMask which was set in API AddNewFilterRule.

### ExcludeFileFilterMask

The file filter mask to be excluded.

For example:

```
FilterMask = *.txt
```

```
ExcludeFileFilterMask = c:\windows\*
```

# EaseFilter File System Filter SDK Manual

---

The filter driver target file is all the files with extension .txt except the files in folder c:\windows and its subfolders.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

This is the API to add the exclude file filter mask for the filter rule which was set in AddNewFilterRule.

## **BOOL**

### ***AddIncludeProcessIdToFilterRule(***

```
    IN    WCHAR* FilterMask,  
    IN    ULONG IncludeProcessId  
)
```

## **Parameters**

### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

### **IncludeProcessId**

The process Id to be included by filter driver.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

This is the API to add the include process Id for the filter rule which was set in AddNewFilterRule, only the files opened by the processes in the included process Ids and process names will be monitored by the filter driver.

## **BOOL**

### ***AddExcludeProcessIdToFilterRule(***

```
    IN    WCHAR* FilterMask,  
    IN    ULONG ExcludeProcessId  
)
```

## **Parameters**

### **FilterMask**

# EaseFilter File System Filter SDK Manual

---

The FilterMask which was set in API AddNewFilterRule.

## **ExcludeProcessId**

The process Id to be excluded by filter driver.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

This is the API to add the exclude process Id for the filter rule which was set in AddNewFilterRule, all the files were opened by the processes in the excluded process Ids and process names won't be monitored by the filter driver.

**BOOL**

## ***AddIncludeProcessNameToFilterRule(***

```
    IN    WCHAR* FilterMask,  
    IN    WCHAR* IncludeProcessName
```

)

## **Parameters**

### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

### **IncludeProcessName**

The process name to be included by filter driver.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

This is the API to add the include process name for the filter rule which was set in AddNewFilterRule, only the files opened by the processes in the included process Ids and process names will be monitored by the filter driver.

**BOOL**

## ***AddExcludeProcessNameToFilterRule(***

```
    IN    WCHAR* FilterMask,  
    IN    WCHAR* ExcludeProcessName
```

)



# EaseFilter File System Filter SDK Manual

---

## Parameters

### FilterMask

The FilterMask which was set in API AddNewFilterRule.

### ExcludeProcessName

The process name to be excluded by filter driver.

### Return Value

Return true if it succeeds, else return false.

## Comments

This is the API to add the exclude process name for the filter rule which was set in AddNewFilterRule, all the files were opened by the processes in the excluded process Ids and process names won't be monitored by the filter driver.

## BOOL

### *AddIncludeUserNameToFilterRule(*

```
    IN    WCHAR* FilterMask,  
    IN    WCHAR* IncludeUserName  
)
```

## Parameters

### FilterMask

The FilterMask which was set in API AddNewFilterRule.

### IncludeUserName

The user name to be included by filter driver.

### Return Value

Return true if it succeeds, else return false.

## Comments

This is the API to add the include user name for the filter rule which was set in AddNewFilterRule, only the files were opened by the users in the included user names will be monitored by the filter driver.

## BOOL

# EaseFilter File System Filter SDK Manual

---

## ***AddExcludeProcessNameToFilterRule(***

```
    IN    WCHAR* FilterMask,  
    IN    WCHAR* ExcludeUserName  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

#### **ExcludeUserName**

The process name to be excluded by filter driver.

### **Return Value**

Return true if it succeeds, else return false.

### **Comments**

This is the API to add the exclude process name for the filter rule which was set in AddNewFilterRule, all the files were opened by the users in the excluded user names won't be monitored by the filter driver.

### **BOOL**

## ***AddProcessRightsToFilterRule(***

```
    IN    WCHAR* FilterMask,  
    IN    WCHAR* ProcessName,  
    IN    ULONG  AccessFlag  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

#### **ProcessName**

The process name to be set access rights.

#### **AccessFlag**

The access rights for the process.

### **Return Value**

Return true if it succeeds, else return false.

### **Comments**

# EaseFilter File System Filter SDK Manual

---

This is the API to add the access flags to the specific process, when the process accesses the files, it needs to check permission from access flags first to allow or deny the file access.

**BOOL**

## ***AddUserRightsToFilterRule(***

```
    IN   WCHAR* FilterMask,  
    IN   WCHAR* UserName,  
    IN   ULONG  AccessFlag  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

#### **UserName**

The user name to be set access rights.

#### **AccessFlag**

The access rights for the user.

### **Return Value**

Return true if it succeeds, else return false.

### **Comments**

This is the API to add the access flags to the specific user, when the user accesses the files, it needs to check permission from access flags first to allow or deny the file access.

**BOOL**

## ***AddIncludeUserNameToFilterRule(***

```
    IN   WCHAR* FilterMask,  
    IN   WCHAR* IncludeUserName  
)
```

### **Parameters**

#### **FilterMask**

The FilterMask which was set in API AddNewFilterRule.

# EaseFilter File System Filter SDK Manual

---

## **IncludeUserName**

The user name to be included by filter driver.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

This is the API to add the include user name for the filter rule which was set in AddNewFilterRule, only the files were opened by the users in the included user names will be monitored by the filter driver.

*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*

*AddIncludedProcessId(ULONG ProcessId)*

## **Parameters**

### **ProcessId**

The process Id you want to be included by filter.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

# EaseFilter File System Filter SDK Manual

---

This API let the filter driver only intercept the I/O for the included processes, discard all other I/O from other processes, you can add multiple process Id.

**BOOL**

*RemoveExcludeProcessId(ULONG ProcessId)*

**Parameters**

**ProcessId**

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

**Return Value**

Return true if it succeeds, else return false.

**Comments**

This API removes the included process Id from filter.

**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**

# EaseFilter File System Filter SDK Manual

---

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 removes the excluded process Id from filter.

**BOOL**

## *AddProtectedProcessId(ULONG ProcessId)*

## **Parameters**

### **ProcessId**

The process Id you want to be protected by filter.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

This API can prevent the process being terminated, you can add multiple process Id, this API is supported in OS vista or later versions.

**BOOL**

## *RemoveProtectedProcessId(ULONG ProcessId)*

## **Parameters**

### **ProcessId**

The process Id you want to remove which set by *AddProtectedProcessId* API.

## **Return Value**

Return true if it succeeds, else return false.

## **Comments**

This API removes the protected procss Id.

**BOOL**

# EaseFilter File System Filter SDK Manual

---

## *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 requests. 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.

# EaseFilter File System Filter SDK Manual

---

## 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 system will allocate a unique IV for the file.

### IV

The initial vector, when IVLength 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.



# EaseFilter File System Filter SDK Manual

---

## Comments

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

## BOOL

### ***AESEncryptFileToFile(***

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

## Parameters

### **SourceFileName**

The source file name to be encrypted.

### **DestFileName**

The target file name was 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 system will allocate a unique IV for the file.

### **IV**

The initial vector, when IVLength 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.

# EaseFilter File System Filter SDK Manual

---

## Comments

AESEncryptFileToFile 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**

## ***AESDecryptFileToFile(***

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

### **Parameters**

#### **SourceFileName**

The encrypted file name.

#### **DestFileName**

The target file name was 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**

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

# EaseFilter File System Filter SDK Manual

---

**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.

# EaseFilter File System Filter SDK Manual

---

## 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.

# EaseFilter File System Filter SDK Manual

---

## 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](mailto: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 = AddNewFilterRule(AccessFlags,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 FilterAPI.cs to your folder. FilterAPI.cs has the structures and APIs used for connecting to the filter driver.