File Expander
Application Programming Interface

Version 3.02

August 26, 2022

Table of Contents

Introduction

File Expander was developed to dig deep into complex file formats and extract data objects for detailed analysis. Such objects are often translated, compressed, encrypted and/or fragmented within the parent file. It is File Expander's job to eliminate any complex storage algorithms and produce data objects that can be studied and processed by the user. This Application Programming Interface is provided for software developers to include this functionality in automated data processing systems and end-user tools.

A Forensic Innovations Dark Data Detective - Objects viewer application will soon be available for end-users to utilize this functionality and enable the user to better search for data in files. This application will initially be provided for MS Windows.

This Application Programming Interface provides you with everything that you need. The license allows you to distribute the required compiled libraries to your users. Sample programs show you how to implement each interface and use the data returned from them. This document explains how the File Expander Engine works and what to expect when utilizing the returned data.

How to Purchase the File Expander Application Programming Interface

The latest purchasing information and product details are available at https://www.FID3.com. It is recommended that you visit this site for updates on this product. You are welcome to use this product for testing purposes, but you must purchase a license before you distribute this product in any way. The latest version of this document is available at https://www.fid3.com/download/feapiman.pdf

How the File Expander Engine Works

The File Expander Engine Windows Dynamically Linked Library (FEEnginew32.dll or FEEnginew64.dll), and Linux Shared Object Library (FEEnginel64.so) interprets files and extracts their individual objects. When you provide the path, filename and object index, of a file to be analyzed, you receive the details of the object and/or a copy of the object as a separate file.

The Application Programming Interface includes sample applications for MS Windows written for MS Visual C#, MS Visual C++.NET, MS Visual Basic.NET and MS Visual C++ 6.0, as well as Linux GCC (compiled in Red Hat Linux 5).

Usage

In this section, we will describe what you are receiving in the File Expander Application Programming Interface (FEAPI), how to install the files, the use of the MS Windows Registry, the sample application and other background information.

Files Included in the API

The FEAPIxxxxx.ZIP Archive Contains:

Filename	Description
changes.txt	List of changes since previous releases
feenginew32.dll	File Expander Engine 32-bit Windows DLL
feenginew64.dll	File Expander Engine 64-bit Windows DLL
feenginel64.so	File Expander Engine 64-bit Linux Library
feengine.h	File Expander C header file for sample applications
fefile.cpp	Example application (unmanaged C++) source code
fefile.vcproj	Example application (unmanaged C++) MS VS 2012 project
fefilew32.exe	Example 32-bit C++ application for Windows (requires feenginew32.dll)
fefilew64.exe	Example 64-bit C++ application for Windows (requires feenginew64.dll)
fefilecpp.cpp	Example application (managed C++.NET) source code
fefilecpp.vcproj	Example application (managed C++.NET) MS VS 2012 project
resource.h	Example application (managed C++.NET) MS VS 2012 header
fefilecppw32.exe	Example 32-bit C++.NET application for Windows (requires feenginew32.dll & fewrpnetw32.dll)
fefilecppw64.exe	Example 64-bit C++.NET application for Windows (requires feenginew64.dll & fewrpnetw64.dll)
fefilecs.cs	Example application (C#) source code
fefilecs.csproj	Example application (C#) MS VS 2012 project
AssemblyInfo.cs	Example application (C#) MS VS 2012 assembly information
fefilecsw32.exe	Example 32-bit C# application for Windows (requires feenginew32.dll & fewrpnetw32.dll)
fefilecsw64.exe	Example 64-bit C# application for Windows (requires feenginew64.dll & fewrpnetw64.dll)
fefilel.cpp	Example Linux application (C++) source code
fefilel64	Example 64-bit C++ application for Linux (requires feenginel64.so)
build	Example Linux application (C++) build script
fefilevb.vb	Example application (Visual BASIC.NET) source code
fefilevb.vbproj	Example application (Visual BASIC.NET) MS VS 2012 project
My Project*.*	Example application (Visual BASIC.NET) MS VS 2012 files
fefilevbw32.exe	Example 32-bit Visual BASIC.NET application for Windows (requires feenginew32.dll & fewrpnetw32.dll)
fefilevbw64.exe	Example 64-bit Visual BASIC.NET application for Windows (requires feenginew64.dll & fewrpnetw64.dll)
fewrpnetw32.dll	File Expander Engine 32-bit Windows.NET DLL Wrapper (uses FEEnginew32.dll)
fewrpnetw64.dll	File Expander Engine 64-bit Windows.NET DLL Wrapper (uses FEEnginew64.dll)
readme.txt	Overview of product and archive contents
unrar	Linux RAR utility (used by feenginel64.so)
unzip.exe	Info-ZIP's UnZip EXE (used by feenginew64.dll & feenginew32.dll)
unzip32.dll	Info-ZIP's UnZip DLL (used by feenginew32.dll for compressed objects)

Install the API

Extract the API Files

The FEAPIxxxxx.ZIP file was created with PKZIP version 2.04g, and contains all of the API files. You can find file extracting software at www.PKWare.com. The files do not need to be copied to any specific directory, but they should all be kept in one directory.

System Requirements

The feenginew??.dll and sample applications require MS Windows 200x/NT/XP/Vista/8.x/10/11 and Internet Explorer 4.0 or later. A Pentium processor or higher and 32MB of RAM is recommended. The feenginel64.so and sample application require a 64-bit Linux operating system and 32MB of RAM is recommended.

Windows Registry

All Forensic Innovations, Inc. consumer applications update their version in the Windows Registry when they execute. The only exceptions are the included sample applications. When fefilecpp.exe is executed, it loads fiwrpnet.dll and feengine.dll into memory. At that point the feengine.dll updates its version at HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Versions. A hexadecimal value of 0x01020304 translates to version 1.02.03.04. When the feengine.dll is instructed to scan for the *.fel libraries, they are each loaded and they also update their binary version values under the same registry key.

Example:

HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Versions		
Archive Library	REG_DWORD	0x01000002
Compound Files Library	REG_DWORD	0x01000002
Engine	REG_DWORD	0x01000002

When the libraries are instructed to Register, they populate the HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Expanders\Names and HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Expanders\Formats keys with the file formats that they support. The library names are linked to the actual *.fel file names and the File Investigator file type index values are linked to the library names.

Example:

HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Expanders\Names		
Archives	REG_SZ	fearchive.fel
Compound Files	REG_SZ	fecompound.fel

HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Expanders\Formats		
12	REG_SZ	Archives
229	REG_SZ	Compound Files

When the libraries are instructed to enter and store the product registration key, they populate the HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Expanders\Validation key with the registration key for the “Engine”

Example:

HKEY_LOCAL_MACHINE\SOFTWARE\Forensic Innovations\File Expander\Validation		
Engine	REG_SZ	RegistrationKey

Sample Application

File Expander File.NET for MS Visual Studio

These C# and managed C++ sample applications are designed for implementing the API to obtain the detailed information returned on each file. While the example source code includes a user interface and everything else needed to create a program, we will only be reviewing the most pertinent portions here. Due to the common .NET framework, this sample application can easily be ported to J# and Visual Basic.NET languages. Forensic Innovations, Inc. does not yet provide .NET sample applications in other programming languages. This sample application and the fiwrpnetw??.dll were built with Visual Studio 2012, targeting .NET 4.0. The included FEENGINEW??.DLL and FEWRPNETW??.DLL assemblies are required for both sample applications to run.

For MS Windows and Linux unmanaged C code samples, the steps will be added soon. For now, you can use the following as an example. Linux doesn't require a wrapper library. MS Windows C also doesn't require the fewrnetw??.dll either, but it may require one or more of the following included MS redistributable libraries (included in our provided feapi.zip archive): mscvp110.dll, msvcr110.dll & vcomp110.dll. These files may already be provided with your Windows OS installation, but if they aren't then they should be in the same working directory with feenginew??.dll. Our feapi.zip archive includes w32 & w64 with matching sets of these files, but built for 32-bit or 64-bit uses.

Getting started

Step 1: Copy/Unzip the archived files to a directory to work in.
Step 2: Open fefilecs.csproj (or fefilecpp.vcproj) with MS Visual Studio
Step 3: Check that the directory locations are all pointing to the same place that you put the source files in
Step 4: Use the 'Build>Build fefilecs' (or fefilecpp) menu option to build the executable
Step 5: Open a DOS Shell (Start >> All Programs >> Accessories >> Command Prompt)
Step 6: Change to the directory that you previously put the source files in
Step 7: Execute the program (`>fefilecs.exe /H` or `>fefilecpp.exe /H`) to see the sample usage
Step 8: Execute the program on a file (`>fefilecs.exe FEAPIxxxx.zip` or `>fefilecpp.exe FEAPIxxxx.zip`) to see the results

The application is configured to display the help screen when no command line parameters are provided. Its primitive command line parameter handling always expects parameters, and returns an error when no parameters are provided. This is as designed. Here are some examples of the output you will see. These examples are from the fifilecpp sample application, but the fifilecs sample application output is identical.

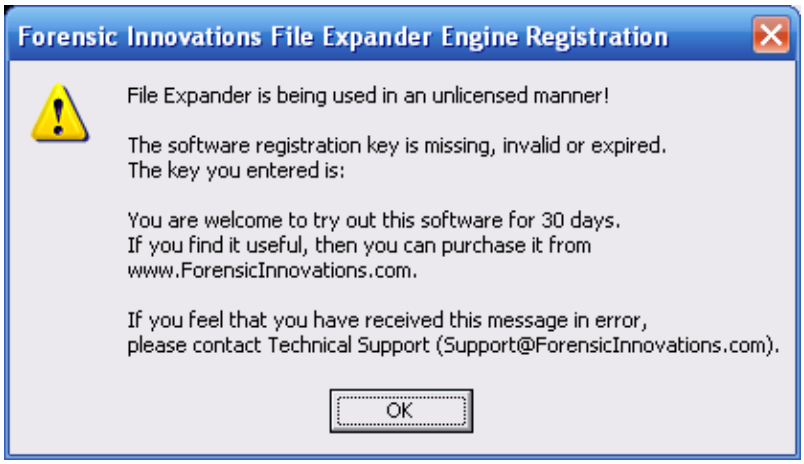
Sample Output

```
>fefilecpp.exe /H
```



```
File Expander File CPP for Windows.NET 98SE/Me/NT/200x/XP/Vista    Ver 3.02
Copyright (C) 2022 Forensic Innovations, Inc.; ALL RIGHTS RESERVED

Summary:     This utility is included in the File Expander API as an
             example application.  It can be used to analyze a single file
             and display details about that file.  It may not be distributed
             to non-File Expander API owners.

Usage:       FEFILECPP.EXE [drive:path]filename [options]

             FEFILECPP.EXE [options]

Options:
  /E         Extract the object(s) found.
  /H         Display this help screen.
  /D<path>   Provide the destination path to expand files to. (ex: /DC:\temp)
  /K<key>    Provide a registration key to prevent the nag screen. (ex: /kKey)
  /L         Display lists of the file formats supported by File Expander.
  /O<number> Display information/extract the specified object. (by index #)
  /P         Pause the display for each screen.
  /W<path>   Provide the working folder to find FEEngine support files.
```

If you do not yet have a valid registration key, then you will see a nag screen (pictured to the right) for the first file that feengine.dll is requested to process. If an application keeps the feengine.dll loaded and sends multiple file process requests, then the nag screen only appears for the first file process request.



As a first test, try running the sample program on the FEAPIxxxx.zip archive file that you downloaded from us. You will see much of the information that this API provides, and you can extract the objects by adding the `/E/D<path>` parameters.

```
>fefilecpp.exe FEAPI100.zip
```

```
File Expander File CPP for Windows.NET 98SE/Me/NT/200x/XP/Vista    Ver 3.02
Copyright (C) 2022 Forensic Innovations, Inc.; ALL RIGHTS RESERVED

GetFileInfo:
  Input File----------] c:\data\robware\source\bu\FEAPI100b1.zip
  File Format ID------] 12
  Total Objects-------] 37
  Filtered Objects----] 37
  Data Object Names---] unzip32.dll
                        fefilecpp.vcproj
                        changes.txt
                        fefilecpp.exe
                        file_id.diz
```

```
                          readme.txt
                          fiwrpnet.dll
                          fefilecpp.cpp
                          fearchive.fel
                          fecompound.fel
                          feengine.dll
                          FIEngine.fid

GetObjectInfo:
  Input File----------] c:\data\robware\source\bu\FEAPI100b1.zip

  Object #------------] 0
  Object Name---------] Local File Header #0
  Object Attributes---] FileHeader, ObjectHeader
  Offset--------------] 0x00000000 + 0 bits
  Information Offset--] 0x00000000 + 0 bits
  Uncompressed Size---] 0x00000029 + 0 bits
  Storage Condition---] Consecutive
  Expansion Support---] Copy

  Object #------------] 1
  Object Name---------] unzip32.dll
  Object Attributes---] ObjectData, ObjectIsFile
  Offset--------------] 0x00000029 + 0 bits
  Information Offset--] 0x00000000 + 0 bits
  Uncompressed Size---] 0x00019000 + 0 bits
  Compressed Size-----] 0x0000baea + 0 bits
  Storage Condition---] Consecutive
  Compression Algorithm] PKWare
  Expansion Support---] Compression, Copy

  .
  .
  .

  Object #------------] 24
  Object Name---------] Central Directory Structure #0
  Object Attributes---] Index, ObjectData
  Offset--------------] 0x0005472b + 0 bits
  Information Offset--] 0x00000000 + 0 bits
  Uncompressed Size---] 0x00000039 + 0 bits
  Storage Condition---] Consecutive
  Expansion Support---] Copy
  .
  .
  .
  Object #------------] 36
  Object Name---------] End of Central Directory Record
  Object Attributes---] FileFooter
  Offset--------------] 0x000549e3 + 0 bits
  Information Offset--] 0x00000000 + 0 bits
  Uncompressed Size---] 0x00000016 + 0 bits
  Storage Condition---] Consecutive
  Expansion Support---] Copy

StopFile() returned: EndOfList (31)
```


Next, take a look at a list of all of the file formats supported, and lists of some flag values provided to help classify each file. Please execute this command on your own, to obtain the most recent lists.

```
>fefilecpp.exe /L/P
```

```
File Expander File CPP for Windows.NET 98SE/Me/NT/200x/XP/Vista    Ver 3.02
Copyright (C) 2022 Forensic Innovations, Inc.; ALL RIGHTS RESERVED

pFEWrap->FormatContent Flag Values:      pFEWrap->FormatStorage Flag Values:

Flag     Type                            Flag Type
-------- ------------------------------  ---- ------------------------------
       0 N/A                                0 N/A
       1 Video                              1 Cabinet/Archive
       2 Database                           2 Binary
       4 Database Hybrid                    4 Bitmap/Raster
       8 Document                           8 Digital Audio
      10 Font                              10 Music Notes
      20 Game Data                         20 Text
      40 Graphic Image                     40 Translated
      80 Graphic Metafile                  80 Vector
     100 Hypertext                        100 Floating Header
     200 Personal/User Data
     400 Icon
     800 Library of Functions            pFEWrap->FormatPlatform Flag Values:
    1000 Macro/Script
    2000 Program Data                    Flag Type
    4000 Program Executable              ---- ------------------------------
    8000 Raw Printer Data                   0 N/A
   10000 ROM/RAM Image                      1 Commodore Amiga
   20000 Shortcut/Link                      2 IBM OS/2
   40000 Sound/Audio                        4 IBM PC Compatible
   80000 Sound Metafile                     8 Apple Macintosh
  100000 Source Code                       10 MS Windows 3.x
  200000 Spreadsheet                       20 MS Windows 95/98/Me/NT/2000/XP
  400000 Template                          40 MS/PC DOS
  800000 Text                              80 Sun OS
 1000000 Text Hybrid                      100 UNIX
 2000000 Virtual Environment              200 Atari
 4000000 Malicious/Virus                  400 Apple II
                                          800 MS Windows CE/Pocket PC
                                         1000 Palm OS
                                         2000 Alpha (64 bit)
                                         4000 Linux

pFEWrap->FormatName Formats Supported:

Index File Format Name                                    Ver. Added Updated
----- ------------------------------------------------- ---------- ----------
    0 Unidentified                                      0.00.00.00 0.00.00.00
       File Extension(s):     .*
       Platform(s):           N/A (0x00)
       Storage Method(s):     N/A (0x00)
       Content Type(s):       N/A (0x00)

   12 PK Zip Archive                                    1.00.00.01 1.00.00.01
       File Extension(s):     .ZIP, .JAR, .WMZ
       MIME Label(s):         application/zip,
                              application/x-zip-compressed,
                              application/x-compressed,
                              application/x-zip
       Platform(s):           IBM OS/2 (0x2),
                              IBM PC Compatible (0x4),
                              Apple Macintosh (0x8),
                              MS Windows 3.x (0x10),
                              MS Windows 95/98/Me/NT/2000/XP (0x20),
                              MS/PC DOS (0x40)
```

```
      Storage Method(s):     Cabinet/Archive (0x1),
                             Binary (0x2)
      Content Type(s):       N/A (0x00)
      File Expander Library: fearchive.fel

  111 MS Excel Worksheet/Add-In/Template                  1.00.00.01 1.00.01.00
      File Extension(s):     .XLS, .XLA, .XLT, .XLB, .WWS
      MIME Label(s):         application/vnd.ms-excel,
                             application/x-excel,
                             application/x-xlt,
                             application/x-msexcel
      Platform(s):           IBM PC Compatible (0x4),
                             MS Windows 3.x (0x10),
                             MS Windows 95/98/Me/NT/2000/XP (0x20)
      Storage Method(s):     Binary (0x2)
      Content Type(s):       Personal/User Data (0x200),
                             Spreadsheet (0x200000),
                             Text (0x800000)
      File Expander Library: fecompound.fel
.
.
.

  KEY:  ?, *, n  = File Extension wildcards indicate a space or word that can
                   be any character and a space that can be 0-9.

  110 File Formats Supported
```

Finally, when you have decided to purchase the product registration key, enter it and instruct the application to store it to the Windows Registry. The actual key is emailed to you once your payment is confirmed.

```
>fefilecpp.exe /Kregistrationkey
```


The Source Code

.NET Wrapper Library

The fewrpnetw32.dll & fewrpnetw64.dll assemblies provide the following enumeration and flag values.

Enumerators and Flags

```
namespace ForensicInnovations
{
    namespace FileExpander
    {
        // Enums
        // Enum values used with all functions
        public __value enum ErrorReturnCodes
        {
            Success                             = 0,
            Failure,
            FileNotFound,
            StringNotFound,
            ObjectsNotFound,
            CreateFileFailed,
            EndOfFile,
            CreateDirectoryFailed               = 20,
            FormatNotSupported                  = 21,
            BadOrEmptyParameter                 = 22,
            BadRegistrationKey                  = 23,
            LibraryInterfaceMissing             = 24,
            LibraryVersionOld                   = 25,
            StopCommandUsed                     = 26,
            ReadingFileFailed                   = 27,
            LoadingLibraryFailed                = 28,
            AccessingRegistryFailed             = 29,
            WrongFormatID                       = 30,
            EndOfList                           = 31,
            FileCorrupted                       = 32,
            UnknownObject                       = 33,
            WritingFileFailed                   = 34,
            AccessingMemory                     = 35,
            BadObjectOffset                     = 36,
            WrongObjectSize                     = 37,
            DecompressingFileFailed             = 38
        };

        // Enum values used with feGetString() for Type
        public __value enum GetStringOptions
        {
            Content                             = 1,
            Storage,
            Platform,
            Error                               = 20
        };

        // Flag values used with feStartEngine() for Instructions
        [FlagsAttribute]
        public __value enum StartEngineFlags
        {
            All                                 = 0,
            RegisterDLL                         = 0x01 << 0,
            ScanFELibraries                     = 0x01 << 1,
            EnterRegistrationKey                = 0x01 << 2,
            StoreRegistrationKey                = 0x01 << 3,
```



```
    ReplaceWorkingDirectory             = 0x01 << 5
};

// Flag values used with feGetFileInfo() for FileFieldsRequested
[FlagsAttribute]
public __value enum FileFieldsRequestFlags
{
    All                                 = 0,
    FileFormatID                        = 0x01 << 0,
    TotalObjects                        = 0x01 << 1,
    FilteredObjects                     = 0x01 << 2,
    ObjectNames                         = 0x01 << 3,
    ObjectsExpandable                   = 0x01 << 6,
    UncompressedSize                    = 0x01 << 7,
    ResetDebugLog                       = 0x01 << 10,
    UseDebugLog                         = 0x01 << 11
};

// Flag values used with feGetObjectInfo() for ObjectFieldsRequested
[FlagsAttribute]
public __value enum ObjectFieldsRequestFlags
{
    All                                 = 0,
    FileFormatID                        = 0x01 << 0,
    AbsoluteIndex                       = 0x01 << 1,
    ObjectName                          = 0x01 << 3,
    TranslationAlgorithm                = 0x01 << 4,
    CompressionAlgorithm                = 0x01 << 5,
    EncryptionAlgorithm                 = 0x01 << 6,
    ExpansionSupport                    = 0x01 << 8,
    AttributeFilter                     = 0x01 << 13,
    ObjectSize                          = 0x01 << 18,
    ObjectSizeCompressed                = 0x01 << 19,
    CRC                                 = 0x01 << 21,
    Path                                = 0x01 << 23,
    ObjectOffset                        = 0x01 << 24,
    ObjectInformationOffset             = 0x01 << 25,
    Storage                             = 0x01 << 26,
    UseDebugLog                         = 0x01 << 28
};

// Flag values used with feGetObjectInfo() for ExpansionInstructions
[FlagsAttribute]
public __value enum ObjectExpansionFlags
{
    None                                = 0,
    ExpandToFile                        = 0x01 << 0,
    CopyWithoutExpansion                = 0x01 << 1,
    CopyIfExpansionFails                = 0x01 << 2
};

// Enum values used with feGetObjectInfo() for TranslationAlgorithm
public __value enum ObjectTranslationAlgorithms
{
    None                                = 0,
    Unknown,
    MIME,
    CB64,
    UUencode
};
```



```
// Enum values used with feGetObjectInfo() for CompressionAlgorithm
public __value enum ObjectCompressionAlgorithms
{
    None                                = 0,
    Unknown,
    PKWare,
    WinZip,
    Tokenizing,
    Deflate64,
    BZIP2,
    IBMTerse,
    IBMLZ77z,
    WavPack,
    PPMdvI,
    zlib
};

// Enum values used with feGetObjectInfo() for EncryptionAlgorithm
public __value enum ObjectEncryptionAlgorithms
{
    None                                = 0,
    Unknown,
    PKWare,
    WinZipAES
};

// Flag values used with feGetObjectInfo() for ExpansionSupport
[FlagsAttribute]
public __value enum ExpansionSupportFlags
{
    None                                = 0,
    Translation                         = 0x01 << 0,
    Compression                         = 0x01 << 1,
    Encryption                          = 0x01 << 2,
    Copy                                = 0x01 << 3
};

// Flag values used with feGetObjectInfo() for ObjectAttributes
[FlagsAttribute]
public __value enum ObjectAttributeFlags
{
    None                                = 0,
    FileHeader                          = 0x01 << 0,//0x0001
    Index                               = 0x01 << 1,//0x0002
    ObjectHeader                        = 0x01 << 2,//0x0004
    ObjectData                          = 0x01 << 3,//0x0008
    ObjectFooter                        = 0x01 << 4,//0x0010
    FileFooter                          = 0x01 << 5,//0x0020
    ObjectIsFile                        = 0x01 << 6,//0x0040
    UnknownObject                       = 0x01 << 7,//0x0080
};

// Flag values used with feGetObjectInfo() for ObjectStorageCondition
[FlagsAttribute]
public __value enum ObjectStorageConditionFlags
{
    Unknown                             = 0,
    Consecutive                         = 0x01 << 0,
    Fragmented                          = 0x01 << 1,
    Incomplete                          = 0x01 << 2,// Deleted & over written
    Corrupted                           = 0x01 << 4
};
```



```
        // Flag values used with feGetFormatInfo() for FieldsRequested
        [FlagsAttribute]
        public __value enum FormatFieldsRequestFlags
        {
            All                                 = 0,
            Name                                = 0x01 << 0,
            Extensions                          = 0x01 << 1,
            MIME                                = 0x01 << 2,
            Platform                            = 0x01 << 3,
            Storage                             = 0x01 << 4,
            Content                             = 0x01 << 5,
            VersionAdded                        = 0x01 << 6,
            VersionUpdated                      = 0x01 << 7,
            AttributeFilterFlags                = 0x01 << 8,
            AlgorithmFlags                      = 0x01 << 9,
            UseDebugLog                         = 0x01 << 12,
            LibraryFilename                     = 0x01 << 13
        };
    };// namespace FileExpander
}// namespace ForensicInnovations
```

The following public members are used to configure the fiwrnnet.dll assembly and access the results.

Public Member Variables

```
namespace ForensicInnovations
{
    namespace FileExpander
    {
        public __gc class Engine
        {
        public:
            // Engine Instructions (Set before calling feStartEngine)
            String *                EngineRegistrationKey;
            String *                EngineWorkingDirectory;

            // File Instructions (Set before calling feGetFileInfo)
            String *                FullPath;
            String *                SearchFilespec;

            ObjectAttributeFlags    AttributeFilter;
            unsigned __int64        FormatAttributeFilter;
            FileFieldsRequestFlags  FileFieldsRequested;

            // File Results (Check after calling feGetFileInfo)
            unsigned long           FileFormatID;
            unsigned long           FileTotalObjects;
            unsigned long           FileFilteredObjects;
            String *                FileObjectNames __gc[];
            unsigned long           FileObjectsExpandable;
            unsigned __int64        FileUncompressedSize;

            // Object Instructions (Set before calling feGetObjectInfo)
            ObjectFieldsRequestFlags ObjectFieldsRequested;
            unsigned long           ObjectIndexStart;
            unsigned long           ObjectIndexStop;
            ObjectExpansionFlags    ObjectExpansionInstructions;
            String *                ObjectExpandToPath;
            String *                ObjectExpandToFilespec;

            // Object Results (Check after calling feGetObjectInfo)
            unsigned long           ObjectAbsoluteIndex;
```



```
            String *                        ObjectName;
            ObjectTranslationAlgorithms     ObjectTranslationAlgorithm;
            ObjectCompressionAlgorithms     ObjectCompressionAlgorithm;
            ObjectEncryptionAlgorithms      ObjectEncryptionAlgorithm;
            ExpansionSupportFlags           ObjectExpansionSupport;
            ObjectAttributeFlags            ObjectAttributes;
            unsigned __int64                ObjectSize;
            unsigned long                   ObjectSizeBitsAdded;
            unsigned __int64                ObjectSizeCompressed;
            unsigned long                   ObjectSizeCompressedBitsAdded;
            unsigned long                   ObjectCRC;
            String *                        ObjectPath;
            unsigned __int64                ObjectOffset;
            unsigned long                   ObjectOffsetBitsAdded;
            unsigned __int64                ObjectInfoOffset;
            unsigned long                   ObjectInfoOffsetBitsAdded
            ObjectStorageConditionFlags     ObjectStorageCondition;

            // Format Results (Check after calling feGetFormatInfo)
               // Fields obtained from the File Investigator database
            FormatFieldsRequestFlags        FormatFieldsRequested;
            String *                        FormatName;
            String *                        FormatExtensions __gc[];
            String *                        FormatMIMEs __gc[];
            unsigned long                   FormatPlatform;
            unsigned long                   FormatStorage;
            unsigned long                   FormatContent;

               // FEEngine specific fields
            String *                        FormatLibraryFilename;
            unsigned long                   FormatVersionAdded;
            unsigned long                   FormatVersionUpdated;
         };// public __gc class Engine
      };// namespace FileExpander
   }// namespace ForensicInnovations
```

Method Declarations

```
namespace ForensicInnovations
{
   namespace FileExpander
   {
      public __gc class Engine
      {
      public:
         // Methods
         ErrorReturnCodes StartEngine(StartEngineFlags Instructions);
         ErrorReturnCodes GetFileInfo(String * InputFilename,
            long InputFileFormatID);
         ErrorReturnCodes GetObjectInfo(String * InputFilename,
            long InputFileFormatID);
         ErrorReturnCodes GetFormatInfo(long InputFileFormatID);
         String *         GetString(GetStringOptions Type, long StringID,
            ErrorReturnCodes Error);
         ErrorReturnCodes StopFile();

         Engine();
      };// public __gc class Engine
   };// namespace FileExpander
}// namespace ForensicInnovations
```


Command Order

Definition to simplify nomenclature

C#
```
using FE = ForensicInnovations.FileExpander;
```

Managed C++
```
#define FE ForensicInnovations::FileExpander
```

Get a pointer to a copy of the File Expander Engine Class

C#
```
FE.Engine FEWrap = new FE.Engine();
```

Managed C++
```
FE::Engine^ pFEWrap = gcnew FE::Engine;
```

One Time Tasks

These are tasks that you only need to perform once after you copy the File Expander files to a folder to start using them, or when you update one or more of the files. You will want to perform this step with just `ReplaceWorkingDirectory` every time you load the library if you are going to use the FEDebug.log option.

Fill in the Engine input values - located in the example ProcessCommandLine()

C#
```
String    RegistrationKey;

RegistrationKey = "The Key You Buy" // Differs from the sample application
FEWrap.EngineRegistrationKey = String.Copy(RegistrationKey);
```

Managed C++
```
String ^  RegistrationKey;

RegistrationKey = "The Key You Buy" // Differs from the sample application
pFEWrap->EngineRegistrationKey = String::Copy(RegistrationKey);
```

Fill in the request flags for starting the Engine - located in the example StartFEEngine()

C#
```
FE.StartEngineFlags    StartEngineInstructions;

StartEngineInstructions =
    (FE.StartEngineFlags) (int)
    (FE.StartEngineFlags.RegisterDLL |      // After installation of a new version
    FE.StartEngineFlags.ScanFELibraries |   // When adding/updating *.FEL libraries
    FE.StartEngineFlags.EnterRegistrationKey |
    FE.StartEngineFlags.StoreRegistrationKey |     // Store key in Registry
    FE.StartEngineFlags.ReplaceWorkingDirectory);
```


Managed C++
```
FE::StartEngineFlags    StartEngineInstructions;

StartEngineInstructions = (FE::StartEngineFlags) (int)
    (FE::StartEngineFlags::RegisterDLL |    // After installation of a new version
    FE::StartEngineFlags::ScanFELibraries | // When adding/updating *.FEL libraries
    FE::StartEngineFlags::EnterRegistrationKey |
    FE::StartEngineFlags::StoreRegistrationKey |   // Store key in Registry
    FE::StartEngineFlags::ReplaceWorkingDirectory);
```

Call the StartEngine method - located in the example StartFEEngine()

C#
```
FE.ErrorReturnCodes Return = FEWrap.StartEngine(StartEngineInstructions);
```

Managed C++
```
FE::ErrorReturnCodes Return = pFEWrap->StartEngine(StartEngineInstructions);
```

Check that the library version is current - located in the example StartFEEngine()

C#
```
long    llTemp = 0L;

RegistryKey rkVersions = Registry.LocalMachine.OpenSubKey("Software\\Forensic
    Innovations\\File Expander\\Versions");
if (rkVersions != null)
{
    String valueEngine = rkVersions.GetValue("Engine").ToString();
    llTemp = Convert.ToInt64(valueEngine);
    if (llTemp/16777216*100 + (llTemp%16777216/65536)<100)// 100=v1.00; 102=v1.02
    {
        Console.Write("\r\nERROR: The FEEngine.DLL is too old!  Please install a
    newer version.\r\n");
        Return = FE.ErrorReturnCodes.LibraryVersionOld;
    }
}
else Return = FE.ErrorReturnCodes.AccessingRegistryFailed;
```

Managed C++
```
long long llTemp = 0L;

RegistryKey ^ rkVersions = Registry::LocalMachine->OpenSubKey("Software\\Forensic
Innovations\\File Expander\\Versions");
if (rkVersions)
{
    String ^ valueEngine = rkVersions->GetValue("Engine")->ToString();
    llTemp = Convert::ToInt64(valueEngine);
    if (llTemp/16777216*100+(llTemp%16777216/65536) < 100)// 100=v1.00, 102=v1.02
    {
        Console::Write("\r\nERROR: The FEEngine.DLL is too old!  Please install a
newer version.\r\n");
        Return = FE::ErrorReturnCodes::LibraryVersionOld;
    }
}
else Return = FE::ErrorReturnCodes::AccessingRegistryFailed;
```


Summarize Each File

This step only needs to be performed when you want a summary of a file's contents.

Fill in the GetFileInfo input values - located in the example ProcessCommandLine()

C#

```
String    File;
String    WorkingFolder = "";

File = "c:\\fitest\\FEAPI100b2.zip";       // Differs from the sample application
WorkingFolder = "c:\\fitest\\";            // Differs from the sample application

FEWrap.FullPath = String.Copy(File);
FEWrap.EngineWorkingDirectory = String.Copy(WorkingFolder);
```

Managed C++

```
String ^  File;
String ^  WorkingFolder;

File = "c:\\fitest\\FEAPI100b2.zip";       // Differs from the sample application
WorkingFolder = "c:\\fitest\\";            // Differs from the sample application

pFEWrap->FullPath = String::Copy(File);
pFEWrap->EngineWorkingDirectory = String::Copy(WorkingFolder);
```

Fill in the GetFileInfo request flags - located in the example ShowFileInfo()

C#

```
String    SearchFilespec      = "*.*";

FEWrap.SearchFilespec  = String.Copy(SearchFilespec);
FEWrap.AttributeFilter = (FE.ObjectAttributeFlags) (int)
    (FE.ObjectAttributeFlags.FileHeader |
    FE.ObjectAttributeFlags.Index |
    FE.ObjectAttributeFlags.ObjectIsFile |
    FE.ObjectAttributeFlags.ObjectHeader |
    FE.ObjectAttributeFlags.ObjectData |
    FE.ObjectAttributeFlags.ObjectFooter |
    FE.ObjectAttributeFlags.FileFooter |
    FE.ObjectAttributeFlags.UnknownObject);
FEWrap.FileFieldsRequested = (FE.FileFieldsRequestFlags) (int)
    (FE.FileFieldsRequestFlags.AlgorithmsSupported |
    FE.FileFieldsRequestFlags.AlgorithmsUsed |
    FE.FileFieldsRequestFlags.FileFormatID |
    FE.FileFieldsRequestFlags.FilteredObjects |
    FE.FileFieldsRequestFlags.ObjectNames |
    FE.FileFieldsRequestFlags.ObjectsExpandable |
    FE.FileFieldsRequestFlags.TotalObjects |
    FE.FileFieldsRequestFlags.UncompressedSize |
    FE.FileFieldsRequestFlags.UseDebugLog);
```


Managed C++
```
String ^  SearchFilespec      = "*.*";

pFEWrap->SearchFilespec= String::Copy(SearchFilespec);
pFEWrap->AttributeFilter = (FE::ObjectAttributeFlags) (int)
    (FE::ObjectAttributeFlags::FileHeader |
    FE::ObjectAttributeFlags::Index |
    FE::ObjectAttributeFlags::ObjectIsFile |
    FE::ObjectAttributeFlags::ObjectHeader |
    FE::ObjectAttributeFlags::ObjectData |
    FE::ObjectAttributeFlags::ObjectFooter |
    FE::ObjectAttributeFlags::FileFooter |
    FE::ObjectAttributeFlags::UnknownObject |
    FE::ObjectAttributeFlags::PrependedData |
    FE::ObjectAttributeFlags::AppendedData |
    FE::ObjectAttributeFlags::SlackSpace |
    FE::ObjectAttributeFlags::ObjectIsFolder);
pFEWrap->FileFieldsRequested = (FE::FileFieldsRequestFlags) (int)
    (FE::FileFieldsRequestFlags::AlgorithmsSupported |
    FE::FileFieldsRequestFlags::AlgorithmsUsed |
    FE::FileFieldsRequestFlags::FileFormatID |
    FE::FileFieldsRequestFlags::FilteredObjects |
    FE::FileFieldsRequestFlags::ObjectNames |
    FE::FileFieldsRequestFlags::ObjectsExpandable |
    FE::FileFieldsRequestFlags::TotalObjects |
    FE::FileFieldsRequestFlags::UncompressedSize |
    FE::FileFieldsRequestFlags::UseDebugLog);
```

Call the GetFileInfo() method - located in the example ShowFileInfo()

C#
```
FE.ErrorReturnCodes Return Result = FEWrap.GetFileInfo(FEWrap.FullPath, 0);
// Use 0L (Unknown) index unless you are sure of the file format
```

Managed C++
```
FE::ErrorReturnCodes Result = pFEWrap->GetFileInfo(pFEWrap->FullPath, 0L);
// Use 0L (Unknown) index unless you are sure of the file format
```

Display the results from the GetFileInfo() method - located in the example ShowFileInfo()

C#
```
Console.Write("  File Format ID-------] {0}\r\n", FEWrap.FileFormatID.ToString());
Console.Write("  Total Objects--------] {0}\r\n",
    FEWrap.FileTotalObjects.ToString());
Console.Write("  Filtered Objects-----] {0}\r\n",
    FEWrap.FileFilteredObjects.ToString());
if (FEWrap.FileObjectNames != null) {
    Console.Write("  Data Object Names----]");
    for (int counter=0; counter < FEWrap.FileObjectNames.Length; counter++) {
        if (counter > 0)
            Console.Write("\r\n                          ");
        Console.Write(" {0}", FEWrap.FileObjectNames[counter]);
    }
    Console.Write("\r\n");
}
```


Managed C++

```
Console::Write("  File Format ID-------] {0}\r\n",
    pFEWrap->FileFormatID.ToString());
Console::Write("  Total Objects--------] {0}\r\n",
    pFEWrap->FileTotalObjects.ToString());
Console::Write("  Filtered Objects-----] {0}\r\n",
    pFEWrap->FileFilteredObjects.ToString());
if (pFEWrap->FileObjectNames) {
    Console::Write("  Data Object Names----]");
    for (int counter=0; counter < pFEWrap->FileObjectNames->Length; counter++) {
    if (counter > 0)
        Console::Write("\r\n                          ");
        Console::Write(" {0}", pFEWrap->FileObjectNames[counter]);
    }
    Console::Write("\r\n");
}
```

Process Each Object

This step must be performed for each object that you would like to process in the file.

Fill in the GetObjectInfo input values - located in the example ProcessCommandLine()

C#

```
String    File;
String    WorkingFolder;
String    ExpandToPath;

File = "c:\\fitest\\FEAPI100b2.zip";       // Differs from the sample application
WorkingFolder = "c:\\fitest\\";            // Differs from the sample application
ExpandToPath = "c:\\fitest\\expand\\";     // Differs from the sample application

FEWrap.FullPath = String.Copy(File);
FEWrap.EngineWorkingDirectory = String.Copy(WorkingFolder);
FEWrap.ObjectExpandToPath = String.Copy(ExpandToPath);
```

Managed C++

```
String ^  File;
String ^  WorkingFolder;
String ^  ExpandToPath;

File = "c:\\fitest\\FEAPI100b2.zip";       // Differs from the sample application
WorkingFolder = "c:\\fitest\\";            // Differs from the sample application
ExpandToPath = "c:\\fitest\\expand\\";     // Differs from the sample application

pFEWrap->FullPath = String::Copy(File);
pFEWrap->EngineWorkingDirectory = String::Copy(WorkingFolder);
pFEWrap->ObjectExpandToPath = String::Copy(ExpandToPath);
```


Fill in the GetObjectInfo request flags - located in the example ShowObjectInfo()

C#

```
String    SearchFilespec      = "*.*";

FEWrap.SearchFilespec   = String.Copy(SearchFilespec);
FEWrap.AttributeFilter = (FE.ObjectAttributeFlags) (int)
    (FE.ObjectAttributeFlags.FileHeader |
    FE.ObjectAttributeFlags.Index |
    FE.ObjectAttributeFlags.ObjectHeader |
    FE.ObjectAttributeFlags.ObjectData |
    FE.ObjectAttributeFlags.ObjectFooter |
    FE.ObjectAttributeFlags.FileFooter |
    FE.ObjectAttributeFlags.ObjectIsFile |
    FE.ObjectAttributeFlags.UnknownObject);
FEWrap.ObjectFieldsRequested  = (FE.ObjectFieldsRequestFlags) (int)
    (FE.ObjectFieldsRequestFlags.AbsoluteIndex |
    FE.ObjectFieldsRequestFlags.AttributeFilter |
    FE.ObjectFieldsRequestFlags.CompressionAlgorithm |
    FE.ObjectFieldsRequestFlags.CRC |
    FE.ObjectFieldsRequestFlags.EncryptionAlgorithm |
    FE.ObjectFieldsRequestFlags.ExpansionSupport |
    FE.ObjectFieldsRequestFlags.FileAttributes |
    FE.ObjectFieldsRequestFlags.FileFormatID |
    FE.ObjectFieldsRequestFlags.FormatAttributeFilter |
    FE.ObjectFieldsRequestFlags.ObjectInformationOffset |
    FE.ObjectFieldsRequestFlags.ObjectName |
    FE.ObjectFieldsRequestFlags.Path |
    FE.ObjectFieldsRequestFlags.ObjectOffset |
    FE.ObjectFieldsRequestFlags.ObjectSize |
    FE.ObjectFieldsRequestFlags.ObjectSizeCompressed |
    FE.ObjectFieldsRequestFlags.Storage |
    FE.ObjectFieldsRequestFlags.TranslationAlgorithm |
    FE.ObjectFieldsRequestFlags.UseDebugLog);
FEWrap.ObjectExpansionInstructions = (FE.ObjectExpansionFlags) (int)
    (FE.ObjectExpansionFlags.CopyIfExpansionFails |
    FE.ObjectExpansionFlags.CopyWithoutExpansion |
    FE.ObjectExpansionFlags.ExpandToFile);
if (FEWrap.ObjectExpandToPath == "")
    FEWrap.ObjectExpandToPath  = "C:\\temp";
FEWrap.ObjectExpandToFilespec = "%filename%-#%index%-0x%offset%-%opath%-%object%";
```


Managed C++

```
String ^  SearchFilespec      = "*.*";
pFEWrap->SearchFilespec= String::Copy(SearchFilespec);
pFEWrap->AttributeFilter = (FE::ObjectAttributeFlags) (int)
    (FE::ObjectAttributeFlags::FileHeader |
    FE::ObjectAttributeFlags::Index |
    FE::ObjectAttributeFlags::ObjectIsFile |
    FE::ObjectAttributeFlags::ObjectHeader |
    FE::ObjectAttributeFlags::ObjectData |
    FE::ObjectAttributeFlags::ObjectFooter |
    FE::ObjectAttributeFlags::FileFooter |
    FE::ObjectAttributeFlags::UnknownObject);
pFEWrap->ObjectFieldsRequested = (FE::ObjectFieldsRequestFlags) (int)
    (FE::ObjectFieldsRequestFlags::AbsoluteIndex |
    FE::ObjectFieldsRequestFlags::AttributeFilter |
    FE::ObjectFieldsRequestFlags::CompressionAlgorithm |
    FE::ObjectFieldsRequestFlags::EncryptionAlgorithm |
    FE::ObjectFieldsRequestFlags::ExpansionSupport |
    FE::ObjectFieldsRequestFlags::FileAttributes |
    FE::ObjectFieldsRequestFlags::FileFormatID |
    FE::ObjectFieldsRequestFlags::FormatAttributeFilter |
    FE::ObjectFieldsRequestFlags::ObjectFormatID |
    FE::ObjectFieldsRequestFlags::ObjectInformationOffset |
    FE::ObjectFieldsRequestFlags::ObjectName |
    FE::ObjectFieldsRequestFlags::Path |
    FE::ObjectFieldsRequestFlags::ObjectOffset |
    FE::ObjectFieldsRequestFlags::ObjectSize |
    FE::ObjectFieldsRequestFlags::ObjectSizeCompressed |
    FE::ObjectFieldsRequestFlags::Storage |
    FE::ObjectFieldsRequestFlags::TranslationAlgorithm |
    FE::ObjectFieldsRequestFlags::UseDebugLog);
pFEWrap->ObjectExpansionInstructions = (FE::ObjectExpansionFlags) (int)
    (FE::ObjectExpansionFlags::CopyIfExpansionFails |
    FE::ObjectExpansionFlags::CopyWithoutExpansion |
    FE::ObjectExpansionFlags::ExpandToFile);
pFEWrap->ObjectExpandToFilespec= "%filename%-#%index%-0x%offset%-%opath%-%object%";
```


Call the GetObjectInfo() method - located in the example ShowObjectInfo()

C#

```
FEWrap.ObjectIndexStart = 0;
FEWrap.ObjectIndexStop = 0;
// Note: You can process more than 1 object at a time by increasing pFEWrap-
>ObjectIndexStop more than pFEWrap->ObjectIndexStart, but only information about
the last processed object are returned.  The advantage is to expand/extract
multiple objects faster in a single pass.

do {
    String InputFile = FEWrap.FullPath;
    Result = FEWrap.GetObjectInfo(InputFile, 0);
    // Use 0 (Unknown) index unless you are sure of the file format
    pFEWrap->ObjectIndexStart++;
    pFEWrap->ObjectIndexStop++;
} while ((LocalValues.Exit == 0) && ((Result == FE.ErrorReturnCodes.Success) ||
    (Result == FE.ErrorReturnCodes.CreateFileFailed) ||
    (Result == FE.ErrorReturnCodes.Failure) ||
    (Result == FE.ErrorReturnCodes.UnknownObject) ||
    (Result == FE.ErrorReturnCodes.WritingFileFailed) ||
    (Result == FE.ErrorReturnCodes.WrongFormatID) ||
    (Result == FE.ErrorReturnCodes.BadObjectOffset) ||
    (Result == FE.ErrorReturnCodes.DecompressingFileFailed) ||
    (Result == FE.ErrorReturnCodes.WrongObjectSize)));
```

Managed C++

```
pFEWrap->ObjectIndexStart = 0L;
pFEWrap->ObjectIndexStop = 0L;
// Note: You can process more than 1 object at a time by increasing pFEWrap-
>ObjectIndexStop more than pFEWrap->ObjectIndexStart, but only information about
the last processed object are returned.  The advantage is to expand/extract
multiple objects faster in a single pass.

do {
    String ^     InputFile = pFEWrap->FullPath;
    Result = pFEWrap->GetObjectInfo(InputFile, 0L);
    // Use 0L (Unknown) index unless you are sure of the file format
    pFEWrap->ObjectIndexStart++;
    pFEWrap->ObjectIndexStop++;
} while ((!LocalValues->Exit) && ((Result == FE::ErrorReturnCodes::Success) ||
    (Result == FE::ErrorReturnCodes::CreateFileFailed) ||
    (Result == FE::ErrorReturnCodes::Failure) ||
    (Result == FE::ErrorReturnCodes::UnknownObject) ||
    (Result == FE::ErrorReturnCodes::WritingFileFailed) ||
    (Result == FE::ErrorReturnCodes::WrongFormatID) ||
    (Result == FE::ErrorReturnCodes::BadObjectOffset) ||
    (Result == FE::ErrorReturnCodes::DecompressingFileFailed) ||
    (Result == FE::ErrorReturnCodes::WrongObjectSize)));
```


Display the results from the GetObjectInfo() method - located in the example ShowObjectInfo()

You will need to either make a copy of the results on each pass of the Do loop, or display these results right after the data is returned for each object.

C#

```
Console.Write("  Object #-------------] {0,-6:G}\r\n", FEWrap.ObjectAbsoluteIndex);
Console.Write("  Object Name----------] {0}\r\n", FEWrap.ObjectName);
if ((FEWrap.ObjectPath != null) && (FEWrap.ObjectPath.Length != 0L))
   Console.Write("  Object Path----------] {0}\r\n", FEWrap.ObjectPath);
Console.Write("  Object Attributes----] {0:F}\r\n", FEWrap.ObjectAttributes);
Console.Write("  Offset---------------] 0x{0,8} + {1,1:D} bits\r\n",
   FEWrap.ObjectOffset.ToString("x8"), FEWrap.ObjectOffsetBitsAdded);
Console.Write("  Information Offset---] 0x{0,8} + {1,1:D} bits\r\n",
   FEWrap.ObjectInfoOffset.ToString("x8"), FEWrap.ObjectInfoOffsetBitsAdded);
Console.Write("  Uncompressed Size----] 0x{0,8} + {1,1:D} bits\r\n",
   FEWrap.ObjectSize.ToString("x8"), FEWrap.ObjectSizeBitsAdded);
Console.Write("  Compressed Size------] 0x{0,8} + {1,1:D} bits\r\n",
   FEWrap.ObjectSizeCompressed.ToString("x8"),
   FEWrap.ObjectSizeCompressedBitsAdded);
Console.Write("  Storage Condition----] {0:F}\r\n", FEWrap.ObjectStorageCondition);
Console.Write("  Translation Algorithm] {0:F}\r\n",
   FEWrap.ObjectTranslationAlgorithm);
Console.Write("  Compression Algorithm] {0:F}\r\n",
   FEWrap.ObjectCompressionAlgorithm);
Console.Write("  Encryption Algorithm-] {0:F}\r\n",
   FEWrap.ObjectEncryptionAlgorithm);
Console.Write("  Expansion Support----] {0:F}\r\n", FEWrap.ObjectExpansionSupport);
Console.Write("  CRC------------------] 0x{0,8}\r\n",
   FEWrap.ObjectCRC.ToString("x8"));
```

Managed C++

```
Console::Write("  Object #-------------] {0,-6:G}\r\n",
   pFEWrap->ObjectAbsoluteIndex);
Console::Write("  Object Name----------] {0}\r\n",
   pFEWrap->ObjectName);
if ((pFEWrap->ObjectPath) && (pFEWrap->ObjectPath->Length != 0L))
   Console::Write("  Object Path----------] {0}\r\n", pFEWrap->ObjectPath);
Console::Write("  Object Attributes----] {0:F}\r\n", pFEWrap->ObjectAttributes);
Console::Write("  Offset---------------] 0x{0,8} + {1,1:D} bits\r\n",
   pFEWrap->ObjectOffset.ToString("x8"), pFEWrap->ObjectOffsetBitsAdded);
Console::Write("  Information Offset---] 0x{0,8} + {1,1:D} bits\r\n",
   pFEWrap->ObjectInfoOffset.ToString("x8"), pFEWrap->ObjectInfoOffsetBitsAdded);
Console::Write("  Uncompressed Size----] 0x{0,8} + {1,1:D} bits\r\n",
   pFEWrap->ObjectSize.ToString("x8"), pFEWrap->ObjectSizeBitsAdded);
Console::Write("  Compressed Size------] 0x{0,8} + {1,1:D} bits\r\n",
   pFEWrap->ObjectSizeCompressed.ToString("x8"),
   pFEWrap->ObjectSizeCompressedBitsAdded);
Console::Write("  Format ID------------] {0}\r\n", pFEWrap->ObjectFormatID);
Console::Write("  Storage Condition----] {0:F}\r\n",
   pFEWrap->ObjectStorageCondition);
Console::Write("  Translation Algorithm] {0:F}\r\n",
   pFEWrap->ObjectTranslationAlgorithm);
Console::Write("  Compression Algorithm] {0:F}\r\n",
   pFEWrap->ObjectCompressionAlgorithm);
Console::Write("  Encryption Algorithm-] {0:F}\r\n",
   pFEWrap->ObjectEncryptionAlgorithm);
Console::Write("  Expansion Support----] {0:F}\r\n",
   pFEWrap->ObjectExpansionSupport);
Console::Write("  CRC------------------] 0x{0,8}\r\n",
   pFEWrap->ObjectCRC.ToString("x8"));
```


Halt All Processing

If you want to stop the processing of all files, in a multitasking/multithreaded environment, you can call StopFile(). Each of the affected thread calls will then return in a controlled manner, with the error result indicating that the file processing was stopped.

Call the StopFile() method - located in the example main()

C#
```
FEWrap.StopFile();
```

Managed C++
```
pFEWrap->StopFile();
```

File Format Details

This step can be performed when you want detailed information about a file format.

Fill in the GetFormatInfo request flags - located in the example ShowLists()

C#
```
FEWrap.FormatFieldsRequested = (FE.FormatFieldsRequestFlags) (int)
    (FE.FormatFieldsRequestFlags.Content |
    FE.FormatFieldsRequestFlags.Extensions |
    FE.FormatFieldsRequestFlags.MIME |
    FE.FormatFieldsRequestFlags.Name |
    FE.FormatFieldsRequestFlags.Platform |
    FE.FormatFieldsRequestFlags.Storage |
    FE.FormatFieldsRequestFlags.LibraryFilename |
    FE.FormatFieldsRequestFlags.VersionAdded |
    FE.FormatFieldsRequestFlags.VersionUpdated);
```

Managed C++
```
pFEWrap->FormatFieldsRequested = (FE::FormatFieldsRequestFlags) (int)
    (FE::FormatFieldsRequestFlags::Content |
    FE::FormatFieldsRequestFlags::Extensions |
    FE::FormatFieldsRequestFlags::MIME |
    FE::FormatFieldsRequestFlags::Name |
    FE::FormatFieldsRequestFlags::Platform |
    FE::FormatFieldsRequestFlags::Storage |
    FE::FormatFieldsRequestFlags::LibraryFilename |
    FE::FormatFieldsRequestFlags::VersionAdded |
    FE::FormatFieldsRequestFlags::VersionUpdated);
```

Call the GetFormatInfo() method - located in the example ShowLists()

C#
```
int formatID = 12; // The File Investigator ID for PK Zip archive
FE.ErrorReturnCodes Result = FEWrap.GetFormatInfo(formatID);
```

Managed C++
```
int formatID = 12; // The File Investigator ID for PK Zip archive
FE::ErrorReturnCodes Result = pFEWrap->GetFormatInfo(formatID);
```


Display the results from the GetFormatInfo() method - located in the example ShowLists()

C#

```
int       iUseComma;

Console.Write("{0,5} {1,-51}\r\n", stringID, FEWrap.FormatName);
Console.Write("{0,1}.{1,2:D2}.{2,2:D2}.{3,2:D2}\r\n",
    FEWrap.FormatVersionAdded/16777216,
    (FEWrap.FormatVersionAdded%16777216)/65536,
    (FEWrap.FormatVersionAdded%65536)/256,
    FEWrap.FormatVersionAdded%256);
Console.Write("{("{0,1}.{1,2:D2}.{2,2:D2}.{3,2:D2}\r\n",
    FEWrap.FormatVersionUpdated/16777216,
    (FEWrap.FormatVersionUpdated%16777216)/65536,
    (FEWrap.FormatVersionUpdated%65536)/256,
    FEWrap.FormatVersionUpdated%256);
if ((FEWrap.FormatExtensions != null) && (FEWrap.FormatExtensions.Length > 0)) {
    Console.Write("       File Extension(s):     ");
    for (iCounter=0; iCounter < FEWrap.FormatExtensions.Length; iCounter++) {
        if (iCounter > 0) Console.Write(", ");
        Console.Write(".{0}", FEWrap.FormatExtensions[iCounter]);
    }
    Console.Write("\r\n");
}
if ((FEWrap.FormatMIMEs != null) && (FEWrap.FormatMIMEs.Length > 0)) {
    Console.Write("       MIME Label(s):         ");
    iUseComma=0;
    for (iCounter=0; iCounter < FEWrap.FormatMIMEs.Length; iCounter++) {
        if (iCounter > 0) Console.Write(",\r\n                                ");
        Console.Write(FEWrap.FormatMIMEs[iCounter]);
    }
    Console.Write("\r\n");
}
Console.Write("       Platform(s):           ");
iUseComma=0;
if (FEWrap.FormatPlatform == 0)
    Console.Write("{0} (0x00)", FEWrap.GetString(FE.GetStringOptions.Platform, 0,
    Error));
else for (iCounter = 0; (iCounter < 32) && (Error == FE.ErrorReturnCodes.Success);
    iCounter++) {
    uint    uiCounter = (uint) (1 << iCounter);
    if ((FEWrap.FormatPlatform & uiCounter) > 0) {
        if (iUseComma > 0) Console.Write(",\r\n                                ");
        Console.Write("{0} (0x{1:X})",
            FEWrap.GetString(FE.GetStringOptions.Platform, iCounter + 1, Error),
               uiCounter);
        iUseComma++;
    }
}
Console.Write("\r\n");
Console.Write("       Storage Method(s):     ");
iUseComma=0;
if (FEWrap.FormatStorage == 0)
    Console.Write("{0} (0x00)", FEWrap.GetString(FE.GetStringOptions.Storage, 0,
    Error));
else for (iCounter = 0; (iCounter < 32) && (Error == FE.ErrorReturnCodes.Success);
    iCounter++) {
    uint uiCounter = (uint)(1 << iCounter);
    if ((FEWrap.FormatStorage & uiCounter) > 0) {
        if (iUseComma > 0) Console.Write(",\r\n                                ");
        Console.Write("{0} (0x{1:X})", FEWrap.GetString(FE.GetStringOptions.Storage,
            iCounter + 1, Error), uiCounter);
        iUseComma++;
```



```
    }
}
Console.Write("\r\n");
Console.Write("       Content Type(s):       ");
iUseComma=0;
if (FEWrap.FormatContent == 0)
    Console.Write("{0} (0x00)", FEWrap.GetString(FE.GetStringOptions.Content, 0,
    Error));
else for (iCounter = 0; (iCounter < 32) && (Error == FE.ErrorReturnCodes.Success);
    iCounter++) {
    uint uiCounter = (uint)(1 << iCounter);
    if ((FEWrap.FormatContent & uiCounter) > 0) {
        if (iUseComma > 0) Console.Write(",\r\n                                   ");
        Console.Write("{0} (0x{1:X})", FEWrap.GetString(FE.GetStringOptions.Content,
            iCounter + 1, Error), uiCounter);
        iUseComma++;
    }
}
Console.Write("\r\n");
if ((FEWrap.FormatLibraryFilename != null) &&
    (FEWrap.FormatLibraryFilename.Length > 0))
    Console.Write("       File Expander Library: {0}\r\n",
        FEWrap.FormatLibraryFilename);
```

Managed C++

```
int    iUseComma; // counter to decide where to place commas in a list

Console::Write("{0,5} {1,-51}\r\n", stringID, pFEWrap->FormatName);
Console::Write("       Version Added:
    {0,1}.{1,2:D2}.{2,2:D2}.{3,2:D2}\r\n",
    pFEWrap->FormatVersionAdded/16777216,
    (pFEWrap->FormatVersionAdded%16777216)/65536,
    (pFEWrap->FormatVersionAdded%65536)/256,
    pFEWrap->FormatVersionAdded%256);
Console::Write("       Version Updated:
    {0,1}.{1,2:D2}.{2,2:D2}.{3,2:D2}\r\n",
    pFEWrap->FormatVersionUpdated/16777216,
    (pFEWrap->FormatVersionUpdated%16777216)/65536,
    (pFEWrap->FormatVersionUpdated%65536)/256,
    pFEWrap->FormatVersionUpdated%256);
if ((pFEWrap->FormatExtensions) && (pFEWrap->FormatExtensions->Length > 0)) {
    Console::Write("       File Extension(s):     ");
    for (iCounter=0; iCounter < pFEWrap->FormatExtensions->Length; iCounter++) {
        if (iCounter > 0) Console::Write(", ");
        Console::Write(".{0}", pFEWrap->FormatExtensions[iCounter]);
    }
    Console::Write("\r\n");
}
if ((pFEWrap->FormatMIMEs) && (pFEWrap->FormatMIMEs->Length > 0)) {
    Console::Write("       MIME Label(s):         ");
    iUseComma=0;
    for (iCounter=0; iCounter < pFEWrap->FormatMIMEs->Length; iCounter++) {
        if (iCounter > 0) Console::Write(",\r\n                                   ");
        Console::Write(pFEWrap->FormatMIMEs[iCounter]);
    }
    Console::Write("\r\n");
}
Console::Write("       Platform(s):           ");
iUseComma=0;
if (pFEWrap->FormatPlatform == 0)
    Console::Write("{0} (0x00)", pFEWrap->GetString(FE::GetStringOptions::Platform,
    0, Error));
else for (iCounter=0; (iCounter < 32) && (Error == FE::ErrorReturnCodes::Success);
```



```
    iCounter++)
    if (pFEWrap->FormatPlatform & (1<<iCounter)) {
        if (iUseComma) Console::Write(",\r\n                                ");
        Console::Write("{0} (0x{1:X})",
            pFEWrap->GetString(FE::GetStringOptions::Platform, iCounter+1, Error),
            1<<iCounter);
        iUseComma++;
    }
Console::Write("\r\n");
Console::Write("       Storage Method(s):     ");
iUseComma=0;
if (pFEWrap->FormatStorage == 0)
    Console::Write("{0} (0x00)", pFEWrap->GetString(FE::GetStringOptions::Storage,
        0, Error));
else for (iCounter=0; (iCounter < 32) && (Error == FE::ErrorReturnCodes::Success);
        iCounter++)
    if (pFEWrap->FormatStorage & (1<<iCounter)) {
        if (iUseComma) Console::Write(",\r\n                                ");
        Console::Write("{0} (0x{1:X})",
            pFEWrap->GetString(FE::GetStringOptions::Storage, iCounter+1, Error),
            1<<iCounter);
        iUseComma++;
    }
Console::Write("\r\n");
Console::Write("       Content Type(s):       ");
iUseComma=0;
if (pFEWrap->FormatContent == 0)
    Console::Write("{0} (0x00)", pFEWrap->GetString(FE::GetStringOptions::Content,
        0, Error));
else for (iCounter=0; (iCounter < 32) && (Error == FE::ErrorReturnCodes::Success);
        iCounter++)
    if (pFEWrap->FormatContent & (1<<iCounter)) {
        if (iUseComma) Console::Write(",\r\n                                ");
        Console::Write("{0} (0x{1:X})",
            pFEWrap->GetString(FE::GetStringOptions::Content, iCounter+1, Error),
                1<<iCounter);
        iUseComma++;
    }
Console::Write("\r\n");
if ((pFEWrap->FormatAttributeFilterFlags) &&
    (pFEWrap->FormatAttributeFilterFlags->Length > 0)) {
    Console::Write("       Atribute Filter Flags: ");
    for (iCounter=0; iCounter < pFEWrap->FormatAttributeFilterFlags->Length;
        iCounter++) {
        if (iCounter > 0) Console::Write(",\r\n                                  ");
        Console::Write("{0} (0x{1:X})",
            pFEWrap->FormatAttributeFilterFlags[iCounter], 1<<iCounter);
    }
    Console::Write("\r\n");
}
if ((pFEWrap->FormatLibraryFilename) &&
    (pFEWrap->FormatLibraryFilename->Length > 0))
    Console::Write("       File Expander Library: {0}\r\n",
        pFEWrap->FormatLibraryFilename);
```


Appendix A: File Formats Supported

The list of supported file formats for File Expander is 302 entries. We are gradually catching up on adding the file formats to File Expander that are already supported in File Investigator. These lists are very long and updated more often than this manual. The File Expander formats list is provided here, but we recommend that you get the latest version at https://www.FID3.com/lists/formats-fe.html.

Description	File Extensions	Added	Updated	Index
4X Movie Format	4XM	2.05.00	2.05.00	2098
Adobe Portable Document (MacBinary)	PDF	2.05.00	2.06.00	1331
Adobe Portable Document Format (PDF/A)	PDF	2.05.00	2.06.00	4065
Adobe Portable Document Format (PDF/A-1a)	PDF	2.05.00	2.06.00	4062
Adobe Portable Document Format (PDF/A-1b)	PDF	2.05.00	2.06.00	4063
Adobe Portable Document Format (PDF/A-2a)	PDF	3.02.00	3.02.00	4516
Adobe Portable Document Format (PDF/A-2b)	PDF	3.02.00	3.02.00	4517
Adobe Portable Document Format (PDF/A-2u)	PDF	3.02.00	3.02.00	4521
Adobe Portable Document Format (PDF/A-3a)	PDF	3.02.00	3.02.00	4522
Adobe Portable Document Format (PDF/A-3b)	PDF	3.02.00	3.02.00	4523
Adobe Portable Document Format (PDF/A-3u)	PDF	3.02.00	3.02.00	4524
Adobe Portable Document Format (PDF/E)	PDF	3.02.00	3.02.00	4572
Adobe Portable Document Format (PDF/E-1)	PDF	3.02.00	3.02.00	4573
Adobe Portable Document Format (PDF/UA)	PDF	3.02.00	3.02.00	4574
Adobe Portable Document Format (PDF/VT)	PDF	3.02.00	3.02.00	4583
Adobe Portable Document Format (PDF/VT-1)	PDF	3.02.00	3.02.00	4584
Adobe Portable Document Format (PDF/VT-2)	PDF	3.02.00	3.02.00	4585
Adobe Portable Document Format (PDF/VT-2s)	PDF	3.02.00	3.02.00	4586
Adobe Portable Document Format (PDF/X)	PDF	2.05.00	2.06.00	4064
Adobe Portable Document Format (PDF/X-1:1999)	PDF	3.02.00	3.02.00	4683
Adobe Portable Document Format (PDF/X-1:2001)	PDF	3.02.00	3.02.00	4684
Adobe Portable Document Format (PDF/X-1a:2001)	PDF	3.02.00	3.02.00	4575
Adobe Portable Document Format (PDF/X-1a:2003)	PDF	3.02.00	3.02.00	4686
Adobe Portable Document Format (PDF/X-2)	PDF	3.02.00	3.02.00	4685
Adobe Portable Document Format (PDF/X-3:2002)	PDF	3.02.00	3.02.00	4576
Adobe Portable Document Format (PDF/X-3:2003)	PDF	3.02.00	3.02.00	4687
Adobe Portable Document Format (PDF/X-4)	PDF	3.02.00	3.02.00	4577
Adobe Portable Document Format (PDF/X-4p)	PDF	3.02.00	3.02.00	4578
Adobe Portable Document Format (PDF/X-5)	PDF	3.02.00	3.02.00	4579
Adobe Portable Document Format (PDF/X-5g)	PDF	3.02.00	3.02.00	4580

Adobe Portable Document Format (PDF/X-5n)	PDF	3.02.00	3.02.00	4581
Adobe Portable Document Format (PDF/X-5pg)	PDF	3.02.00	3.02.00	4582
Adobe Portable Document Format v1.0	PDF	2.05.00	2.06.00	258
Adobe Portable Document Format v1.1	PDF	3.02.00	3.02.00	4676
Adobe Portable Document Format v1.2	PDF	3.02.00	3.02.00	4677
Adobe Portable Document Format v1.3	PDF	3.02.00	3.02.00	4678
Adobe Portable Document Format v1.4	PDF	3.02.00	3.02.00	4679
Adobe Portable Document Format v1.5	PDF	3.02.00	3.02.00	4680
Adobe Portable Document Format v1.6	PDF	3.02.00	3.02.00	4681
Adobe Portable Document Format v1.7	PDF	3.02.00	3.02.00	4682
Aegis Animation	.	2.05.00	2.05.00	2325
Amiga Contiguous Bitmap	ACBM	2.05.00	2.05.00	2629
Amiga IFF/16SVX Sound	SND,IFF,LBM,SVX	2.05.00	2.05.00	1522
Amiga IFF/8SVX Sound	SND,IFF,LBM,8SV,SVX,8SVX	2.05.00	2.05.00	25
Amiga Interchange File Format Image	LBM,IFF,ILM,BBM,ILBM,BLK	2.05.00	2.05.00	6
Amiga Metafile Format	AMF,AMFF	2.05.00	2.05.00	1819
Amiga SoundTracker Music	EMD	2.05.00	2.05.00	1602
Apple Email Message	EML,TXT,EMLX	2.06.00	2.06.00	536
ASDG Split File	SPLT	2.05.00	2.05.00	2639
Atlantis Facsimile Image	FAXX	2.05.00	2.05.00	2628
Audio IFF Compressed Sound	AIFC,AIFF,AIF,IFF,SND	2.05.00	2.05.00	1693
Audio IFF Sound	AIFF,AIFC,AIF,IFF,SND	2.05.00	2.05.00	323
Audio Manager Module (RIFF)	AMM	2.05.00	2.05.00	2653
AXS Realtime Analog Synthesizer Module	AXS	2.05.00	2.05.00	2658
Berkeley UNIX Mailbox Format	MBOX,MBX	2.06.00	2.06.00	862
Capture Classic Filler Template	JDT	2.00.00	3.02.00	2514
Cinema 4D V4 Data	MC4D,CINEMA4D	2.05.00	2.05.00	2290
Common Musical Score	CMUS	2.05.00	2.05.00	2631
Computerized Speech Lab Audio	NSP	2.05.00	2.05.00	1717
Concatenate (CAT) Interchange File Format	IFF,FTXT,FTX,RBS	2.05.00	2.05.00	3966
Corel ClipArt ScrapBook	SRB,SRI	1.00.00	3.02.00	2203
Corel Draw Raster (Intel)	CDR,PAT,CDT	2.05.00	2.05.00	241
Corel Draw Raster (Motorola)	CDR,PAT	2.05.00	2.05.00	2597
Corel Metafile Exchange Image (Intel)	CMX,PAT,CDR	2.05.00	2.05.00	631
Corel Metafile Exchange Image (Motorola)	CMX,PAT,CDR	2.05.00	2.05.00	2279
Corel Presents Presentation	PQF,SHW,CPR	3.00.00	3.02.00	634
Creative SoundFont Manager Data	SFM	2.05.00	2.05.00	1414

Crystal Report	RPT	1.00.00	3.02.00	1020
DAKX Compressed Audio	DAX	2.05.00	2.05.00	1710
Deluxe Paint Perspective Move	PRSP	2.05.00	2.05.00	2637
Deluxe Video Construction Set Video	.	2.05.00	2.05.00	3251
DesignCAD Drawing	DCD	2.00.00	3.02.00	2513
Digital Sheet Music (Motorola)	MTD	2.05.00	2.05.00	1918
Direct3D Audio Driver	CSP	2.05.00	2.05.00	3809
DjVu Document Image	DJVU,DJV	2.05.00	2.05.00	1826
Document Style Definition	STYLEDEF	1.00.01	3.02.00	582
DOS Sound Interface Kit Module Music	DSM	2.05.00	2.05.00	2574
Draw Native Drawing (Compress+Preview)	ZMF	1.00.00	1.00.00	2137
eGram Telegram Message	.	2.06.00	2.06.00	3320
Electric Image 3D Image	FACT	2.05.00	2.05.00	1483
Eudora Email Message	EML,TXT	2.06.00	2.06.00	933
Evolution Email Message	EML,TXT	2.06.00	2.06.00	538
Family Tree Database	FTW,FBK	1.00.00	3.02.00	1359
Family Tree System File	BIN	1.00.00	3.02.00	1970
Fanta Vision Movie (IFF)	FANT	2.05.00	2.05.00	2601
Flash Movie	FLA	1.00.00	3.02.00	727
FlashPix Bitmap	FPX,CPX,FMP	1.00.00	3.02.00	740
Flow Document	.	2.05.00	2.05.00	2632
Framer Animation	.	2.05.00	2.05.00	2630
Global Message Exchange Email Message	EML,TXT	2.06.00	2.06.00	540
Gmail Email Message	EML,TXT	2.06.00	2.06.00	539
Hallmark Card Studio Card	HMK	3.00.00	3.02.00	3633
Home World 2 ROT Graphic Image	ROT	2.05.00	2.05.00	1866
Horde Internet Messaging Program (IMP) Email Message	EML,TXT	2.06.00	2.06.00	542
Hotmail Email Message	EML,TXT	2.06.00	2.06.00	578
IFF 3D Object	.	2.05.00	2.05.00	2634
IFF Raster Font	.	2.05.00	2.05.00	3248
IFF Retargetable Graphic Image	RGFX,RGX	2.05.00	2.05.00	2652
IFF Sound Sample	SAMP	2.05.00	2.05.00	2638
IFF Tracker Song	TRKR	2.05.00	2.05.00	2642
IFF Vector Font	.	2.05.00	2.05.00	3249
IFF YUV Image	YUVN	2.05.00	2.05.00	2643
Image Professional Compressed Bitmap	PMBC	2.05.00	2.05.00	2636
Imagine 3D Object	TDDD,TDD	2.05.00	2.05.00	2293

Imagine Staging Stage Editor File	ISTG	2.05.00	2.05.00	1881
Impress Presentation / Template	SDD,VOR	1.00.01	3.02.00	2540
Info-Zip Self Extracting Archive	EXE	2.01.00	2.01.00	3165
InstallShield Log	ILG	1.00.00	3.02.00	1024
Interchange File Format	IFF	2.05.00	2.05.00	233
Internet Message	EML,TXT	2.06.00	2.06.00	703
Internet Message (MIME)	EML,TXT,MHT,MIM,MME,MIME,MSG	2.06.00	2.06.00	1262
Internet Message (UU-Encoded)	EML,TXT	2.06.00	2.06.00	1263
Java Archive	JAR,ZIP,BASE,MZP,*	1.00.00	1.00.00	1432
JungUm Office Document	GUL,BKG	1.00.00	3.02.00	2510
Karbon Document	KARBON	1.00.00	1.00.01	2475
KChart Document	CHRT	1.00.00	1.00.01	2476
KFormula Document	KFO	1.00.00	1.00.01	2477
Kivio Document	FLW	1.00.00	1.00.01	2478
Knowledge Interchange File Format Image	KIF	2.05.00	2.05.00	3876
Kontour Document	KON	1.00.00	1.00.01	2479
KPresenter Slide Show (XML)	KPR,KPT	1.00.00	1.00.01	2480
KSpread Document	KSP	1.00.00	1.00.01	2481
KWord Document / Template	KWD,KWT	1.00.00	1.00.01	2482
LA Lossless Compressed Audio	LA	2.05.00	2.05.00	1701
LightWave 3D FPBM Image	FPBM	2.05.00	2.05.00	2275
LightWave 3D Layered Object	LWLO	2.05.00	2.05.00	3244
LightWave 3D Object	LWOB,LWO	2.05.00	2.05.00	845
LightWave 3D Scene	LWSC,LWS	2.05.00	2.05.00	847
Link Notebook Chart	LNB,ANB	1.00.00	3.02.00	1238
LIST Interchange File Format	IFF,FTXT,FTX	2.05.00	2.05.00	3967
Lotus Approach Document	APR,MPR	2.00.00	3.02.00	2512
Low Bitrate Packer Compressed Audio	LB	2.05.00	2.05.00	1714
Lucas Arts Sound (SAD)	SAD	2.05.00	2.05.00	70
Macintosh IFF Picture	.	2.05.00	2.05.00	3250
Macromedia Director Cast	CST	2.05.00	2.05.00	3758
Macromedia Director File (Intel)	DCR	2.05.00	2.05.00	3970
Macromedia Director File (Macintosh)	DCR	2.05.00	2.05.00	285
Macromedia Director Movie (Intel)	DXR,CXT,DIR	2.05.00	2.05.00	271
Macromedia Director Movie (Macintosh)	DXR,CXT,DIR	2.05.00	2.05.00	169
MathVision Matrix	.	2.05.00	2.05.00	2633
MAUD Audio Sample	MAUD	2.05.00	2.05.00	2586

Microsoft Mail Email Message	EML,TXT	2.06.00	2.06.00	545
Microsoft Outlook 2000 IMO Email Message	EML,TXT	2.06.00	2.06.00	574
MIME HTML Web Page Archive	MHT,EML,HTM,MAFF,MHTML	2.06.00	2.06.00	1910
Mozilla Browser Extension Package	XPI	1.00.00	1.00.00	2511
MS Access Database Project	ADP	1.00.00	3.02.00	483
MS Access Database Template (Open XML)	ACCDT	1.00.00	1.00.00	2396
MS Access Database Wizard Template	MDZ	1.00.00	3.02.00	327
MS Access Report / Snapshot	RPT,SNP	1.00.00	3.02.00	1521
MS ActiveMovie Graph	GRF	1.00.00	3.02.00	769
MS Agent/Assistant Character	ACS,ACG,ACF	1.00.00	3.02.00	1207
MS Audio/Visual Interleave (Intel)	AVI,VFW,DA2	2.05.00	2.05.00	234
MS Audio/Visual Interleave (Motorola)	AVI,VFW	2.05.00	2.05.00	2599
MS ClipArt Gallery	CAG	1.00.00	3.02.00	328
MS Compound Document (OLE) (General)	OBD,PPT,DOC,XLS	1.00.00	3.02.00	274
MS Data Transformation Services	DTS	1.00.00	3.02.00	2194
MS Developer Studio Workspace Options	OPT	1.00.00	3.02.00	914
MS Excel 2007-2010 Spreadsheet (Open XML)	XLSX,XLAM	1.00.00	1.00.00	2209
MS Excel Graph	GRA	1.00.00	3.02.00	768
MS Excel Spreadsheet (Open XML,Binary)	XLSB	2.00.00	2.00.00	2626
MS Excel Worksheet/Template (OLE)	XLS,XLA,XLT,XLB,WWS,XLM	1.00.00	3.02.00	111
MS Management Console	MSC	1.00.00	3.02.00	2199
MS MIDI Instrument Definition (Motorola)	IDD,IDF	2.05.00	2.05.00	2267
MS Midi Musical Instrument Song (Intel)	RMI	2.05.00	2.05.00	311
MS Midi Musical Musical Instrument Song (Motorola)	RMI	2.05.00	2.05.00	2270
MS Multi-Media Movie	MMM	2.05.00	2.05.00	324
MS Network Shortcut	MCC	1.00.00	3.02.00	237
MS Office Binder Document/Template/Wizard	OBD,OBT,OBZ,BIN	1.00.00	3.02.00	326
MS Office Data (Open XML) (General)	XLSX,DOCX,PPTX,XLSM,DOCM,PPTM,???X,???M	1.00.00	1.00.00	2207
MS Office Macro Reference (OLE)	MSO	1.00.01	3.02.00	591
MS Office Outlook 2003 Email Message	EML,TXT	2.06.00	2.06.00	571
MS Outlook Express Email Message	EML,TXT	2.06.00	2.06.00	575
MS Outlook Form Template	OFT	1.00.00	3.02.00	909
MS Outlook Message	MSG	1.00.00	3.02.00	148
MS Outlook Send/Receive Settings	SRS	1.00.00	3.02.00	2204
MS Picture It! Multilayer Picture	MIX	1.00.00	3.02.00	1358
MS Pictures	PCS	1.00.00	3.02.00	329
MS PipeLine Component File	PCF	1.00.00	3.02.00	1464

MS PowerPoint 2007-2010 Presentation (Open XML)	PPTX	1.00.00	1.00.00	2210
MS PowerPoint Slides (OLE)	PPT,PPA,POT,PPS	1.00.00	3.02.00	164
MS PowerPoint Wizard	PWZ	1.00.00	3.02.00	985
MS Project	MPP,MPT	1.00.00	3.02.00	348
MS Publisher Document	PUB	1.00.00	3.02.00	1356
MS Publisher Job Submission Publication	JSP	1.00.00	3.02.00	127
MS Speech Recognition Engine Articles	ART	1.00.00	3.02.00	2198
MS Visio 3/4 Document/Drawing/Shapes/Template	VSD,VSS,VST,VT	1.00.00	3.02.00	1130
MS Visual Basic Active Document	VBD	1.00.00	3.02.00	1357
MS Visual Studio Macro	VSMACRO,VSMACROS	1.00.00	3.02.00	2197
MS Visual Studio Solution User Options	SUO	1.00.00	3.02.00	2195
MS Windows Animated Cursor (Intel)	ANI,CON	2.05.00	2.05.00	308
MS Windows Animated Cursor (Motorola)	ANI,CON	2.05.00	2.05.00	2262
MS Windows Color Palette (Intel)	PAL	2.05.00	2.05.00	320
MS Windows Color Palette (Motorola)	PAL	2.05.00	2.05.00	2263
MS Windows Color Scheme	SCM	1.00.00	3.02.00	2202
MS Windows Installer Package / Wizard (OLE)	MSI,MST,DAT,MSM,WID,WIM,CUB,*	1.00.00	3.02.00	1211
MS Windows Installer/Upgrade Patch	MSP,DAT	1.00.00	3.02.00	1361
MS Windows Movie Maker Project	MSWMM	1.00.00	3.02.00	2200
MS Windows Sound Mix	MIX	2.05.00	2.05.00	254
MS Windows Wave Sound (Intel)	WAV	2.05.00	2.05.00	166
MS Windows Wave Sound (Motorola)	WAV	2.05.00	2.05.00	2272
MS Word 2007-2010 Document (Open XML)	DOCX	1.00.00	1.00.00	2208
MS Word 97-2003 Document (OLE)	DOC,DOT,WIZ,WZS,WRI,WBK	1.00.00	3.02.00	229
MS Works Database (OLE)	WDB,DB	2.00.00	3.02.00	567
MS Works Database Wizard	WWD,WDB	1.00.00	3.02.00	2392
MS Works for DOS Document	WPS,WWP	1.00.00	3.02.00	134
MS Works for Mac 4 Database/Doc/Sheet (OLE)	DB4,WP4,SS4	2.00.00	3.02.00	2757
MS Works for Windows 3 Document (OLE)	WPS,WWP	2.00.00	3.02.00	2624
MS Works Portfolio	WSB	1.00.00	3.02.00	1140
MS Works Spreadsheet (OLE)	WWS,WLR	2.00.00	3.02.00	2625
MS Write / Word Backup	WRI,WBK,TMP,DOC	1.00.00	3.02.00	1136
MSN Application Extension	NAV	1.00.01	3.02.00	580
MTV Movie	AMV	2.05.00	2.05.00	2699
Music Construction Set Instrument	*	2.05.00	2.05.00	3941
Music Construction Set Song	*	2.05.00	2.05.00	3942
Mutt Email Message	EML,TXT	2.06.00	2.06.00	546

Netscape Email Message	EML	2.06.00	2.06.00	3563
Norton Virus Definitions (compressed)	.	1.00.00	1.00.00	935
Notation Interchange File Format (Intel)	NIF,RMI	2.05.00	2.05.00	2271
Notation Interchange File Format (Motorola)	NIF,RMI	2.05.00	2.05.00	1926
OmniForm Form	OFM	2.00.00	3.02.00	2515
Open WebMail Email Message	EML,TXT	2.06.00	2.06.00	550
OpenDocument Chart	ODC	1.00.00	1.00.01	2488
OpenDocument Chart Template	OTC	1.00.00	1.00.01	2489
OpenDocument Formula	ODF	1.00.00	1.00.01	2490
OpenDocument Formula Template	OTF	1.00.00	1.00.01	2491
OpenDocument Graphics	ODG	1.00.00	1.00.01	2492
OpenDocument Graphics Template	OTG	1.00.00	1.00.01	2493
OpenDocument Image	ODI	1.00.00	1.00.01	2494
OpenDocument Image Template	OTI	1.00.00	1.00.01	2495
OpenDocument Office Document	ODT,O?I,O?G,O?F,O?C,O?T,OD?,OT?	1.00.01	1.00.01	1499
OpenDocument Presentation	ODP	1.00.00	1.00.01	2496
OpenDocument Presentation Template	OTP	1.00.00	1.00.01	2497
OpenDocument Spreadsheet	ODS	1.00.00	1.00.01	2498
OpenDocument Spreadsheet Template	OTS	1.00.00	1.00.01	2499
OpenDocument Text	ODT	1.00.00	1.00.01	2500
OpenDocument Text Master	OTM	1.00.00	1.00.01	2501
OpenDocument Text Template	OTT	1.00.00	1.00.01	2502
OpenDocument Text Web	OTH	1.00.00	1.00.01	2503
OpenOffice Database	ODB	1.00.01	1.00.01	2615
OpenOffice Drawing / Template	SXD,STD	1.00.01	1.00.01	2610
OpenOffice Impress Presentation / Template	SXI,STI	1.00.01	1.00.01	2612
OpenOffice Math Formula	SXM	1.00.01	1.00.01	2611
OpenOffice Spreadsheet / Template	SXC,STC	1.00.01	1.00.01	2613
OpenOffice Text Document / Template	SXW,STW	1.00.01	1.00.01	2614
PageMaker 6.5 Document	P65	1.00.00	3.02.00	2196
Pagestream Document	DOC	2.05.00	2.05.00	2315
PhotoImpact Graphic Image	UFO	1.00.00	3.02.00	2205
Pine Email Message	EML,TXT	2.06.00	2.06.00	577
PKCS #7 Secure MIME Message	P7M,EML,TXT	2.06.00	2.06.00	1810
PKZip Archive	ZIP,JAR,WMZ	1.00.00	1.00.00	12
PKZip Archive (MacBinary)	ZIP	2.01.00	2.01.00	1735
PKZip Archive Split File	CA1,CA2,CA3,CA4,CA5,CA6,CA7,CAn	1.00.00	1.00.00	1272

PKZip Self Extracting Archive	EXE,ZIP	2.01.00	2.01.00	283
Program Traceback	PGTB	2.05.00	2.05.00	2635
PROP Interchange File Format	*	2.05.00	2.05.00	3968
Protracker 3 Module Music	MOD	2.05.00	2.05.00	2580
Provector 2D Image	DR2D	2.05.00	2.05.00	2278
ProWrite Document	WOR,IFF	2.05.00	2.05.00	228
PSpice Capture Design/Library/Symbols	DSN,DBK,OLB,OBK	1.00.00	3.02.00	331
PureVoice Audio	QCP	2.05.00	2.05.00	1090
QED Symbolic Math Data Structure	TREE	2.05.00	2.05.00	2641
Quatro Pro Data	QPW,CLP	1.00.00	3.02.00	155
RAR Compressed Archive	RAR,Rnn,Snn	2.07.00	2.07.00	1252
RAR Self Extracting Archive	EXE	2.07.00	2.07.00	3164
Real 3D Rendering	.	2.05.00	2.05.00	3245
RealLegal Binder Document	PEX	1.00.00	1.00.00	1802
Reason Song	RPS	2.05.00	2.05.00	1724
ReBirth Song	RBS	2.05.00	2.05.00	1720
ReSOF Compressed Archive	SOF	1.00.00	1.00.00	1636
Resource Interchange File Format (Intel)	RIFF,RIF,AVI,WAV,BND	2.05.00	2.05.00	212
Resource Interchange File Format (Motorola)	RIFX,RIF,AVI,WAV,BND,RIFF	2.05.00	2.05.00	3969
Sculpt3D Scene Model	SCENE	2.05.00	2.05.00	1034
Simple Musical Score IFF MIDI	SMU,IFF,SMUS,MUS	2.05.00	2.05.00	198
SolidWorks 2001 Assembly	SLDASM	3.00.00	3.02.00	3668
SolidWorks 2001 Drawing	SLDDRW	3.00.00	3.02.00	4346
SolidWorks 2001 Part	SLDPRT	3.00.00	3.02.00	3669
Sonic Skin	SKN	1.00.00	1.00.00	2389
Sound Smith Sound	*	2.05.00	2.05.00	3940
StarCalc Spreadsheet / Template	SDC,VOR	1.00.01	3.02.00	2607
StarDraw Image / Template	SDA,VOR	1.00.01	3.02.00	2608
StarMath Formula	SMF	1.00.01	3.02.00	2541
StarWriter Document / Template	SDW,VOR	1.00.01	3.02.00	1052
Super Smooth Animation	SSA	2.05.00	2.05.00	3247
TextCraft Document	FTXT,FTX,IFF	2.05.00	2.05.00	199
The Bat! Email Message	EML,TXT	2.06.00	2.06.00	2604
Thumbs Plus Database	DB,TDB	1.00.00	3.02.00	1089
Thunderbird Email Message	EML,TXT	2.06.00	2.06.00	2605
ToolMaker User Interface Design	TMUI	2.05.00	2.05.00	2640
Turbo Silver 12bit RGB Image	RGBN	2.05.00	2.05.00	2594

Turbo Silver 24bit RGB Image	RGB8	2.05.00	2.05.00	2593
TV Paint Image	.	2.05.00	2.05.00	827
Uhuru Sound Software Musical Score	.	2.05.00	2.05.00	3253
Uhuru Sound Software Voice	.	2.05.00	2.05.00	3252
Wing Commander III MVE Movie	MVE	2.05.00	2.05.00	2105
WinZip Self Extracting Archive	EXE,ZIP	2.01.00	2.01.00	584
WordPerfect Document	DOC,WP,WKB,WPD,WPT,WP#,*	1.00.00	3.02.00	157
WordPerfect Document Template	WPX	1.00.00	3.02.00	1360
WordPerfect Graphic Image	WPG	1.00.00	3.02.00	160
Workshare DeltaView File	WDF	1.00.00	3.02.00	1255
XML Paper Specification Document (Open XML)	XPS	2.00.00	2.00.00	2148
YAFA animation	.	2.05.00	2.05.00	3246
Yahoo! Mail Email Message	EML,TXT	2.06.00	2.06.00	2606

Appendix B: Interface Methods

GetFileInfo

Extracts summary information about a file.

ErrorReturnCodes GetFileInfo(String * InputFilename, long InputFileFormatID)

Routine	Compatibility
GetFileInfo	MS C#, MS Visual C++ (managed)

Return Value

An error value is returned to indicate whether the method executed successfully. An enum value of Success (0) indicates that the method executed successfully. See ErrorReturnCodes in Appendix C, for a list of other possible error return codes.

Parameters

InputFilename

The full path and filename of a file to be analyzed.

InputFileFormatID

Description Database index value that indicates the file format that the specified file is believed to be. If unknown, this parameter may be assigned the value zero (0L). See Appendix A for a list of possible index values.

Public Input Variables

AttributeFilter

An object type flag filter assigned before calling GetFileInfo, to specify what types of objects to include in the analysis. (ex: FE::ObjectAttributeFlags::ObjectData) See ObjectAttributeFlags in Appendix C, for a list of other possible flag values.

FileFieldsRequested

A flag value used to determine what information fields are populated when the method returns. (ex: FE::FileFieldsRequestFlags::TotalObjects) See FileFieldsRequestFlags in Appendix C, for a list of other possible flag values.

SearchFilespec

A filename string filter assigned before calling GetFileInfo, in order to filter which object names are included in the analysis. (ex: “*.*”)

Public Output Variables

FileFilteredObjects

A total of all of the objects that passed through the specified filter in the analyzed file.

FileFormatID

Description Database index value that indicates the database entry verified to be the correct file format for the file analyzed. See Appendix A for a list of possible index values.

FileObjectNames

A list of the object names that passed through the specified filter in the analyzed file.

FileTotalObjects

A total of all of the objects found in the analyzed file.

FileUncompressedSize;

The total uncompressed size, in bytes, for all objects found in the file

Remarks

The GetFileInfo function analyzes a file in order to confirm/identify what type of file it is and extract summary information about the objects that it contains.

GetFormatInfo

Queries the Description Database for details about a specific file format.

ErrorReturnCodes GetFormatInfo(long InputFileFormatID)

Routine	Compatibility
GetFormatInfo	MS C#, MS Visual C++ (managed)

Return Value

An error value is returned to indicate whether the method executed successfully. An enum value of Success (0) indicates that the method executed successfully. See ErrorReturnCodes in Appendix C, for a list of other possible error return codes.

Parameters

InputFileFormatID

Description Database index value that indicates the database entry requested. See Appendix A for a list of possible index values.

Public Input Variable

FormatFieldsRequested

A flag value used to determine what information fields are populated when the method returns. (ex: FE::FormatFieldsRequestFlags::Name) See FormatFieldsRequestFlags in Appendix C, for a list of other possible flag values.

Public Output Variables

FormatContent

A flag value used to determine what types of content can be found in the file format type. See ContentFlags in Appendix C, for a list of possible flag values.

FormatExtensions

A list of the valid file extensions that can be used in a filename for files of the analyzed file format type. These are typically the file extensions that will allow applications to open and use the file correctly.

FormatLibraryFilename

A string value that identifies which File Expander library was used to identify and analyze the file. (ex: fearchive.fel)

FormatMIMEs

A list of the valid MIME types that can be used to identify files of the analyzed file format type in email and web page documents.

FormatName

A string value, obtained from the Description Database, that provides a short descriptive name for the file format. See Appendix A for a list of possible name/description values.

FormatPlatform

A flag value used to determine what platform(s)/operating system(s) the file format type is typically found on. See PlatformFlags in Appendix C, for a list of possible flag values.

FormatStorage

A flag value used to determine what storage methods are used in the file format type. See StorageFlags in Appendix C, for a list of possible flag values.

FormatVersionAdded

A number value that contains the version of File Expander that the file format was initially added in. This is a 32bit value in which each 8 bits specify a part of the version. (ex: 0x01020304 = version 01.02.03.04)

FormatVersionUpdated

A number value that contains the version of File Expander that the file format was last updated in. This is a 32bit value in which each 8 bits specify a part of the version. (ex: 0x01020304 = version 01.02.03.04)

Remarks

The GetFormatInfo method queries a database entry structure from the Descriptions Database in order to obtain a detailed description of the specified type of file.

GetObjectInfo

Extracts objects and information about the object in a file.

ErrorReturnCodes GetObjectInfo(String * InputFilename, long InputFileFormatID)

Routine	Compatibility
GetObjectInfo	MS C#, MS Visual C++ (managed)

Return Value

An error value is returned to indicate whether the method executed successfully. An enum value of Success (0) indicates that the method executed successfully. See ErrorReturnCodes in Appendix C, for a list of other possible error return codes.

Parameters

InputFilename

The full path and filename of a file to be analyzed.

InputFileFormatID

Description Database index value that indicates the file format that the specified file is believed to be. If unknown, this parameter may be assigned the value zero (0L). See Appendix A for a list of possible index values.

Public Input Variables

AttributeFilter

An object type flag filter assigned before calling GetObjectInfo, to specify what types of objects to include in the analysis. (ex: FE::ObjectAttributeFlags::ObjectData) See ObjectAttributeFlags in Appendix C, for a list of other possible flag values.

ObjectExpandToFilespec

A string value that specifies how to create the filename for each new file created from expanding an object. (ex: "%filename%-#%index%-0x%offset%-%opath%-%object%")

Variables available:

%filename%	Filename of the original file being analyzed.
%index%	Zero based index value of the object within the parent file.
%object%	Object name.
%offset%	Hexadecimal byte offset of the object within the parent file.
%opath%	Object's path within the parent file. This includes any hierarchal relationship between objects.

ObjectExpandToPath

A string value that specifies where to create the file(s) that the object(s) is/are expanded into. (ex: "c:\temp")

ObjectExpansionInstructions

A flag value used to determine what whether to expand each object to a file. (ex: FE::ObjectExpansionFlags::ExpandToFile) See ObjectExpansionFlags in Appendix C, for a list of other possible flag values.

ObjectFieldsRequested

A flag value used to determine what information fields are populated when the method returns. (ex: FE::ObjectFieldsRequestFlags::ObjectName) See ObjectFieldsRequestFlags in Appendix C, for a list of other possible flag values.

ObjectIndexStart

A zero based object index number value that specifies which object to start processing from the specified file. All prior objects in the file will be ignored.

ObjectIndexStop

A zero based object index number value that specifies the last object to process in the specified file. This value must be equal to or greater that the value provided in ObjectIndexStart. When the two index values are equal, only one object will be processed. When ObjectIndexStop is greater that ObjectIndexStart, then multiple objects will be processed, and optionally expanded to files, but the information fields returns will only contain information about the last object processed. All remaining objects in the file will be ignored.

SearchFilespec

A filename string filter assigned before calling GetObjectInfo, in order to filter which object names are included in the analysis. (ex: "*.*")

Public Output Variables

ObjectAbsoluteIndex

A zero based index value that indicates how many objects appeared in the file before the object being reported on, regardless of any filters used.

ObjectCompressionAlgorithm

An enumeration value that represents what compression algorithm is being used to store the object in the parent file. It is assumed that the object is not using any compression algorithm unless otherwise indicated in this field. (ex: PKWare)

Possible values for type ObjectCompressionAlgorithms:

None (0x00)	The object is not using any compression algorithm.
Unknown (0x01)	The compression algorithm could not be discovered.
PKWare (0x02)	PKWare is being used to store the object in the file.
WinZip (0x03)	WinZip is being used to store the object in the file.
Tokenizing (0x04)	Tokenizing is being used to store the object in the file.
Deflate64 (0x05)	Deflate64 is being used to store the object in the file.
BZIP2 (0x06)	BZIP2 is being used to store the object in the file.
IBMTerse (0x07)	IBMTerse is being used to store the object in the file.
IBMLZ77z (0x08)	IBMLZ77z is being used to store the object in the file.
WavPack (0x09)	WavPack is being used to store the object in the file.
PPMdvI (0x0A)	PPMdvI is being used to store the object in the file.
zlib (0x0B)	zlib is being used to store the object in the file.

ObjectCRC

A cyclic redundancy code calculated to be used in comparing the content of multiple objects.

ObjectEncryptionAlgorithm

An enumeration value that represents what encryption algorithm is being used to store the object in the parent file. It is assumed that the object is not using any encryption algorithm unless otherwise indicated in this field. (ex: WinZipAES)

Possible values for type ObjectEncryptionAlgorithms:

None (0x00)	The object is not using any encryption algorithm.
Unknown (0x01)	The encryption algorithm could not be discovered.
PKWare (0x02)	PKWare is being used to store the object in the file.
WinZipAES (0x03)	WinZipAES is being used to store the object in the file.

ObjectExpansionSupport

A flag value that indicates what methods of expansion the File Expander Engine is capable of using on the object in the parent file. It is assumed that the object can at least be copied (as is) unless the Copy flag is not present. If other fields indicate that the object is compressed, translated and/or encrypted, but that support doesn't appear in this field, then the object can only be copied as is. (ex: Compression)

Possible values for type ExpansionSupportFlags:

None (0x00)	The object cannot be copied, translated, unencrypted nor uncompressed.
Translation	The object can be translated.
Compression	The object can be uncompressed.
Encryption	The object can be unencrypted.
Copy	The object can be copied as is.

ObjectInfoOffset

A zero based number value that represents how many bytes preceded the start of a block of information about the object in the parent file.

ObjectInfoOffsetBitsAdded

A zero based number value that represents how many bits need to be added to the number of bytes in ObjectInfoOffset to represent the actual bit location of the start of the information in the parent file.

ObjectName

A string value containing the name of the object that passed through the specified filter in the analyzed file. If the object is an archived file, then this will be the archived file's original filename.

ObjectOffset

A zero based number value that represents how many bytes preceded the start of the object in the parent file.

ObjectOffsetBitsAdded

A zero based number value that represents how many bits need to be added to the number of bytes in ObjectOffset to represent the actual bit location of the start of the object in the parent file.

ObjectPath

A string value containing the path/hierarchy of the object that passed through the specified filter in the analyzed file. If the object is an archived file, then this will be the archived file's original path. Otherwise, this will indicate the hierarchal relationship between this object and other objects located in the parent file.

ObjectSize

A zero based number value that represents how many bytes are required to store the object in a new file, once the object in uncompressed, unencrypted, translated or simply copied.

ObjectSizeBitsAdded

A zero based number value that represents how many bits need to be added to the number of bytes in ObjectSize to represent the actual size in bits of the object.

ObjectSizeCompressed

A zero based number value that represents how many bytes the object is using in the parent file, if that size is different from the uncompressed, unencrypted translated size provided in ObjectSize. If the compressed size is no different, then this variable is set to zero.

ObjectSizeCompressedBitsAdded

A zero based number value that represents how many bits need to be added to the number of bytes in ObjectSizeCompressed to represent the actual size in bits of the object.

ObjectStorageCondition

A flag value that represents what condition(s) the object is in, while stored in the parent file. It is assumed that the object is not fragmented, incomplete nor corrupted and is completely contained within the file being analyzed, unless otherwise indicated in this field. (ex: Consecutive)

Possible values for type ObjectStorageConditionFlags:

Unknown (0x00)	The condition of the object could not be discovered.
Consecutive (0x01)	All blocks of the object are stored consecutively (one right after the other) within the file
Fragmented (0x02)	One or more of the object's blocks is/are not stored consecutively within the file being analyzed. A fragmented block may be located later in the file or closer to the beginning of the file, depending on the ability of the parent file's file format to cope with fragmented data blocks.
Incomplete (0x04)	One or more of the object's blocks is/are missing due to deletion and/or corruption in the parent file being analyzed.
SpanFiles (0x08)	One or more of the object's blocks is/are located in one or more different parent file(s).
Corrupted (0x10)	One or more of the object's blocks is/are corrupted.

ObjectTranslationAlgorithm

An enumeration value that represents what translation algorithm is being used to store the object in the parent file. It is assumed that the object is not using any translation algorithm unless otherwise indicated in this field. (ex: MIME)

Possible values for type ObjectTranslationAlgorithms:

None (0x00)	The object is not using any translation algorithm.
Unknown (0x01)	The translation algorithm could not be discovered.
MIME (0x02)	MIME is being used to store the object in the file.
CB64 (0x03)	CB64 is being used to store the object in the file.
UUencode (0x04)	UUencode is being used to store the object in the file.

Remarks

The GetFileInfo function analyzes a file in order to confirm/identify what type of file it is and extract summary information about the objects that it contains.

GetString

Queries a string value from the Description Database.

String * Engine::GetString(GetStringOptions Type, long StringID, ErrorReturnCodes Error)

Routine	Compatibility
GetString	MS C#, MS Visual C++ (managed)

Return Value

A String variable containing the requested string value, or NULL if the string is not available.

Parameters

Type – type: GetStringOptions

Content (0x01)	The type of contents in a file (ex: Video, Database, Document, Font, Graphic Image, Personal/User Data, Macro/Script, Program Executable, Sound, Text, …)
Storage (0x02)	The type of storage used in the file (ex: Archive, Binary, Bitmap, Digital Audio, Music Notes, Text, Translated, Vector, Floating Header)
Platform (0x03)	The operating system(s) that the type of file is typically found on (ex: Amiga, IBM PC, Apple Macintosh, MS Windows, Sun OS, UNIX, Atari, Palm OS, Linux, …)
Error (0x14)	An error string related to a specific error return code.

StringID

Address of the index value for a string identifier.

Error

An error value is returned to indicate whether the method executed successfully. An enum value of Success (0) indicates that the method executed successfully. See ErrorReturnCodes in Appendix C, for a list of other possible error return codes.

Remarks

The GetString function queries a string value from the Descriptions Database in order to better represent the metadata and Description Database information obtained for the analyzed file.

StartEngine

Initializes the FE Engine library.

ErrorReturnCodes StartEngine(StartEngineFlags Instructions)

Routine	**Compatibility**
StartEngine	MS C#, MS Visual C++ (managed)

Return Value

An error value is returned to indicate whether the method executed successfully. An enum value of Success (0) indicates that the method executed successfully. See ErrorReturnCodes in Appendix C, for a list of other possible error return codes.

Parameters

Instructions – type: StartEngineFlags

All (0x00)	All of the steps below are executed.
RegisterDLL (0x01)	Write the version information, for the feengine.dll, to the MS Windows Registry. This step is only necessary when first installing or updating the feengine.dll.
ScanFELibraries (0x02)	Search for File Expander Libraries (*.fel), located in the current working directory, and record their file formats supported into the MS Windows Registry.
EnterRegistrationKey (0x04)	Read the EngineRegistrationKey string, and disable the nag screen.
StoreRegistrationKey (0x08)	Record the EngineRegistrationKey string into the MS Windows Registry for permanent use.

PreLoadLibraries (0x10)	Future feature – load all of the *.fel libraries into memory and don't unload them until the feengine.dll is unloaded. Otherwise the *.fel libraries are only loaded when they are needed and then promptly unloaded. This feature will be implemented to provide higher efficiency.
ReplaceWorkingDirectory (0x11)	Read the EngineWorkingDirectory string for the location to save the FIDebug.log file and find the FIEngine.fid database as well as all of the *.fel libraries. If you change the current working directory to this same directory before you load feengine.dll or load feengine.dll directly from an executable, then you probably won't need to use this flag for anything other than the FIDebug.log.

Remarks

The StartEngine method initializes the feengine.dll library as well as all of the recorded *.fel libraries, and must be the first function called.

StopFile

Halts the processing of all files currently being processed for the FE Engine.

ErrorReturnCodes StopFile()

Routine	Compatibility
StopFile	MS C#, MS Visual C++ (managed)

Return Value

An error value is returned to indicate whether the method executed successfully. An enum value of Success (0) indicates that the method executed successfully. See ErrorReturnCodes in Appendix C, for a list of other possible error return codes.

Remarks

The StopFile function instructs the library to halt all file processing currently in process. This is useful when running in a multitasking environment.

Appendix C: Public Types

All of these types are defined in the ForensicInnovations::FileExpander namespace.

ErrorReturnCodes

Used as method return values throughout File Expander

```
public __value enum ErrorReturnCodes {
   Success                    = 0,
   Failure                    = 1,
   FileNotFound               = 2,
   StringNotFound             = 3,
   ObjectsNotFound            = 4,
   CreateFileFailed           = 5,
   EndOfFile                  = 6,
   CreateDirectoryFailed      = 20,
   FormatNotSupported         = 21,
   BadOrEmptyParameter        = 22,
   BadRegistrationKey         = 23,
   LibraryInterfaceMissing    = 24,
   LibraryVersionOld          = 25,
   StopCommandUsed            = 26,
   ReadingFileFailed          = 27,
   LoadingLibraryFailed       = 28,
   AccessingRegistryFailed    = 29,
   WrongFormatID              = 30,
   EndOfList                  = 31,
   FileCorrupted              = 32,
   UnknownObject              = 33,
   WritingFileFailed          = 34,
   AccessingMemory            = 35,
   BadObjectOffset            = 36,
   WrongObjectSize            = 37,
   DecompressingFileFailed    = 38};
```

GetStringOptions

Used with GetString() to specify the string type to retrieve

```
public __value enum GetStringOptions {
   Content                    = 1,
   Storage                    = 2,
   Platform                   = 3,
   Error                      = 20};
```


StartEngineFlags

Used with StartEngine() to specify the steps to perform

```
[FlagsAttribute] public __value enum StartEngineFlags {
   All                        = 0,
   RegisterDLL                = 0x01 << 0,
   ScanFELibraries            = 0x01 << 1,
   EnterRegistrationKey       = 0x01 << 2,
   StoreRegistrationKey       = 0x01 << 3,
   ReplaceWorkingDirectory    = 0x01 << 5};
```

FileFieldsRequestFlags

Used with GetFileInfo() for FileFieldsRequested to specify the fields to populate

```
[FlagsAttribute] public __value enum FileFieldsRequestFlags {
   All                        = 0,
   FileFormatID               = 0x01 << 0,
   TotalObjects               = 0x01 << 1,
   FilteredObjects            = 0x01 << 2,
   ObjectNames                = 0x01 << 3,
   ObjectsExpandable          = 0x01 << 6,
   UncompressedSize           = 0x01 << 7,
   ResetDebugLog              = 0x01 << 10,
   UseDebugLog                = 0x01 << 11};
```

ObjectFieldsRequestFlags

Used with GetObjectInfo() for ObjectFieldsRequested to specify the fields to populate

```
[FlagsAttribute] public __value enum ObjectFieldsRequestFlags {
   All                        = 0,
   FileFormatID               = 0x01 << 0,
   AbsoluteIndex              = 0x01 << 1,
   ObjectName                 = 0x01 << 3,
   TranslationAlgorithm       = 0x01 << 4,
   CompressionAlgorithm       = 0x01 << 5,
   EncryptionAlgorithm        = 0x01 << 6,
   ExpansionSupport           = 0x01 << 8,
   AttributeFilter            = 0x01 << 13,
   FormatAttributeFilter      = 0x01 << 14,
   ObjectSize                 = 0x01 << 18,
   ObjectSizeCompressed       = 0x01 << 19,
   CRC                        = 0x01 << 21,
   Path                       = 0x01 << 23,
   ObjectOffset               = 0x01 << 24,
   ObjectInformationOffset    = 0x01 << 25,
   Storage                    = 0x01 << 26,
   UseDebugLog                = 0x01 << 28};
```


ObjectExpansionFlags

Used with GetObjectInfo() for ExpansionInstructions to specify how to expand an object

```
[FlagsAttribute] public __value enum ObjectExpansionFlags {
   None                        = 0,
   ExpandToFile                = 0x01 << 0,
   CopyWithoutExpansion        = 0x01 << 1,
   CopyIfExpansionFails        = 0x01 << 2};
```

ObjectTranslationAlgorithms

Used with GetObjectInfo() for TranslationAlgorithm to describe how an object is translated

```
public __value enum ObjectTranslationAlgorithms {
   None                        = 0,
   Unknown                     = 1,
   MIME                        = 2,
   CB64                        = 3,
   UUencode                    = 4};
```

ObjectCompressionAlgorithms

Used with GetObjectInfo() for CompressionAlgorithm to describe how an object is compressed

```
public __value enum ObjectCompressionAlgorithms {
   None                        = 0,
   Unknown                     = 1,
   PKWare                      = 2,
   WinZip                      = 3,
   Tokenizing                  = 4,
   Deflate64                   = 5,
   BZIP2                       = 6,
   IBMTerse                    = 7,
   IBMLZ77z                    = 8,
   WavPack                     = 9,
   PPMdvI                      = 10,
   Zlib                        = 11};
```

ObjectEncryptionAlgorithms

Used with GetObjectInfo() for EncryptionAlgorithm to describe how an object is encrypted

```
public __value enum ObjectEncryptionAlgorithms {
   None                        = 0,
   Unknown                     = 1,
   PKWare                      = 2,
   WinZipAES                   = 3};
```


ExpansionSupportFlags

Used with GetObjectInfo() for ExpansionSupport to describe expansion options the File Expander Engine offers for the object

```
[FlagsAttribute] public __value enum ExpansionSupportFlags {
   None                    = 0,
   Translation             = 0x01 << 0,
   Compression             = 0x01 << 1,
   Encryption              = 0x01 << 2,
   Copy                    = 0x01 << 3};
```

ObjectAttributeFlags

Used with GetObjectInfo() for ObjectAttributes to configure the object type filter

```
[FlagsAttribute] public __value enum ObjectAttributeFlags {
   None                    = 0,
   FileHeader              = 0x01 << 0,//0x0001
   Index                   = 0x01 << 1,//0x0002
   ObjectHeader            = 0x01 << 2,//0x0004
   ObjectData              = 0x01 << 3,//0x0008
   ObjectFooter            = 0x01 << 4,//0x0010
   FileFooter              = 0x01 << 5,//0x0020
   UnknownObject           = 0x01 << 7,//0x0080
```

ObjectStorageConditionFlags

Used with GetObjectInfo() for ObjectStorageCondition to describe the state of the object in the parent file

```
[FlagsAttribute] public __value enum ObjectStorageConditionFlags {
   Unknown                 = 0,
   Consecutive             = 0x01 << 0,
   Fragmented              = 0x01 << 1,
   Incomplete              = 0x01 << 2,// Deleted & part over written
   SpanFiles               = 0x01 << 3,
   Corrupted               = 0x01 << 4};
```

FormatFieldsRequestFlags

Used with GetFormatInfo() for FieldsRequested to specify which fields to populate

```
[FlagsAttribute] public __value enum FormatFieldsRequestFlags {
   All                     = 0,
   Name                    = 0x01 << 0,
   Extensions              = 0x01 << 1,
   MIME                    = 0x01 << 2,
   Platform                = 0x01 << 3,
   Storage                 = 0x01 << 4,
   Content                 = 0x01 << 5,
   VersionAdded            = 0x01 << 6,
   VersionUpdated          = 0x01 << 7,
   UseDebugLog             = 0x01 << 12,
   LibraryFilename         = 0x01 << 13};
```


Appendix D: Public Variables

All of these variables are defined in the ForensicInnovations::FileExpander namespace, Engine class.

```
public __gc class Engine {public:
```

StartEngine() Input Fields

FE Engine Instructions set before calling StartEngine()

```
String * EngineRegistrationKey;
```
A 15 character registration key that prevents the nag dialog

```
String * EngineWorkingDirectory;
```
The directory that contains the FEEngine.dll, FIWrpNET.dll, FIEngine.FID & *.fel files

GetFileInfo() Input Fields

File instructions, set before calling GetFileInfo()

```
ObjectAttributeFlags AttributeFilter;
```
Flag(s) that specify the types of objects to filter for. See ObjectAttributeFlags in Appendix C for possible values. (Used for GetFileInfo & GetObjectInfo)

```
FileFieldsRequestFlags FileFieldsRequested;
```
Flag(s) that specify the File Information fields to be populated by GetFileInfo(). See FileFieldsRequestFlags in Appendix C for possible values.

```
String * FullPath;
```
Path+Filename of the file to analyze (Used for GetFileInfo & GetObjectInfo)

```
String * SearchFilespec;
```
Path+Filespec of the object(s) within the parfent file (Used for GetFileInfo & GetObjectInfo; Default: *.*; ex: \testdir*.*)

GetFileInfo() Output Fields

File results, available after calling GetFileInfo()

```
unsigned long FileFilteredObjects;
```
Number of objects, in the file, that passed through the specified filter

```
unsigned long FileFormatID;
```
File Investigator File Description index number (Used for GetFileInfo & GetObjectInfo). The default value is 0L, before calling GetFileInfo, but this field can be assigned a value to expedite the file format verification stage. See Appendix A for index values to use.

```
String * FileObjectNames __gc[];
```
List of all object names that passed through the filter

`unsigned long FileObjectsExpandable;`
Number of objects found, that are expandable by the File Expander Engine

`unsigned long FileTotalObjects;`
Total number of objects found in the file

`unsigned __int64 FileUncompressedSize;`
Total uncompressed size, in bytes, for all objects found in the file

GetObjectInfo() Input Fields

Object instructions, set before calling GetObjectInfo()

`String * ObjectExpandToFilespec;`
A string value that specifies how to create the filename for each new file created from expanding an object. (ex: "%filename%-#%index%-0x%offset%-%opath%-%object%")

Variables available:

%filename%	Filename of the original file being analyzed.
%index%	Zero based index value of the object within the parent file.
%object%	Object name.
%offset%	Hexadecimal byte offset of the object within the parent file.
%opath%	Object's path within the parent file. This includes any hierarchal relationship between objects.

`String * ObjectExpandToPath;`
A string value that specifies where to create the file(s) that the object(s) is/are expanded into. (ex: "c:\temp")

`ObjectExpansionFlags ObjectExpansionInstructions;`
A flag value used to determine what whether to expand each object to a file. (ex: FE::ObjectExpansionFlags::ExpandToFile) See ObjectExpansionFlags in Appendix C, for a list of other possible flag values.

`ObjectFieldsRequestFlags ObjectFieldsRequested;`
A flag value used to determine what information fields are populated when the method returns. (ex: FE::ObjectFieldsRequestFlags::ObjectName) See ObjectFieldsRequestFlags in Appendix C, for a list of other possible flag values.

`unsigned long ObjectIndexStart;`
A zero based object index number value that specifies which object to start processing from the specified file. All prior objects in the file will be ignored.

`unsigned long ObjectIndexStop;`
A zero based object index that specifies the last object to process in the specified file. This value must be equal to or greater that the value provided in ObjectIndexStart. When ObjectIndexStop is greater that ObjectIndexStart, then multiple objects will be processed, and optionally expanded to files, but the information fields returned will only contain information about the last object processed.

GetObjectInfo() Output Fields

Object results, available after calling GetObjectInfo()

`unsigned long ObjectAbsoluteIndex;`

A zero based index value that indicates how many objects appeared in the file before the object being reported on, regardless of any filters used.

`ObjectAttributeFlags ObjectAttributes;`

A flag value that provides the object attributes used to describe the current object. (ex: ObjectData) See ObjectAttributeFlags in Appendix C, for possible values used.

`ObjectCompressionAlgorithms ObjectCompressionAlgorithm;`

An enumeration value that represents what compression algorithm is being used to store the object in the parent file. It is assumed that the object is not using any compression algorithm unless otherwise indicated in this field. (ex: PKWare)

Possible values for type ObjectCompressionAlgorithms:

None (0x00)	The object is not using any compression algorithm.
Unknown (0x01)	The compression algorithm could not be discovered.
PKWare (0x02)	PKWare is being used to store the object in the file.
WinZip (0x03)	WinZip is being used to store the object in the file.
Tokenizing (0x04)	Tokenizing is being used to store the object in the file.
Deflate64 (0x05)	Deflate64 is being used to store the object in the file.
BZIP2 (0x06)	BZIP2 is being used to store the object in the file.
IBMTerse (0x07)	IBMTerse is being used to store the object in the file.
IBMLZ77z (0x08)	IBMLZ77z is being used to store the object in the file.
WavPack (0x09)	WavPack is being used to store the object in the file.
PPMdvI (0x0A)	PPMdvI is being used to store the object in the file.
zlib (0x0B)	zlib is being used to store the object in the file.

`unsigned long ObjectCRC;`

A cyclic redundancy code calculated to be used in comparing the content of multiple objects.

`ObjectEncryptionAlgorithms ObjectEncryptionAlgorithm;`

An enumeration value that represents what encryption algorithm is being used to store the object in the parent file. It is assumed that the object is not using any encryption algorithm unless otherwise indicated in this field. (ex: WinZipAES)

Possible values for type ObjectEncryptionAlgorithms:

None (0x00)	The object is not using any encryption algorithm.
Unknown (0x01)	The encryption algorithm could not be discovered.
PKWare (0x02)	PKWare is being used to store the object in the file.
WinZipAES (0x03)	WinZipAES is being used to store the object in the file.

`ExpansionSupportFlags ObjectExpansionSupport;`

A flag value that indicates what methods of expansion the File Expander Engine is capable of using on the object in the parent file. It is assumed that the object can at least be copied (as is) unless the Copy flag is not present. If other fields indicate that the object is compressed, translated and/or encrypted, but that support doesn't appear in this field, then the object can only be copied as is. (ex: Compression)

Possible values for type ExpansionSupportFlags:

None (0x00)	The object cannot be copied, translated, unencrypted nor uncompressed.
Translation (0x01)	The object can be translated.
Compression (0x02)	The object can be uncompressed.
Encryption (0x04)	The object can be unencrypted.
Copy (0x05)	The object can be copied as is.

`unsigned __int64 ObjectInfoOffset;`

A zero based number value that represents how many bytes preceded the start of a block of information about the object in the parent file.

`unsigned long ObjectInfoOffsetBitsAdded;`

A zero based number value that represents how many bits need to be added to the number of bytes in ObjectInfoOffset to represent the actual bit location of the start of the information in the parent file.

`String * ObjectName;`

A zero based index value that indicates how many objects appeared in the file before the object being reported on, regardless of any filters used.

`unsigned __int64 ObjectOffset;`

A zero based number value that represents how many bytes preceded the start of the object in the parent file.

`unsigned long ObjectOffsetBitsAdded;`

Bits A zero based number value that represents how many bits need to be added to the number of bytes in ObjectOffset to represent the actual bit location of the start of the object in the parent file.

`String * ObjectPath;`

A string value containing the path/hierarchy of the object that passed through the specified filter in the analyzed file. If the object is an archived file, then this will be the archived file's original path. Otherwise, this will indicate the hierarchal relationship between this object and other objects located in the parent file.

`unsigned __int64 ObjectSize;`

A zero based number value that represents how many bytes are required to store the object in a new file, once the object in uncompressed, unencrypted, translated or simply copied.

`unsigned long ObjectSizeBitsAdded;`

A zero based number value that represents how many bits need to be added to the number of bytes in ObjectSize to represent the actual size in bits of the object.

`unsigned __int64 ObjectSizeCompressed;`

A zero based number value that represents how many bytes the object is using in the parent file, if that size is different from the uncompressed, unencrypted translated size provided in ObjectSize. If the compressed size is no different, then this variable is set to zero.

`unsigned long ObjectSizeCompressedBitsAdded;`

A zero based number value that represents how many bits need to be added to the number of bytes in ObjectSizeCompressed to represent the actual size in bits of the object.

`ObjectStorageConditionFlags ObjectStorageCondition;`

A flag value that represents what condition(s) the object is in, while stored in the parent file. It is assumed that the object is not fragmented, incomplete nor corrupted and is completely contained within the file being analyzed, unless otherwise indicated in this field. (ex: Consecutive)

Possible values for type ObjectStorageConditionFlags:

Unknown (0x00)	The condition of the object could not be discovered.
Consecutive (0x01)	All blocks of the object are stored consecutively (one right after the other) within the file
Fragmented (0x02)	One or more of the object's blocks is/are not stored consecutively within the file being analyzed. A fragmented block may be located later in the file or closer to the beginning of the file, depending on the ability of the parent file's file format to cope with fragmented data blocks.
Incomplete (0x04)	One or more of the object's blocks is/are missing due to deletion and/or corruption in the parent file being analyzed.
SpanFiles (0x08)	One or more of the object's blocks is/are located in one or more different parent file(s).
Corrupted (0x10)	One or more of the object's blocks is/are corrupted and may contain partial or no valid data relating to the object.

`ObjectTranslationAlgorithms ObjectTranslationAlgorithm;`

An enumeration value that represents what translation algorithm is being used to store the object in the parent file. It is assumed that the object is not using any translation algorithm unless otherwise indicated in this field. (ex: MIME)

Possible values for type ObjectTranslationAlgorithms:

None (0x00)	The object is not using any translation algorithm.
Unknown (0x01)	The translation algorithm could not be discovered.
MIME (0x02)	MIME is being used to store the object in the file.
CB64 (0x03)	CB64 is being used to store the object in the file.
UUencode (0x04)	UUencode is being used to store the object in the file.

GetFormatInfo() Input Fields

Format instructions, set before calling GetFormatInfo()

`FormatFieldsRequestFlags FormatFieldsRequested;`

A flag value used to determine what information fields are populated when the method returns. See FormatFieldsRequestFlags in Appendix C, for a list of possible flag values.

GetFormatInfo() Output Fields

Format results, available after calling GetFormatInfo()

`unsigned long FormatContent;`

A flag value used to determine what types of content can be found in the file format type. See ContentFlags in Appendix C, for a list of possible flag values.

`String * FormatExtensions __gc[];`

A list of the valid file extensions that can be used in a filename for files of the analyzed file format type. These are typically the file extensions that will allow applications to open and use the file correctly.

`String * FormatLibraryFilename;`

A flag value used to determine what types of content can be found in the file format type. See ContentFlags in Appendix C, for a list of possible flag values.

`String * FormatMIMEs __gc[];`

A list of the valid MIME types that can be used to identify files of the analyzed file format type in email and web page documents.

`String * FormatName;`

A string value, obtained from the Description Database, that provides a short descriptive name for the file format. See Appendix A for a list of possible name/description values.

`unsigned long FormatPlatform;`

A flag value used to determine what platform(s)/operating system(s) the file format type is typically found on. See PlatformFlags in Appendix C, for a list of possible flag values.

`unsigned long FormatStorage;`

A flag value used to determine what storage methods are used in the file format type. See StorageFlags in Appendix C, for a list of possible flag values.

`unsigned long FormatVersionAdded;`

A number value that contains the version of File Expander that the file format was initially added in. This is a 32bit value in which each 8 bits specify a part of the version. (ex: 0x01020304 = version 01.02.03.04)

`unsigned long FormatVersionUpdated;`

A number value that contains the version of File Expander that the file format was last updated in. This is a 32bit value in which each 8 bits specify a part of the version. (ex: 0x01020304 = version 01.02.03.04)

