

KeyView

Software Version 12.4

Filter SDK C Programming Guide



Document Release Date: October 2019
Software Release Date: October 2019

Legal notices

Copyright notice

© Copyright 2016-2019 Micro Focus or one of its affiliates.

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for updated documentation, visit <https://www.microfocus.com/support-and-services/documentation/>.

Support

Visit the [MySupport portal](#) to access contact information and details about the products, services, and support that Micro Focus offers.

This portal also provides customer self-solve capabilities. It gives you a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the MySupport portal to:

- Search for knowledge documents of interest
- Access product documentation
- View software vulnerability alerts
- Enter into discussions with other software customers
- Download software patches
- Manage software licenses, downloads, and support contracts
- Submit and track service requests
- Contact customer support
- View information about all services that Support offers

Many areas of the portal require you to sign in. If you need an account, you can create one when prompted to sign in. To learn about the different access levels the portal uses, see the [Access Levels descriptions](#).

Contents

Part I: Overview of Filter SDK	11
Chapter 1: Introducing Filter SDK	12
Overview	12
Features	12
Platforms, Compilers, and Dependencies	13
Supported Platforms	13
Supported Compilers	14
Software Dependencies	14
Windows Installation	15
UNIX Installation	16
Package Contents	17
License Information	18
Enable Advanced Document Readers	18
Update License Information	18
Directory Structure	19
Chapter 2: Getting Started	21
Architectural Overview	21
Enhance Performance	23
File Caching	23
Filtering	23
Subfile Extraction	24
Memory Abstraction	24
Use the C-Language Implementation of the API	24
Input/Output Operations	25
Filtering in File Mode	25
Filtering in Stream Mode	26
Multithreaded Filtering	27
The Filter Process Model	28
Filter API	28
File Extraction API	28
Persist the Child Process	29
In the API	29
In the formats.ini File	29
Run Filter In Process	29
In the API	30
In the formats.ini File	30
Run File Extraction Functions Out of Process	30
Restart the File Extraction Server	30
Out-of-Process Logging	31
Enable Out-of-Process Logging	31
Set the Verbosity Level	31

Enable Windows Minidump	32
Keep Log Files	32
Run File Detection In or Out of Process	32
Specify the Process Type In the formats.ini File	33
Specify the Process Type In the API	33
Part II: Use Filter SDK	34
Chapter 3: Use the File Extraction API	35
Introduction	35
Extract Subfiles	36
Sanitize Absolute Paths	37
Extract Images	38
Recreate a File's Hierarchy	38
Create a Root Node	38
Recreate a File's Hierarchy—Example	39
Extract Mail Metadata	40
Default Metadata Set	40
Extract the Default Metadata Set	41
Extract All Metadata	41
Microsoft Outlook (MSG) Metadata	41
Extract MSG-Specific Metadata	42
Microsoft Outlook Express (EML) and Mailbox (MBX) Metadata	43
Extract EML- or MBX-Specific Metadata	43
Lotus Notes Database (NSF) Metadata	44
Extract NSF-Specific Metadata	44
Microsoft Personal Folders File (PST) Metadata	45
MAPI Properties	45
Extract PST-Specific Metadata	46
Exclude Metadata from the Extracted Text File	46
Extract Subfiles from Outlook Files	47
Extract Subfiles from Outlook Express Files	47
Extract Subfiles from Mailbox Files	47
Extract Subfiles from Outlook Personal Folders Files	47
Choose the Reader to use for PST Files	48
MAPI Attachment Methods	49
Open Secured PST Files	50
Detect PST Files While the Outlook Client is Running	50
Extract Subfiles from Lotus Domino XML Language Files	51
Extract .DXL Files to HTML	51
Extract Subfiles from Lotus Notes Database Files	52
System Requirements	52
Installation and Configuration	53
Windows	53
Solaris	53
AIX 5.x	54

- Linux 54
- Open Secured NSF Files 54
- Format Note Subfiles 55
- Extract Subfiles from PDF Files 55
 - Improve Performance for PDFs with Many Small Images 55
- Extract Embedded OLE Objects 55
- Extract Subfiles from ZIP Files 56
- Default File Names for Extracted Subfiles 56
 - Default File Name for Mail Formats 56
 - Default File Name for Embedded OLE Objects 57
- Chapter 4: Use the Filter API 58
 - Generate an Error Log 58
 - Enable or Disable Error Logging 59
 - Use the API 59
 - Use Environment Variables 59
 - Change the Path and File Name of the Log File 59
 - Report Memory Errors 60
 - Use the API 60
 - Use Environment Variables 60
 - Specify a Memory Guard 60
 - Report the File Name in Stream Mode 60
 - Report Extended Error Codes 61
 - Specify the Maximum Size of the Log File 61
 - Extract Metadata 61
 - Extract Metadata for File Filtering 62
 - Extract Metadata for Stream Filtering 62
 - Example 62
 - Convert Character Sets 63
 - Determine the Character Set of the Output Text 64
 - Guidelines for Character Set Conversion 64
 - Set the Character Set During Filtering 65
 - Set the Character Set During Subfile Extraction 65
 - Prevent the Default Conversion of a Character Set 65
 - Extract Deleted Text Marked by Tracked Changes 66
 - Filter PDF Files 66
 - Filter PDF Files to a Logical Reading Order 66
 - Enable Logical Reading Order 68
 - Use the C API 68
 - Use the formats.ini File 68
 - Rotated Text 69
 - Extract Custom Metadata from PDF Files 69
 - Extract Custom Metadata by Tag 69
 - Extract All Custom Metadata 70
 - Filter Tagged PDF Content 70
 - Skip Embedded Fonts 71
 - Use the formats.ini File 71

Use the C API	72
Control Hyphenation	72
Use the formats.ini File	72
Use the C API	72
Filter Spreadsheet Files	73
Filter Worksheet Names	73
Filter Hidden Text in Microsoft Excel Files	73
Specify Date and Time Format on UNIX Systems	73
Filter Very Large Numbers in Spreadsheet Cells to Precision Numbers	74
Extract Microsoft Excel Formulas	74
Standardize Cell Formats	76
Numbers	76
Text	77
Dates	77
Filter XML Files	77
Configure Element Extraction for XML Documents	78
Modify Element Extraction Settings	78
Modify Element Extraction Settings in the kvxconfig.ini File	79
Specify an Element's Namespace and Attribute	81
Add Configuration Settings for Custom XML Document Types	81
Configure Headers and Footers	82
Filter Hidden Data	83
Hidden Data in Microsoft Excel Documents	83
Example	84
Toggle Hidden Excel Data Settings in the formats.ini File	84
Hidden Data in HTML Documents	84
Tab Delimited Output for Embedded Tables	85
Table Detection for PDF Files	85
Exclude Japanese Guide Text	86
Source Code Identification	86
Chapter 5: Sample Programs	88
Introduction	88
tstextract	88
filter	90
Part III: C API Reference	93
Chapter 6: File Extraction API Functions	94
KVGetExtractInterface()	94
fpCloseFile()	95
fpExtractSubFile()	96
fpFreeStruct()	97
fpGetMainFileInfo()	98
fpGetSubFileInfo()	99
fpGetSubFileMetaData()	100
fpOpenFile()	102

fpSetExtractionTimeout()	103
Chapter 7: File Extraction API Structures	105
KVCredential	105
KVCredentialComponent	106
KVExtractInterface	106
KVExtractSubFileArg	107
KVGetSubFileMetaArg	110
KVMainFileInfo	111
KVMetadataElem	112
KVMetaName	113
KVOpenFileArg	114
KVOutputStream	115
KVSubFileExtractInfo	115
KVSubFileInfo	116
KVSubFileMetaData	119
Chapter 8: Filter API Functions	121
KV_GetFilterInterfaceEx()	122
fpCanFilterFile()	124
fpCanFilterStream()	125
fpCloseStream()	126
fpFiletoInputStreamCreate()	127
fpFileToInputStreamFree()	128
fpFilterConfig()	129
fpFilterFile()	134
fpFilterStream()	135
fpFreeFilterOutput()	136
fpFreeOLESummaryInfo()	137
fpFreeXmplInfo()	138
fpGetDocInfoFile()	139
fpGetDocInfoStream()	140
fpGetKvErrorCodeEx()	141
fpGetOLESummaryInfo()	142
fpGetOLESummaryInfoFile()	143
fpGetTrgCharSet()	144
fpGetXmplInfo()	145
fpGetXmplInfoFile()	146
fpInit()	147
fpInitWithLicenseData()	149
fpOpenStream()	152
fpOpenStreamEx2()	153
fpRefreshFilterKVOOP()	154
fpSetReplacementChar()	155
fpSetSrcCharSet()	156
fpSetTimeout()	157
fpShutdown()	158

Chapter 9: Filter API Structures	159
KVFitInterfaceEx	160
ADDOCINFO	163
KV_CONFIG_Arg	164
KVFilterOutput	165
KVInputStream	166
KVMemoryStream	167
KVStructHead	168
KVSumInfoElemEx	169
KVSummaryInfoEx	170
KVXConfigInfo	171
KVXmplInfo	173
KVXmplInfoElems	174
Chapter 10: Enumerated Types	175
Introduction	175
Programming Guidelines	176
KVCredKeyType	176
KVErrorCode	177
KVErrorCodeEx	178
KVMetadataType	182
KVMetaNameType	183
KVSumInfoType	184
KVSumType	185
LPDF_DIRECTION	188
Appendixes	190
Appendix A: Supported Formats	191
Supported Formats	191
Archive Formats	192
Binary Format	195
Computer-Aided Design Formats	196
Database Formats	197
Desktop Publishing	198
Display Formats	198
Graphic Formats	199
Mail Formats	203
Multimedia Formats	206
Presentation Formats	209
Spreadsheet Formats	212
Text and Markup Formats	214
Word Processing Formats	215
Appendix B: Detected Formats	221
Key to Detected Formats Table	221
Detected Formats	223

Appendix C: Character Sets	275
Multibyte and Bidirectional Support	275
Coded Character Sets	283
Appendix D: Extract and Format Lotus Notes Subfiles	289
Overview	289
Customize XML Templates	289
Use Demo Templates	290
Use Old Templates	290
Disable XML Templates	290
Template Elements and Attributes	291
Conditional Elements	291
Control Elements	292
Data Elements	293
Date and Time Formats	296
Lotus Notes Date and Time Formats	296
KeyView Date and Time Formats	297
Appendix E: File Format Detection	302
Introduction	302
Extract Format Information	302
Determine Format Support	302
Example formats.ini file entries	303
Refine Detection of Text Files	303
Allow Consecutive NULL Bytes in a Text File	304
Translate Format Information	305
Distinguish Between Formats	305
Determine a Document Reader	306
Category Values in formats.ini	306
Appendix F: List of Required Files for Redistribution	310
Core Files	310
Support Files	311
Document Readers	312
Appendix G: Develop a Custom Reader	319
Introduction	319
How to Write a Custom Reader	320
Naming Conventions	320
Basic Steps	321
Token Buffer	321
Macros	323
Reader Interface	323
Function Flow	324
Example Development of fffFillBuffer()	324
Implementation 1—fpFillBuffer() Function	324
Structure of Implementation 1	325
Problems with Implementation 1	325
Implementation 2—Processing a Large Token Stream	326

Structure of Implementation 2	327
Problems with Implementation 2	327
Boundary Conditions	327
Implementation 3—Interrupting Structured Access Layer Calls	328
Structure of Implementation 3	330
Development Tips	330
Functions	331
xxxsrAutoDet()	331
xxxAllocateContext()	332
xxxFreeContext()	333
xxxInitDoc()	334
xxxFillBuffer()	334
xxxGetSummaryInfo()	335
xxxOpenStream()	336
xxxCloseStream()	337
xxxCharSet()	337
Appendix H: Password Protected Files	339
Supported Password Protected File Types	339
Open Password Protected Container Files	340
Filter Password Protected Files	340
Appendix I: Microsoft Rights Management Service Protected Files	342
Microsoft Rights Management Service	342
Supported Formats	342
Microsoft Office Files	342
Implemented as pFile	343
PDF Files	344
Send documentation feedback	346

Part I: Overview of Filter SDK

This section provides an overview of the Micro Focus KeyView Filter SDK and describes how to use the C implementation of the API.

Chapter 1: Introducing Filter SDK

This section describes the Filter SDK package.

• Overview	12
• Features	12
• Platforms, Compilers, and Dependencies	13
• Windows Installation	15
• UNIX Installation	16
• Package Contents	17
• License Information	18
• Directory Structure	19

Overview

Micro Focus KeyView Filter SDK enables you to incorporate text extraction functionality into your own applications. It extracts text and metadata from a wide variety of file formats on numerous platforms, and can automatically recognize over 1000 document types. It supports both file-based and stream-based I/O operations, and provides in-process or out-of-process filtering.

Filter SDK is part of the KeyView suite of products. KeyView provides high-speed text extraction, conversion to web-ready HTML and well-formed XML, and high-fidelity document viewing.

Features

- Document readers are threadsafe. The benefit of a threadsafe technology is that you can successfully extract text from hundreds of documents simultaneously. Documents are not queued for sequential filtering, but are actually filtered at the same time.
- Filter supports popular word processing, spreadsheet, and presentation formats. Body text, endnotes, footnotes, and additional items such as document metadata are all included as part of the filtering process.
- Sample programs are provided to demonstrate the functionality of the APIs.
- You can extract files embedded within files, such as email attachments or embedded OLE objects, by using the File Extraction API.
- You can configure memory management. If using the C API, you can provide your own memory allocator to the document readers.
- Filter allows for redirected input and output. You can provide an input stream that is not restricted to file system access.

- Filter automatically recognizes the file type being filtered and uses the appropriate filter. Your application does not need to rely on file name extensions to determine file types.
- You can filter documents to specific character encodings, such as Unicode or UTF-8.
- You can use Filter SDK in conjunction with other KeyView technologies, such as the Index, Highlight, and Annotate APIs.
- You can write custom document readers for formats not directly supported by KeyView.

Platforms, Compilers, and Dependencies

This section lists the supported platforms, supported compilers, and software dependencies for the KeyView software.

Supported Platforms

- CentOS 7 x86 and x64
- FreeBSD 8.1 x86
- IBM AIX L6.1 PowerPC 32-bit and 64-bit
- IBM AIX L7.1 PowerPC 32-bit and 64-bit
- Mac OS X Mountain Lion 10.8 or higher on 32- and 64-bit Apple-Intel architecture
- Microsoft Windows Server 2012 x64
- Microsoft Windows Server 2016 x64
- Microsoft Windows Server 2019 x64
- Microsoft Windows 7 x86 and x64
- Microsoft Windows 8 x86 and x64
- Microsoft Windows 10 x64
- Oracle Solaris 10 SPARC
- Oracle Solaris 10 x86 and x64
- Red Hat Enterprise Linux 6 x86 and x64
- Red Hat Enterprise Linux 7 x64
- Red Hat Enterprise Linux 8 x64
- SuSE Linux Enterprise Server 11 x86 and x64
- SuSE Linux Enterprise Server 12 x64
- SuSE Linux Enterprise Server 15 x64

Supported Compilers

Platform	Architecture	Compiler Name	Compiler Version
Microsoft Windows	x86	cl	Microsoft 32-bit C/C++ Optimizing Compiler Version 16.00.30319.01 for x86
	x64	cl	Microsoft C/C++ Optimizing Compiler Version 16.00.30319.01 for x64
Sun Solaris	x86 64-bit	Sun Studio 12	Sun C 5.9 SunOS_i386 Patch 124868-01 2007/07/12
	SPARC 64-bit	Sun Studio 11	Sun C 5.8 Patch 121015-06 2007/10/03
Linux	x86	gcc / g++	3.4.3 (Redhat 4), 4.1.0 (SuSE Linux 10)
	x64	gcc / g++	4.1.0 (Redhat 4), 4.1.0 (SuSE Linux 10)
IBM AIX	Power	xIC_r / cc_r	IBM XL C/C++ Enterprise Edition V8.0
macOS	Apple-Intel 32-bit and 64-bit	LLVM	Apple LLVM 5.1 (clang-503.0.40) (based on LLVM 3.4svn)
FreeBSD	BSD x86	gcc / g++	4.2.1 [FreeBSD] 20070719

Supported Compilers for Java Components

Component	Compiler
Java components	Java 7

Software Dependencies

Some KeyView components require specific third-party software:

- Java Runtime Environment (JRE) or Java Software Developer Kit (JDK) version 7 is required for Java API and graphics conversion in Export SDK.
- Outlook 2002 or later is required to process Microsoft Outlook Personal Folders (PST) files using the MAPI-based reader (pstsr). The native PST readers (pstxsr and pstnsr) do not require Outlook.

NOTE: You must install an edition of Microsoft Outlook (32-bit or 64-bit) that matches the KeyView software. For example, if you use 32-bit KeyView, install 32-bit Outlook. If you use

64-bit KeyView, install 64-bit Outlook.

If the editions do not match, KeyView returns Error 32: KVErrror_PSTAccessFailed and an error message from Microsoft Office Outlook is displayed: Either there is a no default mail client or the current mail client cannot fulfill the messaging request. Please run Microsoft Outlook and set it as the default mail client.

- Lotus Notes or Lotus Domino is required for Lotus Notes database (NSF) file processing. The minimum requirement is 6.5.1, but version 8.5 is recommended.
- The Microsoft .NET Framework is required if you are using the .NET implementation of the API.
- Microsoft Visual C++ 2013 and Microsoft Visual C++ 2010 Redistributables (Windows only).

Windows Installation

To install the SDK on Windows, use the following procedure.

To install the SDK

1. Run the installation program, `KeyViewProductNameSDK_VersionNumber_OS.exe`, where *ProductName* is the name of the product, *VersionNumber* is the product version number, and *OS* is the operating system.

For example:

`KeyViewFilterSDK_12.4_Windows_X86_64.exe`


The installation wizard opens.

2. Read the instructions and click **Next**.

The License Agreement page opens.

3. Read the agreement. If you agree to the terms, click **I accept the agreement**, and then click **Next**.

The Installation Directory page opens.

4. Select the directory in which to install the SDK. To specify a directory other than the default, click  , and then specify another directory. After choosing where to install the SDK, click **Next**.

The License Key page opens.

5. Type the company name and license key that were provided when you purchased KeyView, and then click **Next**.
 - The company name is case sensitive.
 - The license key is a string that contains 31 characters.

NOTE: The installation program validates the company name and license key and generates

the file `install\OS\bin\kv.lic` (where `install` is your chosen installation folder and `OS` is the name of the operating system platform). The license information is validated when the KeyView API is used. If you do not enter a license key at this step, or if you enter invalid information, the KeyView SDK is installed, but the API does not function. When you obtain a valid license key, you can either re-install the KeyView SDK, or manually update the license key file (`kv.lic`) with the new information. For more information, see [License Information, on page 18](#).

The Pre-Installation Summary dialog box opens.

6. Review the settings, and then click **Next**.

The SDK is installed.

7. Click **Finish**.

UNIX Installation

To install the SDK, use one of the following procedures.

To install the SDK from the graphical interface

- Run the installation program and follow the on-screen instructions.

To install the SDK from the console

1. Run the installation program from the console as follows:

```
./KeyViewFilterSDK_VersionNumber_Platform.exe --mode text
```

where:

`VersionNumber` is the product version.

`Platform` is the name of the platform.

2. Read the welcome message and instructions and press `Enter`.

The first page of the license agreement is displayed.

3. Read the license information, pressing `Enter` to continue through the text. After you finish reading the text, and if you accept the agreement, type `y` and press `Enter`.

You are asked to choose an installation folder.

4. Type an absolute path or press `Enter` to accept the default location.

You are asked for license information.

5. At the **Company Name** prompt, type the company name that was provided when you purchased KeyView, and then press `Enter`. The company name is case sensitive.

6. At the **License Key** prompt, type the license key that was provided when you purchased KeyView, and then press `Enter`. The license key is a string that contains 31 characters.

NOTE: The installation program generates the file `install\OS\bin\kv.lic` (where `install` is your chosen installation folder and `OS` is the name of the operating system platform). The license information is validated when the KeyView API is used. If you do not enter a license key at this step, or if you enter invalid information, the KeyView SDK is installed but the API does not function. When you obtain a valid license key, you can either re-install the KeyView SDK, or manually update the license key file (`kv.lic`) with the new information. For more information, see [License Information, on the next page](#).

The Pre-Installation summary is displayed.

7. If you are satisfied with the information displayed in the summary, press `Enter`.

The SDK is installed.

Package Contents

The Filter SDK installation contains:

- All the libraries and executables necessary for extracting text from a wide variety of formats.
- The include files that define the functions and structures used by the application to establish an interface with Filter:

<code>adapi.h</code>	<code>kvfilter.h</code>
<code>adinfo.h</code>	<code>kvioobj.h</code>
<code>kvcfsr.h</code>	<code>kvtoken.h</code>
<code>kverrorcodes.h</code>	<code>kvtypes.h</code>
<code>kvextract.h</code>	<code>kvextract.h</code>
<code>kvfilt.h</code>	<code>kwautdef.h</code>
<code>kvfilt2.h</code>	

- The Java API implemented in the package `com.verity.api.filter` contained in the file `KeyView.jar`.
- The .NET API implemented in the namespace `Autonomy.API.Filter` in the library `FilterDotNet.dll`.
- The C++ SDK, which can be found in the `cppapi` folder.
- Sample programs that demonstrate File Extraction and Filter functionality using the APIs.
- The files necessary to create a custom document reader, and the source for a sample document reader for UTF-8. See [Develop a Custom Reader, on page 319](#).

License Information

During installation, the installation program generates the `install/OS/bin/kv.lic` file, where `install` is the directory in which you installed KeyView, and `OS` is the operating system. This file is opened and validated when the KeyView API is used.

The `kv.lic` file contains the organization name and the 31-digit license key you specified during installation. The contents of a `kv.lic` file looks similar to the following:

```
Company Name  
XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX
```

The license key controls whether the following are enabled:

- the full version of the KeyView SDK
- the trial version of the KeyView SDK
- language detection and advanced document readers—The following components are considered advanced features, and are licensed separately:
 - Microsoft Outlook Personal Folders (PST) readers (`pstsr`, `pstnsr`, and `pstxsr`)
 - Lotus Notes database (NSF) reader (`nsfsr`)
 - Mailbox (MBX) reader (`mbxsr`)
 - Character set detection library (`kvlangdetect`)

If you change the license key at any time, you must update the licensing information in the `kv.lic` file. See [Update License Information](#).

Enable Advanced Document Readers

To enable advanced readers in one of the KeyView SDKs, you must obtain an appropriate license key from Micro Focus and update the installed license key with the new information as described in [Update License Information](#).

If you are enabling the MBX reader in an existing installation of Filter, in addition to updating the license key, change the parameter `208=em1` to `208=mbx` in the `formats.ini` file.

Update License Information

If you currently have an evaluation version of KeyView and have purchased a full version of the SDK, or you are adding a document reader (for example, the PST reader), you must update the license information that was installed with the original version of the KeyView SDK.

If you installed a full version of KeyView, but did not enter licensing information at the time of installation, you must also update the license information.

To update the information, do one of the following:

- Manually update the license information that is stored in the text file named `kv.lic`.
- Re-install the product and enter the new license information when prompted.

To update the KeyView license information

1. Open the license key file, `kv.lic`, in a text editor. The file is in the `install\OS\bin` directory, where `install` is the directory in which you installed KeyView, and `OS` is the operating system. The file contains the following text:

```
COMPANY NAME  
XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX
```

2. Replace the text `COMPANY NAME` with the company name that appears at the top of the License Key Sheet provided by Micro Focus. Enter the text exactly as it appears in the document.
3. Replace the characters `XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX` with the appropriate license key from the License Key Sheet provided by Micro Focus. The license key is listed in the **Key** column in the **Standalone Products** table. The key is a string that contains 31 characters, for example, `2TQD22D-2M6FV66-2KPF23S-2GEM5AB`. Enter the characters exactly as they appear in the document, including the dashes, but do not include a leading or trailing space.
4. The finished `kv.lic` file looks similar to the following:

```
Autonomy  
24QD22D-2M6FV66-2KPF23S-2G8M59B
```

5. Save the `kv.lic` file.

Directory Structure

The following table describes the directories created during the Filter SDK installation. The variable `install` is the path name of the Filter installation directory (for example, `/usr/autonomy/KeyviewFilterSDK` on UNIX, or `C:\Program Files\Autonomy\KeyviewFilterSDK` on Windows).

The variable `OS` is the operating system for which the SDK is installed. For example, the `bin` directory on a standard 32-bit Windows installation would be located at `C:\Program Files\Autonomy\KeyviewFilterSDK\WINDOWS\bin`.

Installed directory structure

Directory	Description
<code>install\OS\bin</code>	Contains the libraries, the format detection file <code>formats.ini</code> , the license key file <code>kv.lic</code> , and other supporting files.
<code>install\OS\lib</code>	(Solaris installations only) Contains the redistributable <code>libstlport.so.1</code> library, which is required to run KeyView on Solaris platforms.

Installed directory structure, continued

Directory	Description
<code>install\dotnetapi</code>	Contains the source files for the .NET API.
<code>install\dotnetapi\dotnethelp</code>	Contains the help for the .NET API.
<code>install\dotnetapi\sample</code>	Contains the sample programs for the .NET API.
<code>install\cppapi</code>	Contains the source files for the C++ API.
<code>install\cppapi\sample</code>	Contains the sample programs for the C++ API.
<code>install\guide</code>	Contains the KeyView Filter SDK programming guides in PDF and HTML format.
<code>install\include</code>	Contains the header files required for Filter.
<code>install\javaapi\javadoc</code>	Contains the Javadoc for the Java API.
<code>install\javaapi\sample</code>	Contains the source files and sample programs for the Java API.
<code>install\rel_notes</code>	Contains the <i>KeyView Filter SDK Release Notes</i> in PDF format.
<code>install\samples\filter</code>	Contains a sample program demonstrating the Filter interface for the C API.
<code>install\samples\filterca</code>	Contains a C sample program demonstrating extraction of a content access stream.
<code>install\samples\pdfini</code>	Contains the initialization file used to extract custom metadata from PDF documents.
<code>install\samples\tstextract</code>	Contains a C sample program demonstrating the File Extraction interface.
<code>install\samples\utf8sr</code>	Contains the source for the sample document reader for UTF-8 files. You can use this to create your own custom document readers.
<code>install\samples\utf8sr\bin</code>	Contains the C program <code>filtertest</code> . You can use this program to test your custom document readers. See Develop a Custom Reader, on page 319 .

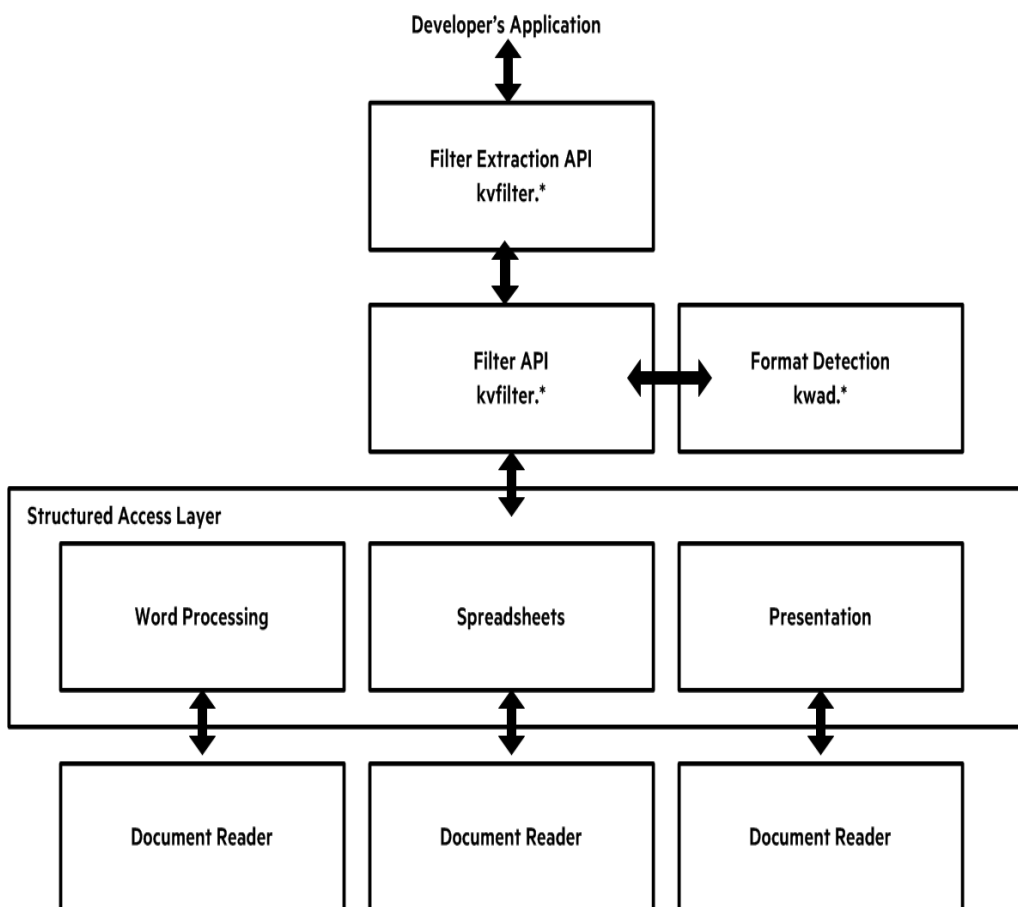
Chapter 2: Getting Started

This section provides an overview of Filter SDK, and describes how to use the C implementation of the API.

- [Architectural Overview](#)21
- [Enhance Performance](#) 23
- [Filtering](#)23
- [Subfile Extraction](#) 24
- [Memory Abstraction](#) 24
- [Use the C-Language Implementation of the API](#) 24
- [The Filter Process Model](#) 28
- [Run File Detection In or Out of Process](#) 32

Architectural Overview

The general architecture of the KeyView Filter technology is the same across all supported platforms and is illustrated in the following diagram:



Each component is described in the following table.

Architectural Components

Component	Description
Developer's Application	The developer's application interfaces directly with the Filter API through either a C-language or Java implementation.
File Extraction API	The File Extraction API opens a file and extracts the file's subfiles so that they are exposed for filtering. See Use the File Extraction API, on page 35 .
Filter API	The Filter API exposes the filtering functionality and controls all other modules during the filtering process. See Use the Filter API, on page 58 .
Format Detection	This module determines the file type of the input stream, allowing the Filter API to return that information to the developer's application, or to load the appropriate structured access layer for further processing. See File Format Detection, on page 302 for more information on format detection.

Architectural Components , continued

Component	Description
Structured Access Layer	There are three modules that reside in the structured access layer—one each for word processing, spreadsheet, and presentation formats. The file detection result determines which structured access layer module is used during the filtering process. That module loads the appropriate document reader and proceeds with text extraction or metadata retrieval.
Document Readers	Each document reader reads a specific file format and sends a text stream of the document to the structured access layer. Each filter is loaded as required by the structured access layer. See Document Readers, on page 312 for a complete list of document readers.

Enhance Performance

KeyView is designed for optimal performance out of the box. However, there are some parameters that you can adjust to improve system performance according to your needs.

File Caching

To reduce the frequency of I/O operations, and consequently improve performance, the KeyView readers load file data into memory. The readers then read the data from the cache rather than the physical disk. You can configure the amount of memory used for file caching through the `formats.ini` file. Generally, when you increase the memory, performance improves.

By default, KeyView uses a maximum of 1 MB of memory for each thread—assuming a thread contains only one instance of `pContext` that is returned from the session initialization (see [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149). If the file data is larger than 1 MB, up to 1 MB of data is cached and the data beyond 1 MB is read from disk. The minimum amount of memory that can be used for file caching is 64 KB.

To determine a reasonable value, divide the maximum amount of memory you want KeyView to use for file caching by the total number of threads. For example, if you want KeyView to use a maximum of 50 MB of memory and have 10 threads, set the value to 5 MB.

To modify the memory allocated for file caching, change the value for the following parameter in the `[DiskCache]` section of the `formats.ini` file:

```
DiskCacheSize=1024
```

The value is in kilobytes. If this parameter is not set or is set to 0 (zero), the minimum value of 64 KB is used.

Filtering

Filter SDK enables you to *filter* many different types of documents. Filtering is the process of extracting the text from a document without the application-specific markup. However, the filtering process can

also include the following:

- Subfile extraction—this process exposes all subfiles for filtering. See [Use the File Extraction API, on page 35](#).
- File format extraction—this process detects a file's format, and reports the information to the API, which in turn reports the information to the developer's application. See [File Format Detection, on page 302](#).
- Metadata extraction—this process extracts selected metadata (document properties) from a file. See [Extract Metadata, on page 61](#).
- Character set conversion—this process controls the character set of both the input and the output text. See [Convert Character Sets, on page 63](#).

Subfile Extraction

To filter a file, you must first determine whether the file contains any subfiles (attachments, embedded OLE objects, and so on). A file that contains subfiles is called a *container* file. Archive files (such as ZIP), mail messages with attachments (such as Microsoft Outlook Express), mail stores (such as Microsoft Outlook Personal Folders), and compound documents with embedded OLE objects (such as a Microsoft Word document with an embedded Excel chart) are examples of container files.

If the file is a container file, the container must be opened and its subfiles extracted using the File Extraction interface. The extraction process is done repeatedly until all subfiles are extracted and exposed for filtering. After a subfile is extracted, you can use the Filter API to filter the file.

If a file is not a container, you should pass it directly to the Filter API for filtering without extraction.

The [tstxtract](#) sample program demonstrates the application logic for extracting and filtering files. See [Use the File Extraction API, on page 35](#) for more information.

Memory Abstraction

Dynamic memory allocations in the Filter modules are abstracted through a C interface. This memory allocation interface is defined in the `KVMemoryStream` structure in `kvtypes.h`. You can override all memory allocations by providing a C structure that contains pointers to functions identical in nature to their standard ANSI C counterpart.

Use the C-Language Implementation of the API

The C-language implementation of the Filter API is divided into the following function suites:

- [File Extraction API Functions](#)—Open and extract subfiles in a container file. These functions also extract metadata and file format information, and control character set conversion on extraction. The [tstxtract](#) sample program demonstrates these functions.
- [Filter API Functions](#)—Extract document information (metadata character set, format), create an input/output stream, and filter a file or stream. The [filter](#) sample program demonstrates these functions.

Input/Output Operations

In Filter, the source input can be either a physical file accessed through a file path, or a filter stream created from a data source. A *filter stream* in the C API implementation is a C data structure that contains pointers to I/O functions similar to their standard ANSI C counterparts. This structure is passed to filter functions in place of the standard input source. The input stream is defined by the [KVInputStream](#) structure in `kvtypes.h`.

You can create an input stream by using the [fpFiletoInputStreamCreate\(\)](#) function, or by using code similar to the code in the Filter sample program. The `fpFiletoInputStreamCreate()` function assigns C equivalent I/O functions to `fpOpen()`, `fpRead()`, `fpSeek()`, `fpTell()`, and `fpClose()`. The code in the Filter sample program is shown below. This code assigns the file I/O functions (`myOpen`, `myRead`, and so on) to `KVInputStream`.

```
typedef struct
{
    char *pszName;
    FILE *fp;
}
MyOpenInfo;

KVInputStream IO;
MyOpenInfo o;

/* Initialize the input stream */
o.pszName = pszFileIn;
IO.pInputStreamPrivateData = (void *)&o;
IO.fpOpen = myOpen;
IO.fpRead = myRead;
IO.fpSeek = mySeek;
IO.fpTell = myTell;
IO.fpClose = myClose;
```

The output for extracted content is either a physical file accessed through a file path and specified in the call to [fpFilterFile\(\)](#), or an *output buffer* specified in the call to [fpFilterStream\(\)](#). The buffer is defined by the [KVFilterOutput](#) data structure in `kvtypes.h`.

Filtering in File Mode

To use the Filter file-based I/O

1. Load the `kvfilter` library and obtain the `KV_GetFilterInterfaceEx()` entry point by calling [KV_GetFilterInterfaceEx\(\)](#). The filter sample program contains sample code for all platforms.
2. Initialize a filter session by calling [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149. This function's return value, `pContext`, is passed as the first argument to the File Extraction interface and all other Filter functions.

3. Pass the context pointer from [fplnit\(\), on page 147](#) or [fplnitWithLicenseData\(\), on page 149](#) and the address of a structure that contains pointers to the File Extraction API functions in the call to [KVGetExtractInterface\(\)](#).
4. Declare the file path in the [KVOpenFileArg](#) structure.
5. Open the file by calling [fpOpenFile\(\)](#) and passing the [KVOpenFileArg](#) structure. This call defines the parameters necessary to open a file for extraction.
6. Determine whether the source file is a container file (that is, whether it contains subfiles) by calling [fpGetMainFileInfo\(\)](#).
7. If the call to [fpGetMainFileInfo\(\)](#) determined that the source file contains subfiles, proceed to step 8; otherwise, proceed to step 11.
8. Determine whether the subfile is a container file by calling [fpGetSubFileInfo\(\)](#).
9. Extract the subfile or subfiles to a file by calling [fpExtractSubFile\(\)](#) and setting `filePath` and `extractDir` in the [KVExtractSubFileArg](#) structure.
10. If the call to [fpGetSubFileInfo\(\)](#) determined that the subfile is a container file, repeat step 4 through step 9 until all subfiles are extracted; otherwise, proceed to step 11.
11. Filter the file by calling [fpFilterFile\(\)](#).
12. Close the file by calling [fpCloseFile\(\)](#).
13. Repeat step 4 through step 12 as required for additional source files.
14. Terminate the filter session by calling [fpShutdown\(\)](#).

Filtering in Stream Mode

To use the Filtering stream-based I/O

1. Load the `kvfilter` library and obtain the [KV_GetFilterInterfaceEx\(\)](#) entry point. The `filter` sample program contains sample code for all platforms.
2. Initialize a filter session by calling [fplnit\(\), on page 147](#) or [fplnitWithLicenseData\(\), on page 149](#). This function's return value, `pContext`, is passed as the first argument to all other Filter functions.
3. Pass the context pointer from [fplnit\(\), on page 147](#) or [fplnitWithLicenseData\(\), on page 149](#) and the address of a structure that contains pointers to the File Extraction API functions in the call to [KVGetExtractInterface\(\)](#). See [KVGetExtractInterface\(\), on page 94](#).
4. Create an input stream (`KVInputStream`) by calling [fpFiletoInputStreamCreate\(\)](#) or by using code similar to the example code in the `Filter` sample program.
5. Open the stream by calling [fpOpenStream\(\)](#).
6. Declare the input stream in the [KVOpenFileArg](#) structure.
7. Open the source file by calling [fpOpenFile\(\)](#) and passing the [KVOpenFileArg](#) structure. This call defines the parameters necessary to open a file for extraction.

8. Determine whether the source file is a container file (that is, whether it contains subfiles) by calling `fpGetMainFileInfo()`.
9. If the call to `fpGetMainFileInfo()` determined that the source file is a container file, proceed to step 10; otherwise, proceed to step 13.
10. Determine whether the subfile is a container file by calling `fpGetSubFileInfo()`.
11. Extract the subfile to a stream by calling `fpExtractSubFile()`.
12. If the call to `fpGetSubFileInfo()` determined that the subfile is a container file, repeat step 4 through step 11 until all subfiles are extracted; otherwise, proceed to step 13.
13. Filter the stream by calling `fpFilterStream()`. Call `fpFilterStream()` repeatedly until the entire output buffer is processed. After each call to `fpFilterStream()`, call `fpFreeFilterOutput()` to free the text buffer returned.
14. Close the stream by calling `fpCloseStream()`.
15. Free the memory allocated for the input stream by calling `fpFileToInputStreamFree()`.
16. Close the file by calling `fpCloseFile()`.
17. Repeat [Step 4](#) through [Step 16](#) as required for additional source files.
18. Terminate the filter session by calling `fpShutdown()`.

Multithreaded Filtering

To make sure that multithreaded filter processes are thread-safe, you must create a unique context pointer for every thread by calling `fpInit()`, on [page 147](#) or `fpInitWithLicenseData()`, on [page 149](#). In addition, threads must not share context pointers, and the same context pointer must be used for all API calls in the same thread. This applies to in-process and out-of-process API calls. Creating a context pointer for every thread does not affect performance because the context pointer uses minimal resources.

For example, C code for file filtering must have the following logic in a thread:

```
fpInit()  
    KVGetExtractInterface()  
    fpOpenFile()  
    fpGetMainFileInfo()          /* container file */  
    fpGetSubFileInfo()  
    fpExtractSubFile  
    fpGetSubFileMetadata()  
    fpFilterFile()  
    fpCloseFile()  
  
    fpOpenFile()  
    fpGetMainFileInfo()          /* not a container file */  
    fpGetDocInfoFile()  
    fpGetOLESummaryInfoFile()  
    fpFilterFile()  
    fpCloseFile()
```

```
...  
fpShutdown()
```

The Filter Process Model

By default, Filter runs independently from the calling application process. This is called *out-of-process* filtering. Out-of-process filtering protects the stability of the calling application in the rare case when a malformed document causes Filter to fail. You can configure Filter to run in the same process as the calling application. This is called *in-process* filtering. However, Micro Focus strongly recommends that you run Filter out of process whenever possible.

With the exception of Solaris and AIX, the creation of child processes on UNIX adheres to Portable Operating System Interface (POSIX) standards. Solaris and AIX use thread semantics. If required, a version of `kvfilter` with POSIX thread semantics is available for Solaris and AIX. For Solaris, the file is `kvfilter_posix.so`. For AIX, the file is `kvfilter_nsl.a`. These files must be renamed `kvfilter.so` or `kvfilter.a` to be used by Filter.

To monitor and debug filtering operations during out-of-process filtering, you can generate an error log at run time. See [Generate an Error Log, on page 58](#).

The following functions can run both in process or out of process:

Filter API

<code>fpCanFilterFile()</code>	<code>fpCanFilterStream()</code>
<code>fpFilterFile()</code>	<code>fpFilterStream()</code>
<code>fpGetDocInfoFile()</code>	<code>fpGetDocInfoStream()</code>
<code>fpGetOLESummaryInfo()</code>	<code>fpGetOLESummaryInfoFile()</code>
<code>fpGetDocInfoFile()</code>	<code>fpGetDocInfoStream()</code>

File Extraction API

<code>fpCloseFile()</code>	<code>fpExtractSubFile()</code>
<code>fpFreeStruct()</code>	<code>fpGetMainFileInfo()</code>
<code>fpGetSubFileInfo()</code>	<code>fpGetSubFileMetaData()</code>
<code>fpOpenFile()</code>	<code>KVGetExtractInterface()</code>

Other Filter API functions always run in process.

Persist the Child Process

By default, in out-of-process filtering, the parent process maintains a persistent connection with the child server after each file is filtered. When the connection is preserved in this way, subsequent filtering requests are processed more quickly because the server is already prepared to receive data.

You can restart the server at regular intervals by using a function or a configuration setting.

In the API

To force KeyView to restart, call the [fpRefreshFilterKVOOP\(\)](#) function.

In the formats.ini File

To control whether Filter persists the server, use the `kvoopRefresh` parameter in the `[FilterSDK_Config]` section of the `formats.ini` file:

`kvoopRefresh= 0` When you set `kvoopRefresh` to **0** (zero), the connection to the server persists for as long as the parent process is running or until the server fails. This is the default.

`kvoopRefresh= n` When you set `kvoopRefresh` to *n* (where *n* is a positive number), the connection persists for *n* filter requests. After the *n*th request, the server is shut down and restarted before processing the next request.

For example, if you set `kvoopRefresh` to **5**, the connection to the server persists for five filter requests. For the sixth request, the server is shut down and restarted.

To control whether the parent process attempts to filter a file after the file has caused the server to fail, use the `kvoopRetry` parameter in the `[FilterSDK_Config]` section of the `formats.ini` file:

`kvoopRetry= 0` When you set `kvoopRetry` to **0** and the server fails, the parent process does not resend the file to a new server.

`kvoopRetry= n` When you set `kvoopRetry` to *n* (where *n* is a positive number) and the server fails, the parent process resends the file to a new server *n* times. By default, `kvoopRetry` is set to **1**, and the file is resent to a server once.

NOTE: The `kvoopRefresh` and `kvoopRetry` parameters do not apply when you run the File Extraction functions out of process. See [Run File Extraction Functions Out of Process, on the next page](#).

Run Filter In Process

By default, Filter runs out of process. However, you can enable in-process filtering through the API or in the `formats.ini` file. If the type of process is not specified in the `formats.ini` or in the API, Filter is run out of process. If the type of process is specified in the `formats.ini` *and* in the API, the setting in the API takes precedence.

In the API

To run Filter in process

1. Set the final argument (`dwFlags`) of either [fplnit\(\)](#), on page 147, [fplnitWithLicenseData\(\)](#), on page 149 or [fpOpenStreamEx2\(\)](#) to `KVF_INPROCESS`.
2. `dwFlags |= KVF_INPROCESS`
3. Call a filtering function or a metadata extraction function. See [Filter API Functions](#), on page 121.
4. Optionally, call a metadata extraction function if a filter function was called in the previous step. See [fpGetDocInfoFile\(\)](#), on page 139 or [fpGetDocInfoStream\(\)](#), on page 140.

In the formats.ini File

To run Filter in process, set the `default_inprocess` parameter in the `[FilterSDK_Config]` section of the `formats.ini` file to `1`.

By default this parameter is set to `0` (zero), which enables out-of-process filtering.

Run File Extraction Functions Out of Process

The out-of-process setting specified in the call to [fplnit\(\)](#), on page 147 or [fplnitWithLicenseData\(\)](#), on page 149 or in the `formats.ini` file is automatically propagated to the File Extraction API in the call to `KVGetExtractInterface()`. In `KVGetExtractInterface()`, you pass a context pointer from [fplnit\(\)](#), on page 147 or [fplnitWithLicenseData\(\)](#), on page 149 and the address of a structure that contains pointers to the File Extraction functions.

When you extract subfiles from container files and pass the files for filtering out of process, Filter generates a server called `kvoop.exe` for filtering and a duplicate server (also called `kvoop.exe`) for file extraction. These servers are independent, so that if the filtering service stops responding, the file extraction service can continue extracting files.

Restart the File Extraction Server

If the file extraction server fails with either the `KVError_InvalidOopDriverSignature` error, or the `KVError_InvalidOopServiceSignature` error, you must restart the server by calling `KVGetExtractInterface()` and passing the original extraction structure. (Restarting the server in this way does not affect performance beyond the cost of restarting the server.)

If you restart the file extraction server before the recursive extraction of subfiles is complete, the new server has no history of the subfiles extracted prior to the restart. If you then call a File Extraction function on one of the extracted files, the `KVError_InvalidOopServiceSignature` error is generated, because the server that extracted the files is no longer running and was replaced with a new `kvoop` server. Micro Focus recommends that you do not make calls to the File Extraction functions by using an invalid container context structure (`KVContainerContext`) after you restart the server.

NOTE: Micro Focus recommends that whenever possible you restart the file extraction server only

after the file recursion is complete. There must be only one out-of-process session per file recursion.

Out-of-Process Logging

Logging is available for out-of-process filtering. The kvoop server can now create a log file that captures information on the files being processed, storing one entry per process. The generated log file is called `xxxx_kvoop.log`, where `xxxx` is a unique number identifying the process.

In the rare case when the kvoop server fails, you can use the log files to determine which file caused the failure. After processing is complete and the system shuts down, the logs are automatically deleted. To keep the log files after processing is successfully completed, see [Keep Log Files, on the next page](#).

NOTE: Out-of-process logging is not supported on AIX.

Enable Out-of-Process Logging

To enable out-of-process logging, set the `KVOOP_LOGS_DIR` environment variable to the directory in which you want the log files to be stored. By default, logging is not enabled.

On UNIX, set the variable as follows:

```
setenv KVOOP_LOGS_DIR /tmp
```

On Windows, set the variable as follows:

```
set KVOOP_LOGS_DIR=c:\tmp
```

The following log file is created in the directory:

```
process_id_kvoop.log
```

where *process_id* is a numeric value that represents the logged process. New messages are appended to the file, and truncation is disabled by default.

If KeyView terminates unexpectedly and Windows minidump is enabled, a *process_id_crash_info.txt* file is generated (see [Enable Windows Minidump, on the next page](#)). If logging was not enabled at the time of termination, this file contains instructions on how to enable logging.

Set the Verbosity Level

You can control how much information is written to the file by setting the `KVOOP_LOG_VERBOSITY` environment variable.

Set the variable to one of the following options:

- 1 Include only error messages.
- 2 Include errors and warnings.
- 3 Include errors, warnings, and general information. This is the default.
- 4 Include all possible information. This setting is useful for debugging purposes.

Enable Windows Minidump

KeyView can use the Windows minidump feature to provide additional logging information, which can be useful for debugging purposes.

The Windows minidump is disabled by default. To enable the Windows minidump, set `KV00P_DUMP_ENABLE` to `1`. If an unexpected termination occurs after the minidump is enabled, three files are generated:

- **`process_id_crash_info.txt`**. This file contains KV00P state and runtime information at the time of termination. If logging was not enabled at the time of termination, this file contains instructions on how to enable logging.
- **`process_id_process_list.txt`**. This file contains information from the DLLs that were loaded at the time of the termination.
- **`process_id_report.dmp`**. The Windows dump file, which contains further information about the termination. You can open it with either a Windows debugger or `autnhe1per.exe` (you must copy this file to the same directory).

You can control the amount of information presented in the Windows dump file by creating the following files in the directory:

```
dumper.NORMAL  
dumper.WITHDATASEGS  
dumper.WITHFULLMEMORY  
dumper.WITHHANDLEDATA
```

Keep Log Files

After processing is complete and the system is shut down, the log files are automatically deleted from the directory. To keep the log files after a successful run, set the `KV00P_KEEP_LOGS` environment variable.

On UNIX, set the variable as follows:

```
setenv KV00P_KEEP_LOGS 1
```

On Windows, set the variable as follows:

```
set KV00P_KEEP_LOGS=1
```

Run File Detection In or Out of Process

By default, detection runs in out-of-process mode. However, you can enable in-process detection through the API or in the `formats.ini` file. If the type of process is not specified in the `formats.ini` or in the API, detection runs in out-of-process mode. If the type of process is specified in the `formats.ini` *and* in the API, the setting in the API takes precedence.

Specify the Process Type In the formats.ini File

Add the `default_detect_inprocess` flag to a `[FilterSDK_Config]` section in the `formats.ini` file to control the default behavior for detection. Set the flag to `0` for out-of-process detection, and `1` for in-process detection. For example,

```
[FilterSDK Config]
default_detect_inprocess=0
```

If this flag is not specified, the file detection behavior is determined by the `default_inprocess` flag for filtering. For example, if you set `default_inprocess` to `1`, filtering and file detection runs in in-process mode by default; if you set `default_inprocess` to `0`, filtering and file detection runs in out-of-process mode by default.

If you set both the `default_inprocess` and `default_detect_inprocess` flags, `default_inprocess` controls the default filtering behavior and `default_detect_inprocess` controls the default file detection behavior.

Specify the Process Type In the API

Set the final argument (`dwFlags`) of either [fplnit\(\)](#), on page 147, [fplnitWithLicenseData\(\)](#), on page 149 or [fpOpenStreamEx2\(\)](#) to `KVF_DETECT_INPROCESS` or `KVF_DETECT_OUTOFPROCESS`.

Part II: Use Filter SDK

This section explains how to perform some basic tasks by using the File Extraction and Filter APIs, and describes the sample programs.

Chapter 3: Use the File Extraction API

This section describes how to extract subfiles from a container file by using the File Extraction API.

• Introduction	35
• Extract Subfiles	36
• Extract Images	38
• Recreate a File's Hierarchy	38
• Extract Mail Metadata	40
• Extract Subfiles from Outlook Files	47
• Extract Subfiles from Outlook Express Files	47
• Extract Subfiles from Mailbox Files	47
• Extract Subfiles from Outlook Personal Folders Files	47
• Extract Subfiles from Lotus Domino XML Language Files	51
• Extract Subfiles from Lotus Notes Database Files	52
• Extract Subfiles from PDF Files	55
• Extract Embedded OLE Objects	55
• Extract Subfiles from ZIP Files	56
• Default File Names for Extracted Subfiles	56

Introduction

To filter a file, you must first determine whether the file contains any subfiles (attachments, embedded OLE objects, and so on). A file that contains subfiles is called a *container* file. A container file has a main file (parent) and subfiles (children) embedded in the main file.

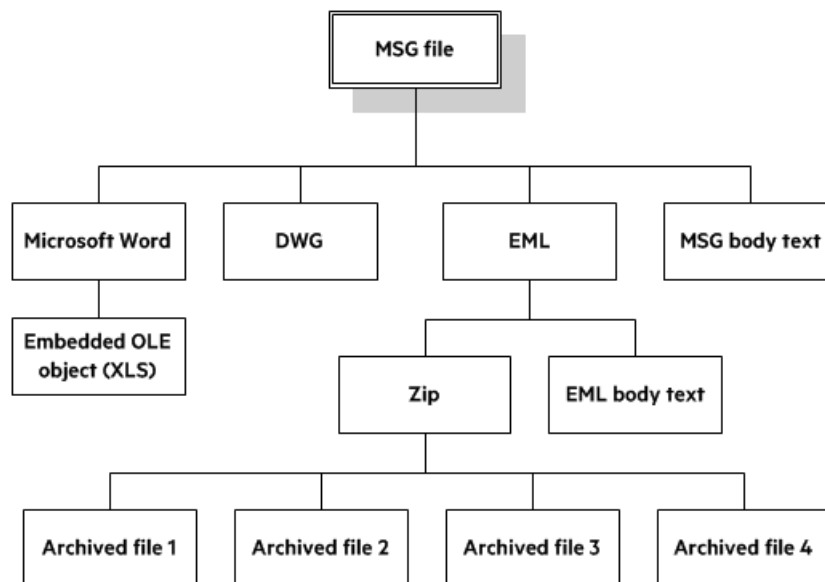
The following are examples of container files:

- Archive files such as ZIP, TAR, and RAR.
- Mail messages such as Outlook (MSG) and Outlook Express (EML).
- Mail stores such as Microsoft Outlook Personal Folders (PST), Mailbox (MBX), and Lotus Notes database (NSF).
- PDF files that contain file attachments.
- Compound documents with embedded OLE objects such as a Microsoft Word document with an embedded Excel chart.

NOTE: [Supported Formats, on page 191](#) indicates which formats are treated as container files and are supported by the File Extraction API.

The subfiles might also be container files, creating a file hierarchy of multiple levels. For example, an MSG file (the root parent) might contain three attachments:

- a Microsoft Word document that contains an embedded Microsoft Excel spreadsheet.
- an AutoCAD drawing file (DWG).
- an EML file with an attached Zip file, which in turn contains four archived files.



The parent MSG file contains four first-level children. The body text of a message file, although not a standalone file in the container, is considered a child of the parent file.

Extract Subfiles

To filter all files in a container file, you must open the container and extract its subfiles by using the *File Extraction API*. The extraction process is done repeatedly until all subfiles are extracted and exposed for filtering. After a subfile is extracted, you can call Filter API functions to filter the file.

If you want to filter a container file and its subfiles to a single file, you must extract all files from the container, filter the files, and then append each filtered output file to its parent.

To extract subfiles

1. Pass the context pointer from `fpInit()` or `fpInitwithLicenseData()` and the address of a structure that contains pointers to the File Extraction API functions in the call to `KVGetExtractInterface()`.
2. Declare the input stream or file name in the `KVOpenFileArg` structure.
3. Open the source file by calling `fpOpenFile()` and passing the `KVOpenFileArg` structure. This call defines the parameters necessary to open a file for extraction.

4. Determine whether the source file is a container file (that is, whether it contains subfiles) by calling [fpGetMainFileInfo\(\)](#).
5. If the call to [fpGetMainFileInfo\(\)](#) determined that the source file is a container file, proceed to step 6; otherwise, filter the file.
6. Determine whether the subfile is itself a container (that is, whether it contains subfiles) by calling [fpGetSubFileInfo\(\)](#).
7. Extract the subfile by calling [fpExtractSubFile\(\)](#).
8. If the call to [fpGetSubFileInfo\(\)](#) determined that the subfile is a container file, repeat step 2 through step 7 until all subfiles are extracted and the lowest level of subfiles is reached; otherwise, filter the file.

Sanitize Absolute Paths

When you extract a subfile from a container and write it to disk, you specify an extract directory and a path to extract the file to.

To set the path, you might use the path in the container file that you are extracting from, as returned from the function [fpGetSubFileInfo\(\)](#), on page 99. However, if the path is an absolute path, the file could be created outside the directory you have chosen as the extract directory. Your application might then contain a vulnerability that could be exploited to write files to unexpected locations in the file system. This section discusses some KeyView features that can help you secure your application by sanitizing paths.

KeyView always sanitizes relative paths that you pass in when extracting files, so that the paths remain within the extract directory you specify. For example, KeyView does not allow the use of "." to move outside the extract directory.

KeyView can update absolute paths so that they remain within the extract directory. You can instruct KeyView to sanitize absolute paths programmatically (through the API), or by setting a parameter in the configuration file.

The following table shows the effect on some example paths.

Requested path	Path of extracted file (not sanitized)	Path of extracted file (sanitized)
file.txt	<i>extractDir/file.txt</i>	<i>extractDir/file.txt</i>
dir/file.txt	<i>extractDir/dir/file.txt</i>	<i>extractDir/dir/file.txt</i>
../file.txt	<i>extractDir/file.txt</i>	<i>extractDir/file.txt</i>
/dir/file.txt	<i>/dir/file.txt</i>	<i>extractDir/dir/file.txt</i>

To sanitize absolute paths

- In the [KVExtractSubFileArg](#) struct that you pass in to [fpExtractSubFile](#), set the flag `KVExtractionFlag_SanitizeAbsolutePaths`. When KeyView sanitizes a path and the resulting directory does not exist, extraction fails unless you instruct KeyView to create the directory, so

you might also want to set the flag `KVExtractionFlag_CreateDir`. You can find the path that a file was actually extracted to from the [KVSubFileExtractInfo](#) structure.

To sanitize absolute paths (through configuration)

- In the `formats.ini` configuration file, set the parameter `SanitizeAbsoluteExtractPaths`, for example:

```
[Options]
SanitizeAbsoluteExtractPaths=TRUE
```

Extract Images

You can use the File Extraction API to extract images within the file by specifying the following in the `formats.ini` file:

```
[Options]
ExtractImages=TRUE
```

If you set this option, images within the file behave in the same way as any other subfile. Extracted images have the name `image[X].[Y]`, where `[X]` is an integer, and `[Y]` is the extension. The format of the image is the same as the format in which it is stored in the document.

This option can also be enabled by passing `KVFLT_EXTRACTIMAGES` to the `fpFilterConfig` function.

NOTE: Turning on `ExtractImages` can reduce the speed of the filtering operation.

Recreate a File's Hierarchy

When you extract a container file, any relationships between the subfiles in the container are not maintained. However, the File Extraction interface provides information that enables you to recreate the hierarchy. You can use the hierarchy to create a directory structure in a file system, or to categorize documents according to their relationship to each other. For example, if you use `KeyView` to generate text for a search engine, the hierarchical information enables your users to search for a document based on the document's parent or sibling. In addition, when the document is returned to the user, the parent and sibling documents can be returned as recommendations.

The information needed to recreate a file's hierarchy is provided in the call to [fpGetSubFileInfo\(\)](#). The members `KVSubFileInfo->parentIndex` and `KVSubFileInfo->childArray` provide information about a subfile's parent and children. Because you can only retrieve the first-level children in the subfile, you must call `fpGetSubFileInfo()` repeatedly until information for the leaf-node children is extracted.

Create a Root Node

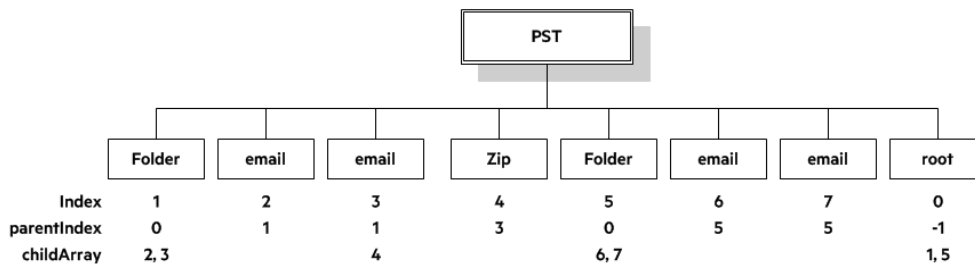
Because of their structure, some container files do not contain a subfile or folder which acts as a root directory on which the hierarchy can be based. For example, subfiles in a Zip archive can be extracted, but none of the subfiles represent the root of the hierarchy. In this case, you must create an artificial

root node at the top of the file hierarchy as a point of reference for each child, and ultimately to recreate the relationships. This artificial root node is an internal object, and is extracted to disk as a directory called *root*. Its index number is 0.

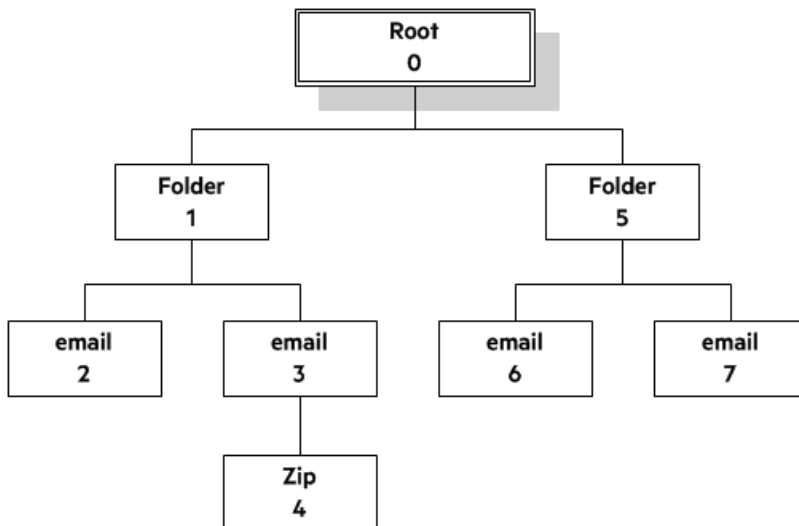
To create the root node, set `openFlag` to `KVOpenFileFlag_CreateRootNode` in the call to `fpOpenFile()`. When you create a root node, the value of `numSubFiles` in `KVMainFileInfo` includes the root node. For example, when you call `fpGetMainFileInfo()` on a Microsoft Word document with three embedded OLE objects and the root node is disabled, `numSubFiles` is 3. If you create a root node, `numSubFiles` is 4.

Recreate a File’s Hierarchy—Example

For example, you might extract a PST file that contains seven subfiles with a root node enabled. The call to `fpGetMainFileInfo()` returns the number of subfiles as eight (seven subfiles and one root node). The following diagram shows the structure and the available hierarchy information after the subfiles are extracted:



The `parentIndex` specifies the index number of a subfile’s parent. The `childArray` specifies an array of a subfile’s children. With this information, you can recreate the hierarchy shown in the following diagram.



Extract Mail Metadata

You can extract metadata, such as subject, sender, and recipient, from subfiles of mail formats, by calling the `fpGetSubFileMetaData()` function. You can extract a predefined set of common metadata fields, a list of metadata fields by their names or MAPI properties, or, for some subfile types, all the metadata in the file.

Default Metadata Set

KeyView internally defines a set of common mail metadata fields that you can extract as a group from mail formats. This default metadata set is listed in the following table.

Default Mail Metadata List

Field Name (string to specify)	Description
From	The display name and email address of the sender.
Sent	The time that the message was sent.
To	The display names and email addresses of the recipients.
Cc	The display names and email addresses of recipients who receive copies of the email.
Bcc	The display names and email addresses of recipients who received blind copies of the email.
Subject	The text in the subject line of the message.
Priority	The priority applied to the message.

Because mail formats use different terms for the same fields, the format's reader maps the default field name to the appropriate format-specific name. For example, when retrieving the default metadata set, the NSF field *Importance* is mapped to the name *Priority* and is returned.

You can also extract the default field names individually by passing the field name (such as *From*, *To*, and *Subject*); however, in this case, the string is not mapped to the format-specific name. For example, if you pass *Priority* in the call, you retrieve the contents of the *Priority* field from an MBX file, but do not retrieve the contents of the *Importance* field from an NSF file.

You cannot pass the field names listed in the table individually for PST files. However, you can pass either the MAPI tag number or the MAPI tag name as integers. See [Microsoft Personal Folders File \(PST\) Metadata, on page 45](#).

Extract the Default Metadata Set

To extract the default metadata set, call the [fpGetSubFileMetaData\(\)](#) function, and pass in 0 for metaArg->metaNameCount, and NULL for metaArg->metaNameArray.

```
KVGetSubFileMetaArgRec metaArg;  
KVSubFileMetaData pMetaData = NULL;  
KVStructInit(&metaArg);  
  
metaArg.index = subFileIndex;  
metaArg.metaNameCount = 0;  
metaArg.metaNameArray = NULL;  
  
error = extractInterface->fpGetSubFileMetaData(pFile, &metaArg, &pMetaData);  
...  
extractInterface->fpFreeStruct(pFile, pMetaData);  
pMetaData = NULL;
```

Extract All Metadata

KeyView can extract all metadata from EML, MBX, MIME, NSF, ICS, and DXL subfiles. You can extract all metadata in a similar way to extracting the default metadata set, but when you call the [fpGetSubFileMetaData\(\)](#) function, pass in -1 for metaArg->metaNameCount and NULL for metaArg->metaNameArray.

Microsoft Outlook (MSG) Metadata

In addition to the default metadata set, you can extract the metadata fields listed in the following table for MSG files. You must pass the field name to metaNameArray in the call to the fpGetSubFileMetadata() function.

MSG-specific Metadata List

Field Name (string to specify)	Description
AttachFileName	An attachment's long file name and extension, excluding the path.
ConversationTopic	The topic of the first message in a conversation thread. A conversation thread is a series of messages and replies. This is the first message's subject with any prefix removed.
CreationTime	The time that the message or attachment was created. This value is displayed in the Sent field in the message's Properties dialog in Outlook.
InternetMessageID	The identifier for messages that come in over the Internet. This is the MAPI property PR_INTERNET_MESSAGE_ID. This property is not in the MAPI headers or MAPI documentation.

MSG-specific Metadata List, continued

Field Name (string to specify)	Description
LastModificationTime	The time that the message or attachment was last modified. This value is displayed in the Modified field in the message's Properties dialog in Outlook.
Location	The physical location of the event specified in the Outlook calendar entry.
MessageID	The message transfer system (MTS) identifier for the message transfer agent (MTA). This value is displayed on the Message ID tab in the message's Properties dialog in Outlook.
Received	The date and time a message was delivered. This value is displayed in the Received field in the message's Properties dialog in Outlook.
Sender	The name and email address of the message sender. This value is a concatenation of two MAPI properties in the following format: "PR_SENDER_NAME" <PR_SENDER_EMAIL_ADDRESS> The Sender value might be the same as or different than the default metadata From value (see Default Metadata Set, on page 40), depending on which MAPI properties exist in the MSG file.
Sensitivity	The value indicating the message sender's opinion of the sensitivity of a message. For example, Personal, Private, or Confidential. This value is displayed in the Sensitivity field in the message's Properties dialog in Outlook.
TransportMsgHeaders	Transport-specific message envelope information. This value corresponds to the MAPI property PR_TRANSPORT_MESSAGE_HEADERS.
StartDate	An appointment start date. This value corresponds to the PR_START_DATE MAPI property.
EndDate	An appointment end date. This value corresponds to the PR_END_DATE MAPI property.

Extract MSG-Specific Metadata

To extract specific metadata fields from an MSG file, call the [fpGetSubFileMetaData\(\)](#) function, and pass the field name defined in [Default Metadata Set, on page 40](#) to metaNameArray (the string is not case sensitive).

For example, the following code extracts the contents of the ConversationTopic and MessageID fields:

```
KVGetSubFileMetaArgRec metaArg;  
KVSubFileMetaData pMetaData = NULL;  
KVStructInit(&metaArg);
```

```
KVMetaNameRec names[2];
KVMetaName     pname[2];

names[0].type = KVMetaNameType_String;
names[0].name.sname = "conversationtopic";
names[1].type = KVMetaNameType_String;
names[1].name.sname = "MessageID";

pname[0] = &names[0];
pname[1] = &names[1];

metaArg.metaNameCount = 2;
metaArg.metaNameArray = pname;
metaArg.index = subFileIndex;

error = extractInterface->fpGetSubFileMetaData(pFile, &metaArg, &pMetaData);
...
extractInterface->fpFreeStruct(pFile, pMetaData);
pMetaData = NULL;
```

Microsoft Outlook Express (EML) and Mailbox (MBX) Metadata

In addition to the default metadata set, you can extract any metadata field that exists in the header of an EML or MBX file by passing the field's name. If the name is a valid field in the file, the content of the field is returned. For example, to retrieve the name of the last mail server that received the message before it was delivered, you can pass the string "Received".

Extract EML- or MBX-Specific Metadata

To extract specific metadata fields from an EML or MBX file, call the [fpGetSubFileMetaData\(\)](#) function, and pass the metadata name to `metaNameArray` (the string is *not* case sensitive).

For example, the following code extracts the contents of the Received and Mime-version fields:

```
KVGetSubFileMetaArgRec metaArg;
KVSubFileMetaData pMetaData = NULL;
KVStructInit(&metaArg);
KVMetaNameRec names[2];
KVMetaName     pname[2];

names[0].type = KVMetaNameType_String;
names[0].name.sname = "Received";
names[1].type = KVMetaNameType_String;
names[1].name.sname = "Mime-version";

pname[0] = &names[0];
pname[1] = &names[1];

metaArg.metaNameCount = 2;
```

```
metaArg.metaNameArray = pname;  
metaArg.index = subFileIndex;  
error = extractInterface->fpGetSubFileMetaData(pFile, &metaArg, &pMetaData);  
...  
extractInterface->fpFreeStruct(pFile,pMetaData);  
pMetaData = NULL;
```

Lotus Notes Database (NSF) Metadata

In addition to the default metadata set, you can extract any Lotus field name that exists in an NSF file by passing the field's name. (You can extract fields from mail NSF files and non-mail NSF files.) If the name is a valid field in the file, the field is returned. For example, to retrieve the date when a document in an NSF file was last accessed, you would pass the string "\$LastAccessedDB".

A complete list of NSF fields is provided in the Lotus Notes file `stdnames.h`. This header file is available in the Lotus API Toolkit.

Extract NSF-Specific Metadata

To extract specific metadata fields from an NSF file, call the [fpGetSubFileMetaData\(\)](#) function, and pass the metadata name to `metaNameArray` (the string is *not* case sensitive).

For example, the following code extracts the contents of the `Description` and `Categories` fields:

```
KVGetSubFileMetaArgRec metaArg;  
KVSubFileMetaData pMetaData = NULL;  
KVStructInit(&metaArg);  
KVMetaNameRec names[2];  
KVMetaName pname[2];  
  
names[0].type = KVMetaNameType_String;  
names[0].name.sname = "description";  
names[1].type = KVMetaNameType_String;  
names[1].name.sname = "Categories";  
  
pname[0] = &names[0];  
pname[1] = &names[1];  
  
metaArg.metaNameCount = 2;  
metaArg.metaNameArray = pname;  
metaArg.index = subFileIndex;  
  
error = extractInterface->fpGetSubFileMetaData(pFile, &metaArg, &pMetaData);  
...  
extractInterface->fpFreeStruct(pFile,pMetaData);  
pMetaData = NULL;
```

Microsoft Personal Folders File (PST) Metadata

In addition to the default metadata set, you can extract Messaging Application Programming Interface (MAPI) properties from a PST file. These properties describe all elements of an Outlook item in a PST file (such as subject, sender, recipient, and message text). Because the properties are stored in the PST file itself, you can retrieve them *before* you extract the contents of the PST. This enables you to determine whether an Outlook item should be extracted based on its attributes. Some MAPI properties are also stored for Outlook attachments that are *not* mail messages (such as an attached Microsoft Word document or Lotus 1-2-3 file).

Because all elements of a message (except non-mail attachments) are represented by MAPI properties, you can extract all components of a subfile, including the header and message text, by calling the `fpGetSubFileMetadata()` function.

MAPI Properties

Each MAPI property is identified by a property tag, which is a constant that contains the property type and a unique identifier. For example, the property that indicates whether a message has attachments has the following components:

Property	PR_HASATTACH
Identifier	0x0E1B
Property type	PT_BOOLEAN (000B)
Property tag	0x0E1B000B

The Microsoft MAPI documentation on the Microsoft Developer Network website lists all available MAPI properties, their tags, and types.

You can retrieve any MAPI property that is of one of the MAPI property types listed below:

PT_I2	PT_DOUBLE	PT_STRING8
PT_I4	PT_FLOAT	PT_TSTRING
PT_BINARY	PT_LONG	PT_SYSTIME
PT_BOOLEAN	PT_SHORT	PT_UNICODE

Properties with a `PT_TSTRING` type have the property type recompiled to either a Unicode string (`PT_UNICODE`) or to an ANSI string (`PT_STRING8`) depending on the operating system's character set. To retrieve the Unicode property, pass in the Unicode version of the tag. For example, the property tag for `PR_SUBJECT` is either `0x0037001E` for an ANSI string, or `0x0037001F` for a Unicode string.

Extract PST-Specific Metadata

In the call to extract subfile metadata, you can pass either the MAPI tag number (such as 0x0070001e) or the MAPI tag name (such as PR_CONVERSATION_TOPIC). If you specify the MAPI tag name, you must include the `mapitags.h` and `mapidefs.h` Windows header files, in which the MAPI tag name is defined as a tag number.

To extract specific MAPI properties from a PST file, call the `fpGetSubFileMetaData()` function, and pass the property tag to `metaNameArray`. The tag is passed as an integer.

For example, the following code extracts the MAPI properties `PR_SUBJECT` and `PR_ALTERNATE_RECIPIENT`:

```
KVGetSubFileMetaArgRec metaArg;
KVSubFileMetaData pMetaData = NULL;
KVMetaNameRec names[2];
KVMetaName pName[2];

names[0].type = KVMetaNameType_Integer;
names[0].name.iname = PR_SUBJECT;

names[1].type = KVMetaNameType_Integer;
names[1].name.iname = 0x3A010102;

pName[0] = &names[0];
pName[1] = &names[1];

KVStructInit(&metaArg);

metaArg.metaNameCount = 2;
metaArg.metaNameArray = pName;
metaArg.index = SubFileIndex;

error = extractInterface->fpGetSubFileMetaData (pFile,&metaArg,&pMetaData);
...
extractInterface->fpFreeStruct(pFile,pMetaData);

pMetaData = NULL;
```

You must include the `mapitags.h` and `mapidefs.h` Windows header files, in which `PR_SUBJECT` is defined as `0x0037001E`.

Exclude Metadata from the Extracted Text File

When you extract a mail message, the message text and header information (To, From, Sent, and so on) is also extracted. You can prevent the header information from appearing in the text file.

To exclude the header information, set `extractFlag` to `KVExtractionFlag_ExcludeMailHeader` in the call to `fpExtractSubFile()`.

Extract Subfiles from Outlook Files

When you extract an Outlook file (MSG) to disk, the message text and header information (To, From, Sent, and so on) is extracted to a text file. (If you do not want the header information to appear in the text file, see [Exclude Metadata from the Extracted Text File, on the previous page.](#)) If the Outlook file contains a non-mail attachment, the attachment is extracted in its native format to a subdirectory. If the Outlook file contains a mail attachment, the attachment's message text is extracted to a subdirectory.

Extract Subfiles from Outlook Express Files

When you extract an Outlook Express (EML) file to disk, the message text and header information (To, From, Sent, and so on) is extracted to a text file. (If you do not want the header information to appear in the text file, see [Exclude Metadata from the Extracted Text File, on the previous page.](#)) If the Outlook file contains a non-mail attachment, the attachment is extracted in its native format to the same directory as the message text file. If the Outlook file contains a mail attachment, the complete attachment (including message text and attachments), the message text file, and any non-mail attachments are extracted to the same directory as the main message.

When the MBX reader (`mbxsr`) is enabled, it is used to filter MBX *and* EML files. If the MBX reader is not enabled, the EML reader (`eml1sr`) is used.

Extract Subfiles from Mailbox Files

A Mailbox (MBX) file is a collection of individual emails compiled with RFC 822 and RFC 2045 - 2049 (MIME), and divided by message separators. There are many mail applications that export to an MBX format, such as Eudora Email and Mozilla Thunderbird.

When an MBX file is extracted to disk, the message text and header information (To, From, Sent, and so on) from each mail file is extracted to text files. (If you do not want the header information to appear in the text file, see [Exclude Metadata from the Extracted Text File, on the previous page.](#))

In Eudora MBX files, attachments are inserted as a link and are stored externally from the message. These attachments are not extracted, but the path to the attachment is returned in the call to the [fpGetSubFileInfo\(\)](#) function. You can write code to retrieve the attachment based on the returned path.

For MBX files from other clients, KeyView extracts attachments when they are embedded in the message.

The Mailbox (MBX) reader is an advanced feature and is sold and licensed separately. To enable this reader in a KeyView SDK, you must obtain the appropriate license key from Micro Focus. See [Update License Information, on page 18](#) for information on adding a new license key to an existing installation.

Extract Subfiles from Outlook Personal Folders Files

KeyView can extract Outlook items such as messages, appointments, contacts, tasks, notes, and journal entries from a PST file. When a PST file is extracted to disk, the text and header information

(To, From, Sent, and so on) from each Outlook item is extracted to a text file. (If you do not want the header information to appear in the text file, see [Exclude Metadata from the Extracted Text File, on page 46.](#))

You can also extract messages from PST files as MSG files, including all their attachments, by setting the `KVExtractionFlag_SaveAsMSG` flag in the `KVExtractSubFileArg` structure when you call `fpExtractSubFile()`.

If an Outlook item contains a non-mail attachment, the attachment is extracted in its native format to a subdirectory. If an Outlook item contains an Outlook attachment, the attached item's text and any attachments are extracted to a subdirectory.

The Microsoft Outlook Personal Folders (PST) readers are an advanced feature and are sold and licensed separately. To enable these readers in a KeyView SDK, you must obtain an appropriate license key from Micro Focus. For information about adding a new license key to an existing installation, see [Update License Information, on page 18.](#)

Choose the Reader to use for PST Files

KeyView provides several ways of processing PST files:

- Indirectly, using the Microsoft Messaging Application Programming Interface (MAPI). MAPI is a Microsoft interface that enables different applications to exchange messages and attachments with each other. MAPI allows KeyView to open a PST file, traverse the folders, and extract items. The `pstsr` reader uses MAPI, but works only on Windows and requires that Microsoft Outlook is installed.
- Directly, without relying on the Microsoft interface to the PST format. Accessing the file directly does not require Microsoft Outlook. The `pstxsr` reader is available for Windows (32-bit and 64-bit) and Linux (64-bit only). The `pstnsr` reader is an alternative native reader, for the platforms not supported by `pstxsr`.

On Windows, the MAPI-based reader is used by default but you can choose `pstxsr` if you prefer. On UNIX platforms, only one of the native readers is available (`pstxsr` on Linux x64 and `pstnsr` on other platforms).

The differences between the readers are summarized in the following table.

Feature	Native Reader (pstxsr)	Native Reader (pstnsr)	MAPI-based Reader (pstsr)
Platforms supported	Windows x86 and x64 Linux x64	All platforms not supported by <code>pstxsr</code>	Windows x86 and x64
Outlook required	No	No	Yes
MAPI properties supported	Yes. All properties defined in <code>mapitags.h</code> . Object properties are not supported.		
Password protection supported	Yes	Yes	Yes (using <code>KVCredential</code> structure)

Feature	Native Reader (pstxsr)	Native Reader (pstnsr)	MAPI-based Reader (psts)
Compressible encryption supported	Yes	Yes	Yes
High encryption supported	No	No	Yes

To change the reader used to process PST files, change the PST entry (file category value 297) in the `formats.ini` file. For example, to use `pstxsr`:

297=**pstx**

You must make sure that the PST that you are extracting is not open in the Outlook client, and that the Outlook process is not running.

When extracting subfiles from PST files, information on the distribution list used in an email is extracted to a file called `emailname.dist`. This applies to the MAPI reader (`psts`) only.

System Requirements

MAPI is supported on Windows platforms only and relies on functionality in Outlook. If you want to use the MAPI-based reader, `psts`, Microsoft Outlook must be installed on the same machine as your application. Outlook must also be the default email application. KeyView supports the following PST formats and Outlook clients:

- Outlook 97 or later PST files

The Outlook client must be the same version as, or newer than, the version of Outlook that generated the PST file.

- Outlook 2002 or later clients

NOTE: You must install an edition of Microsoft Outlook (32-bit or 64-bit) that matches the KeyView software. For example, if you use 32-bit KeyView, install 32-bit Outlook. If you use 64-bit KeyView, install 64-bit Outlook.

If the editions do not match, KeyView returns Error 32: `KVError_PSTAccessFailed` and an error message from Microsoft Office Outlook is displayed: Either there is a no default mail client or the current mail client cannot fulfill the messaging request. Please run Microsoft Outlook and set it as the default mail client.

MAPI Attachment Methods

The way in which you can access the contents of a PST message attachment is determined by the MAPI *attachment method* applied to the attachment. For example, if the attachment is an embedded OLE object, it uses the `ATTACH_OLE` attachment method. KeyView can access message attachments that use the following attachment methods:

ATTACH_BY_VALUE
ATTACH_EMBEDDED_MSG
ATTACH_OLE
ATTACH_BY_REFERENCE
ATTACH_BY_REF_ONLY
ATTACH_BY_REF_RESOLVE

Attachments using the ATTACH_BY_VALUE, ATTACH_EMBEDDED_MSG, or ATTACH_OLE attachment methods are extracted automatically when the PST file is extracted. An "attach by reference" method means that the attachment is not in Outlook, but Outlook contains an absolute path to the attachment. Before you can extract these types of attachments, you must retrieve the path to access the attachment.

To extract "attach by reference" attachments

1. Determine whether the attachment uses an ATTACH_BY_REFERENCE, ATTACH_BY_REF_ONLY, or ATTACH_BY_REF_RESOLVE method by retrieving the MAPI property PR_ATTACH_METHOD.
2. If the attachment uses one of the "attach by reference" methods, get the fully qualified path to the attachment by retrieving the MAPI properties PR_ATTACH_LONG_PATHNAME or PR_ATTACH_PATHNAME.
3. You can then either copy the files from their original location to the path where the PST file is extracted, or use the Filter API functions to filter the attachment.

Open Secured PST Files

KeyView enables you to specify a user name and password to use to open a secured PST file for extraction.

To open password-protected PST files that use high encryption, you must use the MAPI-based PST reader (`pstsr`). The native PST readers (`pstxsr` and `pstnsr`) return the error message `KVERR_PasswordProtected` if a PST file is encrypted with high encryption.

Detect PST Files While the Outlook Client is Running

If you are running an Outlook client while running the File Extraction API, the KeyView format detection module (`kwad`) might not be able to open the PST file to determine the file's format because Outlook has the file locked. In this case, you can do one of the following:

- Close Outlook when using the Extraction API.
- Detect PST files by extension only and bypass the format detection module. To enable this option, add the following lines to the `formats.ini` file:

```
[container_flags]
detectPSTbyExtension=1
```

The `detectPSTbyExtension` option applies only when you are using the MAPI reader (`pstsr`).

If you use this option, you must make sure in your code that valid PST files are passed to `KeyView`, because the format detection module is not available to verify the file type and pass the file to the appropriate reader.

Extract Subfiles from Lotus Domino XML Language Files

When you extract a Lotus Domino XML Language (.DXL) file, the message text and header information (*To*, *From*, *Sent*, and so on) is extracted to a text file.

To prevent header information from being extracted, see [Exclude Metadata from the Extracted Text File, on page 46](#).

You can make sure that dates and times extracted from Lotus Domino .DXL files are displayed in a uniform format.

To extract custom date/time formats

- In the `formats.ini` file, set the `DateTimeFormat` option in the `[dxlsr]` section. For example:

```
[dxlsr]
DateTimeFormat=%m/%d/%Y %I:%M:%S %p
```

In this example, dates and times are extracted in the following format:

```
02/11/2003 11:36:09 AM
```

The format arguments are the same as those for the `strftime()` function. See <http://msdn.microsoft.com/en-us/library/fe06s4ak%28VS.71%29.aspx> for more information.

Extract .DXL Files to HTML

You can use the file extraction API to process .DXL files with an XSLT engine. The XSLT engine then transforms the extracted .DXL to .mail HTML files.

To extract .DXL files to HTML

- Set the following options in the `formats.ini` file:

```
[nsfsr]
ExportDXL=1
ExportDXL_PureXML=1
```

```
[dxlsr]
LNParser=2
```

Extract Subfiles from Lotus Notes Database Files

A Lotus Notes database is a single file that contains multiple documents called *notes*. Notes include design notes (such as forms, views, folders, navigators, outlines, pages, framesets, agents, and resources), data document notes, profile document notes, access control list notes, and collection (index) notes. KeyView can extract text items, attachments, and OLE objects from *data document notes* only. Data document notes include emails, journal entries, discussion threads, documents (Microsoft Office and Lotus SmartSuite), and so on.

All components of a note are prefixed by field names such as "SendTo:", "Subject:", and "Body:". When a note is extracted, the field names are not included in the extracted output; only the field values are extracted.

When a mail message in an NSF file is extracted to disk, the body text and header information (such as the values from the SendTo, From, and DeliveredDate fields) in each message is extracted to a text file. (If you do not want the header information to appear in the message text file, see [Exclude Metadata from the Extracted Text File, on page 46.](#))

The Lotus Notes Database (NSF) reader is an advanced feature and is sold and licensed separately. To enable this reader in a KeyView SDK, you must obtain the appropriate license key from Micro Focus. See [Update License Information, on page 18](#) for information on adding a new license key to an existing installation.

System Requirements

The NSF format is proprietary. Therefore, KeyView accesses NSF files indirectly by using the Lotus Notes API. Because the NSF reader relies on functionality in Lotus Notes, a Lotus Notes client or Lotus Domino server must be installed and configured on the same machine as the application filtering NSF files. On UNIX and Linux, the Lotus Domino server is required. On Windows, the Lotus Notes client or Lotus Domino server is required.

KeyView supports the following Lotus Notes clients and Domino servers:

- Lotus Notes 6.5.1
- Lotus Domino 6.5.1

KeyView supports NSF files on the same platforms supported by Lotus Notes and Lotus Domino:

- Windows XP x86 (Service Pack 1 and 2)
- Windows 2000 x86 (Service Pack 2)
- Solaris 8.0 and 9.0 (built on Solaris 8.0)
- Red Hat Enterprise Linux AS 3.0 (x86)
- SuSE Linux Enterprise Server 8 and 9 (x86)
- IBM AIX 5.1, 5L version 5.2

Installation and Configuration

Before KeyView can filter NSF files, you must set up the Lotus Notes client or Lotus Domino server. Full configuration is not required. The following steps outline the minimal setup for NSF filtering:

Windows

1. Install the Lotus Notes client or Lotus Domino server. You do not need to configure the client or server.
2. Make sure that the `notes.ini` file is in the proper location.
 - If Lotus Notes is installed, the file should appear in the `install\lotus\notes` directory, where `install` is the installation directory.
 - If only Lotus Domino is installed, the file should appear in the `install\lotus\domino` directory, where `install` is the installation directory.

If the file does not exist, create an ASCII file named `notes.ini`, and add the following text:

```
[Notes]
```

3. Add the KeyView bin directory and the `install\lotus\notes` or `install\lotus\domino` directory to the PATH environment variable (the KeyView bin directory must be first in the path). Micro Focus recommends that you add the KeyView bin directory because the Lotus Notes or Domino server installation might contain older KeyView OEM libraries.

Solaris

1. Install Lotus Domino server. You do not need to configure the server.
2. Make sure that the `notes.ini` file is in the `install/lotus/notes/latest/sunspa` directory, where `install` is the directory where Lotus Notes is installed. If the file does not exist, create an ASCII file named `notes.ini`, and add the following text:

```
[Notes]
```

3. Add the `install/lotus/notes/latest/sunspa` directory to the PATH environment variable:

```
setenv PATH install/lotus/notes/latest/sunspa:$PATH
```

4. Add the `install/lotus/notes/latest/sunspa` and the KeyView bin directory to the LD_LIBRARY_PATH environment variable:

```
setenv LD_LIBRARY_PATH keyview_bin:install/lotus/notes/latest/sunspa:$LD_LIBRARY_PATH
```

where `keyview_bin` is the location of the KeyView bin directory. Micro Focus recommends that you add the KeyView bin directory because the Lotus Notes installation might contain older KeyView OEM libraries.

AIX 5.x

1. Install the `bos.iocp.rte` file set if it is not already installed, and reboot the machine. See the Lotus Domino server documentation for more information.
2. Install Lotus Domino server. You do not need to configure the server.
3. Make sure that the `notes.ini` file is in the `install/lotus/notes/latest/ibmpow` directory, where `install` is the directory where Lotus Notes is installed. If the file does not exist, create an ASCII file named `notes.ini`, and add the following text:

```
[Notes]
```

4. Add the `install/lotus/notes/latest/ibmpow` directory to the `PATH` environment variable:

```
setenv PATH install/lotus/notes/latest/ibmpow:$PATH
```
5. Add the `install/lotus/notes/latest/ibmpow` and the `KeyView bin` directory to the `LIBPATH` environment variable:

```
setenv LIBPATH keyview_bin:install/lotus/notes/latest/ibmpow:$LIBPATH
```

where `keyview_bin` is the location of the `KeyView bin` directory. Micro Focus recommends that you add the `KeyView bin` directory because the Lotus Notes installation might contain older `KeyView OEM` libraries.

Linux

1. Install Lotus Domino server. You do not need to configure the server.
2. Make sure that the `notes.ini` file is in the `install/lotus/notes/latest/linux` directory, where `install` is the directory where Lotus Notes is installed. If the file does not exist, create an ASCII file named `notes.ini`, and add the following text:

```
[Notes]
```

3. Add the `install/lotus/notes/latest/linux` directory to the `PATH` environment variable:

```
setenv PATH install/lotus/notes/latest/linux:$PATH
```
4. Add the `install/lotus/notes/latest/linux` and the `KeyView bin` directory to the `LD_LIBRARY_PATH` environment variable:

```
setenv LD_LIBRARY_PATH keyview_bin:install/lotus/notes/latest/linux:$LD_LIBRARY_PATH
```

where `keyview_bin` is the location of the `KeyView bin` directory. Micro Focus recommends that you add the `KeyView bin` directory because the Lotus Notes installation might contain older `KeyView OEM` libraries.

Open Secured NSF Files

KeyView enables you to specify a user ID file and password to use to open a secured NSF file for extraction.

Format Note Subfiles

The KeyView NSF reader uses XML templates to format note subfiles. You can customize the templates to approximate the look and feel of the original notes as closely as possible. For more information, see [Extract and Format Lotus Notes Subfiles, on page 289](#).

Extract Subfiles from PDF Files

KeyView can extract document-level and page-level attachments from a PDF document. Document-level attachments are added by using the **Attach A File** tool, and can include links to or from the parent document or to other file attachments. Page-level attachments are added as comments by using various tools. Page-level or comment attachments display the File Attachment icon or the Speaker icon on the page where they are located. KeyView can also extract the files from Portfolio PDFs.

When a PDF's attachments are extracted to disk, the attachments are saved in their native format.

Improve Performance for PDFs with Many Small Images

To improve performance when processing PDF files that contain many small images, you can choose to ignore images unless they exceed a minimum width and/or height. If an image is smaller than the minimum width or height, KeyView does not extract the image.

For example, to ignore images that are less than 16 pixels wide or less than 16 pixels in height, add the following to the [pdf_flags] section of the formats.ini file:

```
[pdf_flags]
process_images_with_min_width=16
process_images_with_min_height=16
```

Extract Embedded OLE Objects

The File Extraction API can extract embedded OLE objects from the following types of documents:

- Lotus Notes (DXL)
- Microsoft Excel
- Microsoft Word
- Microsoft PowerPoint
- Microsoft Outlook
- Microsoft Visio
- Microsoft Project
- OASIS Open Document
- Rich Text Format (RTF)

When an embedded OLE object is extracted from its parent file, the location of the embedded file in the original document is not available. The parent and child are extracted as separate files.

Extract Subfiles from ZIP Files

You can extract ZIP files that are not password-protected by using the general method (see [Extract Subfiles, on page 36](#)). However, some ZIP files use password protection, in which case you must use a different method to enter the required credentials. See [Password Protected Files, on page 339](#) for more information.

Default File Names for Extracted Subfiles

When you do not specify a file name in the call to `fpExtractSubFile()`, in some cases a default file name is applied to the extracted subfile.

Default File Name for Mail Formats

To avoid naming conflicts and problems with long file names, KeyView applies its own names to the extracted mail items when you do not supply a name in the call to `fpExtractSubFile()`. A non-mail attachment retains its original file name and extension.

When the contents of a mail store or the message body of a mail message are extracted, the extracted file names can include the following:

- The first valid eight characters of the original folder name or "Subject" line of the mail message. If the "Subject" line is empty, the characters `kvext` are used, where `ext` is the format's extension. For example, the characters would be `"kvmsg"` for MSG and `"kvnsf"` for NSF.

For notes, the file name is derived from the first 24 characters of the note text. For contact entries, the file name is derived from the full name of the contact.

The following special characters are considered invalid and are ignored:

any non-printing character with a value less than `0x1F`

angle brackets (< >) double quotation marks (")

asterisk (*) forward slash (/)

back slash (\) pipe (|)

colon (:) question mark (?)

- The characters `_kvn`, where `n` is an integer incremented from 0 for each extracted item.
- One of the following extensions:

Type	File Extension
email message	.mail

Type	File Extension
calendar appointment	.cal
contact entry	.cont
task entry	.task
note	.note
journal entry	.jrn1
distribution list	.dist
posting note	.post

- If the type cannot be determined for an MSG or PST file, the file is given a .mail extension.
- If the type cannot be determined for a NSF file, the file is given a .tmp extension.
- The format of a MAIL file is plain text by default, but can be set to RTF with the `KVExtractionFlag_GetFormattedBody` flag.

For example, an MSG mail message with the subject line *RE: Product roadmap* that contains the Microsoft Excel attachment `release_schedule.xls` is extracted as:

```
RE produ_kv0.mail  
release_schedule.xls
```

If an extracted message contains an embedded OLE object or any attachment that does not have a name, the object or attachment is extracted as `_kv#.tmp`.

Default File Name for Embedded OLE Objects

KeyView can apply a default name to an extracted embedded OLE object when you do not supply a name in the call to `fpExtractSubFile()`. When an embedded OLE object is extracted, the extracted file name can include the following:

- The characters `subfile_kvn`, where *n* is an integer incremented from 0 for each extracted object.
- If KeyView can determine the embedded OLE is a Microsoft Office document, the original extension is used. If the file type cannot be determined, the file is given a .tmp extension.

For example, a Microsoft Word document (`sales_quarterly.doc`) might contain two embedded OLE objects: a Microsoft Excel file called `west_region.xls`, and a bitmap created in the Word document. The embedded objects are extracted as `subfile_kv0.xls` and `subfile_kv1.tmp`.

Chapter 4: Use the Filter API

This section describes how to perform some basic filtering tasks by using the Filter API.

• Generate an Error Log	58
• Extract Metadata	61
• Convert Character Sets	63
• Extract Deleted Text Marked by Tracked Changes	66
• Filter PDF Files	66
• Filter Spreadsheet Files	73
• Filter XML Files	77
• Configure Headers and Footers	82
• Filter Hidden Data	83
• Tab Delimited Output for Embedded Tables	85
• Table Detection for PDF Files	85
• Exclude Japanese Guide Text	86
• Source Code Identification	86

Generate an Error Log

You can monitor and debug filtering operations by enabling a detailed error log. This enables you to see errors that are generated at run time, and to track problem files in stream or file mode.

NOTE: Error logs are not generated when in-process filtering is enabled.

The error log might include the following information:

- Generated error codes.
- A time stamp.
- The path and file name of the file in which the error occurred.
- The length of the file in which the error occurred. If the name of the original file or the name of the temporary file are not obtained in stream mode, the file length is reported.

The following is a sample log file:

```
-KV00PE 12 # Time: 11:14:32 # File Len = 68140
-KV00PE 13 # Time: 11:23:05 # H:\files\WP\Word97\fnldmsa.doc
-KV00PE 5 # Time: 12:15:54 # H:\files\SS\XL2000\corporate.xsl
-KV00PE 5 # Time: 12:45:19 # H:\files\WP\WPerf5\wp501.doc
-KV00PE 12 # Time: 14:25:33 # H:\files\PG\PPoint95\95.ppt
-KV00PE 26 # Time: 16:26:04 # File Len = 19117568
-KV00PE 10 # Time: 20:27:40 # File Len = 19117568
```

You can specify the information that is written to the log file by using either the API or environment variables. To configure a log file for a single filtering session, use environment variables. To configure a log file for all filtering sessions, use the API. Configuring the log file by using the API overrides the same settings in the environment variables. You can also specify additional settings in the `formats.ini` file.

You can configure the following features of the log file:

- Enable or disable logging. See [Enable or Disable Error Logging, below](#).
- Change the default path and file name of the log file. See [Change the Path and File Name of the Log File, below](#).
- Include memory errors in the log file. See [Report Memory Errors, on the next page](#).
- Specify a memory guard that is used to generate memory overwrite errors in the log. See [Specify a Memory Guard, on the next page](#).
- Include the input file name in the log file when filtering a stream. See [Report the File Name in Stream Mode, on the next page](#).
- Include extended error codes that provide more detail on a general error (KVERR_Genera1). See [Report Extended Error Codes, on page 61](#).
- Specify the maximum size of the log file. See [Specify the Maximum Size of the Log File, on page 61](#).

Enable or Disable Error Logging

You can enable or disable error logging by using either the API or environment variables. By default, a file called `kloop.log` is created in the system temporary directory; however, you can change the path and file name of this file (see [Change the Path and File Name of the Log File, below](#)).

Use the API

To enable or disable logging, set the final argument (`dwFlags`) of [fplnit\(\)](#), [on page 147](#), [fplnitWithLicenseData\(\)](#), [on page 149](#) or [fpOpenStreamEx2\(\)](#) to either `KVF_OOPLOGON` or `KVF_OOPLOGOFF`.

Use Environment Variables

To enable logging, add the `KV00PLOGON` environment variable, and set the variable value to `1`. To disable logging, do not set the `KV00PLOGON` environment variable.

Change the Path and File Name of the Log File

You can change the default path and file name of the log file. The default is `C:\temp\kloop.log` on Windows and `/tmp/kloop.log` on UNIX.

To change the path and file name of the log file, add the following to the `formats.ini` file:

```
[klooplog]
KloopLogName=filepath
```

Report Memory Errors

You can report memory leaks and memory overwrites in the log file by enabling the memory trace system, either by using the API or environment variables. If the memory trace system is enabled, the extended error codes for memory leaks and memory overwrites (26 and 27, respectively) are reported in the log file when they are generated. The extended error codes are defined in `KVErrorCodeEx` in `kverrorcodes.h`.

NOTE: To report memory overwrites, you must also set a memory guard. See [Specify a Memory Guard, below](#).

Use the API

To enable or disable the memory trace system, set the final argument (`dwFlags`) of `fpInit()`, on [page 147](#), `fpInitWithLicenseData()`, on [page 149](#) or `fpOpenStreamEx2()` to either `KVF_OOPMEMTRACEON` or `KVF_OOPMEMTRACEOFF`.

Use Environment Variables

To enable the memory trace system, add the `KVOOPMT` environment variable, and set its value to `1`. To disable the memory trace system, do not set the `KVOOPMT` environment variable.

Specify a Memory Guard

To report memory overwrites in the log file, you must set a memory guard that protects against memory overwrites. Normally, this is set in the range of 100-200 bytes. For example, if a memory guard of 100 is set and 20 bytes of memory are specified, a total of 120 bytes of memory are allocated. The additional memory is used to monitor and identify memory overwrites.

To configure the memory guard, add the following section to the `formats.ini` file:

```
[Kvooplog]
mg=100
```

Report the File Name in Stream Mode

When you run Filter in file mode, the file name is always reported in the log file. To report the file name in stream mode, you must extract it through the API.

To add the input file name to the log, call the `fpFilterConfig()` function with the following arguments:

Argument	Parameter
<code>nType</code>	<code>KVFLT_SETOOPSRCFILE</code>
<code>nValue</code>	<code>TRUE</code>
<code>pData</code>	<i>input_filename</i>

For example:

```
char    inputfile[250];  
(*fpFilterConfig)(pKVFilter, KVFLT_SET00PSRCFILE, TRUE, input_filename);
```

Report Extended Error Codes

When a general error (KVERR_General) is generated during out-of-process filtering, *extended* error codes can also be generated and reported in the error log. The extended error codes provide more information about the error, and are defined in [KVErrorCodeEx](#) in `kverrorcodes.h`.

To report extended errors, call the function [fpGetKvErrorCodeEx\(\)](#). Extended error codes are generated in the C sample program, `Filter`.

Specify the Maximum Size of the Log File

You can specify the maximum size of the log file. When this size is reached and new entries are logged, either the first entry in the file is overwritten or the new entries are not reported.

To configure the maximum log size and whether old entries are overwritten, add the following section to the `formats.ini` file:

```
[Kvooplog]  
LogFileSize=10  
OverWriteLog=1
```

Option	Description
LogFileSize	This option specifies the maximum size of the log file in KB. The minimum is 1 K. If you do not specify a size, the default of 2 MB is used.
OverWriteLog	This option determines whether the log file is overwritten when the maximum log file size (LogFileSize) is reached. If you set this option to 1 , the first entry in the log file is overwritten. If you set this option to 0 , new entries are not reported in the log file.

Extract Metadata

When a file format supports metadata, KeyView can extract and process that information. Metadata includes document information fields such as title, author, creation date, and file size. Depending on the file's format, metadata is referred to in a number of ways: for example, "summary information," "OLE summary information," "file information," and "document properties."

The metadata in mail formats (MSG and EML) and mail stores (PST, NSF, and MBX) is extracted differently than other formats. For information on extracting metadata from these formats, see [Extract Mail Metadata, on page 40](#).

NOTE: KeyView can only extract metadata from a document if metadata is defined in the document, and if the document reader can extract metadata for the file format. The section [Supported Formats, on page 191](#) lists the file formats for which metadata can be extracted.

KeyView does not generate metadata automatically from the document contents.

The sample program `filter` demonstrates how to extract metadata. See [Sample Programs](#), on [page 88](#).

Extract Metadata for File Filtering

To extract metadata for file filtering

1. Call `fpFilterFile()`.
2. Declare a pointer to the `KVSummaryInfoEx` structure.
3. Call `fpGetOLESummaryInfoFile()` to extract the metadata.
4. Call `fpFreeOLESummaryInfo()` to free the memory allocated for metadata extraction.

Extract Metadata for Stream Filtering

To extract metadata for stream filtering

1. Call `fpOpenStream()` or `fpOpenStreamEx2()` to open a stream.
2. Call `fpFilterStream()` to filter the stream.
3. Call `fpCloseStream()` to close the input stream.
4. Declare a pointer to the `KVSummaryInfoEx` structure.
5. Call `fpGetOLESummaryInfo()` to extract the metadata.
6. Call `fpFreeOLESummaryInfo()` to free the memory allocated for metadata extraction.

Example

Below is an example of a call to `fpGetOLESummaryInfo()`:

```
{
    KVSummaryInfoEx    si;
    memset( &si, 0, sizeof(si) );
    if ( KVERR_Success != (*pInterface->fpGetOLESummaryInfo)( pKVFilter, pInput, &si
) )
    {
        fprintf( fpOut, "Error obtaining summary information\n" );
        return;
    }
    if ( si.nElem == 0 )
    {
        fprintf( fpOut, "No summary information\n" );
        goto end;
    }
}
```

```
    }  
    PrintSummaryInfo(&si, fpOut);  
end:  
    (*pInterface->fpFreeOLESummaryInfo)( pKVFilter, &si );  
}
```

where:

- pKVFilter** A pointer returned from [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149.
- pInput** A pointer to the developer-assigned instance of [KVInputStream](#). The structure [KVInputStream](#) defines the input stream that contains the source.
- si** Points to the structure [KVSummaryInfoEx](#). In the structure, `nElem` provides a count of the number of metadata elements, and `pElem` points to the first element of the array of individual elements, as defined by the structure [KVSumInfoElemEx](#).

To interpret the metadata after `fpGetOLESummaryInfo()` is called and returns a non-zero status:

- If `si.nElem` is zero, the document did not contain metadata. If `si.nElem` is not zero, `si.nElem` is the number of metadata elements contained in the array.
- Each [KVSumInfoElemEx](#) structure contains the following information for each metadata element:

`si.pElem` Specifies whether the data value is present in the document. 1 specifies that the value is valid. For example, if the "Title" element was not populated in the document, `si.pElem[1].isValid == 0` would evaluate to true.

`si.pElem` Specifies the data type of the metadata element. The types are defined in the structure `[n].type` [KVSumInfoType](#) in `kvtypes.h`.

`si.pElem` A pointer to the content of the element.
`[n].data` If `type` is `KV_Int4` or `KV_Bool`, then `data` contains the actual value. Otherwise, `data` is a pointer to the actual value.

`KV_DateTime` and `KV_IEEE8` point to an 8-byte value.

`KV_String` and `KV_Unicode` point to the beginning of the string that contains the text. `KV_Unicode` is replaced with `KV_String` when the UNICODE value has been character mapped to the desired output character set, as specified in the call to `fpInit()`.

`si.pElem` The name of the metadata field.
`[`
`n].pcType`

Convert Character Sets

Filter can convert the character set of a document to an arbitrary character set specified in the API, or to the character set of the operating system on which the output text is viewed. For this conversion to occur, a source character set *must* be identified. The source character set can either be determined by the document reader, or can be set in the API. The section [Supported Formats, on page 191](#) lists file

formats for which character set information can be determined by the document reader. The character sets are enumerated in `KVCharSet` of `kvtypes.h`.

Determine the Character Set of the Output Text

To determine the output character set of a filtered document, Filter considers the following:

- Whether the document reader can determine the character set of the file format. If the document reader cannot determine the character set information for the document type, set the source character set in the API.
- Whether the *source* character set is specified in the API.
- Whether the *target* character set is specified in the API.

Guidelines for Character Set Conversion

Below are some rules for the determination of character set mapping:

- If the source is not determined by the document reader or configured in the API, the character set of the output text is always unknown, regardless of the target character set configuration. The document cannot be converted to a target character set or the operating system's code page unless the source character set is known.
- If the target character set is *not* specified in the API, and the source character set is identified, the operating system's code page is used for the output text.
- If the source character set is identified, and the target character set is specified in the API, the target character set specified in the API is used for the output text.
- For documents that contain multiple character sets, Micro Focus recommends that the target character set be forced to UNICODE or UTF-8.

The following table illustrates how Filter determines the character set of the output text.

Determining the Output Character Set—Example

Source charset read by Filter	Source charset specified in API	Target charset specified in API	Output charset
No	No	No	no conversion
No	KVCS_936	No	OS code page
No	No	UNICODE	no conversion
No	KVCS_936	UNICODE	UNICODE
Yes	No	No	OS code page

Determining the Output Character Set—Example, continued

Source charset read by Filter	Source charset specified in API	Target charset specified in API	Output charset
Yes	KVCS_936	No	OS code page
Yes	No	UNICODE	UNICODE
Yes	KVCS_936	UNICODE	UNICODE

Set the Character Set During Filtering

You can convert the character set of a file at the time the file is filtered.

To specify the source character set of a file, after calling [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149, call [fpSetSrcCharSet\(\)](#), and set the `eCharSet` argument to any value in the enumerated list in `KVCharSet` of `kvtypes.h`.

To determine the final output character set, call the `fpGetTrgCharSet()` function after filtering is complete.

To specify the target character set, set the `outputCharSet` argument of [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149 to any value in the enumerated list in `KVCharSet` of `kvtypes.h`.

Not all values of the enumerated list can be used as a target character set. [Coded Character Sets](#), on page 283 lists character sets that can be used as output.

Set the Character Set During Subfile Extraction

You can convert the character set of a subfile at the time the subfile is extracted from the container and before it is filtered. This is most often used to set the character set of a mail message's body text. See [Filter PDF Files](#), on the next page for more information.

To specify the source character set of a subfile, call the [fpExtractSubFile\(\)](#) function, and set the `KVExtractSubFileArg->srcCharSet` argument to any value in the enumerated list in `KVCharSet` of `kvtypes.h`.

To specify the target character set of a subfile, call the `fpExtractSubFile()` function, and set the `KVExtractSubFileArg->trgCharSet` argument to any value in the enumerated list in `KVCharSet` of `kvtypes.h`.

Prevent the Default Conversion of a Character Set

You can prevent the default conversion of text to the operating system code page, and specify that Filter retain the original character encoding of the document when it is available. Any document identified as containing more than one character encoding is converted to the first encoding encountered in the file.

To prevent the default conversion, set the flag `KVF_NODEFAULTCHARSETCONVERT` as the last argument of the call to `fpInit()`, on page 147 or `fpInitWithLicenseData()`, on page 149. This setting overrides the source or target character set specified in the API.

This setting overrides the source or target character set specified in the API.

Extract Deleted Text Marked by Tracked Changes

The revision tracking feature in applications—such as Microsoft Word's **Track Changes**—marks changes to a document (typically, strikethrough for deleted text and underline for inserted text) and tracks each change by reviewer name and date.

If revision tracking was enabled when text was deleted from a source document, you can configure Filter to extract the deleted text. Filter does not extract the reviewer name and revision date.

To extract deleted text from a document and include it in the filtered output

1. Call the `fpInit()`, on page 147 or `fpInitWithLicenseData()`, on page 149 function.
2. Call the `fpFilterConfig()` function with the following arguments:

Argument	Parameter
nType	KVFLT_INCLREVISIONMARK
nValue	TRUE
pData	NULL

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_INCLREVISIONMARK, TRUE, NULL);
```

3. Call the `fpFilterFile()` or `fpFilterStream()` function.

Filter PDF Files

Filter has special configuration options that allow greater control over the conversion of Adobe Acrobat PDF files.

Filter PDF Files to a Logical Reading Order

The PDF format is primarily designed for presentation and printing of brochures, magazines, forms, reports, and other materials with complex visual designs. Most PDF files do not contain the *logical structure* of the original document—the correct reading order, for example, and the presence and meaning of significant elements such as headers, footers, columns, tables, and so on.

KeyView can filter a PDF file either by using the file's internal unstructured paragraph flow, or by applying a structure to the paragraphs to reproduce the logical reading order of the visual page. Logical

PDF files that contain languages that read from right-to-left (such as Hebrew and Arabic) in the correct reading direction.

NOTE: The algorithm used to reproduce the reading order of a PDF page is based on common page layouts. The paragraph flow generated for PDFs with unique or complex page designs might not emulate the original reading order exactly.

For example, page design elements such as drop caps, callouts that cross column boundaries, and significant changes in font size might disrupt the logical flow of the output text.

By default, KeyView produces an *unstructured* text stream for PDF files. This means that PDF paragraphs are extracted in the order in which they are stored in the file, not the order in which they appear on the visual page. For example, a three-column article could be output with the headers and title at the end of the output file, and the second column extracted before the first column. Although this output does not represent a logical reading order, it accurately reflects the internal structure of the PDF.

You can configure KeyView to produce a *structured* text stream that flows in a specified direction. This means that PDF paragraphs are extracted in the order (logical reading order) and direction (left-to-right or right-to-left) in which they appear on the page.

The following paragraph direction options are available:

Paragraph Direction Option	Description
Left-to-right	Paragraphs flow logically and read from left to right. You should specify this option when most of your documents are in a language that uses a left-to-right reading order, such as English or German.
Right-to-left	Paragraphs flow logically and read from right to left. You should specify this option when most of your documents are in a language that uses a right-to-left reading order, such as Hebrew or Arabic.
Dynamic	Paragraphs flow logically. The PDF filter determines the paragraph direction for each PDF page, and then sets the direction accordingly. Filter uses this option when a paragraph direction is not specified.

NOTE: Filtering might be slower when logical reading order is enabled. For optimal speed, use an unstructured paragraph flow.

The paragraph direction options control the direction of paragraphs on a page; they do not control the text direction in a paragraph. For example, a PDF file might contain English paragraphs in three columns that read from left to right, but 80% of the second paragraph might contain Hebrew characters. If the left-to-right logical reading order is enabled, the paragraphs are ordered logically in the output—title paragraph, then paragraph 1, 2, 3, and so on—and flow from the top left of the first column to the bottom right of the third column. However, the *text* direction of the second paragraph is determined independently of the page by the PDF filter, and is output from right to left.

NOTE: Extraction of metadata is not affected by the paragraph direction setting. The characters and words in metadata fields are extracted in the correct reading direction regardless of whether

logical reading order is enabled.

Enable Logical Reading Order

You can enable logical reading order by using either the API or the `formats.ini` file. Setting the paragraph direction in the API overrides the setting in the `formats.ini` file.

Use the C API

To enable PDF logical reading order in the C API, call the `fpFilterConfig()` function with the following arguments:

Argument	Parameter
nType	KVFLT_LOGICALPDF
nValue	Set to one of the following flags which are defined in <code>kvtypes.h</code> (see LPDF_DIRECTION , on page 188): <ul style="list-style-type: none">• LPDF_LTR. Logical reading order and left-to-right paragraph direction.• LPDF_RTL. Logical reading order and right-to-left paragraph direction.• LPDF_AUTO. Logical reading order. The PDF reader determines the paragraph direction for each PDF page, and then sets the direction accordingly. Filter uses this option when a paragraph direction is not specified.• LPDF_RAW. Unstructured paragraph flow. This is the default behavior. If logical reading order is enabled, and you want to return to an unstructured paragraph flow, set this flag.
pData	NULL

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_LOGICALPDF, LPDF_LTR, NULL);
```

Use the `formats.ini` File

To enable logical reading order by using the `formats.ini` file

1. Change the PDF reader entry in the `[Formats]` section of the `formats.ini` file as follows:

```
[Formats]  
200=1pdf
```

2. Optionally, add the following section to the end of the `formats.ini` file:

```
[pdf_flags]  
pdf_direction=paragraph_direction
```

where *paragraph_direction* is one of the following:

Flag	Description
LPDF_ LTR	Left-to-right paragraph direction.
LPDF_ RTL	Right-to-left paragraph direction.
LPDF_ AUTO	The PDF reader determines the paragraph direction for each PDF page, and then sets the direction accordingly. Filter uses this option when a paragraph direction is not specified.
LPDF_ RAW	Unstructured paragraph flow. This is the default behavior. If logical reading order is enabled, and you want to return to an unstructured paragraph flow, set this flag.

Rotated Text

When a PDF that contains rotated text is filtered, the rotated text is extracted after the text at the end of the PDF page on which the rotated text appears. If the PDF is filtered with logical order enabled, and the amount of rotated text on a page surpasses a predefined threshold, the page is automatically output as an unstructured text stream. You cannot configure this threshold.

Extract Custom Metadata from PDF Files

You can extract custom metadata from PDF files either by specifying individual metadata tag names, or by extracting all custom metadata at once.

Extract Custom Metadata by Tag

To extract custom metadata by metadata tag, add the custom metadata names to the `pdfsr.ini` file provided, and copy the modified file to the `bin` directory. You can then extract metadata as you normally would.

The `pdfsr.ini` is in the directory `samples\pdfini`, and has the following structure:

```
<META>  
<TOTAL>total_item_number</TOTAL>,  
/metadata_tag_name datatype,  
</META>
```

Parameter	Description
<i>total_item_number</i>	The total number of metadata tags that are listed.
<i>metadata_tag_name</i>	The metadata tag name used in the PDF files.
<i>datatype</i>	The data type of the metadata field. Data types are defined in KVSumInfoType .

For example:

```
<META>  
<TOTAL>4</TOTAL>  
/part_number      INT4  
/volume           INT4  
/purchase_date    DATETIME  
/customer         STRING  
</META>
```

Extract All Custom Metadata

You can extract all metadata through the API.

To extract all metadata by using the API

1. Call the [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149 function.
2. Call the [fpFilterConfig\(\)](#) function with the following arguments:

Argument	Parameter
nType	KVFLT_EXPORTALLMETADATA
nValue	TRUE
pData	NULL

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_EXPORTALLMETADATA, TRUE, NULL);
```

3. Call the [fpGetOLESummaryInfo\(\)](#) or [fpGetOLESummaryInfoFile\(\)](#) function.

Filter Tagged PDF Content

A tagged PDF contains an additional layer of text for visually impaired readers. This text is used in text-to-speech features in various PDF viewing programs. You can enable filtering of tagged PDF text in the API.

Filtering the extra layer of tagged content might result in duplicate text in the output. This is the expected behavior.

To filter tagged PDF content

1. Call the [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149 function.
2. Call the [fpFilterConfig\(\)](#) function with the following arguments:

Argument	Parameter
nType	KVFLT_EXPORTTAGGEDCONTENT
nValue	TRUE
pData	NULL

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_EXPORTTAGGEDCONTENT, TRUE, NULL);
```

Skip Embedded Fonts

Text in PDF files sometimes contains embedded fonts. If you experience difficulties filtering embedded fonts, there are options in the API, the `formats.ini` file, and the filter sample program that enable you to skip this type of text.

NOTE: If you skip embedded fonts, none of the content that contains embedded fonts is included in the output.

Use the `formats.ini` File

When you use `formats.ini` to skip embedded fonts, you can also specify an *embedded font threshold*, which is an arbitrary percentage probability that the glyph in the embedded text maps to a character value in the output character set (ASCII, UTF-8, and so on).

For example, if you specify a threshold of **75**, embedded text glyphs that have a 75% or greater probability of correctly matching the character in the output character set are included in the output; glyphs that have a probability of less than 75% of matching the output character set are omitted from the output.

To skip embedded fonts by using the `formats.ini` file

- Set the following parameters:

```
[pdf_flags]  
skipembeddedfont=TRUE  
embedded_font_threshold=threshold
```

where *threshold* is a value between 0 and 100. A threshold of 100 skips all embedded font text; a threshold of 0 retains all embedded font text. Set `skipembeddedfont` to **TRUE** to enable the `embedded_font_threshold` parameter.

The default value of `embedded_font_threshold` is 100. If you set `skipembeddedfont` to **TRUE** and do not specify the `embedded_font_threshold` parameter, Filter skips all embedded text.

Use the C API

To skip embedded fonts by using the C API, call the [fpFilterConfig\(\)](#) function with the following arguments:

Argument	Parameter
nType	KVFLT_SKIPEMBEDDEDFONT
nValue	TRUE
pData	NULL

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_SKIPEMBEDDEDFONT, TRUE, NULL);
```

Control Hyphenation

There are two types of hyphens in a PDF document:

- A *soft hyphen* is added to a word by a word processor to divide the word across two lines. This is a discretionary hyphen and is used to ensure proper text flow in justified text.
- A *hard hyphen* is intentionally added to a word regardless of the word's position in the text flow. It is required by the rules of grammar or word usage. For example, compound words (such as *three-week vacation* and *self-confident*) contain hard hyphens.

By default, KeyView skips the source document's soft hyphens in the Filter output to provide more searchable text content. However, if you want to maintain the document layout, you can keep soft hyphens in the Filter output. To keep soft hyphens, you must enable the soft hyphen flag in `formats.ini` or in the API.

Use the formats.ini File

To keep soft hyphens by using the `formats.ini` file, set the following parameter:

```
[pdf_flags]  
keepsofthyphen=TRUE
```

Use the C API

To keep soft hyphens by using the C API, call the [fpFilterConfig\(\)](#) function with the following arguments:

Argument	Parameter
nType	KVFLT_KEEPSOFTYPHEN

Argument	Parameter
nValue	TRUE
pData	NULL

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_KEEPSOFTHYPHEN, TRUE, NULL);
```

Filter Spreadsheet Files

Filter has special configuration options that enable greater control over the conversion of spreadsheet files.

Filter Worksheet Names

Normally, Filter does not extract worksheet names from a spreadsheet because it is assumed that the text should not be exposed. To extract worksheet names, add the following lines to the `formats.ini` file:

```
[Options]  
getsheetnames=1
```

Filter Hidden Text in Microsoft Excel Files

Normally, Filter does not filter hidden text from a Microsoft Excel spreadsheet because it is assumed that the text should not be exposed. To extract text from hidden rows, columns, and sheets from Excel spreadsheets, add the following lines to the `formats.ini` file:

```
[Options]  
gethiddeninfo=1
```

NOTE: You can also set an API flag to filter text from hidden sheets. See [Hidden Data in Microsoft Excel Documents](#), on page 83 for more information.

Specify Date and Time Format on UNIX Systems

In Microsoft Excel you can choose to format dates and times according to the system locale. On Windows, KeyView uses the system locale settings to determine how these dates and times should be formatted. In other operating systems, KeyView uses the U.S. short date format (*mm/dd/yyyy*). You can change this by specifying the formats you wish to use in the `formats.ini` file.

To specify the system date and time format on UNIX systems

- In the `formats.ini` file, specify the following options:
 - `SysDateTime`. The format to use when a cell is formatted using the system format including both the date and the time.
 - `SysLongDate`. The format to use when a cell is formatted using the system long date format.
 - `SysShortDate`. The format to use when a cell is formatted using the system short date format.
 - `SysTime`. The format to use when a cell is formatted using the system time format.

NOTE: These values cannot contain spaces.

For example, if you specify `SysDateTime=%d/%m/%Y`, dates and times are extracted in the following format:

`28/02/2008`

The format arguments are the same as those for the `strftime()` function.

See <http://linux.die.net/man/3/strftime> for more information.

Filter Very Large Numbers in Spreadsheet Cells to Precision Numbers

By default, numbers are extracted in the format specified by the Excel file (for example, *General*, *Currency* and *Date*). Spreadsheets might contain cells that have very large numbers in them. Excel displays the numbers in a scientific notation that rounds or truncates the numbers.

To extract numbers without formatting, add the following options in the `formats.ini` file:

```
[Options]
ignoredefnumformats=1
```

Extract Microsoft Excel Formulas

Normally, the actual value of a formula is extracted from an Excel spreadsheet; the formula from which the value is derived is not included in the output. However, `KeyView` enables you to include the value as well as the formula in the output. For example, if `Filter` is configured to extract the formula and the formula value, the output might look like this:

```
245 = SUM(B21:B26)
```

The calculated value from the cell is 245 and the formula from which the value is derived is `SUM(B21:B26)`.

NOTE: Depending on the complexity of the formulas, enabling formula extraction might result in slightly slower performance.

To set the extraction option for formulas

- Add the following lines to the `formats.ini` file:

```
[Options]
getformulastring=option
```

where *option* is one of the following:

Option	Description
0	Extract the formula value only. This is the default. If formula extraction is enabled, and you want to return to the default, set this option.
1	Extract the formula only.
2	Extract the formula and the formula value.

NOTE: You can also set an API flag to filter formulas and formula values. See [Hidden Data in Microsoft Excel Documents, on page 83](#) for more information.

If a function in a formula is not supported or is invalid, and option 1 or 2 is specified, only the calculated value is extracted. See [Supported Microsoft Excel Functions, below](#) for a list of supported functions.

When formula extraction is enabled, Filter can extract Microsoft Excel formulas that contain the functions listed in the following table.

Supported Microsoft Excel Functions

=ABS()	=ACOS()	=AND()	=AREAS()
=ASIN()	=ATAN2()	=ATAN2()	=AVERAGE()
=CELL()	=CHAR()	=CHOOSE()	=CLEAN()
=CODE()	=COLUMN()	=COLUMNS()	=CONCATENATE()
=COS()	=COUNT()	=COUNTA()	=DATE()
=DATEVALUE()	=DAVERAGE()	=DAY()	=DCOUNT()
=DDB()	=DMAX()	=DMIN()	=DOLLAR()
=DSTDEV()	=DSUM()	=DVAR()	=EXACT()
=EXP()	=FACT()	=FALSE()	=FIND()
=FIXED()	=FV()	=GROWTH()	=HLOOKUP()
=HOUR()	=ISBLANK()	=IF()	=INDEX()
=INDIRECT()	=INT()	=IPMT()	=IRR()

=ISERR()	=ISERROR()	=ISNA()	=ISNUMBER()
=ISREF()	=ISTEXT()	=LEFT()	=LEN()
=LINEST()	=LN()	=LOG()	=LOG10()
=LOGEST()	=LOOKUP()	=LOWER()	=MATCH()
=MAX()	=MDTERM()	=MID()	=MIN()
=MINUTE()	=MINVERSE()	=MIRR()	=MMULT()
=MOD()	=MONTH()	=N()	=NA()
=NOT()	=NOW()	=NPER()	=NPV()
=OFFSET()	=OR()	=PI()	=PMT()
=PPMT()	=PRODUCT()	=PROPER()	=PV()
=RATE()	=REPLACE()	=REPT()	=RIGHT()
=ROUND()	=ROUND()	=ROW()	=ROWS()
=SEARCH()	=SECOND()	=SIGN()	=SIN()
=SLN()	=SQRT()	=STDEV()	=SUBSTITUTE()
=SUM()	=SYD()	=T()	=TAN()
=TEXT()	=TIME()	=TIMEVALUE()	=TODAY()
=TRANSPOSE()	=TREND()	=TRIM()	=TRUE()
=TYPE()	=UPPER()	=VALUE()	=VAR()
=VLOOKUP()	=WEEKDAY()	=YEAR()	

Standardize Cell Formats

This options enables the standardization of cell formats within Microsoft Excel files. KeyView formats any cell where a number has been entered according to the following rules.

Numbers

These include:

- rounded numbers
- exponentials
- fractions
- percentages

Numbers are printed to the maximum length entered—that is, the full number put into the cell, without any rounding. Negative numbers are printed with a dash in front of them (as opposed to, for example, bracket form).

Text

All text that is part of the format string is stripped, including currency symbols.

Dates

All dates are printed in full ISO-8601 format (that is YYYY-MM-DDTHH:MM:SS). There are two exceptions to this rule:

- Cases where the date format contains a time delta (that is, "[h]", "[m]", or "[s]"). In this case, the time is displayed as an interval, which is the number of days (where a day is defined as a period of 24 hours). The time is printed in the ISO-8601 time interval form, for example P1.234D.
- Cases where the absolute value of the cell is less than 1.0, and the date format contains only time components. In Excel, values between 0.0 and 1.0 correspond to the fictional date 1900-01-00, and are used to express times without an associated date. For example:

Value	Date format	KeyView output
0.5	hh:mm:ss	12:00:00
0.5	dd hh	1900-01-00 12:00:00
1.5	hh:mm:ss	1900-01-01 12:00:00
1.5	dd hh	1900-01-01 12:00:00

You can enable this option by adding the following to the `formats.ini` file:

```
[Options]  
StandardizeCellFormats=TRUE
```

Alternatively, you can enable this option programmatically by passing `KVFLT_STANDARDIZECELLFORMATS` to `fpFilterConfig`.

Filter XML Files

Filter enables you to extract all or selected content from source XML files. You can specify the elements and attributes to extract from a document by using either the API or an INI file (see [Configure Element Extraction for XML Documents, on the next page](#)). Filter detects the following XML formats:

- generic XML
- Microsoft Office 2003 XML (Word, Excel, and Visio)
- StarOffice/OpenOffice XML (text document, presentation, and spreadsheet)

See [File Format Detection, on page 302](#) for more information on format detection.

Configure Element Extraction for XML Documents

When filtering XML files, you can specify which elements and attributes are extracted according to the file's format ID or *root element*. This is useful when you want to extract only relevant text elements, such as abstracts from reports, or a list of authors from an anthology.

A root element is an element in which all other elements are contained. In the following XML sample, book is the root element:

```
<book>
  <title>XML Introduction</title>
  <product id="33-657" status="draft">XML Tutorial</product>
  <chapter>Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```

For example, you could specify that when filtering files with the root element book, the element title is extracted as metadata, and only product elements with a status attribute value of draft are extracted. When you extract an element, the child elements within the element are also extracted. For example, if you extract the element chapter from the previous sample, the child element para is also extracted.

Filter defines default element extraction settings for the following XML formats:

- generic XML
- Microsoft Office 2003 XML (Word, Excel, and Visio)
- StarOffice/OpenOffice XML (text document, presentation, and spreadsheet)

These settings are defined internally and are used when filtering these file formats; however, you can modify their values.

In addition to the default extraction settings, you can also add custom settings for your own XML document types. If you do not define custom settings for your own XML document types, the settings for the generic XML are used.

Modify Element Extraction Settings

You can modify configuration settings for XML documents through either the API or the `kvxconfig.ini` file.

Use the C API

You can use the C API to modify the settings for the standard XML document types or add configuration settings for your own XML document types.

To modify settings

1. Call the [fpInit\(\)](#), on page 147 or [fpInitWithLicenseData\(\)](#), on page 149 function.
2. Define the [KVXConfigInfo](#) structure.
3. Call the [fpFilterConfig\(\)](#) function with the following arguments:

Argument	Parameter
nType	KVFLT_SETXMLCONFIGINFO
nValue	0
pData	the address of the KVXConfigInfo structure

For example:

```
KVXConfigInfo xinfo; /* populate xinfo */  
  
(*fpFilterConfig)(pKVFilter, KVFLT_SETXMLCONFIGINFO, 0, &xinfo);
```

4. Repeat step 2 and step 3 until the settings for all the XML document types that you want to customize are defined.
5. Call the [fpFilterFile\(\)](#) function.

Use an Initialization File

You can use the initialization file to modify the settings for the standard XML document types or add configuration settings for your own XML document types.

To modify settings

1. Modify the `kvxconfig.ini` file.
2. Use the initialization file when processing the XML file. See [Modify Element Extraction Settings in the kvxconfig.ini File](#), below.

The C sample program (`filter`) demonstrates how to use the initialization file in the filtering process. See [Sample Programs](#), on page 88.

Modify Element Extraction Settings in the kvxconfig.ini File

The `kvxconfig.ini` file contains default element extraction settings for supported XML formats. The file is in the directory `install\OS\bin`, where `install` is the path name of the Filter installation directory and `OS` is the name of the operating system. For example, the following entry defines extraction settings for the Microsoft Visio 2003 XML format:

```
[config3]  
eKVFormat=MS_Visio_XML_Fmt  
szRoot=  
szInMetaElement=DocumentProperties  
szExMetaElement=PreviewPicture
```

```
szInContentElement=Text
szExContentElement=
szInAttribute=
```

The following options are available:

Configuration Option	Description
eKVFormat	<p>The format ID as detected by the KeyView detection module. This determines the file type to which these extraction settings apply. See File Format Detection, on page 302 for more information on format ID values.</p> <p>If you are adding configuration settings for a custom XML document type, this option is not defined.</p>
szRoot	<p>The file's root element. When the format ID is not defined, the root element is used to determine the file type to which these settings apply.</p> <p>To further qualify the element, specify its namespace. See Specify an Element's Namespace and Attribute, on the next page.</p>
szInMetaElement	<p>The elements extracted from the file as metadata. All other elements are extracted as text.</p> <p>Separate multiple entries with commas. To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on the next page.</p>
szExMetaElement	<p>The child elements in the included metadata elements that are not extracted from the file as metadata. For example, the default extraction settings for the Visio XML format extract the DocumentProperties element as metadata. This element includes child elements such as Title, Subject, Author, Description, and so on. However, the child element PreviewPicture is defined in szExMetaElement because it is binary data and should not be extracted.</p> <p>You cannot exclude any metadata elements from the output for StarOffice files. All metadata is extracted regardless of this setting.</p> <p>Separate multiple entries with commas. To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on the next page.</p>
szInContentElement	<p>The elements extracted from the file as content text. Enter an asterisk (*) to extract all elements including child elements.</p> <p>Separate multiple entries with commas. To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on the next page.</p>
szExContentElement	<p>The child elements in the included content elements that are not extracted from the file as content text.</p>

Configuration Option	Description
	Separate multiple entries with commas. To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute , below.
szInAttribute	<p>The attribute values extracted from the file. If attributes are not defined here, attribute values are not extracted.</p> <p>Enter the namespace (if used), element name, and attribute name in the following format:</p> <p><i>namespace:elementname@attributename</i></p> <p>For example:</p> <p>keyview:division@name</p> <p>Separate multiple entries with commas.</p>

Specify an Element's Namespace and Attribute

To further qualify an element, you can specify that the element must exist in a certain namespace, must contain a specific attribute, or both. To define the namespace *and* attribute of an element, enter the following:

```
ns_prefix:elementname@attribname=attribvalue
```

NOTE: Attribute values that contain spaces must be enclosed in quotation marks.

For example, the entry `bg:language@id=xml` extracts a `language` element in the namespace `bg` that contains the attribute name `id` with the value of `"xml"`. This entry extracts the following element from an XML file:

```
<bg:language id="xml">XML is a simple, flexible text format derived from  
SGML</bg:language>
```

but does not extract:

```
<bg:language id="sgml">SGML is a system for defining markup  
languages.</bg:language>
```

or

```
<adv:language id="xml">The namespace should be a Uniform Resource Identifier  
(URI).</adv:language>
```

Add Configuration Settings for Custom XML Document Types

You can define element extraction settings for custom XML document types by adding the settings to the `kvxconfig.ini` file. For example, for files that contain the root element `keyviewxml`, you could add the following section to the end of the initialization file:

```
[config101]
eKVFormat=
szRoot=keyviewxml
szInMetaElement=dc:title,dc:meta@title,dc:meta@name=title
szExMetaElement=

szInContentElement=keyview:division@name=dev,keyview:division@name=export,p@style="
Heading 1"
szExContentElement=
szInAttribute=keyview:division@name
```

The custom extraction settings must be preceded by a section heading named [config*N*], where *N* is an integer starting at 100 and increasing by 1 for each additional file type, for example [config100], [config101], [config102], and so on. The default extraction settings for the supported XML formats are numbered config0 to config99. Currently only 0 to 6 are used.

Because a custom XML document type is not recognized by the KeyView detection module, the format ID is not defined. The file type is identified by the file's root element only.

If a custom XML document type is not defined in the kvxconfig.ini file or by the fpFilterConfig() function, the default extraction settings for a generic XML document are used.

Configure Headers and Footers

You can configure custom header and footer tags for word processing and spreadsheet documents by editing the formats.ini file.

To configure headers and footers

1. Open the formats.ini file.
2. In the [Options] section, add the following items:

```
header_start_tag=HeaderStart
header_end_tag=HeaderEnd
footer_start_tag=FooterStart
footer_end_tag=FooterEnd
```

For example:

```
header_start_tag=<myHeaderTag>
header_end_tag=</myHeaderTag>
footer_start_tag=<myFooterTag>
footer_end_tag=</myFooterTag>
```

NOTE: You must encode custom tags in UTF-8.

Filter Hidden Data

Some documents contain hidden information, which is not filtered by default. Depending on the type of hidden data that you want to filter and the type of document that you are filtering, you can either use the API or set parameters in the `formats.ini` file.

Hidden Data in Microsoft Excel Documents

There are several types of hidden data in Microsoft Excel documents, each of which has a corresponding flag in the `KV_CONFIG_Arg` structure, which you can toggle to determine whether the hidden data is shown.

The following table lists each data type, its default behavior, and its corresponding configuration API flag.

Hidden data settings

Hidden Data Type	Default Behavior	KV_CONFIG_Arg flag
Hidden sheets	Not output	KV_SS_SHOWHIDDENINFOR
Formulas	Calculated value	KV_SS_SHOWFORMULA
Values and formulas	Calculated value	KV_SS_SHOWVALUESANDFORMULAS

To toggle the display of any type of hidden data

1. Define the configurable argument variable to use in the `KV_CONFIG_Arg` structure. For example:

```
KV_CONFIG_Arg setArg = {0}
```

2. Set the `KV_ALL_OVERWRITECONFIGFILE` flag to overwrite the configuration file settings. For example:

```
setArg.keyID = KV_ALLFLAGS;  
setArg.keyType = KV_INT32ARG;  
setArg.keyData.intArg = KV_ALL_OVERWRITECONFIGFILE;
```

NOTE: To re-enable configuration file settings later, set `!KV_ALL_OVERWRITECONFIGFILE`.

3. Assign values to the members of the variable. For example:

```
setArg.keyID = KV_SSFLAGS;  
setArg.keyType = KV_INT32ARG;  
setArg.keyData.intArg = KV_SS_SHOWHIDDENINFOR;
```

4. Call `fpFilterConfig()` with the following arguments to set the variable:

Argument	Parameter
nType	KVFLT_SetConfigurableArguments
nValue	TRUE
pData	The variable defined in step 1.

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_SetConfigurableArguments, TRUE, &setArg)
```

Example

The following example overwrites the configuration file settings and enables filtering of formulas.

```
KV_CONFIG_Arg setArg = {0};
```

```
setArg.keyID = KV_ALLFLAGS;  
setArg.keyType = KV_INT32ARG;  
setArg.keyData.intArg = KV_ALL_OVERWRITECONFIGFILE;
```

```
fpKV_FilterConfig(pFilter, KVFLT_SetConfigurableArguments, TRUE, &setArg);
```

```
setArg.keyID = KV_SSFLAGS;  
setArg.keyType = KV_INT32ARG;  
setArg.keyData.intArg = KV_SS_SHOWFORMULAS;
```

```
fpKV_FilterConfig(pFilter, KVFLT_SetConfigurableArguments, TRUE, &setArg);
```

Toggle Hidden Excel Data Settings in the formats.ini File

You can control Microsoft Excel hidden data settings through parameters in the `formats.ini` file.

To toggle hidden Excel data settings in the formats.ini file

1. Open the `formats.ini` file in a text editor.
2. Under `[Options]`, set one or both of the following parameters.
 - To filter text from hidden sheets, set `gethiddeninfo` to **1**. See [Filter Hidden Text in Microsoft Excel Files, on page 73](#) for more information.
 - To filter formulas and formula values, set `getformulastring` to the appropriate value. See [Extract Microsoft Excel Formulas, on page 74](#) for more information.

Hidden Data in HTML Documents

KeyView can filter comments from HTML documents. To enable comment filtering, you must set a flag in the `formats.ini` file.

To enable filtering of comments from HTML files

1. Open the `formats.ini` file in a text editor.
2. Under `[Options]`, set the following flag.

```
GetHTMLHiddenInfo=1
```

Tab Delimited Output for Embedded Tables

You can use `KeyView` to convert embedded tables in Word Processing documents (for example, Microsoft Word) to tab-delimited form, by specifying the following option in the `formats.ini` file:

```
[Options]  
TabDelimited=TRUE
```

This option inserts a tab character between each cell, and a line break between each row. Tab and line break characters in the cells are replaced with spaces.

Table Detection for PDF Files

PDF files often contain data presented in a tabular form. However, there is no information about the table stored within the PDF itself – the text is simply placed in an arrangement that looks like a table to the human eye. When this data is filtered, it can be very difficult to reconstruct the table.

If table detection is enabled, `KeyView` attempts to recognize tables within PDF pages, and to reconstruct them before they are output. For each page of the document, `KeyView` outputs the contents of each table first, and then outputs all remaining text on the page.

Micro Focus recommends that tab delimited output is also enabled when using table detection. This means that any tables detected appear in the output text in tab delimited format.

To enable table detection and tab delimited output, specify the following in the `formats.ini` file:

```
[Options]  
TableDetection=TRUE  
TabDelimited=TRUE
```

Alternatively, you can enable these options programmatically by setting `KVFLT_TABLEDETECTION` and `KVFLT_TABDELIMITED` to `true` in `fpFilterConfig()`.

NOTE: Table detection is only available with the `pdf2sr` reader. To enable this reader, set the following configuration parameter:

```
[Formats]  
200=pdf2
```

Exclude Japanese Guide Text

This option prevents output of Japanese phonetic guide text when Microsoft Excel (.xlsx) files are processed.

To prevent output of Japanese phonetic guide text

- Set `NoPhoneticGuides` to `TRUE` in the `formats.ini` file:

```
[Options]
NoPhoneticGuides=TRUE
```

You can also enable this option programatically when filtering by passing `KVFLT_NOPHONETICGUIDES` to `fpFilterConfig`.

Source Code Identification

When KeyView auto-detects a file that contains source code, it can attempt to identify the programming language that it is written in.

NOTE: Source code identification is a new, experimental feature in KeyView 12.0. It is available only on Windows 64-bit, Linux 64-bit, and macOS 64-bit platforms.

You can set source code identification to different levels.

Option	Description
<code>KVSOURCECODE_OFF</code>	Do not enable source code identification.
<code>KVSOURCECODE_ENABLED</code>	Enable source code identification for the most common source code formats.
<code>KVSOURCECODE_EXTENDED</code>	Enable source code identification for all supported source code formats. This option might lead to false positives in some cases (for example, a C++ file might get identified as a rarer format).

For the complete list of source code formats supported for both options, see [Detected Formats, on page 223](#).

You can enable source code identification by setting the appropriate level in the `formats.ini` file. For example:

```
[Options]
SourceCodeDetection=KVSOURCECODE_ENABLED
```

You can also enable this option by passing `KVFLT_SOURCECODEIDENTIFICATION` to the `fpFilterConfig()` function. For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_SOURCECODEDETECTION, KVSOURCECODE_ENABLED,  
NULL);
```

Setting the option through `fpFilterConfig` overrides any settings in `formats.ini`.

For more information, see [fpFilterConfig\(\)](#), on page 129.

Chapter 5: Sample Programs

This section describes the sample programs provided with Filter SDK.

- [Introduction](#)88
- [tstxtract](#)88
- [filter](#)90

Introduction

The C sample programs demonstrate how to use the C implementation of the Filter API. The sample code is intended to provide a starting point for your own applications or to be used for reference purposes.

The following C sample programs are provided:

- [tstxtract](#)
- [filter](#)

The source code and makefile (*program_name_platform.mak*) for the programs are in the directory *install*\KeyviewFilterSDK\samples*program_name*, where *install* is the path name of the Filter installation directory, and *program_name* is the name of the sample program.

The executable for the programs is in the directory *install*\KeyviewFilterSDK*OS*\bin, where *OS* is the name of the operating system.

To compile the sample programs, use the makefile provided for the appropriate platform. Make sure that the Filter `include` directory is specified in the include path of the project. After the executable is compiled and built, you must place it in the same directory as the Filter libraries.

tstxtract

The `tstxtract` sample program demonstrates the File Extraction API. It opens a file, extracts subfiles from the file, and repeats the extraction process until all subfiles are extracted. It also demonstrates how to extract the default set of metadata and pass integer or string names to extract specific metadata. After the files are extracted, you can filter the files by using the `filter` sample program. The `filter` sample program demonstrates the functionality of the Filter API.

The source code for the `tstxtract` sample program is the same for the Filter and Export SDKs. A flag in the makefile specifies whether the program is compiled for Filter, HTML Export, or XML Export.

To run `tstxtract`, type the following at the command line:

```
tstxtract [options] input_file output_directory bin_directory
```

where:

- *options* is one or more of the following:

Option	Description
-c charset	Specify the target character set, for example KVCS_SJIS. See Coded Character Sets, on page 283 for a full list of supported character sets.
-cf keyfile1, keyfile2,...	Specify one or more credential files (private keys) to use to decrypt encrypted .EML, .MBX, .PST, or .MSG files.
-l logfile	Specify the path and file name of the log file in which metadata is written.
-lm	Retrieve metadata and write the data to the log file.
-lms metaname1, metaname2,...	Retrieve metadata with string metanames and write the data to the log file for .MSG, .EML, .MBX, and .NSF files.
-lmi metaint1, metaint2,...	Retrieve metadata with integer (hexadecimal) metanames and write the data to the log file for .PST files.
-lma	Retrieve all metadata from an .NSF file and write the data to the log file.
-to <value in seconds>	Specify the timeout value in seconds. This timeout allows for large files that take longer than the default 7 minute timeout.
-i	Run the file extraction in-process.
-r	<p>Recursively extract second-level subfiles to the specified output directory. For example, if a .ZIP file contains a Microsoft Word file and the Word file contains an embedded Microsoft Excel file, set the -r option to extract both the Word and Excel files.</p> <p>If this option is not set, only first-level subfiles are extracted. In this case, only the Word file would be extracted.</p>
-msg	Extract mail messages in a .PST file as an .MSG file, including all of its attachments. If this flag is not set, the mail message is extracted as text. This applies to PST files on Windows only.
-f	Extract the formatted version of the message body (HTML or RTF) from mail files when possible. If neither an HTML nor RTF version of the message body exists in the mail file, it is extracted as plain text. If you do not set this flag, the message body is extracted as plain text when possible.
-e	Run the file extraction in stream mode.
-p password1, password2,...	Specify one or more passwords to open the input or credential file or files.
-t	Preserve the timestamp of embedded files when possible.
-h	Extract hidden text.

- *input_file* is the full path and file name of the source document.
- *output_directory* is the directory to which the files are extracted.
- *bin_directory* is the path to the Filter bin directory. This is required if you do **not** run the program from the *install\Filter SDK\bin* directory.

filter

The *filter* sample program demonstrates the advanced functionality of the Filter API. It is composed of the following files:

- *filter.c*—command line interface
- *filtersupport.c*—contains core functionality, such as file filtering, stream filtering, metadata extraction, and format detection.
- *filtersupport.h*—structure and variable definitions

To run *filter*, type the following at the command line:

```
filter [options] input_file output_file
```

where:

options is one or more of the options listed in [Options for the Filter Sample Program](#), below.

input_file is the full path and file name of the source document.

output_file is the full path and file name of the output file.

Options for the Filter Sample Program

Option	Description
-i	Extract metadata. See Extract Metadata, on page 61 .
-c	Run Filter in the same process as the calling application (in process). See Run Filter In Process, on page 29 .
-e	Run Filter in stream mode. See Filtering in Stream Mode, on page 26 .
-h	Extract headers and footers, as well as the body text. See fpInit(), on page 147 or fpInitWithLicenseData(), on page 149 .
-d	Extract the file format information using the fpGetDocInfoFile() function.
-mt	Enable the memory trace system in error logs. The memory trace system reports memory leaks and memory overwrites in the log file. See Report Memory Errors, on page 60 . Error logs are not generated when in-process filtering is enabled.
-mtN	Disable the memory trace system in error logs. The memory trace system reports memory leaks and memory overwrites in the log file. See Report Memory Errors, on page 60 . Error logs are not generated when in-process filtering is enabled.

Options for the Filter Sample Program , continued

Option	Description
-L	Enable error logging. See Enable or Disable Error Logging, on page 59 . Error logs are not generated when in-process filtering is enabled.
-LN	Disable error logging. See Enable or Disable Error Logging, on page 59 . Error logs are not generated when in-process filtering is enabled.
-AF	Include the input file name in an error log. See Report the File Name in Stream Mode, on page 60 .
-r	Filter a container file and the subfiles in the container file to a single output file. This option uses the Container API, which is obsolete.
-rm	If you set this option, text that was deleted from a document with revision tracking enabled is extracted from the document and included in the filtered output. See Extract Deleted Text Marked by Tracked Changes, on page 66 .
-x <i>xmlconfigfile</i>	Filter an XML file by using customized extraction settings defined in the <i>kvxconfig.ini</i> file. If you do not enter the full path to the INI file, the program looks for the file in the current working directory. See Filter XML Files, on page 77 for more information.
-z <i>tempdirectory</i>	Specify a temporary directory where temporary files generated by the filtering process are stored. The default is the current working directory. On Windows systems, there is a 64 K size limit to the temporary directory. When the limit is reached, you must either create a new directory or delete the contents of the existing directory; otherwise, you might receive an error message.
-ps <i>password</i>	Specify a password to open a password-protected PST file. This option uses the Container API, which is obsolete.
-pdfauto	Specify that PDF files are output in a logical reading order. The PDF filter determines the paragraph direction (left-to-right or right-to-left) for each PDF page, and then sets the direction accordingly. See Filter PDF Files, on page 66 .
-pdfltr	Specify that PDF files are output in a logical reading order, and that the paragraph direction is left to right. See Filter PDF Files, on page 66 .
-pdfrtl	Specify that PDF files are output in a logical reading order, and that the paragraph direction is right to left. See Filter PDF Files, on page 66 .
-pdfraw	Specify that PDF files are output in an unstructured paragraph flow. This is the default option . If logical reading order is enabled, and you want to return to an unstructured paragraph flow, set this flag. See Filter PDF Files, on page 66 .
-xmp	Parse and return XMP metadata as path and value pairs, and include the original XMP packet. See fpGetXmplInfoFile(), on page 146 and fpGetXmplInfo(), on page 145 .

Options for the Filter Sample Program , continued

Option	Description
-xmpr	Return XMP metadata as a raw XMP packet. See fpGetXmpInfoFile() , on page 146 and fpGetXmpInfo() , on page 145.
-embeddedfont	If you use this option, text that contains embedded fonts is not filtered from PDF documents. See fpFilterConfig() , on page 129.

Part III: C API Reference

This section provides detailed reference information for the C-language implementation of the File Extraction and Filter APIs.

Chapter 6: File Extraction API Functions

This section describes the functions in the File Extraction API. The File Extraction functions open a container file, and extract the container's subfiles so that the subfiles are exposed and available for filtering. Subfiles can be files within a Zip archive, messages in a mail store, attachments in a mail message, or OLE objects embedded in a compound document.

Each function appears as a function prototype followed by a description of its arguments, its return value, and a discussion of its use.

• KVGetExtractInterface()	94
• fpCloseFile()	95
• fpExtractSubFile()	96
• fpFreeStruct()	97
• fpGetMainFileInfo()	98
• fpGetSubFileInfo()	99
• fpGetSubFileMetaData()	100
• fpOpenFile()	102
• fpSetExtractionTimeout()	103

KVGetExtractInterface()

This function is the entry point to obtain the file extraction functions. It supplies pointers to the file extraction functions, and in the case of out-of-process mode starts the `kvoop.exe` server and initializes out-of-process extraction services. When `KVGetExtractInterface()` is called, it assigns the function pointers in the structure `KVExtractInterface` to the functions described in this section.

Syntax

```
int pascal KVGetExtractInterface (  
    void *pContext,  
    KVExtractInterface pIextract);
```

Arguments

`pContext` A pointer returned from `fpInit()` or `fpInitWithLicenseData()`.

`pIextract` A pointer to the [KVExtractInterface](#) structure, which contains function pointers that `KVGetExtractInterface()` assigns to all other file extraction functions.

Before you initialize the `KVExtractInterface` structure, use the macro `KVStructInit` to initialize the `KVStructHead` structure.

Returns

- If the call is successful, the return value is `KVERR_Success`.
- If the call is not successful, the return value is an error code.

Example

```
fpKVGetExtractInterface =  
(int (pascal *) ( void *, KVExtractInterface))myGetProcAddress(hKVFilter,  
(char*)"KVGetExtractInterface");  
/*Initialize file extraction interface structure using KVStructInit*/  
KVStructInit(&extractInterface);  
/* Retrieve file extraction interface */  
error = (*fpKVGetExtractInterface)(pFilter,&extractInterface))
```

Discussion

You can define only one extraction structure for one context pointer. For example, the following is not allowed:

```
fpInit()  
    KVGetExtractInterface(pFilter, &extractInterface1)  
  
    fpOpenFile()  
    fpGetMainFileInfo()  
    fpGetSubFileInfo()  
    fpExtractSubFile  
    fpGetSubFileMetadata()  
    fpFilterFile()  
    fpCloseFile()  
    ...  
  
    KVGetExtractInterface(pFilter, &extractInterface2)  
    fpOpenFile()  
    fpGetMainFileInfo()  
    fpGetDocInfoFile()  
    fpGetOLESummaryInfoFile()  
    fpFilterFile()  
    fpCloseFile()  
    ...  
fpShutdown()
```

fpCloseFile()

This function frees the memory allocated by [fpOpenFile\(\)](#) and closes the file.

Syntax

```
int (pascal *fpCloseFile) (void *pFile);
```

Arguments

`pFile` The identifier of the file. This is a file handle returned from `fpOpenFile()`.

Returns

- If the file is closed, the return value is `KVERR_Success`.
- If the file is not closed, the return value is an error code.

Example

```
extractInterface->fpCloseFile(pFile);  
pFile = NULL;
```

fpExtractSubFile()

This function extracts a subfile from a container file to a user-defined path or output stream. This call returns file format information when file is extracted to a path.

Syntax

```
int (pascal *fpExtractSubFile) (  
    void *pFile,   
    KVExtractSubFileArg extractArg,   
    KVSubFileExtractInfo *extractInfo);
```

Arguments

- `pFile` The identifier of the file. This is a file handle returned from `fpOpenFile()`.
- `extractArg` A pointer to the structure `KVExtractSubFileArg`, which defines the subfile to be extracted.
Before you initialize the `KVExtractSubFileArg` structure, use the macro `KVStructInit` to initialize the `KVStructHead` structure.
- `extractInfo` A pointer to the structure `KVSubFileExtractInfo`, which defines information about the extracted subfile.

Returns

- If the subfile is extracted from the container file, the return value is `KVERR_Success`.
- If the subfile is not extracted from the container file, the return value is an error code.

Discussion

- After the file is extracted, call `fpFreeStruct()` to free the memory allocated by this function.
- If the subfile is embedded in the main file as a link and is stored externally, `extractInfo->infoFlag` is set to `KVSubFileExtractInfoFlag_External`.

For example, the subfile might be an object that was embedded in a Word document by using "Link to File," or an attachment that is referenced in an MBX message. This type of subfile cannot be extracted. You must write code to access the subfile based on the path in the member `extractInfo->filePath` or `extractInfo->fileName`. See [KVSubFileExtractInfo, on page 115](#).

Example

```
KVSubFileExtractInfo  extractInfo = NULL;

KVStructInit(&extractArg);

extractArg.index = index;
extractArg.extractionFlag = KVExtractionFlag_CreateDir | KVExtractionFlag_Overwrite;
extractArg.filePath = subFileInfo->subFileName;

/*Extract this subfile*/
error=extractInterface->fpExtractSubFile(pFile,&extractArg,&extractInfo);
if ( error )
{
    extractInterface->fpFreeStruct(pFile,extractInfo);
    subFileInfo = NULL;
}
```

fpFreeStruct()

This function frees the memory allocated by `fpGetMainFileInfo()`, `fpGetSubFileInfo()`, `fpGetSubFileMetadata()`, and `fpExtractSubFile()`.

Syntax

```
int (pascal *fpFreeStruct) (
    void      *pFile,
```

```
void      *obj);
```

Arguments

pFile The identifier of the file. This is a file handle returned from [fpOpenFile\(\)](#).

obj A pointer to the result object returned by [fpGetMainFileInfo\(\)](#), [fpGetSubFileInfo\(\)](#), [fpGetSubFileMetaData](#), or [fpExtractSubFile\(\)](#).

Returns

- If the allocated memory is freed, the return value is `KVERR_Success`.
- Otherwise, the return value is an error code.

Example

The example below frees the memory allocated by [fpGetSubFileInfo\(\)](#):

```
if ( subFileInfo )
{
    extractInterface->fpFreeStruct(pFile,subFileInfo);
    subFileInfo = NULL;
}
```

fpGetMainFileInfo()

This function determines whether a file is a container file—that is, whether it contains subfiles—and should be extracted further.

Syntax

```
int (pascal *fpGetMainFileInfo) (
    void      *pFile,
    KVMainFileInfo *fileInfo);
```

Arguments

pFile The identifier of the file. This is a file handle returned from [fpOpenFile\(\)](#).

fileInfo A pointer to the structure [KVMainFileInfo](#). This structure contains information about the file.

Returns

- If the file information is retrieved, the return value is `KVERR_Success`.
- If the file information is not retrieved, the return value is an error code.

Discussion

- After the file information is retrieved, call `fpFreeStruct()` to free the memory allocated by this function.
- If the file is a container (`fileInfo->numSubFiles` is non-zero), call `fpGetSubFileInfo()` and `fpExtractSubFile()` for each subfile.
- If the file is not a container (`fileInfo->numSubFiles` is 0) and contains text (`fileInfo->infoFlag` is set to `KVMainFileInfoFlag_HasContent`), pass the file directly to the filtering functions.

Example

```
KVMainFileInfo  fileInfo  = NULL;
if( (error=extractInterface->fpGetMainFileInfo(pFile,&fileInfo))
{
    /* Free result object allocated in fileInfo */
    extractInterface->fpFreeStruct(pFile,fileInfo);
    fileInfo = NULL;
}
```

fpGetSubFileInfo()

This function gets information about a subfile in a container file.

Syntax

```
int (pascal *fpGetSubFileInfo) (
    void          *pFile,
    int           index,
    KVSubFileInfo *subFileInfo);
```

Arguments

- `pFile` The identifier of the main file. This is a file handle returned from `fpOpenFile()`.
- `index` The index number of the subfile for which to retrieve information.
- `subFileInfo` A pointer to the `KVSubFileInfo` structure, which defines information about the subfile.

Returns

- If the file information is retrieved, the return value is `KVERR_Success`.
- If the file information is not retrieved, the return value is an error code.

Discussion

- After the subfile information is retrieved, call [fpFreeStruct\(\)](#) to free the memory allocated by this function.
- If the root node is *not* enabled, the first subfile is index 0. If the root node is enabled, the first subfile is index 1. The root node is required to recreate a file's hierarchy. See [Create a Root Node, on page 38](#).
- The members `subFileInfo->parentIndex` and `subFileInfo->childArray` enable you to recreate a file's hierarchy. Because `childArray` retrieves only the first-level children in the subfile, you must call `fpGetSubFileInfo()` repeatedly until information for the leaf-node children is extracted. See [Recreate a File's Hierarchy, on page 38](#).
- If the subfile is embedded in the main file as a link and is stored externally, `subFileInfo->infoFlag` is set to `KVSubFileInfoFlag_External`. For example, the subfile might be an object that was embedded in a Word document by using "Link to File," or an attachment that is referenced in an MBX message. This type of subfile cannot be extracted. You must write code to access the subfile based on the path in the member `subFileInfo->subFileName`. See [KVSubFileInfo, on page 116](#).
- The `KVSubFileInfoFlag_External` flag is not set for an OLE object that is embedded as a link in a Microsoft PowerPoint file. KeyView can detect linked objects in a Microsoft PowerPoint file only when the object is extracted. See [fpExtractSubFile\(\), on page 96](#).

Example

```
KVSubFileInfo    subFileInfo = NULL;
for ( index = 0; index < fileInfo->numSubFiles; index++)
{
    error=extractInterface->fpGetSubFileInfo(pFile,index,&subFileInfo);
    if ( error )
    {
        extractInterface->fpFreeStruct(pFile,subFileInfo);
        subFileInfo = NULL;
    }
}
```

fpGetSubFileMetaData()

This function extracts metadata from mail stores, mail messages, and non-mail items. See [Extract Mail Metadata, on page 40](#).

Syntax

```
int (pascal *fpGetSubFileMetaData) (  
    void *pFile,  
    KVGetSubFileMetaArg metaArg,  
    KVSubFileMetaData *metaData);
```

Arguments

- pFile** The identifier of the file. This is a file handle returned from [fpOpenFile\(\)](#).
- metaArg** A pointer to the [KVGetSubFileMetaArg](#) structure, which defines metadata tags whose values are retrieved.
- Before you initialize the [KVGetSubFileMetaArg](#) structure, use the [KVStructInit](#) macro to initialize the [KVStructHead](#) structure.
- metaData** A pointer to the [KVSubFileMetaData](#) structure, which contains the retrieved metadata values.

Returns

- If the metadata is retrieved, the return value is `KVERR_Success`.
- If the metadata is not retrieved, the return value is an error code.

Discussion

- `KeyView` can extract a predefined set of common subfile metadata fields for all mail formats, and can extract all metadata from EML, MBX, MIME, NSF, ICS, and DXL files. To extract the common metadata fields, pass in `0` for `metaArg->metaNameCount`, and `NULL` for `metaArg->metaNameArray`. To extract all metadata, pass in `-1` for `metaArg->metaNameCount` and `NULL` for `metaArg->metaNameArray`. For more information, see [Extract Mail Metadata, on page 40](#).
- After the metadata is retrieved, call [fpFreeStruct\(\)](#) to free the memory allocated by this function.
- If a field is repeated in an EML or MBX mail header, the values in each instance of the field are concatenated and returned as one field. The values are separated by five pound signs (#####) as a delimiter.

Example

```
KVSubFileMetaData metaData = NULL;  
  
KVStructInit(&metaArg);  
  
/* retrieve all the default metadata elements */
```

```
metaArg.metaNameCount = 0;
metaArg.metaNameArray = NULL;
metaArg.index = Index;

error = extractInterface->fpGetSubFileMetaData(pFile,&metaArg,&metaData);
...

extractInterface->fpFreeStruct(pFile,metaData);
metaData = NULL;

/* retrieve specific metadata fields */
KVMetaName  pName[2];
KVMetaNameRec names[2];

names[0].type = KVMetaNameType_Integer;
names[0].name.iname = KVPR_SUBJECT;

names[1].type = KVMetaNameType_Integer;
names[1].name.iname = KVPR_DISPLAY_T0;

pName[0] = &names[0];
pName[1] = &names[1];

metaArg.metaNameCount = 2;
metaArg.metaNameArray = pName;
metaArg.index = Index;

error = extractInterface->fpGetSubFileMetaData (pFile,&metaArg,&metaData);
...
extractInterface->fpFreeStruct(pFile,metaData);
metaData = NULL;
```

fpOpenFile()

This function opens a file to make the file accessible for subfile extraction or filtering.

Syntax

```
int (pascal *fpOpenFile) (
    void                *pContext,
    KVOpenFileArg       openArg,
    void                **pFile);
```

Arguments

- pContext** A pointer returned from `fpInit()` or `fpInitWithLicenseData()`.
- openArg** A pointer to the [KVOpenFileArg](#) structure. This structure defines the input parameters necessary to open a file for extraction, such as credentials, and the default extraction directory.
- Before you initialize the `KVOpenFileArg` structure, use the macro `KVStructInit` to initialize the `KVStructHead` structure.
- pFile** A handle for the opened file. This handle is used in subsequent file extraction calls to identify the source file.

Returns

- If the file is opened, the return value is `KVERR_Success`.
- If the file is not opened, the return value is an error code and `pFile` is `NULL`.

Discussion

Call [fpCloseFile\(\)](#) to free the memory allocated by this function.

Example

```
KVOpenFileArgRec    openArg;

/*Initialize the structure using KVStructInit*/
KVStructInit(&openArg);
openArg.extractDir = destDir;
openArg.filePath   = srcFile;

/*Open the main file */
if ( (error = extractInterface->fpOpenFile(pFilter,&openArg,&pFile)))
{
    extractInterface->fpCloseFile(pFile);
    pFile = NULL;
}
```

fpSetExtractionTimeout()

This function specifies the length of time that should elapse before assuming that out-of-process extraction has stopped responding.

Syntax

```
BOOL pascal fpSetExtractionTimeout( void *pContext,  
long lTimeout );
```

Arguments

pContext A pointer returned from `fpInit()` or `fpInitWithLicenseData()`.

lTimeout The length of time, in seconds, that must elapse before assuming that out-of-process extraction has stopped responding.

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

Discussion

If this API is not used in out-of-process mode, the filter timeout duration is used on the [fpOpenFile\(\)](#) call. See [fpSetTimeout\(\), on page 157](#).

Example

```
/* set extraction timeouts to 10 minutes */  
  
if (FALSE == extractInterface->fpSetExtractionTimeout(pContext, 600))  
{  
    /* could not set the extraction timeout */  
}
```


Chapter 7: File Extraction API Structures

This section provides information on the structures used by the File Extraction API. These structures define the input and output parameters required to extract subfiles from a container file, and are defined in `kvextract.h`.

- [KVCredential](#) 105
- [KVCredentialComponent](#) 106
- [KVExtractInterface](#) 106
- [KVExtractSubFileArg](#) 107
- [KVGetSubFileMetaArg](#) 110
- [KVMainFileInfo](#) 111
- [KVMetadataElem](#) 112
- [KVMetaName](#) 113
- [KVOpenFileArg](#) 114
- [KVOutputStream](#) 115
- [KVSubFileExtractInfo](#) 115
- [KVSubFileInfo](#) 116
- [KVSubFileMetaData](#) 119

KVCredential

This structure contains a count of the number of credential elements, and a pointer to the first element of the array of individual elements. The structure is initialized by calling `fpOpenFile()`, and is defined in `kvextract.h`.

```
typedef struct tag_KVCredential
{
    int                itemCount;
    KVCredentialComponent *items;
}
KVCredentialRec, *KVCredential;
```

Member Descriptions

- `itemCount` The number of credentials defined for this file.
- `items` A pointer to the [KVCredentialComponent](#) structure. This structure contains the individual credential elements used to open a protected file.

KVCredentialComponent

This structure contains the value of a credential item. The structure is defined in `kvextract.h`.

```
typedef struct tag_KVCredentialComponent
{
    KVCredKeyType    keytype;
    union
    {
        void          *pkey;
        char           *skey;
        unsigned int   ikey;
    }
    keyobj;
}
KVCredentialComponentRec, *KVCredentialComponent;
```

Member Descriptions

keytype The type of credential (such as a user name or password). The types are defined by the [KVCredKeyType](#) enumerated type.

pkey A pointer to a structure defining credentials. Reserved for future use.

skey A pointer to a string credential key.

ikey An integer credential key.

KVExtractInterface

The members of this structure are pointers to the file extraction functions described in [File Extraction API Functions, on page 94](#). When you call the [KVGetExtractInterface\(\)](#) function, this structure assigns pointers to the functions. The structure is defined in `kvextract.h`.

```
typedef struct tag_KVExtractInterface
{
    KVStructHeader;
    int (pascal *fpOpenFile) (void *pContext, KVOpenFileArg openArg, void
**pFileHandle);
    int (pascal *fpCloseFile) (void *pFileHandle);
    int (pascal *fpGetMainFileInfo) (void *pFile, KVMainFileInfo *MainFileInfo);
    int (pascal *fpGetSubFileInfo) (void *pFile, int index, KVSubFileInfo
*subFileInfo);
    int (pascal *fpGetSubFileMetaData) (void *pFile, KVGetSubFileMetaArg metaArg,
KVSubFileMetaData *metaData);
    int (pascal *fpExtractSubFile) (void *pFile, KVExtractSubFileArg extractArg,
KVSubFileExtractInfo *extractInfo);
```

```
    int (pascal *fpFreeStruct) (void *pFile, void *obj);  
}  
KVExtractInterfaceRec, *KVExtractInterface;
```

Member Descriptions

The member functions are described in [File Extraction API Functions, on page 94](#).

Discussion

Before you initialize a File Extraction structure, use the `KVStructInit` macro to initialize the `KVStructHead` structure. This process sets the revision number of the File Extraction API and supports binary compatibility with future releases.

KVExtractSubFileArg

This structure defines the input parameters required to extract a subfile. See [fpExtractSubFile\(\), on page 96](#). The structure is defined in `kvextract.h`.

```
typedef struct tag_KVExtractSubFileArg  
{  
    KVStructHeader;  
    int            index;  
    KVCharSet     srcCharset;  
    KVCharSet     trgCharset;  
    int           isMSBLSB;  
    DWORD         extractionFlag;  
    char          *filePath;  
    char          *extractDir;  
    KVOutputStream *stream;  
}  
KVExtractSubFileArgRec, *KVExtractSubFileArg;
```

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead, on page 168 .
<code>index</code>	The index number of the subfile to be extracted.
<code>srcCharset</code>	Specifies the source character set of the subfile when the file format's reader cannot determine the character set. The character sets are enumerated in <code>KVCharSet</code> of <code>kvtypes.h</code> . See Discussion, on page 109 .
<code>trgCharset</code>	If the file type is <code>KVFileType_Main</code> , this is the target character set of the extracted file. Otherwise, this is ignored. The character sets are enumerated in <code>KVCharSet</code> in <code>kvtypes.h</code> . See Discussion, on page 109 .

- isMSBLSB** This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).
- extractionFlag** A bitwise flag that defines additional parameters for file extraction. The following flags are available:
- **KVExtractionFlag_CreateDir**

This flag indicates whether the directory structure of a subfile should be created. If you set this flag, the path defined in `filePath` is created if it does not already exist. If you do not set this flag, the path is not created, and the function returns `FALSE`.
 - **KVExtractionFlag_Overwrite**

If you set this flag, and the file being extracted has the same name as a file in the target path, the file in the target path is overwritten without warning. If you do not set this flag, and a subfile has the same name as a file in the target path, the error `KVError_OutputFileExists` is generated.
 - **KVExtractionFlag_ExcludeMailHeader**

If you set this flag, header information (To, From, Sent, and so on) in a mail file is not included in the extracted data. If you do not set this flag, the extracted data contains header information and the message's body text. See [Exclude Metadata from the Extracted Text File, on page 46](#).
 - **KVExtractionFlag_GetFormattedBody**

If you set this flag, the formatted version of the message body (HTML or RTF) is extracted from mail files when possible. If neither an HTML nor RTF version of the message body exists in the mail file, it is extracted as plain text. If you do not set this flag, the message body is extracted as plain text when possible.

When an HTML or RTF message body is extracted, the message's mail headers (such as "From," "To," and "Subject,") are extracted, saved in the same format, and added to the beginning of the subfile. This applies to PST (MAPI-based reader), MSG, and NSF files only.
 - **KVExtractionFlag_SaveAsMSG**

If you set this flag, the mail message is extracted as an MSG file, including all of its attachments. If you do not set this flag, the mail message is extracted as text. This applies to PST files on Windows only.

In file mode, when the application sets this flag in `fpExtractSubFile()`, it must also check the `KVSubFileExtractInfo` structure's `filePath` parameter to verify the file name used for extraction.
 - **KVExtractionFlag_SanitizeAbsolutePaths**

If you set this flag, KeyView ensures that the file is extracted to a location within the extract directory (`extractDir`), even if an absolute path is

supplied using `filePath`. When `KeyView` sanitizes a path and the resulting directory does not exist, extraction fails unless you instruct `KeyView` to create the directory, so you might also want to set the flag `KVExtractionFlag_CreateDir`. For more information, see [Sanitize Absolute Paths, on page 37](#).

<code>filePath</code>	A pointer to the suggested path or file name to which the subfile is extracted. This can be a file name, partial path, or full path. You can use this in conjunction with <code>extractDir</code> to create the full output path. See Discussion, below .
<code>extractDir</code>	A pointer to the directory to which subfiles are extracted. This directory must exist. If you set this flag, the path specified in <code>KVOpenFileArg->extractDir</code> is ignored. You can use this in conjunction with <code>filePath</code> to create the full output path.
<code>stream</code>	A pointer to an output stream defined by KVOutputStream . See Discussion, below .

Discussion

- If the document character set is detected and is also specified in `srcCharset`, the detected character set is overridden by the specified character set. If the source character set is *not* detected and is *not* specified, character set conversion does not occur. The [Supported Formats, on page 191](#) section lists the formats for which the source character set can be determined.
- The `KVSubFileExtractInfoFlag_CharsetConverted` flag in the [KVSubFileExtractInfo](#) structure indicates whether the character set of the subfile was converted during extraction.
- The following applies when the output is to a file:
 - If `filePath` is a valid absolute path, the file is extracted to the specified path and `extractDir` is ignored. However, if you have set the flag `KVExtractionFlag_SanitizeAbsolutePaths` the output path is modified to ensure it is within the `extractDir`. For more information, see [Sanitize Absolute Paths, on page 37](#).
 - If `filePath` is a file name or partial path, the target directory specified in either `KVExtractSubFileArg->extractDir` or `KVOpenFileArg->extractDir` is used to create the full path. See [KVOpenFileArg, on page 114](#).
 - If `filePath` is a full path or partial path, and `createDir` is `TRUE`, the directory is created if it does not already exist.
 - If `filePath` is not specified, a default name and the target directory specified in either `KVExtractSubFileArg->extractDir` or `KVOpenFileArg->extractDir` are used to create a full path.
 - If both `filePath` and `extractDir` are not specified or are invalid, an error is returned.
 - If `filePath` is valid, but `extractDir` is not valid, an error is returned.
- The following applies when the output is to a stream:

- Set `filePath` and `extractDir` to `NULL`.
- The file format (`docInfo`) and extraction file path (`filePath`) are not returned in [KVSubFileExtractInfo](#).
- The `KVExtractionFlag_CreateDir` and `KVExtractionFlag_Overwrite` flags are ignored.

KVGetSubFileMetaArg

This structure defines the metadata tags whose values are retrieved by [fpGetSubFileMetaData\(\)](#). This structure is defined in `kvextract.h`.

```
typedef struct tag_KVGetSubFileMetaArg
{
    KVStructHeader;
    int          index;
    int          metaNameCount;
    KVMetaName   *metaNameArray;
    KVCharSet    srcCharset;
    KVCharSet    trgCharset;
    int          isMSBLSB;
}
KVGetSubFileMetaArgRec, *KVGetSubFileMetaArg;
```

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead , on page 168.
<code>index</code>	The index number of the subfile for which metadata is extracted.
<code>metaNameCount</code>	The number of metadata fields to be extracted.
<code>metaNameArray</code>	A pointer to the KVMetaName structure that contains an array of metadata tags whose values are retrieved.
<code>srcCharset</code>	Specifies the source character set of the metadata when the format's reader cannot determine the character set. The character sets are enumerated in <code>KVCharSet</code> of <code>kvtypes.h</code> . See Discussion , below.
<code>trgCharset</code>	The target character set of the extracted metadata. The character sets are enumerated in <code>KVCharSet</code> in <code>kvtypes.h</code> .
<code>isMSBLSB</code>	This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).

Discussion

- If the character set is detected and is also specified in `srcCharset`, the detected character set is overridden by the specified character set. If the source character set is *not* detected and is *not*

specified, character set conversion does not occur. The section [Supported Formats, on page 191](#) lists the formats for which the source character set can be determined.

- KeyView can extract a predefined set of common subfile metadata fields for all mail formats, and can extract all metadata from EML, MBX, MIME, NSF, ICS, and DXL files. To extract the common metadata fields, pass in 0 for `metaArg->metaNameCount`, and NULL for `metaArg->metaNameArray`. To extract all metadata, pass in -1 for `metaArg->metaNameCount` and NULL for `metaArg->metaNameArray`. For more information, see [Extract Mail Metadata, on page 40](#).

KVMainFileInfo

This structure contains information about a main file that is open for extraction. It is initialized by calling [fpGetMainFileInfo\(\)](#). This structure is defined in `kvextract.h`.

```
typedef struct tag_KVMainFileInfo
{
    KVStructHeader;
    int          numSubFiles;
    ADDOCINFO   docInfo;
    KVCharSet   charset;
    int         isMSBLSB;
    unsigned long infoFlag;
}
KVMainFileInfoRec, *KVMainFileInfo;
```

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead, on page 168 .
<code>numSubFiles</code>	The number of subfiles in the main file.
<code>docInfo</code>	The file's major format (such as Microsoft Word or Corel Presentation), as defined by the structure <code>ADDOCINFO</code> . See ADDOCINFO, on page 163 .
<code>charset</code>	The character set of the main file.
<code>isMSBLSB</code>	This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).
<code>infoFlag</code>	A bitwise flag that provides additional information about the main file. The following flag is available: <code>KVMainFileInfoFlag_HasContent</code> —The main file contains text that can be filtered. Below are some examples of how this flag is used: <ul style="list-style-type: none">• For an MSG file without attachments, <code>numSubFiles</code> is 1 (message body text), and this flag is <code>FALSE</code> because the MSG file itself does not contain text.• For a Zip file with three files, <code>numSubFiles</code> is 3, and this flag is <code>FALSE</code>

because a Zip file does not contain text.

- For a Microsoft Word file with an embedded OLE object, `numSubFiles` is 1 (OLE object), and this flag is `TRUE` (Word file contains text to be filtered).

Discussion

- If `numSubFiles` is non-zero, get information on the subfile by calling `fpGetSubFileInfo()`, and then extract the subfiles by using `fpExtractSubFile()`.
- If `numSubFiles` is 0, the file does not contain subfiles and does not need to be extracted further. If the `KVMainInfoFlag_HasContent` flag is set, the file contains body text and can be passed directly to the filtering functions. See [Filter API Functions, on page 121](#).
- If `openFlag` is set to `KVOpenFileFlag_CreateRootNode` in the call to `fpOpenFile()`, `numSubFiles` also includes the root object (index 0) which is created by KeyView for reconstructing the file's hierarchy. See [KVOpenFileArg, on page 114](#).

KVMetadataElem

This structure contains metadata field values extracted from a mail file. This structure is defined in `kvtypes.h`.

```
typedef struct tag_KVMetadataElem
{
    int             isValid;
    int             dataID;
    KVMetadataType dataType;
    char*           strType;
    void*           data;
    int             dataSize;
}
KVMetadataElem;
```

Member Descriptions

<code>isValid</code>	Specifies whether the metadata returned from the API is valid data.
<code>dataID</code>	The integer name of the extracted metadata field.
<code>dataType</code>	The data type of the metadata field. The types are defined in KVMetadataType in <code>kvtypes.h</code> .
<code>strType</code>	A pointer to the string name of the metadata field.
<code>data</code>	The contents of the metadata field. If the type member is <code>KVMetadata_Int4</code> or <code>KVMetadata_Boo1</code> , this member contains the actual value. Otherwise, this member is a pointer to the actual value.

KVMetadata_DateTime points to an 8-byte value.

KVMetadata_String and KVMetadata_Unicode point to the beginning of the string that contains the text. The strings are NULL terminated.

KVMetadata_Binary points to the first element of a byte array.

dataSize The byte count of data when the type is KVMetadata_Binary, KVMetadata_Unicode, or KVMetadata_String.

KVMetaName

This structure defines the names of the metadata fields to be extracted from a mail file. This structure is defined in kvextract.h.

```
typedef struct tag_KVMetaName
{
    KVMetaNameType    type;
    union
    {
        void          *pname;
        int           icode;
        char          *sname;
    }name;
}
KVMetaNameRec, *KVMetaName;
```

Member Descriptions

type The type of metadata name (such as integer or string). The types are defined by the [KVMetaNameType](#) enumerated type.

NOTE: MAPI property names are of type integer.

pname A pointer to a structure defining the metadata fields to be retrieved.

icode The name of a metadata field of type integer.

sname A pointer to the name of a metadata field of type string.

Discussion

If you specify the MAPI tag name (for example, PR_CONVERSATION_TOPIC), you must include the mapitags.h and mapidefs.h Windows header files, in which PR_CONVERSATION_TOPIC is defined as 0x0070001e.

KVOpenFileArg

This structure defines the input arguments necessary to open a file for extraction. It is initialized by calling [fpOpenFile\(\)](#). This structure is defined in `kvextract.h`.

```
typedef struct tag_KVOpenFileArg
{
    KVStructHeader;
    KVCredential    cred;
    KVInputStream   *stream;
    char            *filePath;
    char            *extractDir;
    DWORD           openFlag;
    DWORD           reserved;
    void            *pReserved;
}
KVOpenFileArgRec, *KVOpenFileArg;
```

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead, on page 168 .
<code>cred</code>	The credentials required to open a protected PST or NSF file. This is a pointer to the KVCredential structure. Your application can define multiple credentials to this member for multiple formats.
<code>stream</code>	A pointer to the developer-assigned instance of <code>KVInputStream</code> . The <code>KVInputStream</code> structure defines the input stream that contains the source. See KVInputStream, on page 166 . If you are using a file as input, this is <code>NULL</code> .
<code>filePath</code>	A pointer to the full file path to the source file. If you are using a stream as input, this is <code>NULL</code> .
<code>extractDir</code>	A pointer to the default directory to which subfiles are extracted. This directory must exist. You can use this in conjunction with <code>KVExtractSubFileArg->filePath</code> to create the full output path. See KVExtractSubFileArg, on page 107 .
<code>openFlag</code>	A bitwise flag that defines additional parameters for opening the file. The following flag is available: <code>KVOpenFileFlag_CreateRootNode</code> —If you set this flag, KeyView creates a root object when extracting this file's subfiles. This root node does not have a parent and is at the highest level of the file's tree structure. It is used internally to provide a reference point from which all other child nodes are determined, and the file's hierarchy is created.

If you want to maintain the file's hierarchy when you extract subfiles from a container, you must set this flag. See [Recreate a File's Hierarchy, on page 38](#) for more information.

The root node has an index of zero. Although not all container formats require an artificial root node, the root is created for all container formats regardless of whether the file itself contains a root directory or file.

reserved	Reserved for future use. It must be NULL.
pReserved	Reserved for future use. It must be NULL.

KVOutputStream

This structure defines an output stream for the extracted subfile.

```
typedef struct tag_OutputStream
{
    void *pOutputStreamPrivateData;
    BOOL (pascal *fpCreate)(struct tag_OutputStream *,TCHAR *);
    UINT (pascal *fpWrite) (struct tag_OutputStream *, BYTE *, UINT);
    BOOL (pascal *fpSeek) (struct tag_OutputStream *, long, int);
    long (pascal *fpTell) (struct tag_OutputStream *);
    BOOL (pascal *fpClose) (struct tag_OutputStream *);
}
KVOutputStream;
```

Member Descriptions

All member functions are equivalent to their counterparts in the ANSI standard library.

KVSubFileExtractInfo

This structure contains information about an extracted subfile. It is initialized by calling [fpExtractSubFile\(\)](#). This structure is defined in `kvxtract.h`.

```
typedef struct tag_KVSubFileExtractInfo
{
    KVStructHeader;
    char *filePath;
    char *fileName;
    unsigned long infoFlag;
    ADDOCINFO docInfo;
}
KVSubFileExtractInfoRec, *KVSubFileExtractInfo;
```

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead, on page 168 .
<code>filePath</code>	<p>The full path to which the subfile was extracted.</p> <p>If the subfile is embedded in the main file as a link, this is the external path to the subfile.</p> <p>If you output the data to a stream, the extraction path is not returned.</p>
<code>fileName</code>	<p>The original path, file name, or path and file name of the subfile.</p> <p>If the subfile is embedded in the main file as a link, this is the external path to the subfile.</p>
<code>infoFlag</code>	<p>A bitwise flag that provides additional information about the extracted subfile. The following flags are available:</p> <ul style="list-style-type: none">• <code>KVSubFileExtractInfoFlag_NeedsExtraction</code>—The file might contain subfiles and should be extracted further.• <code>KVSubFileExtractInfoFlag_FileCreated</code>—The file was created on disk.• <code>KVSubFileExtractInfoFlag_CharsetConverted</code>—The subfile's character set was converted.• <code>KVSubFileExtractInfoFlag_External</code>—The subfile is embedded in the main file as a link and is stored externally. For example, the subfile might be an object that was embedded in a Word document using "Link to File," or an attachment that is referenced in an MBX message. This type of file cannot be extracted. You must write code to access the subfile based on the path in the member <code>filePath</code> or <code>fileName</code>.• <code>KVSubFileExtractInfoFlag_FolderCreated</code>—A folder was created.• <code>KVSubFileExtractInfoFlag_NonFormattedBodyExtracted</code>—Indicates that a plain text version of the message was extracted due to an error extracting the formatted version of the message.
<code>docInfo</code>	<p>The file's major format (such as Microsoft Word or Corel Presentation), as defined by the structure <code>ADDDOCINFO</code>. See ADDDOCINFO, on page 163.</p> <p>If you output the data to a stream, the file format is not returned.</p>

KVSubFileInfo

This structure contains information about a subfile in a container file. It is initialized by calling [fpGetSubFileInfo\(\)](#). This structure is defined in `kvxtract.h`.

```
typedef struct tag_KVSubFileInfo
{
    KVStructHeader;
    char          *subFileName;
```

```
    int             subFileType;  
    long           subFileSize;  
    unsigned long  infoFlag;  
    KVCharSet      charset;  
    int            isMSBLSB;  
    BYTE           fileTime[8];  
    int            parentIndex;  
    int            childCount;  
    int            *childArray;  
}  
KVContainerSubFileInfoRec, *KVSubFileInfo;
```

Member Descriptions

- KVStructHeader** The KeyView version of the structure. See [KVStructHead](#), on page 168.
- subFileName** The path, file name, or path and file name of the subfile.
If the subfile is the body text of a mail file or is an embedded OLE object, KeyView provides a default file name. See [Default File Names for Extracted Subfiles](#), on page 56.
- subFileType** The subfile's position in the container file's hierarchy.
- **KVSubFileType_Main** The subfile is at the top level of the main file. This is the default subfile type. See [Discussion](#), on page 119.
 - **KVSubFileType_Attachment** The subfile is an attachment in a file.
 - **KVSubFileType_OLE** The subfile is an embedded OLE object in a compound document.
 - **KVSubFileType_Folder** The subfile is a folder or the artificial root node (see [Create a Root Node](#), on page 38).
 - **KVSubFileType_UncategorisedImage** An embedded image that has not been categorized by the reader.
 - **KVSubFileType_EmbeddedImage** An embedded image.
 - **KVSubFileType_EmbeddedIcon** An icon used to represent an embedded file.
 - **KVSubFileType_EmbeddedContent** An image used to represent content for an embedded file. This could be a preview image of the actual content, or another representation such as an icon.
 - **KVSubFileType_EmbeddedPreview** A preview of an embedded file. This is usually an image that shows part of the embedded file.
 - **KVSubFileType_XrML** The subfile contains the XrML that describes the RMS protection used on an RMS-encrypted main file.

NOTE: The classification of embedded images into images, icons, content, and previews is supported only for some Microsoft Office file formats (DOC, DOCX, XLSX, PPT, PPTX).

subFileSize	<p>The size of the subfile in bytes. This information might be useful if you do not want to extract very large files.</p> <p>This value is approximate and is the maximum size of the subfile. The subfile is usually smaller than this value when it is extracted.</p>
infoFlag	<p>A bitwise flag that provides additional information about the subfile. The following flags are available:</p> <ul style="list-style-type: none">• <code>KVSubFileInfoFlag_NeedsExtraction</code>—The subfile might contain subfiles. It must be extracted further to conclusively determine whether it contains subfiles.• <code>KVSubFileInfoFlag_Secure</code>—The subfile is secured and credentials (such as user name and password) are required to extract it. This flag applies to ZIP, RAR, and PDF files only.• <code>KVSubFileInfoFlag_SMIME</code>—The subfile is S/MIME-encrypted and credentials are required to extract it. This applies to .eml and .pst files only.• <code>KVSubFileInfoFlag_External</code>—The subfile is embedded in the main file as a link and is stored externally. For example, the subfile might be an object that was embedded in a Word document by using "Link to File," or an attachment that is referenced in an MBX message. This type of file cannot be extracted. You must write code to access the subfile based on the path in the member <code>subFileName</code>.• <code>KVSubFileInfoFlag_MailItem</code>—When the subfile type is <code>KVSubFileType_Attachment</code>, this indicates that the attachment is a mail item. This flag applies to PST, MSG, and NSF files only.
charset	<p>If the subfile is not an attachment, this is the character set of the subfile. If the subfile is an attachment, the character set is <code>KVCS_UNKNOWN</code>.</p>
isMSBLSB	<p>This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).</p>
fileTime	<p>When the subfile is a mail message, this is the file's Sent time. Otherwise, it is the last modified time. The file time is not available for the following file types:</p> <ul style="list-style-type: none">• EML attachments• OLE objects in a Microsoft Office document• Embedded images
parentIndex	<p>The index number of this file's parent. For example, the index of a folder in which the subfile is stored, or the file to which the subfile is attached. If a file does not have a parent, the <code>parentIndex</code> is -1.</p>

- `childCount` The number of first-level children in the subfile.
- `childArray` A pointer to an array of first-level children in the subfile.

Discussion

- Embedded images (`subFileType` matching `KVSubFileType_EmbeddedImage`, `KVSubFileType_EmbeddedIcon`, `KVSubFileType_EmbeddedContent`, and `KVSubFileType_EmbeddedPreview` are not extracted unless you set `ExtractImages=TRUE` in the configuration file (or the flag `KVFLT_EXTRACTIMAGES`). However, text contained in these objects *is* present in the filter output from the container file. As a result, if you filter a document but also extract and filter its embedded images, the output from `KeyView` will contain duplicate content.

If you prefer not to see the duplicate content, you can modify your application so that it ignores these sub-files based on their `subFileType`. Alternatively, in the Filter API, you can set the flag `KVFLT_NOEMBEDDEDOBJECT` using the function `fpFilterConfig()`. This instructs `KeyView` not to include information from embedded previews (`subFileType` matching `KVSubFileType_EmbeddedPreview`) in the filter output for the container file.

- The `KVSubFileType_Main` type applies to the following for each file format:

File format	<code>KVSubFileType_Main</code> applies to...
MSG and EML	The message body.
Zip files	A file inside the archive.
PST files	An item that is not an attachment, an OLE object, or a root node.
MBX files	A message in the MBX file.
NSF files	An item that is not an attachment, an OLE object, or a root node.
PDF files	An item that is not an attachment or a root node.

- If you set the `KVSubFileInfoFlag_NeedsExtraction` flag, open the subfile and extract its children. See [fpOpenFile\(\)](#), on page 102 and [fpExtractSubFile\(\)](#), on page 96.
- The `parentIndex` and `childArray` members provide information about the subfile's parent and children. You can use this information to recreate the file hierarchy on extraction. Because `childArray` retrieves only the first-level children in the subfile, you must call `fpGetSubFileInfo()` repeatedly until information for the leaf-node children is extracted. See [Recreate a File's Hierarchy](#), on page 38.

KVSubFileMetaData

This structure contains a count of the number of metadata elements extracted from a mail file, and a pointer to the first element of the array of elements. It is initialized by calling `fpGetSubFileMetaData()`. This structure is defined in `kvextract.h`.

```
typedef struct tag_KVSubFileMetaData  
{
```

```
    KVStructHeader;  
    int             nElem;  
    KVMetadataElem** ppElem;  
    unsigned long   infoFlag;  
}  
KVSubFileMetaDataRec, *KVSubFileMetaData;
```

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead , on page 168.
<code>nElem</code>	The number of metadata fields contained in the array.
<code>ppElem</code>	A pointer to an array of pointers that are the memory addresses of metadata field values in the KVMetadataElem structure.
<code>infoFlag</code>	A bitwise flag that defines additional properties of the extracted metadata. The following flag is available: <code>KVSubFileMetaInfoFlag_CharsetConverted</code> —Indicates that the metadata's character set was converted.

Chapter 8: Filter API Functions

This section describes the functions in the Filter API. Each function appears as a function prototype followed by a description of its arguments, its return value, and a discussion of its use.

• KV_GetFilterInterfaceEx()	122
• fpCanFilterFile()	124
• fpCanFilterStream()	125
• fpCloseStream()	126
• fpFileToInputStreamCreate()	127
• fpFileToInputStreamFree()	128
• fpFilterConfig()	129
• fpFilterFile()	134
• fpFilterStream()	135
• fpFreeFilterOutput()	136
• fpFreeOLESummaryInfo()	137
• fpFreeXmpInfo()	138
• fpGetDocInfoFile()	139
• fpGetDocInfoStream()	140
• fpGetKvErrorCodeEx()	141
• fpGetOLESummaryInfo()	142
• fpGetOLESummaryInfoFile()	143
• fpGetTrgCharSet()	144
• fpGetXmpInfo()	145
• fpGetXmpInfoFile()	146
• fpInit()	147
• fpInitWithLicenseData()	149
• fpOpenStream()	152
• fpOpenStreamEx2()	153
• fpRefreshFilterKVOOP()	154
• fpSetReplacementChar()	155
• fpSetSrcCharSet()	156
• fpSetTimeout()	157
• fpShutdown()	158

KV_GetFilterInterfaceEx()

This function supplies pointers to other Filter functions. When `KV_GetFilterInterfaceEx()` is called, it assigns the function pointers in the structure `KVfltInterfaceEx` to other functions described in this chapter. For example, `KVfltInterfaceEx.fpInit` is assigned to point to the function `Init()`.

NOTE: This is used as an entry point to Filter API versions 7.4 and higher.

Syntax

```
KVErrorCode pascal KV_GetFilterInterfaceEx(  
    KVfltInterfaceEx *pInterfaceEx,  
    int version );
```

Arguments

`pInterfaceEx` A pointer to the structure [KVfltInterfaceEx](#), which contains function pointers that `KV_GetFilterInterfaceEx()` assigns to all other API functions.

`version` The version number of the current Filter interface. This is a symbolic constant (`KVFLTINTERFACE_REVISION`) defined in `kvflt.h`.

Returns

If the revision number of the Filter interface API is unknown, this function returns a general error (`KVERR_General`).

Discussion

- One of the initial steps in using the Filter API is to create an instance of a `KVfltInterfaceEx` structure and use this function to gain access to all other functions. The sample programs provide examples of how to do this.
- You can call the API functions directly. For example, you can call `GetOLESummaryInfo()` instead of using `fpGetOLESummaryInfo()` in `KVfltInterfaceEx`. However, Micro Focus recommends that you assign the function pointers in `KVfltInterfaceEx` to the functions for efficiency.

Example

```
void *pKVFILTER;  
KVfltInterfaceEx FilterFunc;  
KVErrorCode nRet = KVERR_Success;  
KVErrorCode (pascal *fpGetFilterInterfaceEx)( KVfltInterfaceEx *FilterFunc, int  
version );
```

```
pKVFILTER = myLoadLibrary(szDllName);  
  
fpGetFilterInterfaceEx = (KVErrCode (pascal *)( KVFltInterfaceEx *, int ) )  
myGetProcAddress(pKVFILTER, "KV_GetFilterInterfaceEx");
```

fpCanFilterFile()

This function determines whether a file's format is supported by KeyView. The supported formats are listed in [Supported Formats, on page 191](#).

If `KVERR_ General` is returned, you can retrieve the extended error code by using [fpGetKvErrorCodeEx\(\)](#), on page 141.

Syntax

```
KVErrorCode pascal fpCanFilterFile(  
    void      *pContext,  
    char      *szFile );
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

`szFile` The name of the input file to be filtered.

Returns

- If the file format is supported, the return value is `KVERR_Success`.
- If the file format is not supported, the return value is an error code. See [KVErrorCode, on page 177](#).

fpCanFilterStream()

This function determines whether the format of the file to which a stream points is supported by KeyView.

Syntax

```
KVErrorCode pascal fpCanFilterStream(  
    void *pcontext,  
    void *pStreamContext );
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

`pStreamContext` A pointer returned from [fpOpenStream\(\)](#) or [fpOpenStreamEx2\(\)](#).

Returns

- If the file format is supported, the return value is `KVERR_Success`.
- If the file format is not supported, the return value is an error code. See [KVErrorCode](#), on [page 177](#).

fpCloseStream()

This function closes a document stream opened by using `fpOpenStream()`.

Syntax

```
BOOL pascal fpCloseStream( void *pContext, void *pStreamContext );
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

`pStreamContext` A pointer returned from [fpOpenStream\(\)](#) or [fpOpenStreamEx2\(\)](#).

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

Discussion

After filtering is complete, call this function to free the memory allocated by `fpOpenStream()` or `fpOpenStreamEx2()`.

fpFileToInputStreamCreate()

This function creates an input stream from a file.

Syntax

```
BOOL pascal fpFileToInputStreamCreate(  
    void          *pContext,  
    char          *pszFileName,  
    KVInputStream *pInput)
```

Arguments

- pContext** A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).
- pszFileName** A pointer to the name of the input file to be filtered.
- pInput** A pointer to the developer-assigned instance of [KVInputStream](#). The structure [KVInputStream](#) defines the input stream that contains the source.

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

Discussion

- After filtering is complete, call [fpFileToInputStreamFree\(\)](#) to free the memory allocated by this function.
- You can access this function through the [KV_GetFilterInterfaceEx\(\)](#) function, or call it directly.

fpFileToInputStreamFree()

This function frees the memory allocated for the input stream created from a file.

Syntax

```
BOOL pascal fpFileToInputStreamFree(  
    void          *pContext,  
    KVInputStream *pInput)
```

Arguments

pContext A pointer returned from [fpInIt\(\)](#) or [fpInItWithLicenseData\(\)](#).

pInput A pointer to the developer-assigned instance of [KVInputStream](#). The structure [KVInputStream](#) defines the input stream that contains the source.

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

Discussion

- After filtering is complete, call this function to free the memory allocated by [fpFileToInputStreamCreate\(\)](#).
- You can access this function through the [KV_GetFilterInterfaceEx\(\)](#) function, or call it directly.

fpFilterConfig()

This function provides a way to enable and configure various options prior to document filtering, such as providing a password for a file, or enabling hidden text extraction.

Syntax

```
B00L pascal fpFilterConfig(  
    void *pContext,  
    int nType,  
    int nValue,  
    void *pData );
```

Arguments

- pContext** A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).
- nType** The configuration flag. This is a symbolic constant defined in `kvtypes.h`. The available options are described in the [Filter Configuration Flags, below](#) table.
- nValue** The integer value defined for the flags above.
- pData** The data for the configuration flag.

Returns

- If the call is successful, the return value is `TRUE`.
- If the call is unsuccessful, the return value is `FALSE`.

Discussion

- You must call this function after the call to [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#) and before the call to [fpFilterStream\(\)](#), on page 135 or [fpFilterFile\(\)](#), on page 134.
- Although `fpFilterConfig()` does not run out of process, any configuration flags that are set through `fpFilterConfig()` are passed to the out-of-process session.
- The configuration flags are described in the following table.

Filter Configuration Flags

Flag	Description
<code>KVFLT_SET00PSRCFILE</code>	If you set this flag to <code>TRUE</code> , the input file name is reported in the out-of-process error log when the file generates an error in stream mode.

Filter Configuration Flags, continued

Flag	Description
	<p>See Report the File Name in Stream Mode, on page 60. The default is FALSE.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is the name of the input file generating errors.</p>
KVFLT_SETTEMPDIRECTORY	<p>This flag enables you to specify the directory where temporary files created during filtering processes are stored.</p> <p>nValue is set to 0.</p> <p>pData is the path name of the directory where temporary files are stored. This value must be null terminated.</p>
KVFLT_LOGICALPDF	<p>This flag extracts paragraphs from a PDF file in the order in which they appear on the page (logical reading order). The nValue argument specifies the paragraph direction. See Filter PDF Files, on page 66.</p> <p>nValue is one of the paragraph direction options defined in the LPDF_DIRECTION enumerated type in <code>kvtypes.h</code>.</p> <p>pData is NULL.</p>
KVFLT_SETXMLCONFIGINFO	<p>This flag enables you to define which elements and attributes are extracted from XML documents with a specified format ID or root element. You can use this option to override the default settings for the supported XML formats (see Filter XML Files, on page 77), or to define settings for custom XML document types.</p> <p>The settings are defined in the KVXConfigInfo structure. To set custom settings for more than one document type, call the <code>fpFilterConfig()</code> function once for each type.</p> <p>You can also modify element extraction settings by using the <code>kvxconfig.ini</code> file. See Configure Element Extraction for XML Documents, on page 78.</p> <p>nValue is set to 0.</p> <p>pData is a pointer to the KVXConfigInfo structure.</p>
KVFLT_INCLREVISIONMARK	<p>If you set this flag to TRUE, text that was deleted from a document with revision tracking enabled is extracted from the document and included in the filtered output.</p> <p>To reset the flag and exclude deleted text from the filtered output, set the flag to FALSE (the default). See Extract Deleted Text Marked by Tracked Changes, on page 66.</p> <p>nValue is TRUE or FALSE.</p>

Filter Configuration Flags, continued

Flag	Description
	pData is NULL.
KVFLT_SETSRCPASSWORD	<p>This flag enables you to define a password used to open a password-protected file for filtering. See Filter Password Protected Files, on page 340.</p> <p>nValue is the length of the password.</p> <p>pData is the source file password, which can have a maximum length of 255 characters.</p>
KVFLT_NOEMBEDDEDOBJECT	<p>If you set this flag to TRUE, the text of embedded previews in Microsoft Word (DOC, DOCX), Excel (XLSX), and PowerPoint (PPT, PPTX) documents is not included in the filter output.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_SHOWHIDDENTEXT	<p>If you set this flag to TRUE, hidden text from Microsoft Word, Excel, and PowerPoint documents is extracted.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_NOCOMMENTS	<p>If you set this flag to TRUE, comments from Microsoft Word, PowerPoint, or Excel documents are not extracted.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_SKIPEMBEDDEDFONT	<p>If you set this flag to TRUE, text that contains embedded fonts is not filtered from PDF documents. See Filter PDF Files, on page 66.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_SHOWDATEFIELDPCODE	<p>If you set this flag to TRUE, date/time field codes are extracted from Microsoft Word, PowerPoint, and Rich Text Format documents instead of the date/time values.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_SHOWFILENAMEFIELDPCODE	<p>If you set this flag to TRUE, file name field codes are extracted from Microsoft Word documents.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>

Filter Configuration Flags, continued

Flag	Description
KVFLT_KEEPSOFTHYPHEN	<p>If you set this flag to TRUE, soft hyphens are retained when text is filtered from PDF documents. See Filter PDF Files, on page 66.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_EXPORTALLMETADATA	<p>If you set this flag to TRUE, all custom metadata is filtered from PDF documents when the metadata APIs are used. See Extract Custom Metadata from PDF Files, on page 69.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_EXPORTTAGGEDCONTENT	<p>If you set this flag to TRUE, tagged PDF content is filtered from PDF documents. See Filter Tagged PDF Content, on page 70.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_SetConfigurableArguments	<p>If you set this flag to TRUE, the pData is a variable of configurable arguments.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is a variable of configurable arguments.</p>
KVFLT_SETOUTPUTCHARSET	<p>This flag enables the output character set to be changed.</p> <p>pData is one of the character encodings defined in the KVCharSet enumerated type in kvtypes.h.</p>
KVFLT_SHOWHIDDENTEXT	<p>If you set this flag to TRUE, hidden text from Microsoft Word, Excel, PowerPoint, and PDF documents is extracted.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_EXTRACTIMAGES	<p>If you set this flag to TRUE, the extract API also extracts images contained within the file. See Extract Images, on page 38 for more details.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_TABDELIMITED	<p>If you set this flag to TRUE, tables in word processing formats are output in tab delimited formats. See Tab Delimited Output for Embedded Tables, on page 85 for more details.</p> <p>nValue is TRUE or FALSE.</p>

Filter Configuration Flags, continued

Flag	Description
	<p>pData is NULL.</p>
KVFLT_ STANDARDIZECELLFORMATS	<p>If you set this flag to TRUE, standardization of cell formats in Microsoft Excel files is enabled. See Standardize Cell Formats, on page 76.</p> <p>nValue is TRUE or FALSE.</p> <p>pData is NULL.</p>
KVFLT_ SOURCECODEDETECTION	<p>If you enable this option, KeyView attempts to identify the programming language of any source code files that it finds. The nValue argument specifies the level of identification to attempt. See Source Code Identification, on page 86.</p> <p>NOTE: Source code identification is a new, experimental feature in KeyView 12.0.</p> <p>nValue is KVSOURCECODE_OFF, KVSOURCECODE_ENABLED, or KVSOURCECODE_EXTENDED.</p> <p>pData is NULL.</p>

Examples

- To specify a password to open a password-protected file for filtering:

```
(*fpFilterConfig)(pKVFilter, KVFLT_SETSRCPASSWORD, TRUE, password);
```

where *password* is a null-terminated string of 255 or fewer characters.
- To extract hidden text from Microsoft Word, Excel, or PowerPoint files:

```
(*fpFilterConfig)(pKVFilter, KVFLT_SHOWHIDDENTEXT, TRUE, NULL);
```
- To extract all custom metadata fields from PDF documents:

```
(*fpFilterConfig)(pKVFilter, KVFLT_EXPORTALLMETADATA, TRUE, NULL);
```

fpFilterFile()

This function filters text from an input file to an output file.

If the output file path refers to an existing directory, an extended error code is set in pContext and returns KVERR_General. If KVERR_General is returned, you can retrieve the extended error code by using [fpGetKvErrorCodeEx\(\)](#), on page 141.

Syntax

```
KVErrorCode pascal fpFilterFile(  
    void          *pContext,  
    char          *szInputFile,  
    char          *szOutputFile,  
    KVSummaryInfoEx *pSummaryInfo );
```

Arguments

- pContext A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).
- szInputFile A pointer to the input file.
- szOutputFile A pointer to the output file.
- pSummaryInfo This argument is reserved. It must be NULL.

Returns

The return value is an error code. See [KVErrorCode](#), on page 177.

Discussion

This function runs in process or out of process. See [The Filter Process Model](#), on page 28.

Example

```
error = (int)(*pFilterInterface->fpFilterFile)( pFilter, srcFile, outFile, NULL );
```

fpFilterStream()

This function filters text from an input stream to an output buffer.

Syntax

```
KVErrorCode pascal fpFilterStream(  
    void          *pContext,  
    void          *pStreamContext  
    KVFilterOutput *pFilterOutput,  
    KVSummaryInfoEx *pSummaryInfo);
```

Arguments

- | | |
|----------------|--|
| pContext | A pointer returned from fpInit() or fpInitWithLicenseData() . |
| pStreamContext | A pointer returned from fpOpenStream() or fpOpenStreamEx2() . |
| pFilterOutput | A pointer to the KVFilterOutput structure. This structure defines the output buffer. |
| pSummaryInfo | This argument is reserved. It must be NULL. |

Returns

The return value is an error code. See [KVErrorCode](#), on page 177.

Discussion

- This function processes data in chunks. To return the entire output stream, you must call this function repeatedly until the entire output buffer is processed, that is, until the following condition occurs:

```
pFilterOutput-> cbText == 0
```

- This function runs in process or out of process. See [The Filter Process Model](#), on page 28.
- After each call to `fpFilterStream`, once the data in the `pFilterOutput` argument is no longer required, call [fpFreeFilterOutput](#) to free the memory allocated by this function.

Example

```
error = (int)(*pFilterInterface->fpFilterStream)( pFilter, pStream, &filterOut,  
NULL );
```

fpFreeFilterOutput()

This function frees the memory allocated in the `pFilterOutput` argument of [fpFilterStream\(\)](#).

Syntax

```
BOOL pascal fpFreeFilterOutput(  
    void *pContext,  
    KVFilterOutput *pFilterOutput);
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).
`pFilterOutput` A pointer to the [KVFilterOutput](#) structure.

Returns

If the call is successful, the return value is TRUE. If the call is unsuccessful, the return value is FALSE.

Discussion

Call this function after each chunk of filter output is returned by [fpFilterStream](#), after the data is no longer required, before calling [fpFilterStream](#) again. It is not necessary to call this function for the final call to [fpFilterStream](#) (when `pFilterOutput->cbText == 0`).

This function does not free `pFilterOutput` itself.

fpFreeOLESummaryInfo()

This function frees the memory allocated by `fpGetOLESummaryInfoFile()` or `fpGetOLESummaryInfo()` for metadata extraction.

Syntax

```
BOOL pascal fpFreeOLESummaryInfo(  
    void *pContext ,  
    KVSummaryInfoEx *pSummaryInfo );
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

`pSummaryInfo` A pointer to the [KVSummaryInfoEx](#) structure.

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

Discussion

Call this function after `fpGetOLESummaryInfo()` or `fpGetOLESummaryInfoFile()` has successfully filled `pSummaryInfo`, and the data is no longer required.

fpFreeXmpInfo()

This function frees the memory allocated by `fpGetXmpInfoFile()` or `fpGetXmpInfoStream()` for metadata extraction.

Syntax

```
BOOL pascal fpFreeXmpInfo(  
    void *pContext ,  
    KVXmpInfo *pXmpInfo );
```

Arguments

`pContext` A pointer returned from [fpInIt\(\)](#) or [fpInItWithLicenseData\(\)](#).

`pXmpInfo` A pointer to the structure [KVXmpInfo](#).

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

Discussion

Call this function after `fpGetXmpInfoFile()` or `fpGetXmpInfoStream()` has successfully filled `pXmpInfo`, and the data is no longer required.

fpGetDocInfoFile()

This function gets the following format information for a file and populates the ADDOCINFO structure:

- File format
- File class
- Major version
- Other attributes

The possible values are defined in `adinfo.h`.

An extended error code is set in `pContext` if an invalid input file is provided. You can retrieve the error code by using [fpGetKvErrorCodeEx\(\)](#), on page 141.

Syntax

```
BOOL pascal fpGetDocInfoFile(  
    void      *pContext,  
    char      *szFile,  
    ADDOCINFO *pADDocInfo );
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

`szFile` A pointer to the input file.

`pADDOCINFO` The format, file class, and version of the source document. A pointer to the [ADDOCINFO](#) structure. The structure of `ADDOCINFO` is defined in `adinfo.h`.

Returns

- If the format information is extracted, the return value for this function is `TRUE`.
- If the format information is not extracted, the return value is `FALSE`. If `FALSE` is returned, you can retrieve the extended error code by using [fpGetKvErrorCodeEx\(\)](#), on page 141.

Discussion

This function runs in process or out of process. See [The Filter Process Model](#), on page 28.

fpGetDocInfoStream()

This function gets the following format information for a stream and populates the ADDOCINFO structure:

- Format
- File Class
- Major version
- Other attributes

The possible values are defined in `adinfo.h`.

Syntax

```
BOOL pascal fpGetDocInfoStream(  
    void *pContext,  
    KVInputStream *pInput,  
    ADDOCINFO *pADDocInfo );
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

`pInput` A pointer to the input stream.

`pADDOCIINFO` The format, file class, and version of the source document. A pointer to the [ADDOCIINFO](#) structure. The structure of ADDOCINFO is defined in `adinfo.h`.

Returns

- If the format information is extracted, the return value for this function is TRUE.
- If the format information is not extracted, the return value is FALSE.

Discussion

This function runs in process or out of process. See [The Filter Process Model, on page 28](#).

fpGetKvErrorCodeEx()

This function gets an extended error code defined in `KVErrorCodeEx`. It is called to provide additional information when `fpFilterFile()` or `fpFilterStream()` returns the error `KVERR_General`. See [KVErrorCode](#), on page 177.

Syntax

```
KVErrorCodeEx pascal fpGetKvErrorCodeEx ( void *pContext )
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

Returns

The return value is an error code from `KVErrorCodeEx`.

Discussion

You can access this function through the [KV_GetFilterInterfaceEx\(\)](#) interface.

Example

```
KVErrorCode     nReturnCode = 0;  
if ( nReturnCode == KVERR_General )  
{ int kvErrorEx;  
  if ( lsv->fpKV_GetKvErrorCodeEx )  
  {  
    kvErrorEx = (*lsv->fpKV_GetKvErrorCodeEx)( pFilter );  
    if ( kvErrorEx != KVError_Last )  
      printf("KvErrorCodeEx = %d \n ", kvErrorEx );  
  }  
...  
}
```

fpGetOLESummaryInfo()

This function extracts document metadata from an input stream.

Syntax

```
KVErrorCode pascal fpGetOLESummaryInfo(  
    void *pContext,  
    KVInputStream *pInput,  
    KVSummaryInfoEx *pSummaryInfo );
```

Arguments

- | | |
|--------------|---|
| pContext | A pointer returned from fpInit() or fpInitWithLicenseData() . |
| pInput | A pointer to the developer-assigned instance of KVInputStream . The structure KVInputStream defines the input stream that contains the source. |
| pSummaryInfo | A pointer to the structure KVSummaryInfoEx . In the structure, nElem provides a count of the number of metadata elements, and pElem points to the first element of the array of individual elements as defined by the structure KVSumInfoElemEx . |

Returns

The return value is an error code. See [KVErrorCode](#), on page 177.

Discussion

- After the pSummaryInfo argument is successfully filled, and its data is no longer required, call [fpFreeOLESummaryInfo\(\)](#) to free the memory allocated by this function.
- This function runs in process or out of process. See [The Filter Process Model](#), on page 28.

fpGetOLESummaryInfoFile()

This function extracts document metadata from a file.

Syntax

```
KVErrorCode pascal fpGetOLESummaryInfoFile(  
    void          *pContext,  
    char          *szFile,  
    KVSummaryInfoEx *pSummaryInfo);
```

Arguments

- | | |
|---------------------------|---|
| <code>pContext</code> | A pointer returned from fpInit() or fpInitWithLicenseData() . |
| <code>szFile</code> | The name of the input file. |
| <code>pSummaryInfo</code> | A pointer to the KVSummaryInfoEx structure. In the structure, <code>nElem</code> provides a count of the number of metadata elements, and <code>pElem</code> points to the first element of the array of individual elements as defined by the KVSumInfoElemEx structure. |

Returns

The return value is an error code. See [KVErrorCode](#), on page 177.

Discussion

- After the `pSummaryInfo` argument is successfully filled, and its data is no longer required, call [fpFreeOLESummaryInfo\(\)](#) to free the memory allocated by this function.
- This function runs in process or out of process. See [The Filter Process Model](#), on page 28.

fpGetTrgCharSet()

This function verifies that the character set requested was actually used.

Syntax

```
KVCharSet pascal fpGetTrgCharSet(void *pContext);
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

Returns

The return value is one of the character sets listed in `kvtypes.h`.

fpGetXmpInfo()

This function extracts XMP metadata in stream mode.

Syntax

```
KVErrorCode pascal fpGetXmpInfo(  
    void *pContext,  
    KVInputStream *pInput,  
    KVXmpInfo *pXmpInfo,  
    DWORD dwXmpOptions );
```

Arguments

- | | |
|---------------------------|---|
| <code>pContext</code> | The pointer returned by fpInit() or fpInitWithLicenseData() . |
| <code>pInput</code> | A pointer to the input stream. |
| <code>pXmpInfo</code> | A pointer to the KVXmpInfo structure. |
| <code>dwXmpOptions</code> | Set this argument to 1 to return charset information, the raw XMP packet, and the path and value pairs of all XMP elements.

Set this argument to 2 to return the raw XMP packet. |

Returns

The return value is an error code. See [KVErrorCode](#), on page 177.

Discussion

- After the `pXmpInfo` argument is successfully filled, and its data is no longer required, call [fpFreeXmpInfo\(\)](#) to free the memory allocated by this function.
- This function runs in process or out of process. See [The Filter Process Model](#), on page 28.
- XMP extraction is supported only for the PDF, PNG, JPG, TIFF, XML, and pFile formats.
- XMP extraction is supported on the Windows, Linux, AIX, FreeBSD, and macOS platforms.

fpGetXmpInfoFile()

This function extracts XMP metadata from a file.

Syntax

```
KVErrorCode pascal fpGetXmpInfoFile(  
    void          *pMainContext,  
    char          *szInputFile,  
    KVXmpInfo     *pXmpInfo,  
    DWORD         dwXmpOptions );
```

Arguments

- pMainContext** A pointer to the `TPMainContext` structure, which is defined in `kvtypes.h`.
- szInputFile** A pointer to the input file.
- pXmpInfo** A pointer to the [KVXmpInfo](#) structure.
- dwXmpOptions** Set this argument to **1** to return charset information, the raw XMP packet, and the path and value pairs of all XMP elements.
Set this argument to **2** to return the raw XMP packet.

Returns

The return value is an error code. See [KVErrorCode](#), on page 177.

Discussion

- After the `pXmpInfo` argument is successfully filled, and its data is no longer required, call [fpFreeXmpInfo\(\)](#) to free the memory allocated by this function.
- This function runs in process or out of process. See [The Filter Process Model](#), on page 28.
- XMP extraction is supported only for the PDF, PNG, JPG, TIFF, XML, and pFile formats.
- XMP extraction is supported on the Windows, Linux, AIX, FreeBSD, and macOS platforms.

fpInit()

This function initializes a Filter session. Its return value, `pContext`, is passed as the first argument to the File Extraction interface and all other Filter functions.

Syntax

```
void * pascal fpInit(  
    KVMemoryStream *pMemAllocator,  
    KVDynLink *pDynLink,  
    char *pszKeyViewDir,  
    KVCharSet outputCharSet,  
    DWORD dwFlags );
```

Arguments

- `pMemAllocator` A pointer to a developer-defined memory allocator. If NULL is passed, the default C run-time memory allocation is used.
- `pDynLink` This argument is reserved. It must be NULL.
- `pszKeyViewDir` A pointer to the directory where the Filter components (such as the `formats.ini` file, license key file (`kv.lic`), and file filters) are located. This is normally the `install\OS\bin` directory.
- `outputCharSet` The character set to use for textual output when the source character set can be determined from the document or is specified by [fpSetSrcCharSet\(\)](#).
The character sets are enumerated in `KVCharSet` in `kvtypes.h`.
- `dwFlags` Instructions on how to process a file or stream. See [Flags for dwFlags, below](#) for more information.

Flags for dwFlags

- | | |
|-----------------------------------|--|
| <code>KVF_CONTENTACCESS</code> | Reserved for internal use. |
| <code>KVF_METADATA</code> | Reserved for internal use. |
| <code>KVF_OUTOFPROCESS</code> | Enables out-of-process filtering. This is enabled by default. See The Filter Process Model, on page 28 . |
| <code>KVF_INPROCESS</code> | Enables in-process filtering. See The Filter Process Model, on page 28 . |
| <code>KVF_HEADERFOOTERTAGS</code> | Puts tags around header and footer data. |
| <code>KVF_HEADERFOOTER</code> | Extracts headers and footers. |

KVF_UNICODEMSLSB	Uses the byte order for Big Endian systems (MSLSB) for Unicode text. MSLSB is the "Most Significant Byte Least Significant Byte."
KVF_UNICODELSMSB	Uses the byte order for Little Endian systems (LSMSB) for Unicode text. LSMSB is the "Least Significant Byte Most Significant Byte."
KVF_UNICODEMARKER	Generates the byte order marker for Unicode text.
KVF_NOCHARSETCONVERT	Prevents default conversion of document character encoding. See Prevent the Default Conversion of a Character Set, on page 65 .
KVF_OOPLOGON	Enables the out-of-process error log. See Enable or Disable Error Logging, on page 59 .
KVF_OOPMEMTRACEON	Enables memory trace for the out-of-process error log. See Report Memory Errors, on page 60 .
KVF_OOPLOGOFF	Disables the out-of-process error log. Enable or Disable Error Logging, on page 59 .
KVF_OOPMEMTRACEOFF	Disables memory trace for the out-of-process error log. See Report Memory Errors, on page 60 .
KVF_FILTERCONTAINERCONTENT	This flag is used by the Container API which is obsolete. It filters the main file and subfiles of a container file by using the standard filtering functions, and extracts the text to a single file.
KVF_DETECT_OUTOFPROCESS	Set these flags in <code>fpInit()</code> or <code>fpOpenStreamEx2()</code> to specify whether files are detected out of process or in process for a filtering session.
KVF_DETECT_INPROCESS	These flags override the <code>default_detect_inprocess</code> flag in <code>formats.ini</code> . If you set neither of these flags, file detection behavior is determined by the <code>KVF_OUTOFPROCESS</code> or <code>KVF_INPROCESS</code> flags in these calls. If you do not set these flags, behavior is determined by <code>default_detect_inprocess</code> in <code>formats.ini</code> . See Run File Detection In or Out of Process, on page 32 .

Returns

- If the call is successful, the return value is the pointer `pContext` which is passed as the first argument to all other File Extraction API and Filter API functions.
- If the call is unsuccessful, the return value is `NULL`.

Discussion

- If this function returns `NULL`, check `stderr` for the KeyView installation error messages "KeyView Filter SDK License Key has Expired" and "KeyView Filter SDK License Key is Invalid", and pass them to your application.

- To make sure that multithreaded filter processes are thread-safe, you must create a unique context pointer for every thread by calling `fpInit()`. In addition, threads must not share context pointers, and you must use the same context pointer for all API calls in the same thread. Creating a context pointer for every thread does not affect performance because the context pointer uses minimal resources.
- When the filtering context is no longer required, call `fpShutdown()` to terminate it.

fpInitWithLicenseData()

This function initializes a Filter session with license information passed in function parameters rather than a license file. Its return value, `pContext`, is passed as the first argument to the File Extraction interface and all other Filter functions.

This function is similar to `fpInit()`, but it uses a different licensing method. You can use either `fpInit()` or `fpInitWithLicenseData` to initialize your Filter session. However, these functions are mutually exclusive. That is, neither takes the context pointer from the other as an argument. If you call both functions, you initialize two distinct Filter sessions, in the same way as calling `fpInit()` twice.

Syntax

```
void * pascal fpInitWithLicenseData(  
    KVMemoryStream *pMemAllocator,  
    KVDynLink *pDynLink,  
    char *pszKeyViewDir,  
    const char* const pszLicenseOrganization,  
    const char* const pszLicenseKey,  
    KVCharSet outputCharSet,  
    DWORD dwFlags );
```

Arguments

<code>pMemAllocator</code>	A pointer to a developer-defined memory allocator. If NULL is passed, the default C run-time memory allocation is used.
<code>pDynLink</code>	This argument is reserved. It must be NULL.
<code>pszKeyViewDir</code>	A pointer to the directory where the Filter components (such as the <code>formats.ini</code> file and file filters) are located. This is normally the <code>install\OS\bin</code> directory.
<code>pszLicenseOrganization</code>	A pointer to the organization name under which this installation of KeyView is licensed.
<code>pszLicenseKey</code>	A pointer to the license key for this installation of KeyView.
<code>outputCharSet</code>	The character set to use for textual output when the source character set can be determined from the document or is specified by

[fpSetSrcCharSet\(\)](#).

The character sets are enumerated in KVCharSet in kvtypes.h.

dwFlags

Instructions on how to process a file or stream. See [Flags for dwFlags](#), [below](#) for more information.

Flags for dwFlags

KVF_CONTENTACCESS	Reserved for internal use.
KVF_METADATA	Reserved for internal use.
KVF_OUTOFPROCESS	Enables out-of-process filtering. This is enabled by default. See The Filter Process Model , on page 28.
KVF_INPROCESS	Enables in-process filtering. See The Filter Process Model , on page 28.
KVF_HEADERFOOTERTAGS	Puts tags around header and footer data.
KVF_HEADERFOOTER	Extracts headers and footers.
KVF_UNICODEMSLSB	Uses the byte order for Big Endian systems (MSLSB) for Unicode text. MSLSB is the "Most Significant Byte Least Significant Byte."
KVF_UNICODELSMSB	Uses the byte order for Little Endian systems (LSMSB) for Unicode text. LSMSB is the "Least Significant Byte Most Significant Byte."
KVF_UNICODEMARKER	Generates the byte order marker for Unicode text.
KVF_NOCHARSETCONVERT	Prevents default conversion of document character encoding. See Prevent the Default Conversion of a Character Set , on page 65.
KVF_OOPLOGON	Enables the out-of-process error log. See Enable or Disable Error Logging , on page 59.
KVF_OOPMEMTRACEON	Enables memory trace for the out-of-process error log. See Report Memory Errors , on page 60.
KVF_OOPLOGOFF	Disables the out-of-process error log. Enable or Disable Error Logging , on page 59.
KVF_OOPMEMTRACEOFF	Disables memory trace for the out-of-process error log. See Report Memory Errors , on page 60.
KVF_FILTERCONTAINERCONTENT	This flag is used by the Container API which is obsolete. It filters the main file and subfiles of a container file by using the standard filtering functions, and extracts the text to a single file.
KVF_DETECT_OUTOFPROCESS	Set these flags in fpInIt() , fpInitWithLicenseData() or fpOpenStreamEx2() , on page 153 to specify whether files are detected out of process or in process for a filtering session. These flags override the default_detect_inprocess flag in formats.ini.
KVF_DETECT_INPROCESS	

If you set neither of these flags, file detection behavior is determined by the `KVF_OUTOFPROCESS` or `KVF_INPROCESS` flags in these calls. If you do not set these flags, behavior is determined by `default_detect_inprocess` in `formats.ini`.

See [Run File Detection In or Out of Process, on page 32](#).

Returns

- If the call is successful, the return value is the pointer `pContext` which is passed as the first argument to all other File Extraction API and Filter API functions.
- If the call is unsuccessful, the return value is `NULL`.

Discussion

- If this function returns `NULL`, check `stderr` for the KeyView installation error messages "KeyView Filter SDK License Key has Expired" and "KeyView Filter SDK License Key is Invalid", and pass them to your application.
- To make sure that multithreaded filter processes are thread-safe, you must create a unique context pointer for every thread by calling [`fpInit\(\)`, on page 147](#) or `fpInitWithLicenseData()`. In addition, threads must not share context pointers, and you must use the same context pointer for all API calls in the same thread. Creating a context pointer for every thread does not affect performance because the context pointer uses minimal resources.
- When the filtering context is no longer required, call [`fpShutdown\(\)`](#) to terminate it.

fpOpenStream()

This function opens a stream for filtering.

Syntax

```
void * pascal fpOpenStream(  
    void          *pContext,  
    KVInputStream *pInput );
```

Arguments

pContext A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

pInput A pointer to the developer-assigned instance of [KVInputStream](#). The structure [KVInputStream](#) defines the input stream that contains the source.

Returns

- If the call is successful, the return value is a `void *` pointer passed to [fpFilterStream\(\)](#), [fpCanFilterStream\(\)](#), and [fpCloseStream\(\)](#).
- If the call is unsuccessful, the return value is `NULL`.

Discussion

- Before you call this function, you must create an input stream either by using the [fpFiletoInputStreamCreate\(\)](#) function, or by using code similar to the coding example in the Filter sample program. See [Use the C-Language Implementation of the API, on page 24](#) for more information.
- After filtering is complete, call [fpCloseStream\(\)](#) to free the memory allocated by this function.

fpOpenStreamEx2()

This function opens a stream for filtering and enables you to set bitwise flags for each stream.

Syntax

```
void * pascal fpOpenStreamEx2(  
    void          *pContext,  
    KVInputStream *pInput,  
    DWORD         dwFlags);
```

Arguments

- pContext** A pointer returned from [fpInIt\(\)](#) or [fpInItWithLicenseData\(\)](#).
- pInput** A pointer to the developer-assigned instance of [KVInputStream](#). The [KVInputStream](#) structure defines the input stream that contains the source.
- dwFlags** Instructions on how to process a stream. See [Flags for dwFlags, on page 147](#).

Returns

- If the call is successful, the return value is a void * pointer passed to [fpFilterStream\(\)](#), [fpCanFilterStream\(\)](#), and [fpCloseStream\(\)](#).
- If the call is unsuccessful, the return value is NULL.

Discussion

- Before you call this function, you must create an input stream either by using the [fpFiletoInputStreamCreate\(\)](#) function, or by using code similar to the coding example in the Filter sample program. See [Use the C-Language Implementation of the API, on page 24](#) for more information.
- After filtering is complete, call [fpCloseStream\(\)](#) to free the memory allocated by this function.

fpRefreshFilterKVOOP()

This function forces the out-of-process filtering server (kvoop.exe) to restart. This function is optional.

Syntax

```
int (pascal *fpRefreshFilterKVOOP)( void *pContext );
```

Arguments

pContext A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

Returns

- If the restart is successful, the return value is `KVERR_Success`.
- If the restart is unsuccessful, the return value is an error code.

NOTE: There are several different error codes.

fpSetReplacementChar()

This function specifies a replacement character to use when a character cannot be mapped. This function is optional.

Syntax

```
BOOL pascal fpSetReplacementChar( void *pContext, char c );
```

Arguments

- pContext** A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).
- c** The replacement character to use when a character cannot be mapped. If you do not call this function, the default character is used.
The default is a question mark ("?").

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

fpSetSrcCharSet()

This function specifies a character set for the source document. Use this function if the character set information cannot be determined from the source document.

Syntax

```
BOOL pascal fpSetSrcCharSet( void *pContext, KVCharSet eCharSet );
```

Arguments

pContext A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

eCharSet Specifies the source character set when the document reader for the document type cannot determine the character set. The character sets are enumerated in KVCharSet of kvtypes.h.

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

fpSetTimeout()

This function specifies the length of time that should elapse before assuming that the filtering process has stopped responding.

Syntax

```
BOOL pascal fpSetTimeout( void *pContext, long lTimeout );
```

Arguments

pContext A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

lTimeout The length of time, in seconds, that must elapse before assuming that the filtering process has stopped responding.

Returns

- If the call is successful, the return value is TRUE.
- If the call is unsuccessful, the return value is FALSE.

fpShutdown()

This function terminates a filtering session that was initialized by [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#), and frees allocated system resources. It is called when the filtering context is no longer required.

Syntax

```
void pascal fpShutdown( void *pContext );
```

Arguments

`pContext` A pointer returned from [fpInit\(\)](#) or [fpInitWithLicenseData\(\)](#).

Returns

None.

Chapter 9: Filter API Structures

This section describes the data structures used by the Filter API. These structures are defined in `kvflt.h`, `kwautdef.h`, and `adinfo.h`.

• KVFitInterfaceEx	160
• ADDOCINFO	163
• KV_CONFIG_Arg	164
• KVFilterOutput	165
• KVInputStream	166
• KVMemoryStream	167
• KVStructHead	168
• KVSumInfoElemEx	169
• KVSummaryInfoEx	170
• KVXConfigInfo	171
• KVXmplInfo	173
• KVXmplInfoElems	174

KVFltInterfaceEx

The members of this structure are pointers to the functions described in [Filter API Functions, on page 121](#). When you call the `KV_GetFilterInterfaceEx()` function, this structure assigns pointers to the functions. The structure is defined in `kvfilt.h`.

```
typedef struct tag_KVFltInterfaceEx
{
    void *      (pascal *fpInit) ( KVMemoryStream *, KVDynLink *, char *, KVCharSet,
DWORD );
    void      (pascal *fpShutdown) (void *);
    void *      (pascal *fpOpenStream)( void *, KVInputStream * );
    void *      (pascal *fpOpenStreamEx2) (void *, KVInputStream *, DWORD);
    BOOL      (pascal *fpCloseStream)( void *, void * );
    BOOL      (pascal *fpCanFilterCharMap)( void *, adDocDesc * );
    KVErrCode (pascal *fpCanFilterFile)( void *, char * );
    KVErrCode (pascal *fpCanFilterStream) (void *, void *);
    KVErrCode (pascal *fpFilterStream)( void *, void *, KVFilterOutput *,
KVSummaryInfoEx * );
    KVErrCode (pascal *fpFilterFile)( void *, char *, char *, KVSummaryInfoEx * );
    KVErrCode (pascal *fpGetOLESummaryInfo)( void *, KVInputStream *,
KVSummaryInfoEx * );
    KVErrCode (pascal *fpGetOLESummaryInfoFile)( void *, char *, KVSummaryInfoEx *
);
    BOOL      (pascal *fpFreeOLESummaryInfo)( void *, KVSummaryInfoEx * );
    KVCharSet (pascal *fpGetTrgCharSet)( void * );
    BOOL      (pascal *fpSetTimeout)( void *, long );
    BOOL      (pascal *fpSetSrcCharSet)( void *, KVCharSet );
    BOOL      (pascal *fpSetReplacementChar)( void *, char );
    BOOL      (pascal *fpGetDocInfoStream)( void *, KVInputStream *, ADDOCINFO * );
    BOOL      (pascal *fpGetDocInfoFile)( void *, char *, ADDOCINFO * );
    BOOL      (pascal *fpIsArchiveFile)( void *, char * );
    BOOL      (pascal *fpIsArchiveFileSupported)( void *, char * );
    void *      (pascal *fpOpenArchiveFile)( void *, char * );
    int      (pascal *fpGetNumFilesInArchiveFile)( void * );
    KVErrCode (pascal *fpGetArchiveFileInfo)( void *, int, TPArchiveFileInfo * );
    KVErrCode (pascal *fpExtractArchiveFile)( void *, int, char * );
    BOOL      (pascal *fpCloseArchiveFile)( void * );
    /* Revision 1 of Filter Interface API starts here (#define KVFLTINTERFACE_
REVISION). */
    BOOL      (pascal *fpFileToInputStreamCreate)(void *, char *, KVInputStream *);
    BOOL      (pascal *fpFileToInputStreamFree)(void *, KVInputStream *);
    KVErrCode (pascal *fpCanFilterAsContainer)(void *, KVInputStream *);
    void *      (pascal *fpOpenContainerStream)(void *, KVInputStream *);
    BOOL      (pascal *fpCloseContainerStream)( void *, void *);
    int      (pascal *fpGetNumFilesInContainer)( void *, void *);
    KVErrCode (pascal *fpGetContainerSubFileInfo)( void *, void *, int,
TPContainerSubFileInfo *);
};
```



```
    BOOL          (pascal *fpSetExtractionPath)(void *, void *, char *, BOOL);
    void          (pascal *fpSetExtractionOverwrite)( void *, void *, BOOL);
    KVErrCode     (pascal *fpExtractContainerSubFile)( void *, void *, int,
TPContainerSubFileInfo *);
    KVErrCode     (pascal *fpGetContainerContent)( void *, void *, KVFilterOutput *,
    BOOL * );
    KVErrCodeEx  (pascal *fpGetKvErrorCodes)( void *pContext );
    BOOL         (pascal *fpFilterConfig)( void *pContext, int nType, int nValue, void
    *p );
/* Revision 2 of Filter Interface API starts here (#define KVFLTINTERFACE_REVISION)
*/
    KVErrCode     (pascal *fpGetSubFileMetadada)( void *, void *, int, int *, int,
KVSummaryInfoEx *, int );
    KVErrCode     (pascal *fpFreeSubFileMetadada)( void *, void *, KVSummaryInfoEx * );
}
KVfltInterfaceEx;
KVErrCode pascal KV_GetFilterInterfaceEx( KVfltInterfaceEx *pInterfaceEx, int
version );
```

Member Descriptions

The member functions are described in [Filter API Functions](#), on page 121.

Discussion

The following functions are deprecated:

- fpIsArchiveFile
- fpIsArchiveFileSupported
- fpOpenArchiveFile
- fpGetNumFilesInArchiveFile
- fpGetArchiveFileInfo
- fpExtractArchiveFile
- fpCloseArchiveFile
- fpCanFilterCharMap
- fpCanFilterAsContainer
- fpCloseContainerStream
- fpGetNumFilesInContainer
- fpGetContainerSubFileInfo
- fpSetExtractionPath
- fpSetExtractionOverwrite

- `fpExtractContainerSubFile`
- `fpGetContainerContent`
- `fpFreeSubFileMetadada`

ADDOCINFO

This structure contains the format, file class, and version number of the source document. The structure is defined in `adinfo.h`, and is initialized by calling the `fpGetDocInfoFile()` or `fpGetDocInfoStream()` functions.

```
typedef struct
{
    ENdocClass      eClass;
    ENdocFmt       eFormat;
    long           lVersion;
    unsigned long  ulAttributes;
}
ADDOCINFO, *ADDOCINFOPTR;
```

Member Descriptions

<code>eClass</code>	The file class of the source document (for example, spreadsheet, word processor, or encapsulation format), as defined by the enumerated type <code>ENdocClass</code> in <code>adinfo.h</code> .
<code>eFormat</code>	The major format of the source document (for example Microsoft Word XML format or Corel Presentation), as defined by the enumerated type <code>ENdocFmt</code> in <code>adinfo.h</code> . The <code>ENdocFmt</code> type provides a unique ID for each major format.
<code>lVersion</code>	The version number of the file format. The number is multiplied by 1,000 (for example, 1.02 is represented by 1020).
<code>ulAttributes</code>	Other attributes of the document, as defined by the enumerated type <code>ENdocAttributes</code> in <code>adinfo.h</code> .

Discussion

When format detection is enhanced in future releases, new format IDs might be added to the `ENdocFmt` enumerated type. When using this type, your code should ensure binary compatibility with future releases. For example, if you use an array to access format information based on a format ID, your code should check that the format ID is less than `Max_Fmt` before accessing the data. This ensures that new format codes are detected when you add KeyView binary files from new releases to your existing installation.

KV_CONFIG_Arg

This structure defines configurable arguments to use as the data in the [fpFilterConfig\(\)](#) function when you set the `KVFLT_SetConfigurableArguments` flag to `TRUE`. The structure is described in `kvtypes.h`.

Use this structure to control the filtering of hidden data from Microsoft Excel documents. See [Filter Hidden Data, on page 83](#).

```
typedef struct _KV_CONFIG_ARG_TAG
{
    unsigned int    keyID;
    int             keyType;
    KV_CONFIG_DATA keyData;
    unsigned int    keyDataSize;
}
KV_CONFIG_Arg;
```

Member Descriptions

<code>keyID</code>	Determines the kind of configuration flags that you can use as values of <code>keyData</code> . If you use the same <code>keyID</code> more than once, the most recent setting overrides the previous setting.
<code>keyType</code>	The type of data for the <code>keyData</code> element. Set to <code>KV_INT32ARG</code> .
<code>keyData</code>	<code>KV_CONFIG_DATA</code> is a union defined in <code>kvtypes.h</code> . Only <code>intArg</code> is supported, where the value of <code>intArg</code> is one of the flags in the corresponding <code>keyID</code> .
<code>keyDataSize</code>	The size of <code>keyData</code> . This is reserved for future use.

KVFilterOutput

This structure defines an output buffer for filtering. The structure is defined in `kvtypes.h`.

```
typedef struct tag_KVFilterOutput
{
    BYTE      *pcText;
    int       cbText;
}
KVFilterOutput;
```

Member Descriptions

`pcText` A pointer to the text returned from [fpFilterStream\(\)](#).

`cbText` The number of valid bytes in `pcText`.

KVInputStream

This structure defines an input stream for filtering. The structure is defined in `kvtypes.h`.

```
typedef struct tag_InputStream
{
    void *pInputStreamPrivateData;
    long lcbFilesize;
    BOOL (pascal *fpOpen) (struct tag_InputStream *);
    UINT (pascal *fpRead) (struct tag_InputStream *, BYTE *, UINT);
    BOOL (pascal *fpSeek) (struct tag_InputStream *, long, int);
    long (pascal *fpTell) (struct tag_InputStream *);
    BOOL (pascal *fpClose)(struct tag_InputStream *);
}
KVInputStream;
```

Member Descriptions

- All member functions are equivalent to their counterparts in the ANSI standard library, except `fpOpen()`, which returns `FALSE` on failure.
- On `fpOpen()`, if the size of the stream is known, assign that value to `lcbFilesize`. Otherwise, set `lcbFilesize` to `0`.

KVMemoryStream

This structure defines an optional memory allocator to be used by Filter. Behavior for all functions is the same as for their C run-time equivalents. The structure is defined in `kvtypes.h`.

```
typedef struct tag_MemoryStream
{
    void *pMemoryStreamPrivateData;
    void * (pascal *fpMalloc) (struct tag_MemoryStream*, size_t );
    void (pascal *fpFree) (struct tag_MemoryStream*, void *);
    void * (pascal *fpRealloc) (struct tag_MemoryStream*, void *, size_t);
    void * (pascal *fpCalloc) (struct tag_MemoryStream*, size_t, size_t);
}
KVMemoryStream;
```

Member Descriptions

- All member functions are equivalent to their counterparts in the ANSI standard library.
- `fpRealloc()` must handle a NULL pointer.

KVStructHead

This structure contains the current KeyView version number, and is the first member of other structures. It enables Micro Focus to modify the structures in future releases, but to maintain backward compatibility. Before you initialize a structure that contains the KVStructHead structure, use the macro KVStructInit to initialize KVStructHead. The structure and macro are defined in kvtypes.h.

```
typedef struct _KVStructHead
{
    WORD        version;
    WORD        size;
    DWORD       reserved;
    void        *internal;
}
KVStructHeadRec, *KVStructHead;
```

Member Descriptions

- `version` The current KeyView version number. This is a symbolic constant (`KeyviewVersion`) defined in `kvextract.h`. This constant is updated for each KeyView release.
- `size` The size of the `KVStructHeadRec`.
- `reserved` Reserved for internal use.
- `internal` Reserved for internal use.

Example

```
KVOpenFileArgRec    openArg;
KVStructInit(&openArg);
```


KVSumInfoElemEx

This structure contains the individual metadata elements. The structure is defined in `kvtypes.h`.

```
typedef struct tag_KVSumInfoElemEx
{
    int             isValid;
    KVSumInfoType  type;
    void           *data;
    char           *pcType;
}
KVSumInfoElemEx;
```

Member Descriptions

isValid Specifies whether the data value is present in the document. The setting 1 specifies that the value is valid and exists. For example, if the "Title" element is not populated in the document, `pSummaryInfo.pElem[1].isValid == 0` evaluates to true.

type The data type of the metadata element. The types are defined in [KVSumInfoType](#) in `kvtypes.h`.

data The content of the metadata field.

If the `type` member is `KV_Int4`, or `KV_Boo1`, this member contains the actual value. Otherwise, this member is a pointer to the actual value.

`KV_DateTime` and `KV_IEEE8` point to an 8-byte value.

`KV_String` and `KV_Unicode` point to the beginning of the string that contains the text. `KV_Unicode` is replaced with `KV_String` when the UNICODE value has been character mapped to the desired output character set as specified in the call to [fplnit\(\)](#), on page 147 or [fplnitWithLicenseData\(\)](#), on page 149.

pcType A pointer to the name of the metadata field.

KVSummaryInfoEx

This structure contains a count of the number of metadata elements, and a pointer to the first element of the array of individual elements. The structure is defined in `kvtypes.h`.

```
typedef struct tag_KVSummaryInfoEx
{
    int          nElem;
    KVSumInfoElemEx *pElem;
}
KVSummaryInfoEx;
```

Member Descriptions

- `nElem` The number of metadata elements contained in the array. A value of zero indicates that the document did not contain metadata, such as an ASCII text document.
- `pElem` A pointer to the first element of the array of metadata elements defined by the structure [KVSumInfoElemEx](#).

KVXConfigInfo

This structure defines an XML document type and the element extraction settings for that type. You can apply the settings based on the file format ID, or the root element of the file. This structure is in `kvtypes.h`.

```
typedef struct TAG_KVXConfigInfo
{
    ENdocFmt    eKVFormat;
    char*       pszRoot;
    char*       pszInMeta;
    char*       pszExMeta;
    char*       pszInContent;
    char*       pszExContent;
    char*       pszInAttribute;
}
KVXConfigInfo;
```

Member Descriptions

eKVFormat	<p>The format ID as detected by the KeyView detection module. This determines the file type to which these extraction settings apply. The format ID is defined by the enumerated type <code>ENdocFmt</code>. See File Format Detection, on page 302 for more information on format ID values.</p> <p>If you are adding configuration settings for a custom XML document type, this is not defined.</p>
pszRoot	<p>The root element of the file. If the format ID is not defined, the root element is used to determine the file type to which these settings apply.</p> <p>To further qualify the element, specify its namespace. See Specify an Element's Namespace and Attribute, on page 81.</p>
pszInMeta	<p>The elements extracted from the file as metadata. All other elements are extracted as text. Separate multiple entries with commas.</p> <p>To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 81.</p>
pszExMeta	<p>The child elements in the included metadata elements that are not extracted from the file as metadata. For example, the default extraction settings for the Visio XML format extract the <code>DocumentProperties</code> element as metadata. This element includes child elements such as <code>Title</code>, <code>Subject</code>, <code>Author</code>, <code>Description</code>, and so on. However, the child element <code>PreviewPicture</code> is defined in <code>pszExMeta</code> because it is binary data and should not be extracted.</p> <p>You cannot exclude any metadata elements from the output for StarOffice files. All metadata is extracted regardless of this setting.</p>

	To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 81 .
pszInContent	The elements extracted from the file as content text. An asterisk (*) extracts all elements including child elements. To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 81 .
pszExContent	The child elements in the included content elements that are not extracted from the file as content text. To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 81 .
pszInAttribute	The attribute values extracted from the file. If attributes are not defined, attribute values are not extracted. You must define the namespace (if used), element name, and attribute name in the following format: <i>namespace:elementname@attributename</i> For example: microfocus:division@name

KVXmpInfo

This structure contains the XMP metadata, and is defined in `kvtypes.h`.

```
typedef struct tag_KVXmpInfo
{
    KVCharSet      encoding;
    BOOL           bIsLittleEndian;
    UINT           nNoOfElements;
    KVXmpInfoElem *pXmpInfoElems;
    KV_I18NSTR     usXpacketData;
    void           *pExtension;
}
KVXmpInfo;
```

Member Descriptions

<code>encoding</code>	The type of encoding.
<code>bIsLittleEndian</code>	Indicates whether little-endian byte ordering is used.
<code>nNoOfElements</code>	The total number of elements.
<code>pXmpInfoElems</code>	A pointer to the KVXmpInfoElems structure.
<code>usXpacketData</code>	A copy of the XMP data.
<code>pExtension</code>	A reserved pointer.

KVXmpInfoElems

This structure contains the individual XMP metadata elements, and is defined in `kvtypes.h`.

```
typedef struct tag_KVXmpInfoElem
{
    KV_I18NSTR    usXPathToElement;
    KV_I18NSTR    usValue;
}
KVXmpInfoElem;
```

Member Descriptions

`usXPathToElement` The path to the XMP element.

`usValue` The value of the XMP element.

Chapter 10: Enumerated Types

This section provides information on some of the enumerated types used by the Filter API.

- [Introduction](#) 175
- [KVCredKeyType](#) 176
- [KVErrorCode](#) 177
- [KVErrorCodeEx](#) 178
- [KVMetadataType](#) 182
- [KVMetaNameType](#) 183
- [KVSumInfoType](#) 184
- [KVSumType](#) 185
- [LPDF_DIRECTION](#) 188

Introduction

The enumerated types are in `adinfo.h`, `kverrorcodes.h`, `kvtypes.h`, `kv.h`, and `kvextract.h`. These header files are in the `include` directory. The first entry in an enumerated type structure should be set to zero (0). Each subsequent entry is increased by 1. For example, the first five entries of `KVCharSet` in `kvtypes.h` are:

```
KVCS_UNKNOWN
KVCS_SJIS
KVCS_GB
KVCS_BIG5
KVCS_KSC
```

They would be set in the following way:

Enumerated Type	Setting
<code>KVCS_UNKNOWN</code>	0
<code>KVCS_SJIS</code>	1
<code>KVCS_GB</code>	2
<code>KVCS_BIG5</code>	3
<code>KVCS_KSC</code>	4

You can also set many enumerated types by entering the appropriate symbolic constant, or `TRUE` or `FALSE`.

Programming Guidelines

When KeyView is enhanced in future releases, some enumerated types might be expanded. For example, new format IDs might be added to the `ENdocFmt` enumerated type, or new error codes might be added to the `KVErrorCodeEx` enumerated type. When you use these expandable types, your code should ensure binary compatibility with future releases.

For example, if you use an array to access error messages based on an error code, your code should check that the error code is less than `KVError_Last` before accessing the data. This ensures that new error codes are detected when you add KeyView binary files from new releases to your existing installation.

The following enumerated types are expandable:

`KVErrorCodeEx`

`KVMetadataType`

`KVCharSet`

`KVLanguageID`

`KVSubfileType`

`ENdocFmt`

KVCredKeyType

This enumerated type defines the type of credential used to open a protected file. See [KVCredentialComponent](#), on page 106. This enumerated type is defined in `kvextract.h`.

Definition

```
typedef enum tag_KVCredKeyType
{
    KVCredKeyType_UserName,
    KVCredKeyType_UserIdFile,
    KVCredKeyType_Password,
}
KVCredKeyType;
```

Enumerators

`KVCredKeyType_UserName` The credential in `KVCredentialComponent` is a user name.

`KVCredKeyType_UserIdFile` The credential in `KVCredentialComponent` is a path to a file that contains user IDs.

KVCredKeyType_Password The credential in KVCredentialComponent is a password.

KVErrorCode

This enumerated type defines the type of error generated if Filter fails. This enumerated type is defined in `kverrorcodes.h`.

Definition

```
typedef enum tag_KVErrorCode
{
KVERR_Success,                    /* 0 Success*/
KVERR_DLLNotFound,               /* 1 DLL or shared library not found*/
KVERR_OutOfCore,                 /* 2 memory allocation failure*/
KVERR_processCancelled,         /* 3 fpContinue() returns FALSE*/
KVERR_badInputStream,           /* 4 Invalid/corrupt input stream*/
KVERR_badOutputType,           /* 5 Invalid output type requested*/
KVERR_General,                   /* 6 General error... */
KVERR_FormatNotSupported,       /* 7 Format not supported*/
KVERR_PasswordProtected,       /* 8 File is Password Protected*/
KVERR_ADSSNotFound,             /* 9 Adobe Document Server not found*/
KVERR_AutoDetFail,              /* 10 Autodetect error*/
KVERR_AutoDetNoFormat,         /* 11 Unable to detect file format*/
KVERR_ReaderInitError,         /* 12 Error initializing the reader*/
KVERR_NoReader,                 /* 13 No reader available for this format*/
KVERR_CreateOutputFileFailed,   /* 14 Unable to create output file*/
KVERR_CreateTempFileFailed,    /* 15 Unable to create temp file*/
KVERR_ErrorWritingToOutputFile, /* 16 Error writing to output file*/
KVERR_CreateProcessFailed,     /* 17 Error creating a child process*/
KVERR_WaitForChildFailed,      /* 18 Wait for child process failed*/
KVERR_ChildTimeout,            /* 19 Child process hung / timed out*/
KVERR_ArchiveFileNotFound,     /* 20 Attempt to extract nonexistent file*/
KVERR_ArchiveFatalError       /* 21 Fatal error processing archive - should abort*/
}
KVErrorCode;
```

Enumerators

KVERR_SUCCESS	The function completed successfully.
KVERR_DLLNotFound	A DLL or shared library was not found.
KVERR_OutOfCore	Memory allocation failure.
KVERR_processCancelled	The callback function <code>fpContinue()</code> returns <code>FALSE</code> .

KVERR_badInputStream	Invalid or corrupt input stream.
KVERR_badOutputType	Invalid output is requested.
KVERR_General	General error. To return a more detailed message for KVERR_General, call fpGetKvErrorCodeEx() .
KVERR_FormatNotSupported	The file format is not supported.
KVERR_PasswordProtected	The file is encrypted or password-protected. KeyView supports only secure PST files.
KVERR_ADSNotFound	Adobe Document Server not found. This error is obsolete.
KVERR_AutoDetFail	Autodetect error.
KVERR_AutoDetNoFormat	Unable to detect file format.
KVERR_ReaderInitError	Error initializing the reader.
KVERR_NoReader	No reader is available for this format.
KVERR_CreateOutputFileFailed	Unable to create output file. This error is generated if the overwrite flag in KVExtractSubFileArg is FALSE, and a subfile has the same name as a file in the target path.
KVERR_CreateTempFileFailed	Unable to create temporary file.
KVERR_ErrorWritingToOutputFile	There was an error writing to the output file.
KVERR_CreateProcessFailed	There was an error creating a child process.
KVERR_WaitForChildFailed	The wait for child process failed.
KVERR_ChildTimeOut	The child process hung or timed out.
KVERR_ArchiveFileNotFound	Attempt to extract nonexistent file.
KVERR_ArchiveFatalError	A fatal error occurred processing an archive file.

KVErrorCodeEx

This enumerated type defines extended error codes. The type is defined in `kverrorcodes.h`.

Some of these error codes provide more information when `fpFilterFile()` or `fpFilterStream()` returns the error `KVERR_General`. To return these error codes, call [fpGetKvErrorCodeEx\(\)](#).

Definition

```
typedef enum tag_KVErrorCodeEx
{
KVError_OpenStreamFailure = KVERR_ArchiveFatalError + 1, /* 22 KVOpen stream
failure */
KVError_InterfaceFunctionNotFound, /* 23 Interface function not found */
KVError_InputFileNotFound, /* 24 Cannot find input file*/
KVError_OpenOutputFileFailed, /* 25 Cannot open output file*/
KVError_MemoryLeak, /* 26 Memory leak*/
KVError_MemoryOverwrite, /* 27 Memory overwrite*/
KVError_GPF, /* 28 Exception during oop filtering*/
KVError_OopCore, /* 29 Core dump in child process*/
KVError_KVoopLogFailed, /* 30 Creation of oop error log failed*/
KVError_OverNestedFileLimit, /* 31 File exceeds nested file limit*/
KVError_PSTAccessFailed, /* 32 Access failed on PST files*/
KVError_PasswordRequired, /* 33 Password required to access file*/
KVError_InvalidArgs /* 34 Input argument/structure is invalid*/
KVError_ReaderUsageDenied, /* 35 Reader requires a valid license*/
KVError_OopBadConfig, /* 36 Config buffer data was incomplete*/
KVError_OopBrokenPipe, /* 37 Read/write to/from pipe failed*/
KVError_OopPipeOEF, /* 38 Pipe was closed prior to read/write*/
KVError_IPCTimeOut, /* 39 Pipe/socket timed out on poll/select*/
KVError_InvalidOopDriverSignature, /* 40 Client sent request to OOP server but
context driver does not exist on the server*/
KVError_InvalidOopServiceSignature, /* 41 Client sent request to OOP service that
does not exist*/
KVError_ZeroFile, /* 42 Input file is empty or zero bytes */
KVError_CompressionNotSupported /* 43 File or subfile is compressed with
unsupported method */KVError_NoTemplates /* 44 No templates found (nsfsr) */
KVError_NoMainTemplate /* 45 No main template found (nsfsr) */
KVError_InvalidTemplate /* 46 Invalid template (nsfsr) */
KVError_TemplateError /* 47 Template error (nsfsr) */
KVError_IsADirectory /* 48 A directory exists at the given pathname */
KVError_Last /* 49 */
}
KVErrorCodeEx;
```

Enumerators

KVError_OpenStreamFailure = KVERR_ArchiveFatalError +1	Failed to open a stream during out-of-process filtering. This is an extended error for the KVERR_General code.
KVError_ InterfaceFunctionNotFound	An interface function was not found during out-of-process filtering. This is an extended error for the KVERR_General code.

KVError_InputFileNotFound	Could not find the input file during out-of-process filtering. This is an extended error for the KVERR_General code.
KVError_OpenOutputFileFailed	Could not open the output file during out-of-process filtering. This is an extended error for the KVERR_General code.
KVError_MemoryLeak	A memory leak occurred during out-of-process filtering. This is an extended error for the KVERR_General code.
KVError_MemoryOverwrite	A memory overwrite occurred during out-of-process filtering. This is an extended error for the KVERR_General code.
KVError_GPF	An exception occurred during out-of-process filtering. This is an extended error for the KVERR_General code.
KVError_OopCore	A memory dump was generated in a child process during out-of-process filtering. This is an extended error for the KVERR_General code.
KVError_KVoopLogFailed	The creation of the out-of-process error log failed. This is an extended error for the KVERR_General code.
KVError_OverNestedFileLimit	The container file has more than the allowable number of child documents. One or more child documents were not converted. Currently, this enumerator is not used.
KVError_PSTAccessFailed	<p>The PST file could not be converted. This error might be returned when a call to <code>fpOpenFile()</code> returns <code>NULL</code> for one of the following reasons:</p> <ul style="list-style-type: none"> • A Microsoft Outlook client is not installed. • A Microsoft Outlook client is installed, but is not the default email client. • A Microsoft Outlook client is installed, but is not configured correctly. • The PST file is corrupt. • The PST file is read-only (PST files must allow read and write access). • The MAPI call fails. • The bit editions of Microsoft Outlook do not match the bit editions of the KeyView software. <p>For example, if 32-bit KeyView is used, 32-bit Outlook must be installed. If 64-bit KeyView is used, 64-bit Outlook must be installed.</p>
KVError_PasswordRequired	To open the file, you must provide credentials. This error might be returned when a call to <code>fpOpenFile()</code> returns <code>NULL</code> .

KVError_InvalidArgs	The input argument or structure is invalid. This error is generated by the File Extraction APIs.
KVError_ReaderUsageDenied	The current license key does not enable the document reader required to filter the file. This error might be returned when a call to <code>fpOpenFile()</code> returns <code>NULL</code> . Some document readers are considered advanced features and are licensed separately from the KeyView SDK (for example, the PST and MBX readers). Contact your Micro Focus sales representative to get an updated license key.
KVError_OopBadConfig	Information in the <code>kvxconfig.ini</code> file is incomplete and cannot be used to filter the XML file.
KVError_OopBrokenPipe	Data was not transferred between the parent and child processes during out-of-process filtering because either the parent or child failed.
KVError_OopPipeOEF	Data was not transferred between the parent and child processes during out-of-process filtering because the parent process was shut down.
KVError_IPCTimeOut	Either the parent or child process is waiting for a reply or request during out-of-process filtering.
KVError_InvalidOopDriverSignature	A client sent a request to an out-of-process server, but the context driver does not exist on the server.
KVError_InvalidOopServiceSignature	A client sent a request to a File Extraction service that does not exist. If this error is generated on the call to <code>fpClose()</code> , you can ignore it.
KVError_ZeroFile	The input file is empty or zero bytes.
KVError_CompressionNotSupported	The file or subfile is compressed with an unsupported compression method.
KVError_NoTemplates	
KVError_NoMainTemplate	
KVError_InvalidTemplate	
KVError_TemplateError	
KVError_IsADirectory	
KVError_Last	

Discussion

- When error reporting is enhanced in future releases, new error messages might be added to this enumerator type. When you use this type, your code must ensure binary compatibility with future releases. See [Programming Guidelines, on page 176](#).
- If an extended error code is called for a format to which the error does not apply, the `KVError_Last` code is returned.

KVMetadataType

This enumerated type defines the data type of metadata that can be extracted from a subfile in a mail message or mail store. If a metadata field has a corresponding KeyView type in `KVMetadataType`, the metadata is converted to the `KVMetadataElem` structure, and the structure member `isValid` is 1. This enumerated type is defined in `kvtypes.h`.

Definition

```
typedef enum
{
    KVMetadata_Unknown      = 0,
    KVMetadata_Bool        = 1,
    KVMetadata_Binary       = 2,
    KVMetadata_Int4         = 3,
    KVMetadata_UInt4        = 4,
    KVMetadata_Int8         = 5,
    KVMetadata_UInt8        = 6,
    KVMetadata_String       = 7,
    KVMetadata_Unicode      = 8,
    KVMetadata_DateTime     = 9,
    KVMetadata_Float        = 10,
    KVMetadata_Double       = 11,
    KVMetadata_Last
}
KVMetadataType;
```

Enumerators

<code>KVMetadata_Unknown</code>	The value in the property is of an unknown type.
<code>KVMetadata_Bool</code>	The value in the property is a Boolean value. The corresponding MAPI type is <code>PT_BOOLEAN</code> .
<code>KVMetadata_Binary</code>	The value in the property is a byte array. The corresponding MAPI type is <code>PT_BINARY</code> .

KVMetadata_ Int4	The value in the property is a signed 4-byte integer. The corresponding MAPI types are PT_I2, PT_SHORT, PT_I4, and PT_LONG.
KVMetadata_ UInt4	The value in the property is an unsigned 4-byte integer. This type is not currently supported.
KVMetadata_ Int8	The value in the property is a signed 8-byte integer. This type is not currently supported.
KVMetadata_ UInt8	The value in the property is an unsigned 8-byte integer. This type is not currently supported.
KVMetadata_ String	The value in the property is a string. The corresponding MAPI type is PT_STRING8.
KVMetadata_ Unicode	The value in the property is a Unicode string. The corresponding MAPI type is PT_UNICODE.
KVMetadata_ DateTime	The value in the property is a date and time. The corresponding MAPI type is PT_SYSTIME.
KVMetadata_ Float	The value in the property is a 4-byte float. The corresponding MAPI type is PT_FLOAT.
KVMetadata_ Double	The value in the property is an 8-byte double. The corresponding MAPI type is PT_DOUBLE.

Discussion

New types might be added to this enumerated type. When you use this type, your code should ensure binary compatibility with future releases. See [Programming Guidelines, on page 176](#).

KVMetaNameType

This enumerated type defines the type of metadata fields extracted from a subfile in a mail message or mail store. See [KVMetaName, on page 113](#). This enumerated type is defined in `kvextract.h`.

Definition

```
typedef enum
{
    KVMetaNameType_Integer = 0,
    KVMetaNameType_String  = 1
}
KVMetaNameType;
```

Enumerators

`KVMetaNameType_Integer` The metadata field is an integer.

`KVMetaNameType_String` The metadata field is a string.

KVSumInfoType

This enumerated type defines the data type of the metadata field extracted from a document. This enumerated type is defined in `kvtypes.h`.

Definition

```
typedef enum tag_KVSumInfoType
{
    KV_String          = 0x1,
    KV_Int4            = 0x2,
    KV_DateTime       = 0x3,
    KV_ClipBoard      = 0x4,
    KV_Bool           = 0x5,
    KV_Unicode        = 0x6,
    KV_IEEE8          = 0x7,
    KV_Other          = 0x8
}
KVSumInfoType;
```

Enumerators

`KV_String` The value in the metadata field is a string.

`KV_Int4` The value in the metadata field is an integer.

`KV_DateTime` The value in the metadata field is a date and time. This type is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (Windows FILETIME EPOCH). You might need to convert this value into another format.

The Filter sample program demonstrates how to convert this value to another format. The program translates `KV_DATETIME` to a UNIX timestamp, that is, the number of seconds since 00:00:00 (UTC), January 1, 1970. It then uses the `ctime` system library call, which works on UNIX and Windows, to print the date in the following format:

```
Thu Aug 22 16:19:07 2002
```

`KV_ClipBoard` Currently not supported.

KV_Boo1	The value in the metadata field is a Boolean value.
KV_Unicode	The value in the metadata field is a Unicode string.
KV_IEEE8	The value in the metadata field is an IEEE 8-byte integer.
KV_Other	The value in the metadata field is user-defined.

KVSumType

This enumerated type defines the metadata fields that can be extracted from a document. This enumerated type is defined in `kvtypes.h`.

- Types 0 to 34 and type 42 are Office summary fields.
- Types 35 to 40 are computer-aided design (CAD) metadata fields.
- Type 41, `KV_OrigAppVersion`, is shared by Office software and CAD.

Types 43 or greater are reserved for any non-standard metadata field defined in a document.

Definition

```
typedef enum tag_KVSumType
```

```
    KV_CodePage          = 0,  
    KV_Title             = 1,  
    KV_Subject           = 2,  
    KV_Author            = 3,  
    KV_Keywords          = 4,  
    KV_Comments          = 5,  
    KV_Template          = 6,  
    KV_LastAuthor        = 7,  
    KV_RevNumber         = 8,  
    KV_EditTime          = 9,  
    KV_LastPrinted       = 10,  
    KV_Create_DTM        = 11,  
    KV_LastSave_DTM      = 12,  
    KV_PageCount         = 13,  
    KV_WordCount         = 14,  
    KV_CharCount         = 15,  
    KV_ThumbNail         = 16,  
    KV_AppName           = 17,  
    KV_Security          = 18,  
    KV_Category          = 19,  
    KV_PresentationTarget = 20,  
    KV_Bytes              = 21,  
    KV_Lines              = 22,  
    KV_Paragraphs        = 23,
```

```
KV_Slides           = 24,  
KV_Notes           = 25,  
KV_HiddenSlides    = 26,  
KV_MMClips         = 27,  
KV_ScaleCrop       = 28,  
KV_HeadingPairs    = 29,  
KV_TitlesofParts   = 30,  
KV_Manager         = 31,  
KV_Company         = 32,  
KV_LinksUpToDate   = 33,  
KV_HyperlinkBase   = 34,  
KV_Layouts         = 35,  
KV_Objects         = 36,  
KV_FileVersion     = 37,  
KV_LastFileVersion = 38,  
KV_OrigFileVersion = 39,  
KV_OrigFileType    = 40,  
KV_OrigAppVersion  = 41,  
KV_ContentStatus   = 42,  
KV_UserDefined     = 43  
}  
KVSumType;
```

Enumerators

KV_CodePage	The code page of the document.
KV_Title	The contents of the "Title" property field taken from the source document.
KV_Subject	The contents of the "Subject" property field taken from the source document.
KV_Author	The contents of the "Author" property field taken from the source document.
KV_Keywords	The contents of the "Keywords" property field taken from the source document.
KV_Comments	The contents of the "Comments" property field taken from the source document.
KV_Template	The contents of the "Template" property field taken from the source document.
KV_LastSavedby	The contents of the "Last saved by" property field taken from the source document.
KV_RevNumber	The contents of the "Revision number" property field taken from the source document.
KV_EditTime	The contents of the "Total editing time" property field taken from the source document.

KV_LastPrinted	The contents of the "Printed" property field taken from the source document.
KV_Create_DTM	The contents of the "Created" property field taken from the source document.
KV_LastSave_DTM	The contents of the "Modified" property field taken from the source document.
KV_PageCount	The contents of the "Pages" property field taken from the source document. The field provides the number of pages in the document.
KV_WordCount	The contents of the "Words" property field taken from the source document. The field provides the number of words in the document.
KV_CharCount	The contents of the "Characters" property field taken from the source document. The field provides the number of characters in the document.
KV_ThumbNail	A thumbnail image of a document.
KV_AppName	The contents of the "Type" property field taken from the source document. This field identifies the application used to read the document.
KV_Security	The contents of the "Attributes" property field taken from the source document.
KV_Category	The contents of the "Category" property field taken from the source document.
KV_PresentationTarget	The target format for presentations (35mm, printer, video, and so on).
KV_Bytes	The contents of the "Size" property field taken from the source document. The field provides the size of the file in bytes.
KV_Lines	The contents of the "Lines" property field taken from the source document. The field provides the number of lines in the document.
KV_Paragraphs	The contents of the "Paragraphs" property field taken from the source document. The field provides the number of paragraphs in the document.
KV_Slides	The contents of the "Slides" property field taken from a presentation document. The field provides the number of slides in the document.
KV_Notes	The contents of the "Notes" property field taken from a presentation document. The field provides the number of notes in the document.
KV_HiddenSlides	The contents of the "Hidden slides" property field taken from a presentation document. The field provides the number of hidden slides in the document.
KV_MMClips	The contents of the "Multimedia clips" property field taken from a presentation document. The field provides the number of multimedia clips in the document.
KV_ScaleCrop	A Boolean value that specifies whether thumbnails are cropped or scaled.

KV_HeadingPairs	An internally-used property indicating the grouping of different document parts and the number of items in each group.
KV_TitlesofParts	The contents of the "Document Contents" property field taken from the source document. The field contains a list of the parts of the file, such as the names of macro sheets in Microsoft Excel or the headings in Word.
KV_Manager	The contents of the "Manager" property field taken from the source document.
KV_Company	The contents of the "Company" property field taken from the source document.
KV_LinksUpToDate	A Boolean value that specifies whether links in the document are resolved and current.
KV_HyperlinkBase	The base address used for all relative links in the file.
KV_Layouts	The number of layouts in the AutoCAD drawing.
KV_Objects	The approximate number of objects in the AutoCAD drawing.
KV_FileVersion	The AutoCAD version (for example, R13, R14) of the drawing.
KV_LastFileVersion	The AutoCAD version (for example, R13, R14) that the AutoCAD drawing was last saved as.
KV_OrigFileVersion	The AutoCAD version (for example, R13, R14) of the original source file.
KV_OrigFileType	The AutoCAD file type (for example, DWG, DXF, or DWB) of the original source file.
KV_OrigAppVersion	The AutoCAD version (for example, R13, R14) of the application that created the original source file.
KV_ContentStatus	The status of the content, for example <i>Draft</i> , <i>Reviewed</i> , or <i>Final</i> .
KV_UserDefined	The contents of the first entry in the array of non-standard metadata. This could be user-defined metadata, or metadata unique to a file type.

LPDF_DIRECTION

This enumerated type defines the paragraph direction of extracted paragraphs from a PDF file when logical order is enabled. This enumerated type is defined in `kvtypes.h`.

Definition

```
typedef enum{
    LPDF_RAW = 0,
    LPDF_LTR,
    LPDF_RTL,
```

```
    LPDF_AUTO  
} LPDF_DIRECTION ;
```

Enumerators

LPDF_ RAW Unstructured paragraph flow. This is the default behavior.

LPDF_ LTR Logical reading order and left-to-right paragraph direction.

LPDF_ RTL Logical reading order and right-to-left paragraph direction.

LPDF_ AUTO Logical reading order. The PDF reader determines the paragraph direction for each PDF page, and then sets the direction accordingly. This is the default when logical order is enabled.

Appendixes

This section lists supported formats, supported character sets, and redistributed files, and provides information on format detection and developing a custom document reader.

Appendix A: Supported Formats

This section lists the file formats that KeyView can process (either filter, convert, or display).

- [Supported Formats](#) 191

Supported Formats

The tables in this section provide the following information:

- The file formats supported by the Filter API, Export API, Viewing API, and File Extraction API. The supported versions and the format's extension are also listed. All of the formats listed in this section can be detected by the KeyView format detection module (kwad). For a complete list of formats that can be detected, see [Detected Formats, on page 221](#).
- The file formats for which KeyView can detect and extract the character set and metadata information (properties such as title, author, and subject).

Even though a file format might be able to provide character set information, some documents might not contain character set information. Therefore, the document reader would not be able to determine the character set of the document. In this case, either the operating system code page or the character set specified in the API is used.

- The document reader used to filter each format.

Key to Support Tables

Symbol	Description
Y	The format is supported. You can extract metadata for this format. You can determine the character set for this format.
N	The format is not supported. You cannot extract metadata for this format. You cannot determine the character set for this format.
P	Partial metadata is extracted from this format. Some non-standard fields are not extracted.
T	Only text is extracted from this format. Formatting information is not extracted.
M	Only metadata (title, subject, author, and so on) is extracted from this format. Text and formatting information are not extracted.

Archive Formats

Supported Archive Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
7-Zip	4.57	z7zsr, multiarcsr ¹	7Z	N	N	Y	Y	N	n/a	N
AD1	n/a	ad1sr	AD1	N	N	Y	Y	N	n/a	N
ARJ	n/a	multiarcsr	ARJ	N	N	N	Y	N	n/a	N
B1	n/a	b1sr	B1	N	N	Y	Y	N	n/a	N
BinHex	n/a	kvhqxsr	HQX	N	N	Y	Y	N	n/a	N
Bzip2	n/a	bzip2sr	BZ2	N	N	Y	Y	N	n/a	N
CPIO (copy-in-and-out archiver)	n/a	multiarcsr		N	N	N	Y	N	n/a	N
Debian binary package	n/a	multiarcsr	DEB	N	N	N	Y	N	n/a	N
DOS/Windows Object Library	n/a	multiarcsr	LIB, A	N	N	N	Y	N	n/a	N
Expert Witness Compression Format (EnCase)	6	encasesr	E01, L01	N	N	Y	Y	N	n/a	N
	7	encase2sr	Lx01	N	N	Y	Y	N	n/a	N

¹7zip is supported with the multiarcsr reader on some platforms for Extract.

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
GZIP	2	kvgzsr	GZ	N	N	N	Y	N	n/a	N
		kvgz	GZ	N	N	Y	N	N	n/a	N
ISO	n/a	isosr	ISO	N	N	Y	Y	N	n/a	N
Java Archive	n/a	unzip	JAR	N	N	Y	Y	N	n/a	N
Legato EMailXtender Archive	n/a	emxsr	EMX	N	N	Y	Y	N	n/a	N
LZMA compressed data	n/a	multiarcsr	LZMA	N	N	N	Y	N	n/a	N
MacBinary	n/a	macbinsr	BIN	N	N	Y	Y	N	n/a	N
Mac Disk Copy Disk Image	n/a	dmgsr	DMG	N	N	Y	Y	N	n/a	N
Mac OS-X (Mach-O) executable	n/a	multiarcsr		N	N	N	Y	N	n/a	N
Microsoft Backup File	n/a	bkfsr	BKF	N	N	Y	Y	N	n/a	N
Microsoft Cabinet format	1.3	cabsr	CAB	N	N	Y	Y	N	n/a	N
Microsoft Compiled HTML Help	3	chmsr	CHM	N	N	Y	Y	N	n/a	N
Microsoft Compressed Folder	n/a	lzhsr	LZH LHA	N	N	N	Y	N	n/a	N
Microsoft Power BI Desktop format	n/a	unzip	PBIX	N	N	N	Y	N	n/a	N
MSI (Microsoft Installer)	n/a	multiarcsr	MSI	N	N	N	Y	N	n/a	N

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
PKZIP	through 9.0	unzip	ZIP	N	N	Y	Y	N	n/a	N
RAR archive	2.0 through 3.5	rarsr	RAR	N	N	N	Y	N	n/a	N
RAR5 archive	5	multiarcsr	RAR5	N	N	N	Y	N	n/a	N
RPM (package manager file)	n/a	multiarcsr	RPM	N	N	N	Y	N	n/a	N
SUN PEX Binary Archive	n/a	multiarcsr		N	N	Y	Y	N	n/a	N
Tableau Packaged Data Source format	n/a	unzip	TDSX	N	N	N	Y	N	n/a	N
Tableau Packaged Workbook format	n/a	unzip	TWBX	N	N	N	Y	N	n/a	N
Tape Archive	n/a	tarsr	TAR	N	N	Y	Y	N	n/a	N
UNIX Compress	n/a	kvzeesr	Z	N	N	N	Y	N	n/a	N
		kvzee	Z	N	N	Y	N	N	n/a	N
UUEncoding	all versions	uudsr	UUE	N	N	Y	Y	N	n/a	N
XZ	n/a	multiarcsr	XZ	N	N	N	Y	N	n/a	N
Windows Imaging Format	n/a	multiarcsr	WIM	N	N	N	Y	N	n/a	N

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Windows Scrap File	n/a	olesr	SHS	N	N	N	Y	N	n/a	N
WinZip	through 10	unzip	ZIP	N	N	Y	Y	N	n/a	N
XAR (Extensible Archive)	n/a	multiarcsr		N	N	N	Y	N	n/a	N
Zipped Keyhole Markup Language	n/a	unzip	ZIP	N	N	N	Y	N	n/a	N

Binary Format

Supported Binary Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Executable	n/a	exesr	EXE	N	N	Y	N	N	n/a	N
Link Library	n/a	exesr	DLL	N	N	Y	N	N	n/a	N

Computer-Aided Design Formats

Supported CAD Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
AutoCAD Drawing	R13, R14, R15/2000, 2004, 2007, 2010, 2013, 2018	kpODArdr kpDWGrdr ¹	DWG	Y	Y	Y	N	Y	Y	N
AutoCAD Drawing Exchange	R13, R14, R15/2000, 2004, 2007, 2010, 2013	kpODArdr kpDXFrdr ²	DXF	Y	Y	Y	N	Y	Y	N
CATIA formats	5	kpCATrdr	CAT ³	Y	N	N	N	Y	N	N
Microsoft Visio	4, 5, 2000, 2002, 2003, 2007, 2010 ⁴	vsdsr	VSD	Y	Y	Y	Y ⁵	Y	Y	N
		kpVSD2rdr	VSD, VSS VST	Y	Y	Y	N	Y	Y	N

¹The kpODArdr reader can filter, export, and view all versions but is supported only on Windows, Linux, and macOS. The kpDWGrdr reader is used on AIX, FreeBSD, Solaris, and SPARC platforms, but does not support graphics for versions after 2004 or text for versions after 2013.

²The kpODArdr reader can filter, export, and view all versions but is supported only on Windows, Linux, and macOS. The kpDXFrdr reader is used on AIX, FreeBSD, Solaris, and SPARC platforms, but does not support graphics for versions after 2004.

³All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.

⁴Viewing and Export use the graphic reader, kpVSD2rdr for Microsoft Visio 2003, 2007, and 2010, and vsdsr for all earlier versions. Image fidelity in Viewing and Export is therefore only supported for versions 2003 and above. Filter uses the graphic reader kpVSD2rdr for Microsoft Visio 2003, 2007, and 2010, and vsdsr for all earlier versions.

⁵Extraction of embedded OLE objects is supported for Filter on Windows platforms only.

Supported CAD Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
	2013	ActiveX components	VSDM VSSM VSTM VSDX VSSX VSTX	N	N	Y ¹	N	Y	N	N
		kpVSDXrdr	VSDM VSSM VSTM VSDX VSSX VSTX	Y	Y	Y	Y	Y	Y	N
Unigraphics (UG) NX		kpUGrdr	PRT	Y	N	N	N	N	N	N

Database Formats

Supported Database Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
dBase Database	III+, IV	dbfsr	DBF	Y	Y	Y	N	N	N	N

¹Visio 2013 is supported in Viewing only, with the support of ActiveX components from the Microsoft Visio 2013 Viewer. Image fidelity is supported but other features, such as highlighting, are not.

Supported Database Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Access	95, 97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	mdbsr	MDB, ACCDB	Y	T	T	N	N	Y ¹	N
Microsoft Project	2000, 2002, 2003, 2007, 2010, 2013, 2016	mppsr	MPP	Y	Y	Y	Y	Y	Y	N

Desktop Publishing

Supported Desktop Publishing Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Publisher	98 to 2016	mspubsr	PUB	Y	T	T	Y	Y	Y	N

Display Formats

Supported Display Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe PDF	1.1 to 1.7, 2.0	pdfsr	PDF	Y	Y	N	Y ²	Y	Y	N
		pdf2sr	PDF	N	Y	N	N	N	N	N

¹Charset is not supported for Microsoft Access 95 or 97.

²Includes support for extraction of subfiles from PDF Portfolio documents.

Supported Display Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
		kppdfldr	PDF	N	Y	Y	N	N	N	N
		kppdf2rdr ¹	PDF	N	N	Y	N	N	N	N

Graphic Formats

Supported Graphic Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Computer Graphics Metafile	n/a	kpcgmrdr ²	CGM	Y	Y	Y	N	N	N	N
CorelDRAW ³	through 9.0 10, 11, 12, X3	kpcdrdr	CDR	N	Y	Y	N	N	N	N
DCX Fax System	n/a	kpcdxrdr	DCX	N	Y	Y	N	N	N	N
Digital Imaging & Communications in	n/a	dcmsr	DCM	M	N	N	N	Y	N	N

¹kppdf2rdr is an alternate graphic-based reader that produces high-fidelity output but does not support other features such as highlighting or text searching.

²Files with non-partitioned data are supported.

³CDR/CDR with TIFF header.

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Medicine (DICOM)										
Encapsulated PostScript (raster)	TIFF header	kpepsrdr	EPS	N	Y	Y	N	N	N	N
Enhanced Metafile	n/a	kpemfrdr	EMF	Y	Y	Y	N	Y	N	N
GIF	87, 89	kpgifdr	GIF	N	Y	Y	N	N	N	N
		gifsr		M	M	N	N	Y	N	N
ISO-BMFF JPEG 2000 compound image	n/a	kpjp2000rdr	JPM	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
ISO-BMFF JPEG 2000 image	n/a	kpjp2000rdr	JP2	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
ISO-BMFF JPEG 2000 with extensions	n/a	kpjp2000rdr	JPX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
JBIG2	n/a	kpJBIG2rdr	JBIG2	N	Y	Y	N	N	N	N
JPEG	n/a	kpjpgdr	JPEG	N	Y	Y	N	N	N	N
		jpgsr		M	M	N	N	Y	N	N
JPEG 2000	n/a	kpjp2000rdr	JP2, JPF, J2K, JPWL, JPX, PGX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
JPEG 2000 PGX Verification Model image	n/a	kjpg2000rdr	PGX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
Lotus AMIDraw Graphics	n/a	kpsdwrdr	SDW	N	Y	Y	N	N	N	N
Lotus Pic	n/a	kppicrdr	PIC	Y	Y	Y	N	N	N	N
Macintosh Raster	2	kppctrdr	PIC PCT	N	Y	Y	N	N	N	N
MacPaint	n/a	kpmacrdr	PNTG	N	Y	Y	N	N	N	N
Microsoft Office Drawing	n/a	kpmsordr	MSO	N	Y	Y	N	N	N	N
Omni Graffiti	n/a	kpGFLrdr	GRAFFLE	Y	N	N	N	Y	Y	N
PC PaintBrush	3	kppcxrdr	PCX	N	Y	Y	N	N	N	N
Portable Network Graphics	n/a	kppngrdr	PNG	N	Y	Y	N	N	N	N
		pngsr	PNG	M	M	N	N	Y	N	N
Scalable Vector Graphics	n/a	xmlsr	SVG	Y	T	T	N	Y	Y	N
SGI RGB Image	n/a	kpsgirdr	RGB	N	Y	Y	N	N	N	N
Sun Raster Image	n/a	kpsunrdr	RS	N	Y	Y	N	N	N	N

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Tagged Image File	through 6.0 ¹	tifsr	TIFF	M	M	N	N	Y	N	N
		kptifrd	TIFF	N	Y	Y	N	N	N	N
Truevision Targa	2	kpTGArdr	TGA	N	Y	Y	N	N	N	N
Windows Animated Cursor	n/a	kpanirdr	ANI	N	Y	Y	N	N	N	N
Windows Bitmap	n/a	kpbmprdr	BMP	N	Y	Y	N	N	N	N
		bmpsr	BMP	M	M	N	N	Y	N	N
Windows Icon Cursor	n/a	kpicordr	ICO	N	Y	Y	N	N	N	N
Windows Metafile	3	kpwmfrdr	WMF	Y ²	Y	Y	N	N	N	N
WordPerfect Graphics 1	1	kpwpgrdr	WPG	N	Y	Y	N	N	N	N
WordPerfect Graphics 2	2, 7	kpwg2rdr	WPG	N	Y	Y	N	N	N	N

¹The following compression types are supported: no compression, CCITT Group 3 1-Dimensional Modified Huffman, CCITT Group 3 T4 1-Dimensional, CCITT Group 4 T6, LZW, JPEG (only Gray, RGB and CMYK color space are supported), and PackBits.

²Windows Metafiles can contain both raster images (KeyView file class 4) and vector graphics (KeyView file class 5). Filtering is supported only for vector graphics (class 5).

Mail Formats

Supported Mail Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Documentum EMCMF	n/a	msgsr	EMCMF	N	N	Y	Y	Y	Y	N
Domino XML Language ¹	n/a	dxlsr	DXL	N	N	Y	Y	Y	N	N
GroupWise FileSurf	n/a	gwfssr	GWFS	N	N	Y	Y	Y	N	N
Legato Extender	n/a	onmsr	ONM	N	N	Y	Y	Y	N	N
Lotus Notes database	4, 5, 6.0, 6.5, 7.0, 8.0	nsfsr	NSF	N	N	Y	Y	Y	N	N
Mailbox ²	Thunderbird 1.0, Eudora 6.2	mbxsr ³	MBX	N	N	T	Y	Y	Y	N
Microsoft	2004	entsr	various	N	N	Y	Y	Y	Y	N

¹Supports non-encrypted embedded files only.

²KeyView supports MBX files created by Eudora Email and Mozilla Thunderbird. MBX files created by other common mail applications are typically filtered, converted, and displayed.

³This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

Supported Mail Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Entourage Database										
Microsoft Outlook	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016, 2019	msgsr ¹	MSG, OFT	Y	T	T	Y	Y	Y ²	N
Microsoft Outlook DBX	5.0, 6.0	dbxsr	DBX	N	N	Y	Y	Y	Y	N
Microsoft Outlook Express	Windows 6 MacIntosh 5	emlsr ³	EML	Y	T	T	Y	Y	Y	N
		mbxsr ⁴	EML	N	N	T	Y	Y	Y	N
Microsoft Outlook iCalendar	1.0, 2.0	icssr	ICS, VCS	N	N	Y	Y	Y	Y	N
Microsoft Outlook for Macintosh	2011	olmsr	OLM	N	N	Y	Y	N	Y	N
Microsoft Outlook Offline Storage File	97, 2000, 2002, 2003, 2007, 2010, 2013	pffsr ⁵	OST	N	N	Y	Y	Y	Y	N

¹This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

²Returns "Unicode" character set for version 2003 and up, and "Unknown" character set for previous versions.

³This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

⁴This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

⁵The reader pffsr is available only on Windows and Linux.

Supported Mail Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Outlook Personal Folder ¹	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016, 2019	pstsr ²	PST	N	N	Y	Y	Y	N	N
	97, 2000, 2002, 2003, 2007, 2010, 2013	pstnsr	PST	N	N	Y	Y	Y	Y	N
	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016, 2019	pstxsr	PST	N	N	Y	Y	Y	Y	N
Microsoft Outlook vCard Contact	2.1, 3.0, 4.0	vcfsr	VCF	Y	Y	T	N	Y	N	N
Text Mail (MIME)	n/a	emlsr ³	various	Y	T	T	Y	Y	Y	N
		mbxsr ⁴	various	Y	T	T	Y	Y	Y	N
Transport Neutral Encapsulation Format	n/a	tnfsr	various	N	N	Y	Y	Y	Y	N

¹KeyView provides several readers capable of processing PST files. The `pstsr` reader uses the Microsoft Messaging Application Programming Interface (MAPI), works only on Windows, and requires that you have Microsoft Outlook installed. The `pstxsr` reader is available for Windows (32-bit and 64-bit) and Linux (64-bit only) and does not require Microsoft Outlook. The `pstnsr` reader is an alternative reader that does not require Microsoft Outlook, for all platforms not supported by `pstxsr`. For more information about these readers, see "Extract Subfiles from Outlook Personal Folders Files" in Chapter 3.

²This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

³This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

⁴This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

Multimedia Formats

Viewing SDK plays some multimedia files using the Windows Media Control Interface (MCI). MCI is a set of Windows APIs that communicate with multimedia devices.

Supported Multimedia Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
3GPP video file	n/a	mpeg4sr	3GP	M	N	N	N	Y	N	N
3GPP2 video file	n/a	mpeg4sr	3G2	M	N	N	N	Y	N	N
Adobe Flash Player audio	n/a	mpeg4sr	F4A	M	N	N	N	Y	N	N
Adobe Flash Player audio book	n/a	mpeg4sr	F4B	M	N	N	N	Y	N	N
Adobe Flash Player protected video	n/a	mpeg4sr	F4P	M	N	N	N	Y	N	N
Adobe Flash Player video	n/a	mpeg4sr	F4V	M	N	N	N	Y	N	N
Apple ISO-BMFF QuickTime video	n/a	MCI	QT MOV	N	N	Y	N	N	N	N
Apple MPEG-4 Part 14 audio	n/a	mpeg4sr	M4A	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14 audio book	n/a	mpeg4sr	M4B	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14 protected audio	n/a	mpeg4sr	M4P	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14	n/a	mpeg4sr	M4V	M	N	N	N	Y	N	N

Supported Multimedia Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
video										
Audible Enhanced Audiobook	n/a	mpeg4sr	AAX	M	N	N	N	Y	N	N
KDDI video file	n/a	MCI		N	N	Y	N	N	N	N
Advanced Systems Format	1.2	asfsr	ASF WMA WMV	N	N	N	N	Y	N	N
Audio Interchange File Format	n/a	MCI	AIFF	N	N	Y	N	N	N	N
		aifsr	AIFF	M	N	N	N	Y	N	N
ISO-BMFF MPEG-4 with AVC extension	n/a	mpeg4sr		M	N	N	N	Y	N	N
Microsoft Wave Sound	n/a	MCI	WAV	N	N	Y	N	N	N	N
		rifsr	WAV	M	N	N	N	Y	N	N
MIDI	n/a	MCI	MID	N	N	Y	N	N	N	N
Mobile QuickTime video	n/a	mpeg4sr	MQV	M	N	N	N	Y	N	N
Motion JPEG 2000	n/a	kpjp2000rdr	MJ2 MJP2	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
MPEG-1 Audio layer 3	ID3 v1 and v2	MCI	MP3	N	N	Y	N	N	N	N
		mp3sr	MP3	M	M	Y	N	Y	N	N

Supported Multimedia Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
MPEG-1 Video	2, 3	MCI	MPG	N	N	Y	N	N	N	N
MPEG-2 Audio	n/a	MCI	MPEGA	N	N	Y	N	N	N	N
MPEG-21	n/a	mpeg4sr		M	N	N	N	Y	N	N
MPEG-4 Audio	n/a	mpeg4sr	MP4 3GP	M	N	N	N	Y	N	N
Nero AAC audio	n/a	mpeg4sr		M	N	N	N	Y	N	N
Nero MPEG-4 profile	n/a	mpeg4sr		M	N	N	N	Y	N	N
Nero MPEG-4 profile with AVC extension	n/a	mpeg4sr		M	N	N	N	Y	N	N
NeXT/Sun Audio	n/a	MCI	AU	N	N	Y	N	N	N	N
NTT MPEG-4	n/a	mpeg4sr		M	N	N	N	Y	N	N
QuickTime Movie	2, 3, 4	MCI	QT MOV	N	N	Y	N	N	N	N
Sony PSP MPEG-4	n/a	mpeg4sr	MP4	M	N	N	N	Y	N	N
Sony XAVC video	n/a	mpeg4sr		M	N	N	N	Y	N	N
Windows Video	2.1	MCI	AVI	N	N	Y	N	N	N	N

NOTE: Depending on the default multimedia player installed on your computer, the View API might not be able to play some supported multimedia formats. To play multimedia files, the View API uses the Windows Media Control Interface (MCI) to communicate with the multimedia player installed on your computer. If the player does not play a multimedia file that is supported by the Viewing SDK, the View API cannot play the file.

If you cannot play a supported multimedia file by using the View API, install a different multimedia player or compressor/decompressor (codec) component.

Presentation Formats

Supported Presentation Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Keynote	2, 3, '08, '09	kpIWPGGrdr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16, '18 iCloud 2018	kpIWPG13rdr ¹	KEY	Y	T	N	N	N	N	N
Applix Presents	4.0, 4.2, 4.3, 4.4	kpagrdr	AG	Y	Y	Y	N	N	N	N
Corel Presentations	6, 7, 8, 9, 10, 11, 12, X3	kpshwrdr	SHW	Y	Y	Y	N	N	N	N
Extensible Forms Description Language	n/a	kpXFDLrdr	XFD XFDL	Y	Y	Y	N	Y	Y	N
Lotus Freelance Graphics	96, 97, 98, R9, 9.8	kprrzrdr	PRZ	Y	Y	Y	N	N	N	N
Lotus Freelance Graphics 2	2	kprrdr	PRE	Y	Y	Y	N	N	N	N

¹This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Macromedia Flash	through 8.0	swfsr	SWF	Y	Y	Y	N	N	Y ¹	N
Microsoft PowerPoint Macintosh	98	kpp40rdr	PPT	Y	Y	Y	N	N	N	N
	2001, v.X, 2004	kpp97rdr	PPT PPS POT	Y	Y	Y	N	P	Y	N
Microsoft PowerPoint PC	4	kpp40rdr	PPT	Y	Y	Y	N	P	N	N
Microsoft PowerPoint Windows	95	kpp95rdr	PPT	Y	Y	Y	N	P	Y	N
Microsoft PowerPoint Windows	97, 2000, 2002, 2003	kpp97rdr	PPT PPS POT	Y	Y	Y	Y	P	Y	Y ²
Microsoft PowerPoint Windows XML	2007, 2010, 2013, 2016, 2019	kpppxrdr	PPTX PPTM POTX POTM PPSX PPSM PPAM	Y	Y	Y	Y	Y	Y	Y

¹The character set cannot be determined for versions 5.x and lower.

²Slide footers are supported for Microsoft PowerPoint 97 and 2003.

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
OASIS Open Document Format	1, 2 ¹	kpodfrdr	SXD SXI ODG ODP	Y	Y	Y	Y ²	Y	Y	N
OpenOffice Impress, LibreOffice Impress	1 to 5	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N
StarOffice Impress	3, 4, 5	kpsddrdr	SDA SDD	Y	T	N	N	N	N	N
	6, 7, 8, 9	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N

¹Generated by OpenOffice Impress 2.0, StarOffice 8 Impress, and IBM Lotus Symphony Presentation 3.0.

²Supported using the o1esr embedded objects reader.

Spreadsheet Formats

Supported Spreadsheet Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Numbers	'08, '09	iwsssr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16, '18, iCloud 2018	iwss13sr ¹	NUMBERS	Y	T	T	N	N	Y	N
Applix Spreadsheets	4.2, 4.3, 4.4	assr	AS	Y	Y	Y	N	N	Y	N
Comma Separated Values	n/a	csvsr	CSV	Y	Y	Y	N	N	N	N
Corel Quattro Pro	5, 6, 7, 8	qpssr	WB2 WB3	Y	Y	Y	N	P	Y	N
	X4	qpwsr	QPW	Y	N	Y	N	P	Y	N
Data Interchange Format	n/a	difsr		Y	Y	Y	N	N	N	N
Lotus 1-2-3	96, 97, R9, 9.8	l123sr	123	Y	Y	Y	N	P	Y	N
Lotus 1-2-3	2, 3, 4, 5	wkssr	WK4	Y	Y	Y	N	N	Y	N
Lotus 1-2-3 Charts	2, 3, 4, 5	kpchtrdr	123	N	Y	Y	N	N	N	N
Microsoft Excel Charts	2, 3, 4, 5, 6, 7	kpchtrdr	XLS	N	Y	Y	N	N	N	N

¹This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Excel Macintosh	98, 2001, v.X, 2004	xlssr	XLS	Y	Y	Y	Y ¹	Y	Y	N
Microsoft Excel Windows	2.2 through 2003	xlssr	XLS XLW XLT XLA	Y	Y	Y	Y ²	Y	Y	Y
Microsoft Excel Windows XML	2007, 2010, 2013, 2016, 2019	xlsxsr	XLSX XLTX XLSM XLTM XLAM	Y	Y	Y	Y	Y	Y	Y
Microsoft Excel Binary Format	2007, 2010, 2013, 2016	xlsbsr	XLSB	Y	Y	Y	N	Y	N	N
Microsoft Works Spreadsheet	2, 3, 4	mwssr	S30 S40	Y	Y	Y	N	N	Y	N
Microsoft Power BI	1.11	pbixsr	PBIX	Y	T	T	N	N	Y	N
OASIS Open Document Format	1, 2 ³	odfssr	ODS SXC STC	Y	Y	Y	Y ⁴	Y	Y	N

¹Supported using the embedded objects reader o1esr.

²Supported for versions 97 and higher using the embedded objects reader o1esr.

³Generated by OpenOffice Calc 2.0, StarOffice 8 Calc, and IBM Lotus Symphony Spreadsheet 3.0.

⁴Supported using the embedded objects reader o1esr.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
OpenOffice Calc, LibreOffice Calc	1 to 5	sosr	SXC ODS OTS	Y	T	T	N	Y	Y	N
StarOffice Calc	3, 4, 5	starcsr	SDC	Y	T	T	N	N	N	N
	6, 7, 8, 9	sosr	SXC ODS	Y	T	T	N	Y	Y	N

Text and Markup Formats

Supported Text and Markup Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
ANSI	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
ASCII	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
HTML	3, 4	htmsr	HTM	Y	Y	Y	N	P	Y	N
Microsoft Excel Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N
Microsoft Word Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N
Microsoft Visio XML	2003	xmlsr	VDX VTX	Y	T	T	N	Y	Y	N
MIME HTML	n/a	mhtsr	MHT	Y	Y	Y	N	Y	Y	N
Rich Text Format	1 through	rtfsr	RTF	Y	Y	Y	N	P	Y	Y

Supported Text and Markup Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
	1.7									
Tableau Data Source format	n/a	xmlsr	TDS	Y	T	T	N	Y	Y	N
Tableau Map Source format	n/a	xmlsr	TMS	Y	T	T	N	Y	Y	N
Tableau Preferences format	n/a	xmlsr	TPS	Y	T	T	N	Y	Y	N
Tableau Workbook format	n/a	xmlsr	TWB	Y	T	T	N	Y	Y	N
Unicode HTML	n/a	unihtmlsr	HTM	Y	Y	Y	N	Y	Y	N
Unicode Text	3, 4	unistr	TXT	Y	Y	Y	N	N	Y	N
Vector Open Diagnostic Data Exchange Format	n/a	xmlsr	ODX	Y	T	T	N	Y	Y	N
XHTML	1.0	htmlsr	HTM	Y	Y	Y	N	Y	Y	N
XML (generic)	1.0	xmlsr	XML	Y	T	T	N	Y	Y	N

Word Processing Formats

Supported Word Processing Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe FrameMaker Interchange Format	5, 5.5, 6, 7	mifsr	MIF	Y	Y	Y	N	N	Y	N
Apple iChat Log	1, AV 2 AV 2.1, AV 3	ichatsr	ICHAT	Y	Y	Y	N	N	N	N

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Pages	'08, '09	iwwpsr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16, '18 iCloud 2018	iwwp13sr 1	PAGES	Y	T	T	N	N	N	N
Applix Words	3.11, 4, 4.1, 4.2, 4.3, 4.4	awsr	AW	Y	Y	Y	N	N	Y	Y
Corel WordPerfect Linux	6.0, 8.1	wp6sr	WPS	Y	Y	Y	N	P	Y	N
Corel WordPerfect Macintosh	1.02, 2, 2.1, 2.2, 3, 3.1	wpmsr	WPM	Y	Y	Y	N	N	Y	N
Corel WordPerfect Windows	5, 5.1	wosr	WO	Y	Y	Y	N	P	Y	Y
Corel WordPerfect Windows	6, 7, 8, 9, 10, 11, 12, X3	wp6sr	WPD	Y	Y	Y	N	P	Y	Y
DisplayWrite	4	dw4sr	IP	Y	Y	Y	N	N	Y	N
Folio Flat File	3.1	foliosr	FFF	Y	Y	Y	N	Y	Y	Y
Founder Chinese E-	3.2.1	cebsr ²	CEB	Y	N	N	N	N	N	N

¹This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.

²This reader is only supported on Windows 32-bit platforms.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
paper Basic										
Fujitsu Oasys	7	oa2sr	OA2	Y	Y	Y	N	P	N	N
Haansoft Hangul	97	hwpsr	HWP	Y	Y	Y	N	Y	Y	N
	2002, 2005, 2007, 2010	hwposr	HWP	Y	Y	Y	Y	Y	Y	N
Health level7	2.0	hl7sr	HL7	Y	Y	Y	N	Y	Y	N
IBM DCA/RFT (Revisable Form Text)	SC23-0758-1	dcasr	DC	Y	Y	Y	N	N	Y	N
JustSystems Ichitaro	8 to 2013, 2018	jtdsr	JTD	Y	Y	Y	N	P	N	Y
Lotus AMI Pro	2, 3	lasr	SAM	Y	Y	Y	N	P	Y	Y
Lotus AMI Professional Write Plus	2.1	lasr	AMI	Y	Y	Y	N	N	N	Y
Lotus Word Pro	96, 97, R9	lwpsr	LWP	Y	Y	Y	N	P	N	Y
Lotus SmartMaster	96, 97	lwpsr	MWP	Y	Y	Y	N	N	N	N
Microsoft OneNote	2007, 2010, 2013, 2016	kpONErdr	ONE ONETOC2	Y	Y	Y	Y	N	Y	N
Microsoft OneNote Alternate Format	2007, 2010, 2013, 2016	onealtsr	ONE ONETOC2	Y	T	T	Y	N	N	N

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Word Macintosh	4, 5, 6, 98	mbsr	DOC	Y	Y	Y	N	Y	N	Y
	2001, v.X, 2004	mw8sr	DOC DOT	Y	Y	Y	Y ¹	Y	Y	N
Microsoft Word PC	4, 5, 5.5, 6	mwsr	DOC	Y	Y	Y	N	N	N	Y
Microsoft Word Windows	1.0, 2.0	misr	DOC	Y	Y	Y	N	N	N	Y
Microsoft Word Windows	6, 7, 8, 95	mw6sr	DOC	Y	Y	Y	N	Y	Y	Y
Microsoft Word Windows	97, 2000, 2002, 2003	mw8sr	DOC DOT	Y	Y	Y	Y ²	Y	Y	Y
Microsoft Word Windows XML	2007, 2010, 2013, 2016, 2019	mwxsr	DOCX DOCX DOTX DOTM	Y	Y	Y	Y	Y	Y	Y
Microsoft Word Windows Flat XML	2007, 2010, 2013, 2016	mwxsr	XML	Y	Y	Y	Y	Y	Y	Y
Microsoft Works	1, 2, 3, 4	mswsr	WPS	Y	Y	Y	N	N	N	Y

¹Supported using the embedded objects reader olesr.

²Supported using the embedded objects reader olesr.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Works	6, 2000	msw6sr	WPS	Y	Y	Y	N	N	N	Y
Microsoft Windows Write	1, 2, 3	mwsr	WRI	Y	Y	Y	N	N	Y	N
OASIS Open Document Format	1, 2 ¹	odfwpsr	ODT SXW STW	Y	Y	Y	Y ²	Y	Y	Y
Omni Outliner	v3, OPML, OOutline	oo3sr	OO3 OPML OOUTLINE	Y	Y	Y	N	N	Y	N
OpenOffice Writer, LibreOffice Writer	1 to 5	sosr	SXW ODT	Y	T	T	N	Y	Y	N
Open Publication Structure eBook	2.0, 3.0	epubsr	EPUB	Y	Y	Y	N	Y	Y	N
pFiles	n/a	pfilesr	PFILE PBMP PGIF	Y ³	T ⁴	T ⁵	N	Y	N	N

¹Generated by OpenOffice Writer 2.0, StarOffice 8 Writer, and IBM Lotus Symphony Documents 3.0.

²Supported using the embedded objects reader o1esr.

³KeyView filters only the internal redirection text. The underlying document text is not accessible without the decryption key.

⁴KeyView filters only the internal redirection text. The underlying document text is not accessible without the decryption key.

⁵KeyView filters only the internal redirection text. The underlying document text is not accessible without the decryption key.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
			PJPG PPNG PTIF PTXT PXML							
StarOffice Writer	3, 4, 5	starwsr	SDW	Y	T	T	N	N	N	N
	6, 7, 8, 9	sosr	SXW ODT	Y	T	T	N	Y	Y	N
Skype Log	3	skypesr	DBB	Y	Y	Y	N	N	N	N
WordPad	through 2003	rtfsr	RTF	Y	Y	Y	N	P	Y	N
XML Paper Specification	n/a	xpsr	XPS	Y	T	T	N	N	N	N
XyWrite	4.12	xywsr	XY4	Y	Y	Y	N	N	N	N
Yahoo! Instant Messenger	n/a	yimsr ¹	DAT	Y	Y	Y	N	N	N	N

¹To successfully use this reader, you must set the KV_YAH00_ID environment variable to the Yahoo user ID. You can optionally set the KV_OTHER_YAH00_ID environment variable to the other Yahoo user ID. If you do not set it, "Other" is used by default. If you enter incorrect values for the environment variables, erroneous data is generated.

Appendix B: Detected Formats

This section lists the file formats that KeyView can detect.

- [Key to Detected Formats Table](#) 221
- [Detected Formats](#) 223

Key to Detected Formats Table

The detected formats table includes the following information:

Column	Description
Format Name	The format name that is returned by KeyView format detection. <ul style="list-style-type: none">• In the C API, these values are defined in the <code>ENdocFmt</code> enumeration in <code>adDocFmt.h</code>.• In the .NET API these values are defined in the <code>Autonomy.API.Filter.DocFormat</code> enumeration.• In the Java API these values are defined in the <code>com.verity.api.DocFormat</code> enumeration.• In the C++ API these values are defined in <code>keyview::Format</code>, used in <code>DetectionInfo</code> which is returned by <code>Session::detect()</code>.
Number	The format number that is returned by KeyView format detection. This is the value associated with the Format Name in the relevant enumeration.
Category	This value is used in the KeyView configuration file <code>formats.ini</code> to specify the reader to use to filter, export, or view the format. Several formats might have the same category value.
Description	A short description of the file format.
MIME Type	The MIME type (if any).
Extension	A list of common file extensions for the file format. <div style="border: 1px solid #0070C0; padding: 5px; background-color: #F0F8FF;"><p>This is not a complete list of file extensions. KeyView does not distinguish between file types based on their extension. Instead, it detects the file format based on the file content. This is more reliable because content cannot always be predicted from the file extension, and because some file extensions are associated with multiple formats.</p></div>
File Class	The KeyView file class. <ul style="list-style-type: none">• In the C API, these values are defined in the <code>ENdocClass</code> enumeration in

	<p><code>adinfo.h</code>.</p> <ul style="list-style-type: none">• In the .NET API these values are defined in the <code>Autonomy.API.Filter.DocClass</code> enumeration.• In the Java API these values are defined in the <code>com.verity.api.DocClass</code> enumeration.• In the C++ API these values are defined in <code>keyview::Category</code>, used in <code>DetectionInfo</code> which is returned by <code>Session::detect()</code>.
--	---

Detected Formats

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Reserved__Fmt	-1	-1				AutoDetNoFormat
Unknown_Fmt	0	0				AutoDetNoFormat
AES_Multiplus_Comm_Fmt	1	1	Multiplus (AES)		PTF	adWORDPROCESSOR
ASCII_Text_Fmt	2	2	Plain Text file	text/plain	TXT	adWORDPROCESSOR
MSDOS_Batch_File_Fmt	3	2	MS-DOS Batch File	application/x-bat	BAT	adEXECUTABLE
Applix_Alis_Fmt	4	3	APPLIX ASTERIX		AX	adWORDPROCESSOR
BMP_Fmt	5	4	Windows Bitmap Image (BMP)	image/bmp	BMP	adRASTERIMAGE
CT_DEF_Fmt	6	5	Convergent Technologies DEF Comm. Format			adWORDPROCESSOR
Corel_Draw_Fmt	7	6	Corel Draw (up to version 13/X3)	application/coreldraw	CDR	adVECTORGRAPHIC
CGM_ClearText_Fmt	8	8	Computer Graphics Metafile (CGM)		CGM	adVECTORGRAPHIC
CGM_Binary_Fmt	9	8	Computer Graphics Metafile (CGM)	image/cgm	CGM	adVECTORGRAPHIC
CGM_Character_Fmt	10	8	Computer Graphics Metafile (CGM)		CGM	adVECTORGRAPHIC
Word_Connection_Fmt	11	9	Word Connection		CN	adWORDPROCESSOR
COMET_TOP_Word_Fmt	12	10	Nixdorf COMET TOP Financial Accounting software			adWORDPROCESSOR
CEOwrite_Fmt	13	11	CEOwrite		CW	adWORDPROCESSOR
DSA101_Fmt	14	12	DSA101 (Honeywell Bull)			adWORDPROCESSOR
DCA_RFT_Fmt	15	13	DCA-RFT (IBM Revisable Form)	application/dca-rft	RFT, DC	adWORDPROCESSOR
CDA_DDIF_Fmt	16	14	CDA / DDIF		DDIF	adWORDPROCESSOR
DG_CDS_Fmt	17	16	DG Common Data Stream (CDS)		CDS	adWORDPROCESSOR
Micrografx_Draw_Fmt	18	18	Windows Draw (Micrografx)		DRW	adVECTORGRAPHIC
Data_Point_VistaWord_Fmt	19	19	Vistaword		DV	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
DECdx_Fmt	20	20	DECdx		DX	adWORDPROCESSOR
Enable_WP_Fmt	21	21	Enable Word Processing		WPF	adWORDPROCESSOR
EPSF_Fmt	22	22	Encapsulated PostScript	application/postscript	EPS	AutoDetNoFormat
Preview_EPSF_Fmt	23	22	Encapsulated PostScript	application/postscript		AutoDetNoFormat
MS_Executable_Fmt	24	23	MSDOS/Windows Program	application/x-msdownload	EXE	adEXECUTABLE
G31D_Fmt	25	24	CCITT G3 1D			adRASTERIMAGE
GIF_87a_Fmt	26	25	Graphics Interchange Format (GIF87a)	image/gif	GIF	adRASTERIMAGE
GIF_89a_Fmt	27	25	Graphics Interchange Format (GIF89a)	image/gif	GIF	adRASTERIMAGE
HP_Word_PC_Fmt	28	26	HP Word PC		HW	adWORDPROCESSOR
IBM_1403_LinePrinter_Fmt	29	27	IBM 1403 Line Printer		I4	adWORDPROCESSOR
IBM_DCF_Script_Fmt	30	28	DCF Script		IC	adWORDPROCESSOR
IBM_DCA_FFT_Fmt	31	29	DCA-FFT (IBM Final Form)		IF, FFT	adWORDPROCESSOR
Interleaf_Fmt	32	30	Interleaf			adWORDPROCESSOR
GEM_Image_Fmt	33	31	GEM Bit Image		IMG	adRASTERIMAGE
IBM_Display_Write_Fmt	34	32	Display Write		IP	adWORDPROCESSOR
Sun_Raster_Fmt	35	33	Sun Raster	image/x-cmu-raster	RAS	adRASTERIMAGE
Ami_Pro_Fmt	36	35	Lotus Ami Pro	application/x-lotus-amipro	SAM	adWORDPROCESSOR
Ami_Pro_StyleSheet_Fmt	37	35	Lotus Ami Pro Style Sheet			adWORDPROCESSOR
MORE_Fmt	38	36	MORE Database MAC			adOUTLINE
Lyrix_Fmt	39	37	Lyrix Word Processing			adWORDPROCESSOR
MASS_11_Fmt	40	38	MASS-11		M1	adWORDPROCESSOR
MacPaint_Fmt	41	39	MacPaint		PNTG	adRASTERIMAGE
MS_Word_Mac_Fmt	42	40	Microsoft Word for Macintosh (up to version 3)	application/msword	DOC	adWORDPROCESSOR
SmartWare_II_Comm_Fmt	43	41	SmartWare II			adCOMMUNICATION
MS_Word_Win_Fmt	44	42	Microsoft Word for Windows (up to version 6)	application/msword	DOC, WPS	adWORDPROCESSOR
Multimate_Fmt	45	43	MultiMate		MM	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Multimate_Fnote_Fmt	46	43	MultiMate Footnote File			adWORDPROCESSOR
Multimate_Adv_Fmt	47	43	MultiMate Advantage			adWORDPROCESSOR
Multimate_Adv_Fnote_Fmt	48	43	MultiMate Advantage Footnote File			adWORDPROCESSOR
Multimate_Adv_II_Fmt	49	43	MultiMate Advantage II			adWORDPROCESSOR
Multimate_Adv_II_Fnote_Fmt	50	43	MultiMate Advantage II Footnote File		FBX, FNX	adWORDPROCESSOR
Multiplan_PC_Fmt	51	44	Multiplan (PC)			adSPREADSHEET
Multiplan_Mac_Fmt	52	44	Multiplan (Mac)			adSPREADSHEET
MS_RTF_Fmt	53	45	Rich Text Format (RTF)	application/rtf	RTF	adWORDPROCESSOR
MS_Word_PC_Fmt	54	46	Microsoft Word for PC (up to version 6)	application/x-ms-wordpc	MW	adWORDPROCESSOR
MS_Word_PC_StyleSheet_Fmt	55	46	Microsoft Word for PC (up to version 6) Style Sheet			adWORDPROCESSOR
MS_Word_PC_Glossary_Fmt	56	46	Microsoft Word for PC (up to version 6) Glossary			adWORDPROCESSOR
MS_Word_PC_Driver_Fmt	57	46	Microsoft Word for PC (up to version 6) Driver			adWORDPROCESSOR
MS_Word_PC_Misc_Fmt	58	46	Microsoft Word for PC (up to version 6) Miscellaneous File			adWORDPROCESSOR
NBI_Async_Archive_Fmt	59	47	NBI Async Archive Format			adWORDPROCESSOR
Navy_DIF_Fmt	60	48	Navy DIF (document interchange format)		ND	adWORDPROCESSOR
NBI_Net_Archive_Fmt	61	49	NBI Net Archive Format		NN	adWORDPROCESSOR
NIOS_TOP_Fmt	62	50	NIOS TOP			adWORDPROCESSOR
FileMaker_Mac_Fmt	63	51	Filemaker MAC		FP5, FP7	adDATABASE
ODA_Q1_11_Fmt	64	52	ODA / ODIF Q1 11		OD	adWORDPROCESSOR
ODA_Q1_12_Fmt	65	52	ODA / ODIF Q1 12		OD	adWORDPROCESSOR
OLIDIF_Fmt	66	53	OLIDIF (Olivetti)			adWORDPROCESSOR
Office_Writer_Fmt	67	55	Office Writer		OW	adWORDPROCESSOR
PC_Paintbrush_Fmt	68	56	PC Paintbrush Graphics (PCX)	image/vnd.zbrush.pcx	PCX	adRASTERIMAGE
CPT_Comm_Fmt	69	57	CPT Corporation word processor		PF	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Lotus_PIC_Fmt	70	58	Lotus PIC	image/x-pict	PIC	adVECTORGRAPHIC
Mac_PICT_Fmt	71	59	QuickDraw Picture	image/x-pict	PCT	AutoDetNoFormat
Philips_Script_Word_Fmt	72	60	Philips Script			adWORDPROCESSOR
PostScript_Fmt	73	61	PostScript	application/postscript	PS	adVECTORGRAPHIC
PRIMEWORD_Fmt	74	62	PRIMEWORD			adWORDPROCESSOR
Quadratron_Q_One_v1_Fmt	75	63	Q-One V1.93J		Q1, QX	adWORDPROCESSOR
Quadratron_Q_One_v2_Fmt	76	64	Q-One V2.0		Q1, QX	adWORDPROCESSOR
SAMNA_Word_IV_Fmt	77	65	SAMNA Word		SAM	adWORDPROCESSOR
Ami_Pro_Draw_Fmt	78	66	Lotus Ami Pro Draw		SDW	adVECTORGRAPHIC
SYLK_Spreadsheet_Fmt	79	67	SYmbolic LinK (SYLK) format		SLK	adSPREADSHEET
SmartWare_II_WP_Fmt	80	68	Informix SmartWare II word processor		DOC	adWORDPROCESSOR
Symphony_Fmt	81	69	Lotus Symphony spreadsheet		WR1	adSPREADSHEET
Targa_Fmt	82	70	Targa image	image/x-tga	TGA	adRASTERIMAGE
TIFF_Fmt	83	71	Tag Image File Format (TIFF)	image/tiff	TIF, TIFF	adRASTERIMAGE
Targon_Word_Fmt	84	72	Targon Word		TW	adWORDPROCESSOR
Uniplex_Ucalc_Fmt	85	73	Uniplex Ucalc		SS	adSPREADSHEET
Uniplex_WP_Fmt	86	74	Uniplex word processor		UP	adWORDPROCESSOR
MS_Word_UNIX_Fmt	87	75	Microsoft Word UNIX	application/msword		adWORDPROCESSOR
WANG_PC_Fmt	88	76	WANG PC			adWORDPROCESSOR
WordERA_Fmt	89	77	WordERA		DC, GL, FR	adWORDPROCESSOR
WANG_WPS_Comm_Fmt	90	78	WANG WPS		WF	adWORDPROCESSOR
WordPerfect_Mac_Fmt	91	79	WordPerfect MAC	application/x-corel-wordperfect		adWORDPROCESSOR
WordPerfect_Fmt	92	86	WordPerfect version 4	application/x-corel-wordperfect	WP, WP4	adWORDPROCESSOR
WordPerfect_VAX_Fmt	93	139	WordPerfect VAX	application/x-corel-wordperfect		adWORDPROCESSOR
WordPerfect_Macro_Fmt	94	139	WordPerfect Macro	application/vnd.wordperfect	MRS	adWORDPROCESSOR
WordPerfect_Dictionary_Fmt	95	139	WordPerfect Spelling Dictionary	application/vnd.wordperfect	SPW	adWORDPROCESSOR
WordPerfect_Thesaurus_Fmt	96	139	WordPerfect Thesaurus	application/vnd.wordperfect		adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
WordPerfect_Resource_Fmt	97	139	WordPerfect Resource File	application/vnd.wordperfect	WWK, PRS	adWORDPROCESSOR
WordPerfect_Driver_Fmt	98	139	WordPerfect Driver	application/vnd.wordperfect	IRS, VRS	adWORDPROCESSOR
WordPerfect_Cfg_Fmt	99	139	WordPerfect Configuration File	application/vnd.wordperfect	PFX	adWORDPROCESSOR
WordPerfect_Hyphenation_Fmt	100	139	WordPerfect Hyphenation Dictionary	application/vnd.wordperfect	HYC	adWORDPROCESSOR
WordPerfect_Misc_Fmt	101	139	WordPerfect Miscellaneous File	application/vnd.wordperfect		adWORDPROCESSOR
WordMARC_Fmt	102	82	WordMARC Composer	video/x-ms-wm	WM, PW	adWORDPROCESSOR
Windows_Metafile_Fmt	103	83	Windows Metafile	image/wmf	WMF	adVECTORGRAPHIC
Windows_Metafile_NoHdr_Fmt	104	83	Windows Metafile (no header)	image/wmf	WMF	adVECTORGRAPHIC
SmartWare_II_DB_Fmt	105	84	Informix SmartWare II database			adDATABASE
WordPerfect_Graphics_Fmt	106	195	WordPerfect Graphics (version 2 and higher)	application/vnd.wordperfect	WPG, QPG	AutoDetNoFormat
WordStar_Fmt	107	87	WordStar		WS, WSD	adWORDPROCESSOR
WANG_WITA_Fmt	108	88	WANG WITA		WT	adWORDPROCESSOR
Xerox_860_Comm_Fmt	109	89	Xerox 860			adWORDPROCESSOR
Xerox_Writer_Fmt	110	91	Xerox Writer			adWORDPROCESSOR
DIF_SpreadSheet_Fmt	111	92	Data Interchange Format (DIF)	application/dif+xml	DIF	adSPREADSHEET
Enable_Spreadsheet_Fmt	112	93	Enable Spreadsheet	application/vnd.epson.ssf	SSF	adSPREADSHEET
SuperCalc_Fmt	113	94	Sorcim SuperCalc spreadsheet		CAL	adSPREADSHEET
UltraCalc_Fmt	114	95	UltraCalc spreadsheet			adSPREADSHEET
SmartWare_II_SS_Fmt	115	96	Informix SmartWare II spreadsheet			adSPREADSHEET
SOF_Encapsulation_Fmt	116	97	Serialized Object Format (SOF)	application/java-serialized-object	SOF	adENCAPSULATION
PowerPoint_Win_Fmt	117	98	Microsoft PowerPoint PC (up to version 4)	application/x-ms-powerpoint	PPT	adPRESENTATION
PowerPoint_Mac_Fmt	118	99	Microsoft PowerPoint MAC (up to version 4)	application/x-ms-powerpoint	PPT	adPRESENTATION
PowerPoint_95_Fmt	119	212	Microsoft PowerPoint 95	application/x-ms-powerpoint	PPT	adPRESENTATION
PowerPoint_97_Fmt	120	272	Microsoft PowerPoint 97	application/x-ms-powerpoint	PPT	adPRESENTATION
PageMaker_Mac_Fmt	121	100	PageMaker for Macintosh			adDESKTOPPUBLSH
PageMaker_Win_Fmt	122	101	PageMaker for Windows			adDESKTOPPUBLSH

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Works_Mac_WP_Fmt	123	103	Microsoft Works Word Processor for MAC	application/x-msworks	MWK	adWORDPROCESSOR
MS_Works_Mac_DB_Fmt	124	104	Microsoft Works Database for MAC	application/x-msworks		adDATABASE
MS_Works_Mac_SS_Fmt	125	105	Microsoft Works Spreadsheet for MAC	application/x-msworks		adSPREADSHEET
MS_Works_Mac_Comm_Fmt	126	106	Microsoft Works Communication for MAC	application/x-msworks		adCOMMUNICATION
MS_Works_DOS_WP_Fmt	127	107	Microsoft Works Word Processor for DOS	application/x-msworks	WPS	adWORDPROCESSOR
MS_Works_DOS_DB_Fmt	128	108	Microsoft Works Database for DOS	application/x-msworks	WDB	adDATABASE
MS_Works_DOS_SS_Fmt	129	109	Microsoft Works Spreadsheet for DOS	application/x-msworks		adSPREADSHEET
MS_Works_Win_WP_Fmt	130	227	Microsoft Works Word Processor for Windows	application/x-msworks	WPS, W40	adWORDPROCESSOR
MS_Works_Win_DB_Fmt	131	231	Microsoft Works Database for Windows	application/x-msworks		adDATABASE
MS_Works_Win_SS_Fmt	132	228	Microsoft Works Spreadsheet for Windows	application/x-msworks	S30, S40	adSPREADSHEET
PC_Library_Fmt	133	111	DOS/Windows Object Library	application/x-archive	LIB, A	adLIBRARY
MacWrite_Fmt	134	112	MacWrite	application/macwriteii		adWORDPROCESSOR
MacWrite_II_Fmt	135	113	MacWrite II	application/macwriteii		adWORDPROCESSOR
Freehand_Fmt	136	114	Freehand MAC	image/x-freehand		adVECTORGRAPHIC
Disk_Doubler_Fmt	137	115	Disk Doubler			adENCAPSULATION
HP_GL_Fmt	138	116	HP Graphics Language	vector/x-hpgl	HPGL, HPG	adVECTORGRAPHIC
FrameMaker_Fmt	139	136	FrameMaker	application/vnd.frameMaker	FM, FRM	adDESKTOPPUBLSH
FrameMaker_Book_Fmt	140	136	FrameMaker Book	application/vnd.frameMaker	BOOK	adDESKTOPPUBLSH
Maker_Markup_Language_Fmt	141	174	Maker Markup Language	application/vnd.mif		adDESKTOPPUBLSH
Maker_Interchange_Fmt	142	117	Maker Interchange Format (MIF)	application/x-mif	MIF	adWORDPROCESSOR
JPEG_File_Interchange_Fmt	143	118	JPEG Interchange Format	image/jpeg	JPG, JPEG	adRASTERIMAGE
Reflex_Fmt	144	119	Borland Reflex database			adDATABASE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Framework_Fmt	145	276	Framework office suite			adMIXED
Framework_II_Fmt	146	120	Framework II office suite		FW3	adMIXED
Paradox_Fmt	147	121	Borland Paradox database		DB	adDATABASE
MS_Windows_Write_Fmt	148	123	Microsoft Windows Write	application/x-ms-write	WRI	adWORDPROCESSOR
Quattro_Pro_DOS_Fmt	149	124	Quattro Pro for DOS	application/x-quattropro	WQ1	adSPREADSHEET
Quattro_Pro_Win_Fmt	150	184	Quattro Pro for Windows	application/x-quattro-win	WB1, WB2, WB3	adSPREADSHEET
Persuasion_Fmt	151	126	Adobe Persuasion			adPRESENTATION
Windows_Icon_Fmt	152	128	Windows Icon Format	image/ico	ICO	adRASTERIMAGE
Windows_Cursor_Fmt	153	133	Windows Cursor	image/x-win-bitmap	CUR	adRASTERIMAGE
MS_Project_Activity_Fmt	154	129	Microsoft Project (up to version 3) activity file			adSCHEDULE
MS_Project_Resource_Fmt	155	129	Microsoft Project (up to version 3) resource file			adSCHEDULE
MS_Project_Calc_Fmt	156	129	Microsoft Project (up to version 3) calc file			adSCHEDULE
PKZIP_Fmt	157	132	ZIP Archive	application/zip	ZIP, ZIPX	adENCAPSULATION
Quark_Xpress_Fmt	158	134	Quark Xpress MAC			adDESKTOPPUBLSH
ARC_PAK_Archive_Fmt	159	135	PAK/ARC Archive		ARC, PAK	adENCAPSULATION
MS_Publisher_Fmt	160	137	Microsoft Publisher (up to version 3)	application/x-mspublisher	PUB	adDESKTOPPUBLSH
PlanPerfect_Fmt	161	138	PlanPerfect			adSCHEDULE
WordPerfect_Auxiliary_Fmt	162	139	WordPerfect auxiliary file		WPW	adMISC
MS_WAVE_Audio_Fmt	163	141	Microsoft Wave	audio/wav	WAV	adSOUND
MIDI_Audio_Fmt	164	142	MIDI audio	audio/mid	MID, MIDI	adSOUND
AutoCAD_DXF_Binary_Fmt	165	143	AutoCAD DXF	image/x-dxf	DXF	adVECTORGRAPHIC
AutoCAD_DXF_Text_Fmt	166	143	AutoCAD DXF	image/x-dxf	DXF	adVECTORGRAPHIC
dBase_Fmt	167	144	dBase	application/x-dbf	DBF, VCX	adDATABASE
OS_2_PM_Metatile_Fmt	168	145	OS/2 PM Metatile		MET	adVECTORGRAPHIC
Lasergraphics_Language_Fmt	169	146	Lasergraphics Language			adVECTORGRAPHIC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
AutoShade_Rendering_Fmt	170	147	AutoShade Rendering			adVECTORGRAPHIC
GEM_VDI_Fmt	171	148	GEM VDI Metafile image		GEM, GDI	adVECTORGRAPHIC
Windows_Help_Fmt	172	149	Windows Help File	application/winhelp	HLP	adMISC
Volkswriter_Fmt	173	150	Volkswriter word processor		VW4	adWORDPROCESSOR
Ability_WP_Fmt	174	151	Ability Word Processor			adWORDPROCESSOR
Ability_DB_Fmt	175	151	Ability Database			adDATABASE
Ability_SS_Fmt	176	151	Ability Spreadsheet			adSPREADSHEET
Ability_Comm_Fmt	177	151	Ability Presentation			adCOMMUNICATION
Ability_Image_Fmt	178	151	Ability Image			adRASTERIMAGE
XyWrite_Fmt	179	152	XYWrite / Nota Bene		XY4	adWORDPROCESSOR
CSV_Fmt	180	153	CSV (Comma Separated Values)	text/csv	CSV	adSPREADSHEET
IBM_Writing_Assistant_Fmt	181	154	IBM Writing Assistant		IWA	adWORDPROCESSOR
WordStar_2000_Fmt	182	155	WordStar 2000		WS2	adWORDPROCESSOR
HP_PCL_Fmt	183	157	HP Printer Control Language	application/pcl	PCL	adVECTORGRAPHIC
UNIX_Exe_PreSysV_VAX_Fmt	184	158	Unix Executable (PDP-11/pre-System V VAX)	application/octet-stream		adEXECUTABLE
UNIX_Exe_Basic_16_Fmt	185	158	Unix Executable (Basic-16)	application/octet-stream		adEXECUTABLE
UNIX_Exe_x86_Fmt	186	158	Unix Executable (x86)	application/octet-stream		adEXECUTABLE
UNIX_Exe_iAPX_286_Fmt	187	158	Unix Executable (iAPX 286)	application/octet-stream		adEXECUTABLE
UNIX_Exe_MC68k_Fmt	188	158	Unix Executable (MC680x0)	application/octet-stream		adEXECUTABLE
UNIX_Exe_3B20_Fmt	189	158	Unix Executable (3B20)	application/octet-stream		adEXECUTABLE
UNIX_Exe_WE32000_Fmt	190	158	Unix Executable (WE32000)	application/octet-stream		adEXECUTABLE
UNIX_Exe_VAX_Fmt	191	158	Unix Executable (VAX)	application/octet-stream		adEXECUTABLE
UNIX_Exe_Bell_5_Fmt	192	158	Unix Executable (Bell 5.0)	application/octet-stream		adEXECUTABLE
UNIX_Obj_VAX_Demand_Fmt	193	159	Unix Object Module (VAX Demand)			adOBJECTMODULE
UNIX_Obj_MS8086_Fmt	194	159	Unix Object Module (old MS 8086)			adOBJECTMODULE
UNIX_Obj_Z8000_Fmt	195	159	Unix Object Module (Z8000)			adOBJECTMODULE
AU_Audio_Fmt	196	161	NeXT/Sun Audio Data	audio/basic	AU	adSOUND

Format Name	Number	Category	Description	MIME Type	Extension	File Class
NeWS_Font_Fmt	197	162	NeWS bitmap font			adFONT
cpio_Archive_CRCldr_Fmt	198	163	cpio archive (CRC Header)	application/x-cpio		adENCAPSULATION
cpio_Archive_CHRhdr_Fmt	199	163	cpio archive (CHR Header)	application/x-cpio		adENCAPSULATION
PEX_Binary_Archive_Fmt	200	164	SUN PEX Binary Archive			adENCAPSULATION
Sun_vfont_Fmt	201	165	SUN vfont Definition			adFONT
Curses_Screen_Fmt	202	166	Curses Screen Image			adRASTERIMAGE
UUEncoded_Fmt	203	167	UU encoded	text/x-uencode	UUE	adENCAPSULATION
WriteNow_Fmt	204	168	WriteNow MAC			adWORDPROCESSOR
PC_Obj_Fmt	205	169	DOS/Windows Object Module	application/octet-stream	OBJ	adOBJECTMODULE
Windows_Group_Fmt	206	170	Windows Group			adMISC
TrueType_Font_Fmt	207	171	TrueType Font	application/x-font-ttf	TTF	adFONT
Windows_PIF_Fmt	208	172	Program Information File (PIF)	application/octet-stream	PIF	adMISC
MS_COM_Executable_Fmt	209	173	PC (.COM)	application/octet-stream	COM	adEXECUTABLE
Stuffit_Fmt	210	175	Stuffit (MAC)	application/x-stuffit	HQX	adENCAPSULATION
PeachCalc_Fmt	211	176	PeachCalc		CAL	adSPREADSHEET
Wang_GDL_Fmt	212	177	WANG Office GDL Header			adENCAPSULATION
Q_A_DOS_Fmt	213	179	Q & A for DOS			adWORDPROCESSOR
Q_A_Win_Fmt	214	180	Q & A for Windows		JW	adWORDPROCESSOR
WPS_PLUS_Fmt	215	181	WPS-PLUS	application/vnd.ms-wpl	WPL	adWORDPROCESSOR
DCX_Fmt	216	182	DCX FAX Format(PCX images)	image/dcx	DCX	adFAXFORMAT
OLE_Fmt	217	183	OLE Compound Document		OLE	adENCAPSULATION
EBCDIC_Fmt	218	186	EBCDIC Text			adWORDPROCESSOR
DCS_Fmt	219	187	DCS			adWORDPROCESSOR
UNIX_SHAR_Fmt	220	190	SHAR shell archive format	application/x-shar	SHAR	adENCAPSULATION
Lotus_Notes_BitMap_Fmt	221	191	Lotus Notes Bitmap			adRASTERIMAGE
Lotus_Notes_CDF_Fmt	222	193	Lotus Notes CDF	application/cdf	CDF	adWORDPROCESSOR
Compress_Fmt	223	192	Unix Compress	application/x-compress	Z	adENCAPSULATION
GZ_Compress_Fmt	224	198	GZ Compress	application/gzip	GZ	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
TAR_Fmt	225	194	TAR archive	application/tar	TAR	adENCAPSULATION
ODIF_FOD26_Fmt	226	196	Open Document Architecture (ODA / ODIF) FOD26	application/oda	F26	adWORDPROCESSOR
ODIF_FOD36_Fmt	227	196	Open Document Architecture (ODA / ODIF) FOD36	application/oda	F36	adWORDPROCESSOR
ALIS_Fmt	228	197	ALIS			adWORDPROCESSOR
Envoy_Fmt	229	199	WordPerfect Envoy	application/envoy	EVY	adWORDPROCESSOR
PDF_Fmt	230	200	Portable Document Format	application/pdf	PDF	adWORDPROCESSOR
BinHex_Fmt	231	206	BinHex	application/mac-binhex40	HQX	adENCAPSULATION
SMTP_Fmt	232	207	SMTP	message/rfc822	SMTP	adENCAPSULATION
MIME_Fmt	233	208	MIME (EML, MBX email) ¹	message/rfc822	EML, MBX	adENCAPSULATION
USENET_Fmt	234	264	USENET	message/news		adWORDPROCESSOR
SGML_Fmt	235	209	SGML	text/sgml	SGML	adWORDPROCESSOR
HTML_Fmt	236	210	HTML	text/html	HTM, HTML	adWORDPROCESSOR
ACT_Fmt	237	211	ACT! CRM software		ACT	adWORDPROCESSOR
PNG_Fmt	238	213	Portable Network Graphics (PNG)	image/png	PNG	adRASTERIMAGE
MS_Video_Fmt	239	214	Video for Windows (AVI)	video/avi	AVI	adMOVIE
Windows_Animated_Cursor_Fmt	240	215	Windows Animated Cursor		ANI	adRASTERIMAGE
Windows_CPP_Obj_Storage_Fmt	241	216	Windows C++ Object Storage			adMIXED
Windows_Palette_Fmt	242	217	Windows Palette		PAL	adRASTERIMAGE
RIFF_DIB_Fmt	243	218	RIFF Device Independent Bitmap			adRASTERIMAGE
RIFF_MIDI_Fmt	244	219	RIFF MIDI	audio/midi	RMI	adSOUND
RIFF_Multimedia_Movie_Fmt	245	220	RIFF Multimedia Movie			adMOVIE
MPEG_Fmt	246	221	MPEG Movie	video/mpeg		adMOVIE
QuickTime_Fmt	247	222	QuickTime Movie, MPEG-4 audio	video/quicktime	MOV, QT, MP4	adMOVIE
AIFF_Fmt	248	223	Audio Interchange File Format (AIFF)	audio/aiff	AIF, AIFF	adSOUND
Amiga_MOD_Fmt	249	224	Amiga MOD		MOD	adSOUND
Amiga_IFF_8SVX_Fmt	250	225	Amiga IFF (8SVX) Sound	audio/x-8svx	IFF	adSOUND

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Creative_Voice_Audio_Fmt	251	226	Creative Voice (VOC)		VOC	adSOUND
AutoDesk_Animator_FLI_Fmt	252	229	AutoDesk Animator FLIC	video/x-fli	FLI	adANIMATION
AutoDesk_AnimatorPro_FLC_Fmt	253	230	AutoDesk Animator Pro FLIC	video/x-flc	FLC	adANIMATION
Compactor_Archive_Fmt	254	233	Compactor / Compact Pro	application/mac-compactpro		adENCAPSULATION
VRML_Fmt	255	234	VRML	model/vrml	WRL	adVECTORGRAPHIC
QuickDraw_3D_Metafire_Fmt	256	235	QuickDraw 3D Metafile			adVECTORGRAPHIC
PGP_Secret_Keyring_Fmt	257	236	PGP Secret Keyring	application/pgp		adENCAPSULATION
PGP_Public_Keyring_Fmt	258	237	PGP Public Keyring	application/pgp		adENCAPSULATION
PGP_Encrypted_Data_Fmt	259	238	PGP Encrypted Data	application/pgp		adENCAPSULATION
PGP_Signed_Data_Fmt	260	239	PGP Signed Data	application/pgp		adENCAPSULATION
PGP_SignedEncrypted_Data_Fmt	261	240	PGP Signed and Encrypted Data	application/pgp		adENCAPSULATION
PGP_Sign_Certificate_Fmt	262	241	PGP Signature Certificate	application/pgp-signature	SIG	adENCAPSULATION
PGP_Compressed_Data_Fmt	263	246	PGP Compressed Data	application/pgp		adENCAPSULATION
PGP_ASCII_Public_Keyring_Fmt	264	242	ASCII-armored PGP Public Keyring	application/pgp	PGP	adENCAPSULATION
PGP_ASCII_Encoded_Fmt	265	243	ASCII-armored PGP encoded	application/pgp		adENCAPSULATION
PGP_ASCII_Signed_Fmt	266	244	ASCII-armored PGP signed	application/pgp		adENCAPSULATION
OLE_DIB_Fmt	267	245	OLE DIB object			adRASTERIMAGE
SGI_Image_Fmt	268	247	SGI Image	image/sgi	RGB	adRASTERIMAGE
Lotus_ScreenCam_Fmt	269	248	Lotus ScreenCam	application/vnd.lotus-screencam	SCM	adANIMATION
MPEG_Audio_Fmt	270	249	MPEG Audio	audio/mpeg	MPEGA, MPG, MP3	adSOUND
FTP_Software_Session_Fmt	271	250	FTP Session Data		STE	adCOMMUNICATION
Netscape_Bookmark_File_Fmt	272	210	Netscape Bookmark File	text/html		adWORDPROCESSOR
Corel_Draw_CMx_Fmt	273	252	Corel CMX	application/cmx	CMX	adVECTORGRAPHIC
AutoDesk_DWG_Fmt	274	253	AutoDesk Drawing (DWG)	image/x-dwg	DWG	adVECTORGRAPHIC
AutoDesk_WHIP_Fmt	275	254	AutoDesk WHIP		WHP	adVECTORGRAPHIC
Macromedia_Director_Fmt	276	255	Macromedia Director	application/x-director	DCR	adANIMATION
Real_Audio_Fmt	277	256	Real Audio	audio/x-pn-realaudio	RM, RA	adSOUND

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MSDOS_Device_Driver_Fmt	278	257	MSDOS Device Driver	application/octet-stream	SYS	adEXECUTABLE
Micrografx_Designer_Fmt	279	258	Micrografx Designer		DSF	adVECTORGRAPHIC
SVF_Fmt	280	259	Simple Vector Format (SVF)	image/x-svf	SVF	adVECTORGRAPHIC
Applix_Words_Fmt	281	261	Applix Words	application/x-applix-word	AW	adWORDPROCESSOR
Applix_Graphics_Fmt	282	262	Applix Graphics		AG	adPRESENTATION
MS_Access_Fmt	283	263	Microsoft Access (versions 1 and 2)	application/x-msaccess	MDB	adDATABASE
MS_Access_95_Fmt	284	263	Microsoft Access 95	application/msaccess	MDB	adDATABASE
MS_Access_97_Fmt	285	263	Microsoft Access 97	application/msaccess	MDB	adDATABASE
MacBinary_Fmt	286	265	MacBinary	application/x-macbinary	BIN	adENCAPSULATION
Apple_Single_Fmt	287	266	Apple Single			adENCAPSULATION
Apple_Double_Fmt	288	267	Apple Double	multipart/appledouble	AD	adENCAPSULATION
Enhanced_Metafile_Fmt	289	270	Enhanced Metafile	image/x-emf	EMF	adVECTORGRAPHIC
MS_Office_Drawing_Fmt	290	271	Microsoft Office Drawing			adVECTORGRAPHIC
XML_Fmt	291	285	XML	text/xml	XML	adWORDPROCESSOR
DeVice_Independent_Fmt	292	274	DeVice Independent file (DVI)	application/x-dvi	DVI	adVECTORGRAPHIC
Unicode_Fmt	293	275	Unicode text file	text/plain	UNI	adWORDPROCESSOR
Lotus_123_Worksheet_Fmt	294	81	Lotus 1-2-3	application/x-lotus-123	WKS, WK1, WK3, WK4	adSPREADSHEET
Lotus_123_Format_Fmt	295	81	Lotus 1-2-3 Formatting	application/x-123	FM3	adSPREADSHEET
Lotus_123_97_Fmt	296	81	Lotus 1-2-3 97	application/x-lotus-123	123	adSPREADSHEET
Lotus_Word_Pro_96_Fmt	297	268	Lotus Word Pro 96	application/vnd.lotus-wordpro	LWP, MWP	adWORDPROCESSOR
Lotus_Word_Pro_97_Fmt	298	268	Lotus Word Pro 97	application/vnd.lotus-wordpro	LWP, MWP	adWORDPROCESSOR
Freelance_DOS_Fmt	299	140	Lotus Freelance for DOS	application/x-freelance	PRZ	adPRESENTATION
Freelance_Win_Fmt	300	140	Lotus Freelance for Windows	application/x-freelance	PRE	adPRESENTATION
Freelance_OS2_Fmt	301	140	Lotus Freelance for OS/2	application/x-freelance	PRS	adPRESENTATION
Freelance_96_Fmt	302	140	Lotus Freelance 96	application/x-freelance	PRZ	adPRESENTATION
Freelance_97_Fmt	303	140	Lotus Freelance 97	application/x-freelance	PRZ	adPRESENTATION
MS_Word_95_Fmt	304	189	Microsoft Word 95	application/msword	DOC	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Word_97_Fmt	305	269	Microsoft Word 97	application/msword	DOC, WPS, WBK	adWORDPROCESSOR
Excel_Fmt	306	90	Microsoft Excel (up to version 5)	application/x-ms-excel	XLS	adSPREADSHEET
Excel_Chart_Fmt	307	90	Microsoft Excel (up to version 5) chart	application/x-ms-excel	XLC	adSPREADSHEET
Excel_Macro_Fmt	308	90	Microsoft Excel (up to version 5) macro	application/vnd.ms-excel	XLM	adSPREADSHEET
Excel_95_Fmt	309	188	Microsoft Excel 95	application/x-ms-excel	XLS	adSPREADSHEET
Excel_97_Fmt	310	188	Microsoft Excel 97	application/x-ms-excel	XLS	adSPREADSHEET
Corel_Presentations_Fmt	311	127	Corel Presentations	application/x-corelpresentations	XFD, XFDL	adPRESENTATION
Harvard_Graphics_Fmt	312	131	Harvard Graphics		PR4	adPRESENTATION
Harvard_Graphics_Chart_Fmt	313	131	Harvard Graphics Chart		CH3, CHT	adVECTORGRAPHIC
Harvard_Graphics_Symbol_Fmt	314	131	Harvard Graphics Symbol File		SY3	adVECTORGRAPHIC
Harvard_Graphics_Cfg_Fmt	315	131	Harvard Graphics Configuration File			adVECTORGRAPHIC
Harvard_Graphics_Palette_Fmt	316	131	Harvard Graphics Palette			adVECTORGRAPHIC
Lotus_123_R9_Fmt	317	81	Lotus 1-2-3 Release 9	application/x-lotus-123	123	adSPREADSHEET
Applix_Spreadsheets_Fmt	318	278	Applix Spreadsheets	application/x-applix-spreadsheet	AS	adSPREADSHEET
MS_Pocket_Word_Fmt	319	45	Microsoft Pocket Word		PWD	adWORDPROCESSOR
MS_DIB_Fmt	320	279	Microsoft Device Independent Bitmap	image/bmp	DIB	adRASTERIMAGE
MS_Word_2000_Fmt	321	269	Microsoft Word 2000	application/msword	DOC	adWORDPROCESSOR
Excel_2000_Fmt	322	188	Microsoft Excel 2000	application/x-ms-excel	XLS	adSPREADSHEET
PowerPoint_2000_Fmt	323	272	Microsoft PowerPoint 2000	application/x-ms-powerpoint	PPT	adPRESENTATION
MS_Access_2000_Fmt	324	263	Microsoft Access 2000	application/x-msaccess	MDB	adDATABASE
MS_Project_4_Fmt	325	281	Microsoft Project 4		MPP	adSCHEDULE
MS_Project_41_Fmt	326	281	Microsoft Project 4.1		MPP	adSCHEDULE
MS_Project_98_Fmt	327	281	Microsoft Project 98	application/vnd.ms-project	MPP	adSCHEDULE
Folio_Flat_Fmt	328	282	Folio Flat File		FFF	adWORDPROCESSOR
HWP_Fmt	329	283	HWP (Arae-Ah Hangul)	application/x-hwp	HWP	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ICHITARO_Fmt	330	284	ICHITARO (v4-10)		JTD	adWORDPROCESSOR
IS_XML_Fmt	331	273	Extended or Custom XML	text/xml	XML	adWORDPROCESSOR
Oasys_Fmt	332	286	Oasys	application/vnd.fujitsu.oasys	OAS, OA2, OA3	adWORDPROCESSOR
PBM_ASC_Fmt	333	287	Portable Bitmap Utilities ASCII format (PBM)	image/pbm	PBM	adRASTERIMAGE
PBM_BIN_Fmt	334	287	Portable Bitmap Utilities BINARY format (PBM)	image/pbm	PBM	adRASTERIMAGE
PGM_ASC_Fmt	335	288	Portable Greymap Utilities ASCII format (PGM)	image/x-pgm	PGM	adRASTERIMAGE
PGM_BIN_Fmt	336	288	Portable Greymap Utilities BINARY format (PGM)	image/x-pgm	PGM	adRASTERIMAGE
PPM_ASC_Fmt	337	289	Portable Pixmap Utilities ASCII format (PPM)	image/x-portable-pixmap	PPM	adRASTERIMAGE
PPM_BIN_Fmt	338	289	Portable Pixmap Utilities BINARY format (PPM)	image/x-portable-pixmap	PPM	adRASTERIMAGE
XBM_Fmt	339	290	X Bitmap format (XBM)	image/x-xbitmap	XBM	adRASTERIMAGE
XPM_Fmt	340	291	X Pixmap format (XPM)	image/xpm	XPM	adRASTERIMAGE
FPX_Fmt	341	292	Kodak FlashPix FPX Image format	image/fpx	FPX	adRASTERIMAGE
PCD_Fmt	342	293	PCD Image format	image/pcd	PCD	adRASTERIMAGE
MS_Visio_Fmt	343	294	Microsoft Visio (up to version 11)	image/x-vsd	VSD	adPRESENTATION
MS_Project_2000_Fmt	344	281	Microsoft Project 2000	application/vnd.ms-project	MPP	adSCHEDULE
MS_Outlook_Fmt	345	295	Microsoft Outlook message	application/vnd.ms-outlook	MSG, OFT	adENCAPSULATION
ELF_Relocatable_Fmt	346	159	ELF Relocatable	application/octet-stream	O	adOBJECTMODULE
ELF_Executable_Fmt	347	158	ELF Executable	application/octet-stream		adEXECUTABLE
ELF_Dynamic_Lib_Fmt	348	160	ELF Dynamic Library	application/octet-stream	SO	adLIBRARY
MS_Word_XML_Fmt	349	285	Microsoft Word 2003 XML	text/xml	XML	adWORDPROCESSOR
MS_Excel_XML_Fmt	350	285	Microsoft Excel 2003 XML	text/xml	XML	adWORDPROCESSOR
MS_Visio_XML_Fmt	351	285	Microsoft Visio 2003 XML	text/xml	VDX	adWORDPROCESSOR
SO_Text_XML_Fmt	352	314	OpenDocument format (OpenOffice 1/StarOffice 6,7) Text XML	application/vnd.sun.xml.writer	SXW	adWORDPROCESSOR
SO_Spreadsheet_XML_Fmt	353	315	OpenDocument format	application/vnd.sun.xml.calc	SXC, STC	adSPREADSHEET

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			(OpenOffice 1/StarOffice 6,7) Spreadsheet XML			
SO_Presentation_XML_Fmt	354	316	OpenDocument format (OpenOffice 1/StarOffice 6,7) Presentation XML	application/vnd.sun.xml.impress	SXD, SXI	adPRESENTATION
XHTML_Fmt	355	296	XHTML	text/xhtml	XML, ASP	adWORDPROCESSOR
MS_OutlookPST_Fmt	356	297	Microsoft Outlook Personal Folders File (.pst)	application/vnd.ms-outlook-pst	PST	adENCAPSULATION
RAR_Fmt	357	298	RAR archive format	application/x-rar-compressed	RAR	adENCAPSULATION
Lotus_Notes_NSF_Fmt	358	299	IBM Lotus Notes Database NSF/NTF	application/x-lotus-notes	NSF	adENCAPSULATION
Macromedia_Flash_Fmt	359	300	Macromedia Flash (.swf)	application/x-shockwave-flash	SWF, SWD	adWORDPROCESSOR
MS_Word_2007_Fmt	360	301	Microsoft Word 2007 XML - Docx	application/x-ms-word07	DOCX, DOTX	adWORDPROCESSOR
MS_Excel_2007_Fmt	361	302	Microsoft Excel 2007 XML	application/x-ms-excel07	XLSX, XLTX	adSPREADSHEET
MS_PPT_2007_Fmt	362	303	Microsoft PowerPoint 2007 XML	application/x-ms-powerpoint07	PPTX, POTX, PPSX	adPRESENTATION
OpenPGP_Fmt	363	304	OpenPGP Message Format (with new packet format)	application/pgp-encrypted	PGP	adENCAPSULATION
Intergraph_V7_DGN_Fmt	364	305	Intergraph Standard File Format (ISFF) V7 DGN (non-OLE)		DGN	adVECTORGRAPHIC
MicroStation_V8_DGN_Fmt	365	306	MicroStation V8 DGN (OLE)		DGN	adVECTORGRAPHIC
MS_Word_Macro_2007_Fmt	366	307	Microsoft Word Macro 2007 XML	application/x-ms-word07m	DOCM, DOTM	adWORDPROCESSOR
MS_Excel_Macro_2007_Fmt	367	308	Microsoft Excel Macro 2007 XML	application/x-ms-excel07m	XLSM, XLTM, XLAM	adSPREADSHEET
MS_PPT_Macro_2007_Fmt	368	309	Microsoft PPT Macro 2007 XML	application/x-ms-powerpoint07m	PPTM, POTM, PPSM, PPAM	adPRESENTATION
LZH_Fmt	369	310	LZH Archive	application/x-lzh-compressed	LZH, LHA	adENCAPSULATION
Office_2007_Fmt	370	311	Office 2007 document		XLSB	adMISC
MS_XPS_Fmt	371	312	Microsoft XML Paper Specification (XPS)	application/vnd.ms-xpsdocument	XPS	adWORDPROCESSOR
Lotus_Domino_DXL_Fmt	372	313	IBM Domino Data in XML format (.dxl)	text/xml	DXL	adENCAPSULATION
ODF_Text_Fmt	373	314	ODF Text	application/vnd.oasis.opendocument.text	ODT	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ODF_Spreadsheet_Fmt	374	315	ODF Spreadsheet	application/vnd.oasis.opendocument.spreadsheet	ODS	adSPREADSHEET
ODF_Presentation_Fmt	375	316	ODF Presentation	application/vnd.oasis.opendocument.presentation	ODP	adPRESENTATION
Legato_Extender_ONM_Fmt	376	317	Legato Extender Native Message ONM	application/x-lotus-notes	ONM	adENCAPSULATION
bin_Unknown_Fmt	377	318	Bin unknown format (.xxx)			adWORDPROCESSOR
TNEF_Fmt	378	319	Transport Neutral Encapsulation Format (TNEF)	application/vnd.ms-tnef		adENCAPSULATION
CADAM_Drawing_Fmt	379	320	CADAM Drawing		CDD	adVECTORGRAPHIC
CADAM_Drawing_Overlay_Fmt	380	321	CADAM Drawing Overlay		CDO	adVECTORGRAPHIC
NURSTOR_Drawing_Fmt	381	322	NURSTOR Drawing		NUR	adVECTORGRAPHIC
HP_GLP_Fmt	382	323	HP Graphics Language (Plotter)	vector/x-hpgl2	HPG	adVECTORGRAPHIC
ASF_Fmt	383	324	Advanced Systems Format (ASF)	application/x-ms-asf	ASF	adMISC
WMA_Fmt	384	325	Windows Media Audio Format (WMA)	audio/x-ms-wma	WMA	adSOUND
WMV_Fmt	385	326	Windows Media Video Format (WMV)	video/x-ms-wmv	WMV	adMOVIE
EMX_Fmt	386	327	Legato EMailXtender Archives Format (EMX)		EMX	adENCAPSULATION
Z7Z_Fmt	387	328	7 Zip Format (7z)	application/7z	7Z	adENCAPSULATION
MS_Excel_Binary_2007_Fmt	388	329	Microsoft Excel Binary 2007	application/vnd.ms-excel.sheet.binary.macroenabled.12	XLSB	adSPREADSHEET
CAB_Fmt	389	330	Microsoft Cabinet File (CAB)	application/vnd.ms-cab-compressed	CAB	adENCAPSULATION
CATIA_Fmt	390	331	CATIA Formats (CAT*)		CATPART, CATPRODUCT 2	adVECTORGRAPHIC
YIM_Fmt	391	332	Yahoo Instant Messenger History		DAT	adWORDPROCESSOR
ODF_Drawing_Fmt	392	316	ODF Drawing/Graphics	application/vnd.oasis.opendocument.graphics	ODG	adVECTORGRAPHIC
Founder_CEB_Fmt	393	333	Founder Chinese E-paper Basic (ceb)	application/ceb	CEB	adWORDPROCESSOR
QPW_Fmt	394	334	Corel Quattro Pro 9+ for Windows	application/quattro-pro	QPW	adSPREADSHEET
MHT_Fmt	395	335	MHTML format (MHT) ¹	multipart/related	MHT, MHTML	adWORDPROCESSOR
MDI_Fmt	396	336	Microsoft Document Imaging Format	image/vnd.ms-modi	MDI	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
GRV_Fmt	397	337	Microsoft Office Groove Format	application/vnd.groove-injector	GRV	adWORDPROCESSOR
IWWP_Fmt	398	338	Apple iWork Pages format	application/vnd.apple.pages	PAGES	adWORDPROCESSOR
IWSS_Fmt	399	339	Apple iWork Numbers format	application/vnd.apple.numbers	NUMBERS	adSPREADSHEET
IWPG_Fmt	400	340	Apple iWork Keynote format	application/vnd.apple.keynote	KEY	adPRESENTATION
BKF_Fmt	401	341	Windows Backup File		BKF	adENCAPSULATION
MS_Access_2007_Fmt	402	342	Microsoft Access 2007	application/msaccess	ACCDB	adDATABASE
ENT_Fmt	403	343	Microsoft Entourage Database Format			adENCAPSULATION
DMG_Fmt	404	344	Mac Disk Copy Disk Image File	application/x-apple-diskimage	DMG	adENCAPSULATION
CWK_Fmt	405	345	AppleWorks File	application/appleworks	CWK	adWORDPROCESSOR
OO3_Fmt	406	346	Omni Outliner V3 File		OO3	adWORDPROCESSOR
OPML_Fmt	407	347	Omni Outliner OPML File		OPML	adWORDPROCESSOR
Omni_Graffle_XML_Fmt	408	348	Omni Graffle XML File		GRAFFLE	adVECTORGRAPHIC
PSD_Fmt	409	349	Photoshop Document	image/vnd.adobe.photoshop	PSD, PSB	adRASTERIMAGE
Apple_Binary_PLList_Fmt	410	350	Apple Binary Property List format		PLIST	adMISC
Apple_iChat_Fmt	411	351	Apple iChat format		ICHAT	adWORDPROCESSOR
OOUTLINE_Fmt	412	352	OOutliner File		OOUTLINE	adWORDPROCESSOR
BZIP2_Fmt	413	353	Bzip 2 Compressed File	application/x-bzip2	BZ2	adENCAPSULATION
ISO_Fmt	414	354	ISO-9660 CD Disc Image Format	application/x-iso9660-image	ISO	adENCAPSULATION
DocuWorks_Fmt	415	355	DocuWorks Format	application/vnd.fujixerox.docuworks	XDW	adWORDPROCESSOR
RealMedia_Fmt	416	356	RealMedia Streaming Media	application/vnd.rn-realmedia	RM, RA	adMOVIE
AC3Audio_Fmt	417	357	AC3 Audio File Format	audio/ac3	AC3	adSOUND
NEF_Fmt	418	358	Nero Encrypted File		NEF	adENCAPSULATION
SolidWorks_Fmt	419	359	SolidWorks Format Files		SLDASM, SLDPRT, SLDDRW, SLDDRT	adVECTORGRAPHIC
XFDL_Fmt	420	366	Extensible Forms Description Language	application/x-xfdl	XFDL, XFD	adPRESENTATION
Apple_XML_PLList_Fmt	421	367	Apple XML Property List format		PLIST	adMISC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
OneNote_Fmt	422	368	OneNote Note Format	application/onenote	ONE	adWORDPROCESSOR
IFilter_Fmt	423	369	iFilter			adWORDPROCESSOR
Dicom_Fmt	424	370	Digital Imaging and Communications in Medicine (Dicom)	application/dicom	DCM	adRASTERIMAGE
EnCase_Fmt	425	371	Expert Witness Compression Format (EnCase)		E01, L01, Lx01	adENCAPSULATION
Scrap_Fmt	426	372	Shell Scrap Object File		SHS	adENCAPSULATION
MS_Project_2007_Fmt	427	373	Microsoft Project 2007	application/vnd.ms-project	MPP	adSCHEDULE
MS_Publisher_98_Fmt	428	374	Microsoft Publisher from version 98	application/x-mspublisher	PUB	adDESKTOPPUBLSH
Skype_Fmt	429	375	Skype Log File		DBB	adWORDPROCESSOR
HL7_Fmt	430	377	Health level7 message		HL7	adWORDPROCESSOR
MS_OutlookOST_Fmt	431	378	Microsoft Outlook Offline Folders File (OST)	application/vnd.ms-outlook-pst	OST	adENCAPSULATION
Epub_Fmt	432	379	Electronic Publication	application/epub+zip	EPUB	adWORDPROCESSOR
MS_OEDBX_Fmt	433	380	Microsoft Outlook Express DBX Message Database		DBX	adENCAPSULATION
BB_Activ_Fmt	434	381	BlackBerry Activation File		DAT	adWORDPROCESSOR
DiskImage_Fmt	435	382	Disk Image		DMG	adENCAPSULATION
Milestone_Fmt	436	383	Milestone Document		MLS, ML3, ML4, ML5, ML6, ML7, ML8, ML9, MLA	adRASTERIMAGE
E_Transcript_Fmt	437	384	RealLegal E-Transcript File		PTX	adWORDPROCESSOR
PostScript_Font_Fmt	438	385	PostScript Type 1 Font	application/x-font	PFB	adFONT
Ghost_DiskImage_Fmt	439	386	Ghost Disk Image File		GHO, GHS	adENCAPSULATION
JPEG_2000_JP2_File_Fmt	440	387	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	image/jp2	JP2, JPF, J2K, JPWL, JPX, PGX	adRASTERIMAGE
Unicode_HTML_Fmt	441	388	Unicode HTML	text/html	HTM, HTML	adWORDPROCESSOR
CHM_Fmt	442	389	Microsoft Compiled HTML Help	application/x-chm	CHM	adENCAPSULATION
EMCMF_Fmt	443	390	Documentum EMCMF format		EMCMF	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Access_2007_Tmpl_Fmt	444	391	Microsoft Access 2007 Template		ACCDT	adDATABASE
Jungum_Fmt	445	392	Samsung Electronics Jungum Global document		GUL	adWORDPROCESSOR
JBIG2_Fmt	446	393	JBIG2 File Format	image/jbig2	JB2, JBIG2	adRASTERIMAGE
EFax_Fmt	447	394	eFax file		EFX	adRASTERIMAGE
AD1_Fmt	448	395	AD1 Evidence file		AD1	adENCAPSULATION
SketchUp_Fmt	449	396	Google SketchUp		SKP	adVECTORGRAPHIC
GWFS_Email_Fmt	450	397	Group Wise File Surf email		GWFS	adENCAPSULATION
JNT_Fmt	451	398	Windows Journal format		JNT	adWORDPROCESSOR
Yahoo_yChat_Fmt	452	399	Yahoo! Messenger chat log		YCHAT	adWORDPROCESSOR
PaperPort_MAX_File_Fmt	453	400	PaperPort MAX image file	image/max	MAX	adRASTERIMAGE
ARJ_Fmt	454	402	ARJ (Archive by Robert Jung) file format	application/arj	ARJ	adENCAPSULATION
RPMSG_Fmt	455	403	Microsoft Outlook Restricted Permission Message	application/x-microsoft-rpmsg-message	RPMSG	adENCAPSULATION
MAT_Fmt	456	404	MATLAB file format	application/x-matlab-data	MAT, FIG	adWORDPROCESSOR
SGY_Fmt	457	405	SEG-Y Seismic Data format		SGY, SEGY	adWORDPROCESSOR
CDXA_MPEG_PS_Fmt	458	406	MPEG-PS container with CDXA stream	video/mpeg	MPG	adMOVIE
EVT_Fmt	459	407	Microsoft Windows NT Event Log		EVT	adMISC
EVTX_Fmt	460	408	Microsoft Windows Vista Event Log		EVTX	adMISC
MS_OutlookOLM_Fmt	461	409	Microsoft Outlook for Macintosh format		OLM	adENCAPSULATION
WARC_Fmt	462	410	Web ARChive	application/warc	WARC	adENCAPSULATION
JAVACLASS_Fmt	463	411	Java Class format	application/x-java-class	CLASS	adWORDPROCESSOR
VCF_Fmt	464	412	Microsoft Outlook vCard file format	text/vcard	VCF	adWORDPROCESSOR
EDB_Fmt	465	413	Microsoft Exchange Server Database file format		EDB	adENCAPSULATION
ICS_Fmt	466	414	Microsoft Outlook iCalendar file format	text/calendar	ICS, VCS	adENCAPSULATION
MS_Visio_2013_Fmt	467	415	Microsoft Visio 2013	application/vnd.visio	VSDX, VSTX,	adPRESENTATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
					VSSX	
MS_Visio_2013_Macro_Fmt	468	415	Microsoft Visio 2013 macro	application/vnd.visio	VSDM, VSTM, VSSM	adPRESENTATION
ICHITARO_Compr_Fmt	469	417	ICHITARO Compressed format	application/x-js-taro	JTDC	adWORDPROCESSOR
IWWP13_Fmt	470	418	Apple iWork 2013 Pages format		IWA, PAGES	adWORDPROCESSOR
IWSS13_Fmt	471	419	Apple iWork 2013 Numbers format		IWA, NUMBERS	adSPREADSHEET
IWPG13_Fmt	472	420	Apple iWork 2013 Keynote format		IWA, KEY	adPRESENTATION
XZ_Fmt	473	421	XZ archive format	application/x-xz	XZ	adENCAPSULATION
Sony_WAVE64_Fmt	474	422	Sony Wave64 format	audio/wav64	W64	adSOUND
Conifer_WAVPACK_Fmt	475	423	Conifer Wavpack format	audio/x-wavpack	WV	adSOUND
Xiph_OGG_VORBIS_Fmt	476	424	Xiph Ogg Vorbis format	audio/ogg	OGG	adSOUND
MS_Visio_2013_Stencil_Fmt	477	415	MS Visio 2013 stencil format	application/vnd.visio	VSSX	adPRESENTATION
MS_Visio_2013_Stencil_Macro_Fmt	478	415	MS Visio 2013 stencil Macro format	application/vnd.visio	VSSM	adPRESENTATION
MS_Visio_2013_Template_Fmt	479	415	MS Visio 2013 template format	application/vnd.visio	VSTX	adPRESENTATION
MS_Visio_2013_Template_Macro_Fmt	480	415	MS Visio 2013 template Macro format	application/vnd.visio	VSTM	adPRESENTATION
Borland_Reflex_2_Fmt	481	425	Borland Reflex 2 format		R2D	adDATABASE
PKCS_12_Fmt	482	426	PKCS #12 (p12) format	application/x-pkcs12	P12, PFX	adWORDPROCESSOR
B1_Fmt	483	427	B1 format	application/x-b1	B1	adENCAPSULATION
ISO_IEC_MPEG_4_Fmt	484	428	ISO/IEC MPEG-4 (ISO 14496) format	video/mp4	MP4	adMOVIE
RAR5_Fmt	485	429	RAR5 Format	application/x-rar-compressed	RAR	adENCAPSULATION
Unigraphics_NX_Fmt	486	362	Unigraphics (UG) NX CAD Format		PRT	adVECTORGRAPHIC
PTC_Creo_Fmt	487	430	PTC Creo CAD Format		ASM, PRT	adVECTORGRAPHIC
KML_Fmt	488	431	Keyhole Markup Language	application/vnd.google-earth.kml+xml	KML	adWORDPROCESSOR
KMZ_Fmt	489	432	Zipped Keyhole Markup Language	application/vnd.google-earth.kmz	KMZ	adWORDPROCESSOR
WML_Fmt	490	433	Wireless Markup Language	text/vnd.wap.wml	WML	adWORDPROCESSOR
ODF_Formula_Fmt	491	434	ODF Formula	application/vnd.oasis.opendocument.formula	ODF	adWORDPROCESSOR
SO_Text_Fmt	492	435	Star Office 4,5 Writer Text	application/vnd.stardivision.writer	SDW, SGL,	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
					VOR	
SO_Spreadsheet_Fmt	493	436	Star Office 4,5 Calc Spreadsheet	application/vnd.stardivision.calc	SDC	adSPREADSHEET
SO_Presentation_Fmt	494	437	Star Office 4,5 Impress Presentation	application/vnd.stardivision.draw	SDD, SDA	adPRESENTATION
SO_Math_Fmt	495	438	Star Office 4,5 Math	application/vnd.stardivision.math	SMF	adMISC
STEP_Fmt	496	439	ISO 10303-21 STEP format			adMISC
STL_Fmt	497	364	3D Systems STL ASCII format			adMISC
AppleScript_Fmt	498	440	AppleScript Source Code ³	text/x-applescript	APPLESCRIPT	adSOURCECODE
Assembly_Fmt	499	441	Assembly Code ³	text/x-assembly		adSOURCECODE
C_Fmt	500	442	C Source Code ³	text/x-c	C, H	adSOURCECODE
Csharp_Fmt	501	443	C# Source Code ³	text/x-csharp	CS	adSOURCECODE
CPlusPlus_Fmt	502	444	C++ Source Code ³	text/x-c++	CPP, HPP	adSOURCECODE
Css_Fmt	503	445	Cascading Style Sheet ³	text/css	CSS	adSOURCECODE
Clojure_Fmt	504	446	Clojure Source Code ³	text/x-clojure	CLJ, CL2	adSOURCECODE
CoffeeScript_Fmt	505	447	CoffeeScript Source Code ³	text/x-coffeescript	COFFEE, CAKE	adSOURCECODE
Lisp_Fmt	506	448	Common Lisp Source Code ³	text/x-common-lisp	EL	adSOURCECODE
Dockerfile_Fmt	507	449	Dockerfile ³	text/x-dockerfile		adSOURCECODE
Eiffel_Fmt	508	450	Eiffel Source Code ³	text/x-eiffel	E	adSOURCECODE
Erlang_Fmt	509	451	Erlang Source Code ³	text/x-erlang	ERL, ES	adSOURCECODE
Fsharp_Fmt	510	452	F# Source Code ³	text/x-fsharp	FS	adSOURCECODE
Fortran_Fmt	511	453	Fortran Source Code ³	text/x-fortran	F	adSOURCECODE
Go_Fmt	512	454	Go Source Code ³	text/x-go	GO	adSOURCECODE
Groovy_Fmt	513	455	Groovy Source Code ³	text/x-groovy	GRT, GVY	adSOURCECODE
Haskell_Fmt	514	456	Haskell Source Code ³	text/x-haskell	HS	adSOURCECODE
Ini_Fmt	515	457	Initialization (INI) file ³	text/x-ini		adSOURCECODE
Java_Fmt	516	458	Java Source Code ³	text/x-java-source	JAVA	adSOURCECODE
Javascript_Fmt	517	459	Javascript Source Code ³	text/javascript	JS	adSOURCECODE
Lua_Fmt	518	460	Lua Source Code ³	text/x-lua	LUA	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Makefile_Fmt	519	461	Makefile ³	text/x-makefile	MAKE	adSOURCECODE
Mathematica_Fmt	520	462	Wolfram Mathematica Source Code ³	text/x-mathematica	M	adSOURCECODE
ObjC_Fmt	521	464	Objective-C Source Code ³	text/x-objc		adSOURCECODE
ObjCpp_Fmt	522	465	Objective-C++ Source Code ³	text/x-objectivec++		adSOURCECODE
ObjJ_Fmt	523	466	Objective-J Source Code ³	text/x-objectivej	J	adSOURCECODE
PHP_Fmt	524	467	PHP Source Code ³	text/x-php	PHP	adSOURCECODE
PLSQL_Fmt	525	468	PLSQL Source Code ³	text/x-plsql		adSOURCECODE
Pascal_Fmt	526	469	Pascal Source Code ³	text/x-pascal	PASCAL	adSOURCECODE
Perl_Fmt	527	470	Perl Source Code ³	text/x-perl	PL	adSOURCECODE
Powershell_Fmt	528	471	PowerShell Source Code ³	text/x-powershell	PS1	adSOURCECODE
Prolog_Fmt	529	472	Prolog Source Code ³	text/x-prolog	PRO, PROLOG	adSOURCECODE
Puppet_Fmt	530	473	Puppet Source Code ³	text/x-puppet	PP	adSOURCECODE
Python_Fmt	531	474	Python Source Code ³	text/x-python	PY	adSOURCECODE
R_Fmt	532	475	R Source Code ³	text/x-rsrc	R	adSOURCECODE
Ruby_Fmt	533	476	Ruby Source Code ³	text/x-ruby	RB	adSOURCECODE
Rust_Fmt	534	477	Rust Source Code ³	text/x-rust	RS	adSOURCECODE
Scala_Fmt	535	478	Scala Source Code ³	text/x-scala	SC	adSOURCECODE
Shell_Fmt	536	479	Shell Script ³	application/x-sh	SH	adSOURCECODE
Smalltalk_Fmt	537	480	Smalltalk Source Code ³	text/x-stsrc	ST	adSOURCECODE
ML_Fmt	538	481	Standard ML Source Code ³	text/x-ml	ML	adSOURCECODE
Swift_Fmt	539	482	Swift Source Code ³	text/x-swift	SWIFT	adSOURCECODE
Tcl_Fmt	540	483	Tool Command Language (Tcl) Source Code ³	text/x-tcl	TM	adSOURCECODE
Tex_Fmt	541	484	TeX Typesetting File ³	application/x-tex		adSOURCECODE
TypeScript_Fmt	542	485	TypeScript Source Code ³	text/x-typescript	TS	adSOURCECODE
Verilog_Fmt	543	486	Verilog Source Code ³	text/x-verilog	V	adSOURCECODE
YAML_Fmt	544	487	YAML File ³	text/x-yaml	YML	adSOURCECODE
Wiki_Fmt	545	488	MediaWiki File	text/x-mediawiki		adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Word_2007_Flat_XML_Fmt	546	301	Microsoft Word 2007 XML - Flat xml	text/xml	XML	adWORDPROCESSOR
Matroska_Fmt	547	489	Matroska video File	video/x-matroska	MKV	adMOVIE
SVG_Fmt	548	490	Scalable Vector Graphics image	image/svg+xml	SVG	adVECTORGRAPHIC
Shapefile_Fmt	549	491	Shapefile	application/x-shapefile	SHP, SHX	adGIS
Flash_Video_Fmt	550	492	Flash video File	video/x-flv	FLV	adMOVIE
Embedded_OpenType_Fmt	551	493	Embedded OpenType font	application/vnd.ms-fontobject	EOT	adFONT
Web_Open_Font_Fmt	552	494	Web Open Font Format	font/woff	WOFF, WOFF2	adFONT
OpenType_Fmt	553	495	OpenType Font	font/otf	OTF	adFONT
MNG_Fmt	554	496	Multiple-image Network Graphics	video/x-mng	MNG	adANIMATION
JNG_Fmt	555	497	JPEG Network Graphics	image/x-jng	JNG	adRASTERIMAGE
AppleScript_Binary_Fmt	556	498	AppleScript Binary Source Code		SCPT	adSOURCECODE
Maya_Binary_Fmt	557	499	Autodesk Maya binary file		MB	adCAD
Jupiter_Tesselation_Fmt	558	363	UGS Jupiter Tesselation file		JT	adCAD
OGV_Fmt	559	500	Ogg Theora Video format	video/ogg	OGV	adMOVIE
OGG_Container_Fmt	560	501	General Ogg Container format	application/ogg	OGG	adMISC
GNU_Message_Catalog_Fmt	561	502	GNU Message Catalog format		MO	adMISC
Windows_Shortcut_Fmt	562	503	Windows shortcut file	application/x-ms-shortcut	LNK	adMISC
Apple_Typedstream_Fmt	563	504	Apple/NeXT typedstream data format			adMISC
XCF_Fmt	564	505	GIMP XCF image	image/x-xcf	XCF	adRASTERIMAGE
PaintShop_Pro_Fmt	565	506	PaintShop Pro image		PSP, PSPIMAGE	adRASTERIMAGE
SQLite_Database_Fmt	566	507	SQLite database format	application/x-sqlite3	QHC	adDATABASE
MySQL_Table_Fmt	567	508	MySQL table definition file		FRM	adDATABASE
Microsoft_Program_DB_Fmt	568	509	Microsoft Program Database format		PDB	adDATABASE
OpenEXR_Fmt	569	510	OpenEXR image format		EXR	adRASTERIMAGE
XMV_Fmt	570	511	4X Movie File			adMOVIE
AMV_Fmt	571	512	AMV video file		AMV	adMOVIE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
NIFF_Fmt	572	513	Notation Interchange File Format		NIF	adSOUND
CuBase_Fmt	573	514	Steinberg CuBase file			adSOUND
SoundFont_Fmt	574	515	SoundFont file			adSOUND
WebP_Fmt	575	516	WebP image	image/webp	WEBP	adRASTERIMAGE
ICC_Fmt	576	517	International Color Consortium files	application/vnd.iccprofile	ICC, ICM	adMISC
PCF_Fmt	577	518	X11 Portable Compiled Font file	application/x-font-pcf	PCF	adFONT
WebM_Fmt	578	519	WebM video file	video/webm	WEBM	adMOVIE
AMFF_Fmt	579	520	Amiga Metafile		AMF	adVECTORGRAPHIC
ANBM_Fmt	580	521	IFF Animated Bitmap			adRASTERIMAGE
ANIM_Fmt	581	522	IFF Amiga animated raster graphics format			adRASTERIMAGE
DEEP_Fmt	582	523	IFF-DEEP TVPaint image		DEEP	adRASTERIMAGE
FAXX_Fmt	583	524	IFF-FAXX Facsimile image			adRASTERIMAGE
ICON_Fmt	584	525	IFF Glow Icon image			adRASTERIMAGE
ILBM_Fmt	585	526	Interleaved BitMap image		IFF	adRASTERIMAGE
LWOB_Fmt	586	527	LightWave Object format		LWOB	adMISC
MAUD_Fmt	587	528	IFF-MAUD MacroSystem audio format			adSOUND
PBM_Fmt	588	529	IFF Planar BitMap			adRASTERIMAGE
TDDD_Fmt	589	530	IFF TDDD and Imagine Object animation format		TDD	adRASTERIMAGE
DjVu_Fmt	590	531	AT&T DjVu format	image/vnd.djvu	DJVU	adWORDPROCESSOR
InDesign_Fmt	591	532	Adobe InDesign document	application/x-indesign		adDESKTOPPUBLSH
Calamus_Fmt	592	533	Calamus Desktop Publishing			adDESKTOPPUBLSH
Adaptive_MultiRate_Fmt	593	534	Adaptive Multi-Rate audio format	audio/amr	AMR	adSOUND
FLAC_Fmt	594	535	Free Lossless Audio Codec format	audio/flac	FLAC	adSOUND
Ogg_FLAC_Fmt	595	536	Ogg Container FLAC audio format		OGG	adSOUND
SAS7BDAT_Fmt	596	537	SAS7BDAT database storage format		SAS7BDAT	adDATABASE
Design_Web_Format_Fmt	597	538	Autodesk Design Web Format	model/vnd.dwf	DWF	adCAD

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Adobe_Flash_Audio_Book_Fmt	598	539	Adobe Flash Player audio book	audio/mp4	F4B	adSOUND
Adobe_Flash_Audio_Fmt	599	540	Adobe Flash Player audio	audio/mp4	F4A	adSOUND
Adobe_Flash_Protected_Video_Fmt	600	541	Adobe Flash Player protected video	video/mp4	F4P	adMOVIE
Adobe_Flash_Video_Fmt	601	542	Adobe Flash Player video	video/x-f4v	F4V	adMOVIE
Audible_Audiobook_Fmt	602	543	Audible Enhanced Audiobook	audio/vnd.audible.aax	AAX	adSOUND
Canon_Camera_Fmt	603	544	Canon Digital Camera image			adRASTERIMAGE
Canon_Raw_Fmt	604	545	Canon Raw image		CR3	adRASTERIMAGE
Casio_Camera_Fmt	605	546	Casio Digital Camera image			adRASTERIMAGE
Convergent_Design_Fmt	606	547	Convergent Design file			adRASTERIMAGE
DMB_MAF_Audio_Fmt	607	548	DMB MAF audio			adSOUND
DMB_MAF_Video_Fmt	608	549	DMB MAF video			adMOVIE
DMP_Content_Fmt	609	550	Digital Media Project Content Format			adMISC
DVB_Fmt	610	551	Digital Video Broadcast format	video/vnd.dvb.file	DVB	adMOVIE
Dirac_Wavelet_Compression_Fmt	611	552	ISO-BMFF Dirac Wavelet compression			adMISC
HEICS_Image_Sequence_Fmt	612	553	High Efficiency Image Format HEVC image sequence	image/heic-sequence	HEICS	adRASTERIMAGE
HEIC_Image_Fmt	613	554	High Efficiency Image Format HEVC image	image/heic	HEIC	adRASTERIMAGE
HEIFS_Image_Sequence_Fmt	614	555	High Efficiency Image Format image sequence	image/heif-sequence	HEIFS	adRASTERIMAGE
HEIF_Image_Fmt	615	556	High Efficiency Image Format image	image/heif	HEIF	adRASTERIMAGE
ISMACryp_Fmt	616	557	ISMACryp 2.0 Encrypted format			adENCAPSULATION
ISO_3GPP2_Fmt	617	558	3GPP2 video file	video/3gpp2	3G2	adMOVIE
ISO_3GPP_Fmt	618	559	3GPP video file	video/3gpp	3GP	adMOVIE
ISO_JPEG2000_JP2_Fmt	619	560	ISO-BMFF JPEG 2000 image	image/jp2	JP2	adRASTERIMAGE
ISO_JPEG2000_JPM_Fmt	620	561	ISO-BMFF JPEG 2000 compound image	image/jpm	JPM	adRASTERIMAGE
ISO_JPEG2000_JPX_Fmt	621	562	ISO-BMFF JPEG 2000 with extensions	image/jpx	JPX	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ISO_QuickTime_Fmt	622	563	Apple ISO-BMFF QuickTime video	video/quicktime	QT, MOV	adMOVIE
KDDI_Video_Fmt	623	564	KDDI Video file	video/3gpp2		adMOVIE
MAF_Photo_Player_Fmt	624	565	MAF Photo Player			adMISC
MPEG4_AVC_Fmt	625	566	ISO-BMFF MPEG-4 with AVC extension	video/mp4		adMOVIE
MPEG4_M4A_Fmt	626	567	Apple MPEG-4 Part 14 audio	audio/x-m4a	M4A	adSOUND
MPEG4_M4B_Fmt	627	568	Apple MPEG-4 Part 14 audio book	audio/mp4	M4B	adSOUND
MPEG4_M4P_Fmt	628	569	Apple MPEG-4 Part 14 protected audio	audio/mp4	M4P	adSOUND
MPEG4_M4V_Fmt	629	570	Apple MPEG-4 Part 14 video	video/x-m4v	M4V	adMOVIE
MPEG4_Sony_PSP_Fmt	630	571	Sony PSP MPEG-4	audio/mp4	MP4	adSOUND
MPEG_21_Fmt	631	572	MPEG-21	audio/mp4		adMISC
Mobile_QuickTime_Fmt	632	573	Mobile QuickTime video	video/quicktime	MQV	adMOVIE
Motion_JPEG_2000_Fmt	633	574	Motion JPEG 2000	video/mj2	MJ2, MJP2	adMOVIE
NTT_MPEG4_Fmt	634	575	NTT MPEG-4	video/mp4		adMOVIE
Nero_MPEG4_AVC_Profile	635	576	Nero MPEG-4 profile with AVC extension	video/mp4		adMOVIE
Nero_MPEG4_Audio_Fmt	636	577	Nero AAC audio	audio/mp4		adSOUND
Nero_MPEG4_Profile	637	578	Nero MPEG-4 profile	video/mp4		adMOVIE
OMA_DRM_Fmt	638	579	OMA DRM Format			adMISC
Panasonic_Camera_Fmt	639	580	Panasonic Digital Camera image			adRASTERIMAGE
Ross_Video_Fmt	640	581	Ross video			adMOVIE
SDA_Video_Fmt	641	582	SDA SD Memory Card video			adMOVIE
Samsung_Stereoscopic_Fmt	642	583	Samsung stereoscopic stream			adMISC
Sony_XAVC_Fmt	643	584	Sony XAVC video			adMOVIE
JPEG_2000_PGX_Fmt	644	585	JPEG 2000 PGX Verification Model image		PGX	adRASTERIMAGE
Apple_Desktop_Services_Store_Fmt	645	586	Apple Desktop Services Store file		DS_Store	adMISC
Core_Audio_Fmt	646	587	Apple Core Audio Format	audio/x-caf	CAF	adSOUND
VICAR_Fmt	647	588	VICAR image format		IMG	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
FITS_Fmt	648	589	Flexible Image Transport System FITS image	image/fits	FIT	adRASTERIMAGE
DIF_Fmt	649	590	Digital Interface Format (DIF) DV video		DV	adMOVIE
MPEG_Transport_Stream_Fmt	650	591	MPEG Transport Stream data	video/MP2T	TS	adMISC
MPEG_Sequence_Fmt	651	592	MPEG Sequence format	video/mpeg		adMISC
Ogg_OGM_Fmt	652	593	Ogg OGM video format	video/ogg	OGM	adMOVIE
Ogg_Speex_Fmt	653	594	Ogg Speex audio format	audio/ogg	SPX	adSOUND
Ogg_Opus_Fmt	654	595	Ogg Opus audio format	audio/ogg	OGG	adSOUND
Musepack_Audio_Fmt	655	596	Musepack audio format	audio/x-musepack	MPC	adSOUND
ART_Image_Fmt	656	597	ART image format		ART	adRASTERIMAGE
Vivo_Fmt	657	598	Vivo audio-video format	video/vnd.vivo	VIV	adMOVIE
QCP_Fmt	658	599	Qualcomm QCP audio	audio/qcelp	QCP	adSOUND
CSP_Codec_Fmt	659	600	Creative Signal Processor codec		CSP	adMISC
TwinVQ_Fmt	660	601	NTT TwinVQ audio format		VQF	adSOUND
Interplay_MVE_Fmt	661	602	Interplay MVE video format		MVE	adMOVIE
IRIX_Moviemaker_Fmt	662	603	IRIX Silicon Graphics moviemaker video file	video/x-sgi-movie	MV, MOVIE	adMOVIE
Sega_FILM_Fmt	663	604	Sega FILM video format		CPK, CAK	adMOVIE
SMAF_Fmt	664	605	Synthetic music Mobile Application Format	application/vnd.smaf	MMF	adSOUND
NIST_SPHERE_Fmt	665	606	NIST SPeech HEader REsources format		NIST	adSOUND
Chinese_AVS_Fmt	666	607	Chinese AVS video format			adMOVIE
VQA_Fmt	667	608	Westwood Studios Vector Quantized Animation video file		VQA	adANIMATION
YAFA_Fmt	668	609	Wildfire YAFA animation		YAFA	adANIMATION
Origin_MVE_Fmt	669	610	Origin Wing Commander III MVE movie format		MVE	adMOVIE
BBC_Dirac_Fmt	670	611	BBC Dirac video format	video/x-dirac	DRC	adMOVIE
Maya_ASCII_Fmt	671	612	Autodesk Maya ASCII file format		MA	adCAD

Format Name	Number	Category	Description	MIME Type	Extension	File Class
RenderMan_Fmt	672	613	Pixar RenderMan Interface Bytestream file		RIB	adVECTORGRAPHIC
NOFF_Binary_Fmt	673	614	NOFF 3D Object File Format		NOFF	adVECTORGRAPHIC
VTK_ASCII_Fmt	674	615	Visualization Toolkit VTK ASCII format		VTK	adVECTORGRAPHIC
VTK_Binary_Fmt	675	616	Visualization Toolkit VTK Binary format		VTK	adVECTORGRAPHIC
Wolfram_CDF_Fmt	676	617	Wolfram Mathematica Computable Document Format	application/cdf	CDF	adMISC
Wolfram_Notebook_Fmt	677	618	Wolfram Mathematica Notebook Format		NB	adMISC
HDF4_Fmt	678	619	Hierarchical Data Format HDF4	application/x-hdf	HDF, H4	adMISC
HDF5_Fmt	679	620	Hierarchical Data Format HDF5	application/x-hdf	HDF, H5	adMISC
ARMovie_Fmt	680	621	Acorn RISC ARMovie video format		RPL	adMOVIE
Windows_TV_DVR_Fmt	681	622	Windows Television DVR format		WTV	adMOVIE
InstallShield_Z_Fmt	682	623	InstallShield Z archive format	application/x-compress	Z	adENCAPSULATION
MS_DirectDraw_Surface_Fmt	683	624	Microsoft DirectDraw Surface container format		DDS	adENCAPSULATION
Bink_Fmt	684	625	Bink audio-video container format		BIK, BK2	adMOVIE
LZMA_Fmt	685	626	LZMA compressed data format	application/x-lzma	LZMA	adENCAPSULATION
True_Audio_Fmt	686	627	True Audio format	audio/x-tta	TTA	adSOUND
Keepass_Fmt	687	628	Keepass Password file		KDB, KDBX	adMISC
RPM_Fmt	688	629	RPM Package Manager file	application/x-rpm	RPM	adENCAPSULATION
Printer_Font_Metrics_Fmt	689	630	Adobe Printer Font Metrics format	application/x-font-printer-metric	PFM	adFONT
Adobe_Font_Metrics_Fmt	690	631	Adobe Font Metrics ASCII format	application/x-font-adobe-metric	AFM	adFONT
Printer_Font_ASCII_Fmt	691	632	Adobe Printer Font ASCII format	application/x-font-type1	PFA	adFONT
Netware_Loadable_Module_Fmt	692	633	Netware Loadable Module format		NLM	adMISC
TCPdump_pcap_Fmt	693	634	TCPdump packet stream capture savefile format	application/vnd.tcpdump.pcap	PCAP	adMISC
Multiple_Master_Font_Fmt	694	635	Adobe Multiple master font format		MMM	adFONT
TrueType_Font_Collection_Fmt	695	636	TrueType font collection format	application/x-font-ttf	TTC	adFONT

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Shapefile_Spatial_Index_Fmt	696	637	Shapefile binary spatial index format	application/x-shapefile	SBX, SBN	adGIS
Java_Key_Store_Fmt	697	638	Java Key Store format	application/x-java-keystore	KS	adMISC
Java_JCE_Key_Store_Fmt	698	639	Java JCE Key Store format	application/x-java-jce-keystore		adMISC
Quark_Xpress_Intel_Fmt	699	640	QuarkXPress Intel format	application/vnd.quark.quarkxpress	QXB	adDESKTOPPUBLSH
Windows_Imaging_Fmt	700	641	Microsoft Windows Imaging Format WIM		WIM	adMISC
VMware_Virtual_Disk_Fmt	701	642	VMware Virtual Disk Format 5.0	application/x-vmrk	VMDK	adMISC
XPCConnect_Typelib_Fmt	702	643	XPCConnect Typelib Format		XPT	adMISC
MS_DOS_Compression_Fmt	703	644	Microsoft MS-DOS installation 'Quantum' compression		EX_	adENCAPSULATION
DLS_Fmt	704	645	DLS Downloadable Sounds format		DLS	adSOUND
MS_Windows_Registry_Fmt	705	646	Microsoft Windows Registry format			adMISC
Microsoft_Help_2_0_Fmt	706	647	Microsoft Help 2.0 format		HXD, HXW, HXH	adENCAPSULATION
Qt_Translation_Fmt	707	648	Qt binary translation file format		QM	adMISC
PEM_SSL_Certificate_Fmt	708	649	PEM-encoded SSL certificate	application/pkix-cert	CRT, PEM, CER, KEY	adENCAPSULATION
PostScript_Printer_Description_Fmt	709	650	Adobe PostScript Printer Description file	application/vnd.cups-ppd	PPD	adMISC
Speedo_Font_Fmt	710	651	Speedo Font format		SPD	adFONT
InstallShield_Cabinet_Fmt	711	652	InstallShield Cabinet Archive format		CAB, HDR	adENCAPSULATION
InstallShield_Uninstall_Fmt	712	653	InstallShield Uninstall format		ISU	adENCAPSULATION
MS_OEDBX_Folder_Fmt	713	654	Outlook Express DBX folder database format		DBX	adENCAPSULATION
LabVIEW_Fmt	714	655	National Instruments LabVIEW file format		VI	adMISC
SAP_Archive_SAR_Fmt	715	656	SAP compression archive SAR format		SAR	adENCAPSULATION
Netscape_Address_Book_Fmt	716	657	Netscape Address Book format		NAB	adMISC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Universal_3D_Fmt	717	658	Universal 3D file format		U3D	adVECTORGRAPHIC
Open_Inventor_ASCII_Fmt	718	659	Open Inventor ASCII format		IV	adVECTORGRAPHIC
Open_Inventor_Binary_Fmt	719	660	Open Inventor Binary format		IV	adVECTORGRAPHIC
X_Window_Dump_Fmt	720	661	X Window Dump image	image/x-xwindowdump	XWD	adRASTERIMAGE
Git_Packfile_Fmt	721	662	Git Packfile format		PACK	adENCAPSULATION
Xara_Xar_Fmt	722	663	Xara X Xar image format	application/vnd.xara	XAR	adVECTORGRAPHIC
Internet_Archive_ARC_Fmt	723	664	Internet Archive ARC format	application/x-ia-arc	ARC	adENCAPSULATION
Applix_Builder_Fmt	724	665	Applix Builder format		AB	adMISC
Applix_Bitmap_Fmt	725	666	Applix Bitmap image format		IM	adRASTERIMAGE
PEM_RSA_Private_Key_Fmt	726	667	PEM-encoded RSA private key		PEM	adENCAPSULATION
MIFF_Fmt	727	668	Magick Image File Format		MIFF	adRASTERIMAGE
Subversion_Dump_Fmt	728	669	Subversion Dump format			adENCAPSULATION
Virtual_Hard_Disk_Fmt	729	670	Microsoft Virtual Hard Disk format	application/x-vhd	VHD	adENCAPSULATION
Direct_Access_Archive_Fmt	730	671	PowerISO Direct Access Archive format		DAA	adENCAPSULATION
Debian_Binary_Fmt	731	672	Debian binary package format	application/x-debian-package	DEB	adENCAPSULATION
XUL_Fastload_Fmt	732	673	Mozilla XUL Fastload format		MFL	adMISC
Nastran_OP2_Fmt	733	674	Nastran OP2 format		OP2	adCAD
Binary_Logging_Fmt	734	675	CAD Binary Logging Format		BLF	adCAD
Measurement_Data_Fmt	735	676	CAD Measurement Data Format		MDF	adCAD
Abaqus_ODB_Fmt	736	677	Abaqus ODB Format		ODB	adCAD
Open_Diagnostic_Data_Exchange_Fmt	737	678	Vector Open Diagnostic Data Exchange format		ODX	adCAD
Vector_ASCII_Fmt	738	679	Vector CAD ASCII ASC format		ASC	adCAD
LSDYNA_State_Database_Fmt	739	680	LS-DYNA State Database format			adCAD
LSDYNA_Binary_Output_Fmt	740	681	LS-DYNA binary output (binout) format			adCAD
MS_Power_BI_Fmt	741	682	Microsoft Power BI Desktop format		PBIX	adANALYTICS
Tableau_Workbook_Fmt	742	683	Tableau Workbook format		TWB	adANALYTICS
Tableau_Packaged_Workbook_Fmt	743	684	Tableau Packaged Workbook		TWBX	adANALYTICS

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			format			
Tableau_Extract_Fmt	744	685	Tableau Extract format		TDE	adANALYTICS
Tableau_Data_Source_Fmt	745	686	Tableau Data Source format		TDS	adANALYTICS
Tableau_Packaged_Data_Source_Fmt	746	687	Tableau Packaged Data Source format		TDSX	adANALYTICS
Tableau_Preferences_Fmt	747	688	Tableau Preferences format		TPS	adANALYTICS
Tableau_Map_Source_Fmt	748	689	Tableau Map Source format		TMS	adANALYTICS
ABAP_Fmt	749	690	ABAP Source Code ⁴	text/x-abap	ABAP	adSOURCECODE
AMPL_Fmt	750	691	AMPL Source Code ⁴		AMPL	adSOURCECODE
APL_Fmt	751	692	APL Source Code ⁴		APL	adSOURCECODE
ASN1_Fmt	752	693	ASN.1 Source Code ⁴		ASN	adSOURCECODE
ATS_Fmt	753	694	ATS Source Code ⁴			adSOURCECODE
Agda_Fmt	754	695	Agda Source Code ⁴	text/x-agda	AGDA	adSOURCECODE
Alloy_Fmt	755	696	Alloy Source Code ⁴	text/x-alloy	ALS	adSOURCECODE
Apex_Fmt	756	697	Apex Source Code ⁴		CLS	adSOURCECODE
Arduino_Fmt	757	698	Arduino Source Code ⁴	text/x-arduino	INO	adSOURCECODE
AsciiDoc_Fmt	758	699	AsciiDoc Source Code ⁴	text/x-asciidoc	ASC	adSOURCECODE
AspectJ_Fmt	759	700	AspectJ Source Code ⁴	text/x-aspectj	AJ	adSOURCECODE
Awk_Fmt	760	701	Awk Source Code ⁴	text/x-awk	AWK	adSOURCECODE
BlitzMax_Fmt	761	702	BlitzMax Source Code ⁴	text/x-bmx	BMX	adSOURCECODE
Bluespec_Fmt	762	703	Bluespec Source Code ⁴		BSV	adSOURCECODE
Brainfuck_Fmt	763	704	Brainfuck Source Code ⁴	text/x-brainfuck	B, BF	adSOURCECODE
Brightscript_Fmt	764	705	Brightscript Source Code ⁴		BRS	adSOURCECODE
CLIPS_Fmt	765	706	CLIPS Source Code ⁴		CLP	adSOURCECODE
CMake_Fmt	766	707	CMake Source Code ⁴	text/x-cmake	CMAKE	adSOURCECODE
COBOL_Fmt	767	708	COBOL Source Code ⁴	text/x-cobol	CBL, CCP, COB, CPY	adSOURCECODE
CWeb_Fmt	768	709	CWeb Source Code ⁴		W	adSOURCECODE
CartoCSS_Fmt	769	710	CartoCSS Source Code ⁴		MSS	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Ceylon_Fmt	770	711	Ceylon Source Code ⁴	text/x-ceylon	CEYLON	adSOURCECODE
Chapel_Fmt	771	712	Chapel Source Code ⁴		CHPL	adSOURCECODE
Clarion_Fmt	772	713	Clarion Source Code ⁴		CLW	adSOURCECODE
Clean_Fmt	773	714	Clean Source Code ⁴		DCL, ICL	adSOURCECODE
Component_Pascal_Fmt	774	715	Component Pascal Source Code ⁴	text/x-component-pascal	CP	adSOURCECODE
Cool_Fmt	775	716	Cool Source Code ⁴		CL	adSOURCECODE
Coq_Fmt	776	717	Coq Source Code ⁴	text/x-coq	V	adSOURCECODE
Creole_Fmt	777	718	Creole Source Code ⁴		CREOLE	adSOURCECODE
Crystal_Fmt	778	719	Crystal Source Code ⁴		CR	adSOURCECODE
Csound_Fmt	779	720	Csound Source Code ⁴		ORC	adSOURCECODE
Csound_Document_Fmt	780	721	Csound Document Source Code ⁴		CSD	adSOURCECODE
Cuda_Fmt	781	722	Cuda Source Code ⁴	text/x-cuda	CU	adSOURCECODE
D_Fmt	782	723	D Source Code ⁴	text/x-d	DCL, ICL	adSOURCECODE
DIGITAL_Command_Language_Fmt	783	724	DIGITAL Command Language Source Code ⁴		COM	adSOURCECODE
DTrace_Fmt	784	725	DTrace Source Code ⁴		D	adSOURCECODE
Dart_Fmt	785	726	Dart Source Code ⁴	text/x-dart	DART	adSOURCECODE
E_Fmt	786	727	E Source Code ⁴		E	adSOURCECODE
ECL_Fmt	787	728	ECL Source Code ⁴	application/x-ecl	ECL	adSOURCECODE
Elm_Fmt	788	729	Elm Source Code ⁴	text/x-elm	ELM	adSOURCECODE
Emacs_Lisp_Fmt	789	730	Emacs Lisp Source Code ⁴	text/x-emacs-lisp	EL	adSOURCECODE
EmberScript_Fmt	790	731	EmberScript Source Code ⁴		EM	adSOURCECODE
Fantom_Fmt	791	732	Fantom Source Code ⁴	application/x-fantom	FAN	adSOURCECODE
Forth_Fmt	792	733	Forth Source Code ⁴	text/x-forth	FOR, FORTH	adSOURCECODE
FreeMarker_Fmt	793	734	FreeMarker Source Code ⁴		FTL	adSOURCECODE
Frege_Fmt	794	735	Frege Source Code ⁴		FR	adSOURCECODE
G_code_Fmt	795	736	G-code Source Code ⁴		G	adSOURCECODE
GAMS_Fmt	796	737	GAMS Source Code ⁴		GMS	adSOURCECODE
GAP_Fmt	797	738	GAP Source Code ⁴			adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
GDScript_Fmt	798	739	GDScript Source Code ⁴		GD	adSOURCECODE
GLSL_Fmt	799	740	GLSL Source Code ⁴	text/x-glslsrc	GLSL	adSOURCECODE
Game_Maker_Language_Fmt	800	741	Game Maker Language Source Code ⁴		GML	adSOURCECODE
Gnuplot_Fmt	801	742	Gnuplot Source Code ⁴	text/x-gnuplot	GNU, GP	adSOURCECODE
Golo_Fmt	802	743	Golo Source Code ⁴		GOLO	adSOURCECODE
Gosu_Fmt	803	744	Gosu Source Code ⁴	text/x-gosu	GS	adSOURCECODE
Gradle_Fmt	804	745	Gradle Source Code ⁴		GRADLE	adSOURCECODE
GraphQL_Fmt	805	746	GraphQL Source Code ⁴		GRAPHQL	adSOURCECODE
Graphviz_DOT_Fmt	806	747	Graphviz (DOT) Source Code ⁴		DOT	adSOURCECODE
HLSL_Fmt	807	748	HLSL Source Code ⁴		HLSL	adSOURCECODE
Hack_Fmt	808	749	Hack Source Code ⁴			adSOURCECODE
Haml_Fmt	809	750	Haml Source Code ⁴	text/x-haml	HAML	adSOURCECODE
Handlebars_Fmt	810	751	Handlebars Source Code ⁴		HBS	adSOURCECODE
Hy_Fmt	811	752	Hy Source Code ⁴	text/x-hy	HY	adSOURCECODE
IDL_Fmt	812	753	IDL Source Code ⁴	text/x-idl	PRO	adSOURCECODE
IGOR_Pro_Fmt	813	754	IGOR Pro Source Code ⁴	text/ipf	IPF	adSOURCECODE
Idris_Fmt	814	755	Idris Source Code ⁴	text/x-idris	IDR	adSOURCECODE
Inform_7_Fmt	815	756	Inform 7 Source Code ⁴		I7X	adSOURCECODE
Ioke_Fmt	816	757	Ioke Source Code ⁴	text/x-iokesrc	IK	adSOURCECODE
Isabelle_Fmt	817	758	Isabelle Source Code ⁴	text/x-isabelle		adSOURCECODE
J_Fmt	818	759	J Source Code ⁴	text/x-j	IJS	adSOURCECODE
JSONiq_Fmt	819	760	JSONiq Source Code ⁴		JQ	adSOURCECODE
JSX_Fmt	820	761	JSX Source Code ⁴		JSX	adSOURCECODE
Jasmin_Fmt	821	762	Jasmin Source Code ⁴		J	adSOURCECODE
Jolie_Fmt	822	763	Jolie Source Code ⁴			adSOURCECODE
Julia_Fmt	823	764	Julia Source Code ⁴	text/x-julia	JL	adSOURCECODE
KiCad_Layout_Fmt	824	765	KiCad Layout Source Code ⁴			adSOURCECODE
KiCad_Schematic_Fmt	825	766	KiCad Schematic Source Code ⁴		SCH	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Kotlin_Fmt	826	767	Kotlin Source Code ⁴		KT	adSOURCECODE
LFE_Fmt	827	768	LFE Source Code ⁴	text/x-kotlin	LFE	adSOURCECODE
LOLCODE_Fmt	828	769	LOLCODE Source Code ⁴		LOL	adSOURCECODE
Lasso_Fmt	829	770	Lasso Source Code ⁴	text/x-lasso	LAS, LASSO	adSOURCECODE
Limbo_Fmt	830	771	Limbo Source Code ⁴	text/limbo		adSOURCECODE
LiveScript_Fmt	831	772	LiveScript Source Code ⁴	text/x-livescript	LS	adSOURCECODE
M_Fmt	832	773	M Source Code ⁴		M	adSOURCECODE
MAXScript_Fmt	833	774	MAXScript Source Code ⁴		MS	adSOURCECODE
Markdown_Fmt	834	775	Markdown Source Code ⁴		MD	adSOURCECODE
Matlab_Fmt	835	463	Matlab Source Code ⁴	text/x-matlab	M	adSOURCECODE
Max_Code_Fmt	836	776	Max Source Code ⁴		MXT	adSOURCECODE
Mercury_Fmt	837	777	Mercury Source Code ⁴			adSOURCECODE
Modelica_Fmt	838	778	Modelica Source Code ⁴	text/x-modelica	MO	adSOURCECODE
Modula_2_Fmt	839	779	Modula-2 Source Code ⁴	text/x-modula2	MOD	adSOURCECODE
Monkey_Fmt	840	780	Monkey Source Code ⁴	text/x-monkey	MONKEY	adSOURCECODE
Moocode_Fmt	841	781	Moocode Source Code ⁴	text/x-moocode	MOO	adSOURCECODE
NL_Fmt	842	782	NL Source Code ⁴		NL	adSOURCECODE
NSIS_Fmt	843	783	NSIS Source Code ⁴	text/x-nsis	NSI	adSOURCECODE
NetLogo_Fmt	844	784	NetLogo Source Code ⁴		NLOGO	adSOURCECODE
NewLisp_Fmt	845	785	NewLisp Source Code ⁴	text/x-newlisp	NL	adSOURCECODE
Nginx_Fmt	846	786	Nginx Source Code ⁴	text/x-nginx-conf	VHOST	adSOURCECODE
Nix_Fmt	847	787	Nix Source Code ⁴	text/x-nix	NIX	adSOURCECODE
Nu_Fmt	848	788	Nu Source Code ⁴		NU	adSOURCECODE
OCaml_Fmt	849	789	OCaml Source Code ⁴	text/x-ocaml		adSOURCECODE
OpenCL_Fmt	850	790	OpenCL Source Code ⁴		CL	adSOURCECODE
OpenEdge_ABL_Fmt	851	791	OpenEdge ABL Source Code ⁴	text/x-openedge		adSOURCECODE
OpenSCAD_Fmt	852	792	OpenSCAD Source Code ⁴		SCAD	adSOURCECODE
Ox_Fmt	853	793	Ox Source Code ⁴		OX	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Oxygene_Fmt	854	794	Oxygene Source Code ⁴		OXYGENE	adSOURCECODE
Oz_Fmt	855	795	Oz Source Code ⁴		OZ	adSOURCECODE
PAWN_Fmt	856	796	PAWN Source Code ⁴	text/x-pawn	PWN	adSOURCECODE
PLpgSQL_Fmt	857	797	PLpgSQL Source Code ⁴	text/x-plpgsql	PLSQL	adSOURCECODE
Pan_Fmt	858	798	Pan Source Code ⁴		PAN	adSOURCECODE
Parrot_Assembly_Fmt	859	799	Parrot Assembly Source Code ⁴		PASM	adSOURCECODE
PicoLisp_Fmt	860	800	PicoLisp Source Code ⁴			adSOURCECODE
Pike_Fmt	861	801	Pike Source Code ⁴	text/x-pike	PIKE	adSOURCECODE
Pony_Fmt	862	802	Pony Source Code ⁴		PONY	adSOURCECODE
Processing_Fmt	863	803	Processing Source Code ⁴		PDE	adSOURCECODE
PureBasic_Fmt	864	804	PureBasic Source Code ⁴		PB	adSOURCECODE
QMake_Fmt	865	805	QMake File ⁴			adSOURCECODE
RAML_Fmt	866	806	RAML Source Code ⁴		RAML	adSOURCECODE
RDoc_Fmt	867	807	RDoc Source Code ⁴		RDOC	adSOURCECODE
REXX_Fmt	868	808	REXX Source Code ⁴	text/x-rexx	REXX	adSOURCECODE
Racket_Fmt	869	809	Racket Source Code ⁴	text/x-racket		adSOURCECODE
Ragel_Fmt	870	810	Ragel Source Code ⁴			adSOURCECODE
Rascal_Fmt	871	811	Rascal Source Code ⁴		RSC	adSOURCECODE
Rebol_Fmt	872	812	Rebol Source Code ⁴	text/x-rebol	REB, REBOL	adSOURCECODE
Red_Fmt	873	813	Red Source Code ⁴	text/x-red	RED	adSOURCECODE
RenPy_Fmt	874	814	Ren'Py Source Code ⁴		RPY	adSOURCECODE
RenderScript_Fmt	875	815	RenderScript Source Code ⁴		RS	adSOURCECODE
Ring_Fmt	876	816	Ring Source Code ⁴		RING	adSOURCECODE
RobotFramework_Fmt	877	817	RobotFramework Source Code ⁴	text/x-robotframework	ROBOT	adSOURCECODE
SAS_Fmt	878	818	SAS Source Code ⁴		SAS	adSOURCECODE
SPARQL_Fmt	879	819	SPARQL format ⁴	application/sparql-query		adSOURCECODE
SQL_Fmt	880	820	SQL format ⁴	text/x-sql		adSOURCECODE
SQLPL_Fmt	881	821	SQLPL Source Code ⁴			adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
SaltStack_Fmt	882	822	SaltStack Source Code ⁴		SLS	adSOURCECODE
Scheme_Fmt	883	823	Scheme Source Code ⁴	text/x-scheme		adSOURCECODE
Scilab_Fmt	884	824	Scilab Source Code ⁴	text/scilab	SCI	adSOURCECODE
Squirrel_Fmt	885	825	Squirrel Source Code ⁴		NUT	adSOURCECODE
Stan_Fmt	886	826	Stan Source Code ⁴		STAN	adSOURCECODE
Stata_Fmt	887	827	Stata Source Code ⁴			adSOURCECODE
Stylus_Fmt	888	828	Stylus Source Code ⁴		STYL	adSOURCECODE
SuperCollider_Fmt	889	829	SuperCollider Source Code ⁴	text/supercollider	SC	adSOURCECODE
SystemVerilog_Fmt	890	830	SystemVerilog Source Code ⁴	text/x-systemverilog	SV	adSOURCECODE
TXL_Fmt	891	831	TXL Source Code ⁴		TXL	adSOURCECODE
Turing_Fmt	892	832	Turing Source Code ⁴		T	adSOURCECODE
Turtle_Fmt	893	833	Turtle Source Code ⁴	text/turtle	TTL	adSOURCECODE
UrWeb_Fmt	894	834	UrWeb Source Code ⁴		UR, URS	adSOURCECODE
Vim_script_Fmt	895	835	Vim script File ⁴	text/x-vim	VIM	adSOURCECODE
Visual_Basic_Fmt	896	836	Visual Basic Source Code ⁴	text/x-vbasic	VB	adSOURCECODE
WebAssembly_Fmt	897	837	WebAssembly Source Code ⁴		WAT	adSOURCECODE
WebIDL_Fmt	898	838	WebIDL Source Code ⁴		WEBIDL	adSOURCECODE
X10_Fmt	899	839	X10 Source Code ⁴	text/x-x10	X10	adSOURCECODE
XQuery_Fmt	900	840	XQuery Source Code ⁴	text/xquery	XQM	adSOURCECODE
Xojo_Fmt	901	841	Xojo Source Code ⁴			adSOURCECODE
Xtend_Fmt	902	842	Xtend Source Code ⁴	text/x-xtend	XTEND	adSOURCECODE
YANG_Fmt	903	843	YANG Source Code ⁴		YANG	adSOURCECODE
Zephir_Fmt	904	844	Zephir Source Code ⁴		ZEP	adSOURCECODE
eC_Fmt	905	845	eC Source Code ⁴	text/x-ecsrc	EC	adSOURCECODE
reStructuredText_Fmt	906	846	reStructuredText Source Code ⁴	text/x-rst		adSOURCECODE
xBase_Fmt	907	847	xBase Source Code ⁴			adSOURCECODE
Windows_Installer_Fmt	908	848	MSI Windows Installer format	application/x-ole-storage	MSI	adENCAPSULATION
Autodesk_3ds_Max_Fmt	909	849	Autodesk 3ds Max format		MAX	adCAD

Format Name	Number	Category	Description	MIME Type	Extension	File Class
PhotoDraw_Mix_Fmt	910	850	PhotoDraw MIX image	image/vnd.mix	MIX	adRASTERIMAGE
Softimage_SCN_Fmt	911	851	Softimage Scene SCN format		SCN	adCAD
Parasolid_XT_Fmt	912	852	Parasolid ascii XT format		X_T	adCAD
Parasolid_XB_Fmt	913	853	Parasolid binary XB format		X_B	adCAD
IGES_Fmt	914	854	Initial Graphics Exchange Specification format	model/iges	IGS	adCAD
ACE_Archive_Fmt	915	855	ACE archive format	application/x-ace-compressed	ACE	adENCAPSULATION
Grasshopper_GHX_Fmt	916	856	Grasshopper GHX format		GHX	adCAD
MS_FrontPage_Macro_Fmt	917	857	Microsoft FrontPage macro file format		FPM	adWORDPROCESSOR
MS_AtWork_Fax_Fmt	918	858	Microsoft AtWork Fax format		AWD	adFAXFORMAT
MS_Image_Composer_Fmt	919	859	Microsoft Image Composer format		MIC	adRASTERIMAGE
MS_Visual_InterDev_Fmt	920	860	Microsoft Visual InterDev web project items file		WDM	adMISC
Macromedia_Flash_FLA_OLE_Fmt	921	861	Macromedia Flash FLA Project File OLE format		FLA	adWORDPROCESSOR
Corel_Draw_X4_Fmt	922	862	CorelDRAW version X4 onwards	application/x-vnd.corel.zcf.draw.document+zip	CDRX	adVECTORGRAPHIC
Ogg_Daala_Fmt	923	863	Ogg Daala video format	video/daala	OGV	adMOVIE
Ogg_BBC_Dirac_Fmt	924	864	Ogg BBC Dirac video format	video/x-dirac	OGV	adMOVIE
PKCS_7_Fmt	925	865	PKCS #7 cryptographic format	application/pkcs7-signature	P7S	adWORDPROCESSOR
Time_Stamped_Data_Fmt	926	866	Time-stamped data format	application/timestamped-data	TSD	adENCAPSULATION
Sereal_Fmt	927	867	Sereal data serialization format	application/sereal	SRL	adMISC
Associated_Signature_Simple_Fmt	928	868	Associated Signature Container Simple format	application/vnd.etsi.asic-s+zip	ASICS	adENCAPSULATION
Associated_Signature_Extended_Fmt	929	869	Associated Signature Container Extended format	application/vnd.etsi.asic-e+zip	ASICE	adENCAPSULATION
iBooks_Fmt	930	870	Apple iBooks format	application/x-ibooks+zip	IBOOKS	adWORDPROCESSOR
PDF_Forms_Data_Fmt	931	871	PDF Forms Data Format	application/vnd.fdf	fdf	adWORDPROCESSOR
PDF_XML_Forms_Data_Fmt	932	872	PDF XML Forms Data Format	application/vnd.adobe.xfdf	Xfdf	adWORDPROCESSOR
AxCrypt_Fmt	933	873	AxCrypt encrypted document	application/x-axcrypt	AXX	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Unix_Archive_Fmt	934	874	Unix Archive ar format	application/x-archive	AR	adENCAPSULATION
Berkeley_Btree_Database_Fmt	935	875	Berkeley DB btree database format	application/x-berkeley-db	DB	adDATABASE
Berkeley_Hash_Database_Fmt	936	876	Berkeley DB hash database format	application/x-berkeley-db	DB	adDATABASE
Berkeley_Log_Database_Fmt	937	877	Berkeley DB log database format	application/x-berkeley-db		adDATABASE
Berkeley_Queue_Database_Fmt	938	878	Berkeley DB queue database format	application/x-berkeley-db		adDATABASE
BitTorrent_Fmt	939	879	BitTorrent file format	application/x-bittorrent	TORRENT	adMISC
Chrome_Extension_Fmt	940	880	Google Chrome Extension format	application/x-chrome-package	CRX	adENCAPSULATION
Dalvik_Executable_Fmt	941	881	Dalvik Executable dex format	application/x-dex	DEX	adEXECUTABLE
Foxmail_Fmt	942	882	Foxmail email format	application/x-foxmail	BOX	adWORDPROCESSOR
GRIB_Fmt	943	883	General Regularly-distributed Information in Binary form GRIB format	application/x-grib	GRB, GRIB2	adMISC
Zstandard_Fmt	944	884	Zstandard compression format	application/zstd	ZSTD	adENCAPSULATION
LZ4_Fmt	945	885	LZ4 compressed file	application/x-lz4	LZ4	adENCAPSULATION
MS_Money_Fmt	946	886	Microsoft Money format	application/x-msmoney	MNY	adSPREADSHEET
NetCDF_Fmt	947	887	Network Common Data Form NetCDF format	application/x-netcdf	NC	adMISC
SAS6_Data_Fmt	948	888	SAS 6 Data storage format	application/x-sas-data-v6	SD2	adDATABASE
SAS_Transport_Fmt	949	889	SAS Transport File XPORT format	application/x-sas-xport	XPT, XPORT	adDATABASE
Snappy_Framed_Fmt	950	890	Snappy Framed compression format	application/x-snappy-framed	SZ	adENCAPSULATION
Stata_Data_Fmt	951	891	Stata Data Format	application/x-stata-dta	DTA	adDATABASE
SPSS_SAV_Fmt	952	892	SPSS Statistics Data File Format		SAV	adDATABASE
Zoo_Archive_Fmt	953	893	Zoo Compressed Archive Format	application/x-zoo	ZOO	adENCAPSULATION
CDX_Fmt	954	894	ChemDraw CDX format	chemical/x-cdx	CDX	adSCIENTIFIC
CDXML_Fmt	955	895	ChemDraw CDXML format	application/vnd.chemdraw+xml	CDXML	adSCIENTIFIC
BPG_Fmt	956	896	Better Portable Graphics BPG format	image/x-bpg	BPG	adRASTERIMAGE
Apple_Icon_Fmt	957	897	Apple Icon image format	image/icns	ICNS	adRASTERIMAGE
NITF_Fmt	958	898	National Imagery Transmission	image/nitf	NTF, NITF	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			Format NITF image			
ERDAS_Imagine_Fmt	959	899	ERDAS Imagine image format	application/x-erdas-hfa	HFA, RRD, AUX	adRASTERIMAGE
MS_Office_Temporary_Owner_Fmt	960	900	Microsoft Office temporary owner file	application/x-ms-owner		adMISC
EAC3_Audio_Fmt	961	901	Enhanced-AC3 (EAC3) Audio File format	audio/eac3	AC3	adSOUND
COFF_Relocatable_Fmt	962	902	Common Object File Format (COFF) relocatable object	application/x-object-file	O	adOBJECTMODULE
COFF_Executable_Fmt	963	903	Common Object File Format (COFF) executable	application/x-executable-file		adEXECUTABLE
COFF_Dynamic_Lib_Fmt	964	904	Common Object File Format (COFF) dynamic library	application/x-library-file		adLIBRARY
ELF_Core_Fmt	965	905	ELF Core file	application/x-coredump		adMISC
Purify_Fmt	966	906	Rational Purify data file		PFY	adMISC
Kryptel_Fmt	967	907	Kryptel encrypted file		EDC	adENCAPSULATION
Windows_Core_Dump_Fmt	968	908	Windows heap or mini core dump file	application/x-dmp	DMP	adMISC
Qt_Prerendered_Font_Fmt	969	909	Qt Prerendered Font format		QPF2	adFONT
AIX_Relocatable_Fmt	970	910	AIX/RISC COFF relocatable object	application/x-object-file		adOBJECTMODULE
AIX_Executable_Fmt	971	911	AIX/RISC COFF executable	application/x-executable-file		adEXECUTABLE
AIX_Dynamic_Lib_Fmt	972	912	AIX/RISC COFF dynamic library	application/x-library-file	A	adLIBRARY
HPUX_Relocatable_Fmt	973	913	HPUX/PA-RISC COFF relocatable object	application/x-object-file		adOBJECTMODULE
HPUX_Executable_Fmt	974	914	HPUX/PA-RISC COFF executable	application/x-executable-file		adEXECUTABLE
HPUX_Dynamic_Lib_Fmt	975	915	HPUX/PA-RISC COFF dynamic library	application/x-library-file	SL	adLIBRARY
XML_EBCDIC_Fmt	976	916	EBCDIC-encoded XML file	application/xml	XML	adWORDPROCESSOR
MPEG_JVT_H264_Fmt	977	917	MPEG JVT-NAL sequence H264 video	video/h264	264	adMOVIE
Material_Exchange_Fmt	978	918	Material Exchange Format audio-video container format	application/mxf	MXF	adMOVIE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Agent_Character_Fmt	979	919	Microsoft Agent Character file		ACS	adMOVIE
Quicken_Fmt	980	920	Quicken data file		QDF	adMISC
MS_Outlook_Address_Fmt	981	921	Microsoft Outlook address file		WAB	adMISC
MS_Answer_Wizard_Fmt	982	922	Microsoft Answer Wizard file			adMISC
ADX_Fmt	983	923	ADX audio file		ADX	adSOUND
System_Deployment_Image_Fmt	984	924	Microsoft System Deployment Image SDI format		SDI	adMISC
Free_Lossless_Image_Fmt	985	925	Free Lossless Image Format (FLIF)	image/flif	FLIF	adRASTERIMAGE
DPX_Fmt	986	926	Digital Picture Exchange (DPX) image format	image/dpx	DPX	adRASTERIMAGE
Avro_Fmt	987	927	Apache Avro binary format		AVRO	adMISC
InstallShield_Archive_Fmt	988	928	InstallShield archive (early versions) format		EX_	adENCAPSULATION
Mac_Executable_Fmt	989	929	Mac OS-X (Mach-O) executable format			adEXECUTABLE
GDSII_Fmt	990	930	GDSII data format		GDS	adMISC
ActiveMime_Fmt	991	931	Microsoft ActiveMime (mso) documents	application/x-mso	MSO	adMISC
SmartCharts_Fmt	992	932	BizInt SmartCharts data format		CHP, CHRR	adMISC
Webex_ARF_Fmt	993	933	Webex advanced network ARF recordings		ARF	adMOVIE
Webex_WRF_Fmt	994	934	Webex local WRF recordings		WRF	adMOVIE
PGP_NetShare_Fmt	995	935	Symantec PGP NetShare encrypted file			adENCAPSULATION
Ability_WP_OLE_Fmt	996	936	Ability Write later versions format		AWW	adWORDPROCESSOR
Ability_SS_OLE_Fmt	997	937	Ability Spreadsheet later versions format		AWS	adSPREADSHEET
InDesign_IDML_Fmt	998	938	Adobe InDesign IDML format	application/vnd.adobe.indesign-idml-package	IDML	adDESKTOPPUBLSH
Executable_JAR_Fmt	999	939	Executable Java Archive (jar) file	application/java-archive	JAR	adENCAPSULATION
IDOL_IDX_Fmt	1000	940	IDOL Server IDX file		IDX	adENCAPSULATION
Android_Package_Kit_Fmt	1001	941	Android Package Kit (APK) format	application/vnd.android.package-archive	APK	adEXECUTABLE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Android_Binary_XML_Fmt	1002	942	Android Binary XML (compressed by aapt) format	application/xml	XML	adWORDPROCESSOR
Java_WAR_Fmt	1003	943	Java WAR file format		WAR	adENCAPSULATION
Java_EAR_Fmt	1004	944	Java EAR file format		EAR	adENCAPSULATION
Atom_Syndication_Fmt	1005	945	Atom Syndication Format	application/atom+xml	ATOM	adWORDPROCESSOR
RSS_Fmt	1006	946	RSS syndication XML format	application/rss+xml	RSS	adWORDPROCESSOR
SMIL_Fmt	1007	947	Synchronized Multimedia Integration Language (SMIL) XML format	application/smil+xml	SMIL	adWORDPROCESSOR
XSLT_Fmt	1008	948	Extensible Stylesheet Language Transformations (XSLT) format	application/xslt+xml	XSL, XSLT	adWORDPROCESSOR
XML_Shareable_Playlist_Fmt	1009	949	XML Shareable Playlist Format (XSPF)	application/xspf+xml	XSPF	adWORDPROCESSOR
FictionBook_Fmt	1010	950	FictionBook e-book XML format	application/x-fictionbook+xml	FB2	adWORDPROCESSOR
Adobe_Premiere_Project_Fmt	1011	951	Adobe Premiere project format	image/vnd.adobe.premiere	PPJ	adMISC
RDF_XML_Fmt	1012	952	RDF/XML format	application/rdf+xml	RDF	adWORDPROCESSOR
Really_Simple_Discovery_Fmt	1013	953	Really Simple Discovery (RSD) XML format	application/rsd+xml	RSD	adWORDPROCESSOR
SBML_Fmt	1014	954	Systems Biology Markup Language (SBML) XML format	application/sbml+xml	SBML	adWORDPROCESSOR
SRU_Fmt	1015	955	Search/Retrieve via URL (SRU) XML format	application/sru+xml	SRU	adWORDPROCESSOR
SSML_Fmt	1016	956	Speech Synthesis Markup Language (SSML) XML format	application/ssml+xml	SSML	adWORDPROCESSOR
PLS_Fmt	1017	957	Pronunciation Lexicon Specification (PLS) XML format	application/pls+xml	PLS	adWORDPROCESSOR
TEI_Fmt	1018	958	Text Encoding Initiative (TEI) XML format	application/tei+xml	TEI	adWORDPROCESSOR
METS_Fmt	1019	959	Metadata Encoding and Transmission Standard (METS) XML format	application/mets+xml	METS	adWORDPROCESSOR
MODS_Fmt	1020	960	Metadata Object Description Schema (MODS) XML format	application/mods+xml	MODS	adWORDPROCESSOR
Metalink_Fmt	1021	961	Metalink XML format	application/metalink4+xml	METALINK	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Open_eBook_Fmt	1022	962	Open eBook (OEBPS) XML format	application/oebps-package+xml	OPF	adWORDPROCESSOR
SRGS_Fmt	1023	963	Speech Recognition Grammar Specification (SRGS) XML format	application/srgs+xml	SRGS	adWORDPROCESSOR
SPARQL_Results_Fmt	1024	964	SPARQL Query Results XML format	application/sparql-results+xml	SRX	adWORDPROCESSOR
Adobe_XML_Data_Package_Fmt	1025	965	Adobe XML Data Package format	application/vnd.adobe.xdp+xml	XDP	adWORDPROCESSOR
ESzigno_Fmt	1026	966	e-Szigno signed xml document	application/vnd.eszigno3+xml	ES3	adWORDPROCESSOR
Mozilla_XUL_Fmt	1027	967	Mozilla XML User Interface Language (XUL) XML format	application/vnd.mozilla.xul+xml	XUL	adWORDPROCESSOR
SyncML_Fmt	1028	968	Synchronization Markup Language (SyncML) XML format	application/vnd.syncml+xml	XML	adWORDPROCESSOR
VoiceXML_Fmt	1029	969	VoiceXML (VXML) XML format	application/voicexml+xml	VXML	adWORDPROCESSOR
TI_Target_Configuration_Fmt	1030	970	Texas Instruments CCXML target configuration XML format		CCXML	adWORDPROCESSOR
LZFSE_Fmt	1031	971	Lempel-Ziv Finite State Entropy (LZFSE) compression format		LZFSE	adENCAPSULATION
Kindle_eBook_Fmt	1032	972	Amazon Kindle or Mobipocket eBook format	application/vnd.amazon.ebook	AZW, PRC	adWORDPROCESSOR
Oasis_Stream_Fmt	1033	973	Open Artwork System Interchange Standard (OASIS) format		OAS	adMISC
Amazon_KFX_Fmt	1034	974	Amazon KFX eBook format		KFX	adWORDPROCESSOR
KTX_Fmt	1035	975	KTX image format	image/ktx	KTX	adRASTERIMAGE
GMSH_Mesh_Fmt	1036	976	GMSH Mesh polygon format	model/mesh	MSH	adCAD
Collada_DAE_Fmt	1037	977	Collada Digital Asset Exchange (DAE) format	model/vnd.collada+xml	DAE	adCAD
YIN_Fmt	1038	978	YIN XML format	application/yin+xml	YIN	adWORDPROCESSOR
MPEG_Playlist_Fmt	1039	979	MPEG audio playlist format	audio/mpegurl	M3U	adSOUND
Windows_Audio_Playlist_Fmt	1040	980	Windows Audio playlist format	audio/x-ms-wax	WAX	adSOUND
DTS_Audio_Fmt	1041	981	DTS Coherent Acoustics audio format	audio/vnd.dts	DTS	adSOUND
Chemical_Markup_Language_Fmt	1042	982	Chemical Markup Language (CML) XML format	chemical/x-cml	CML	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
CrystalMaker_Fmt	1043	983	CrystalMaker chemical format	chemical/x-cmdf	CMDF	adSCIENTIFIC
VTK_XML_Fmt	1044	984	Visualization Toolkit VTK XML format	model/vnd.vtu	VTU	adVECTORGRAPHIC
IPFIX_Fmt	1045	985	IP Flow Information Export (IPFIX) format	application/ipfix	IPFIX	adMISC
Portable_Font_Resource_Fmt	1046	986	Portable Font Resource font format	application/font-tdpfr	PFR	adFONT
MARC_Fmt	1047	987	Machine-Readable Cataloging (MARC21) format	application/marc	MARC	adDATABASE
MARC_XML_Fmt	1048	988	Machine-Readable Cataloging (MARC) XML format	application/marcxml+xml	XML	adWORDPROCESSOR
XAR_Fmt	1049	989	Extensible Archive (XAR) format			adENCAPSULATION
Symbian_Installer_Fmt	1050	990	Symbian installer format	application/vnd.symbian.install	SIS	adENCAPSULATION
SO_Drawing_XML_Fmt	1051	316	OpenDocument format (OpenOffice 1/StarOffice 6.7) Drawing XML	application/vnd.sun.xml.draw	SXD	adVECTORGRAPHIC
SO_Text_Global_XML_Fmt	1052	991	OpenDocument format (OpenOffice 1/StarOffice 6.7) Writer Master document XML	application/vnd.sun.xml.writer.global	SXG	adWORDPROCESSOR
ODF_Chart_Fmt	1053	992	ODF Chart	application/vnd.oasis.opendocument.chart	ODC	adVECTORGRAPHIC
ODF_Database_Fmt	1054	993	ODF Database	application/vnd.sun.xml.base	ODB	adDATABASE
ODF_Image_Fmt	1055	994	ODF Image	application/vnd.oasis.opendocument.image	ODI	adRASTERIMAGE
ODF_Text_Master_Fmt	1056	995	ODF Text Master	application/vnd.oasis.opendocument.text-master	ODM	adWORDPROCESSOR
ODF_Text_Web_Fmt	1057	996	ODF Text Web	application/vnd.oasis.opendocument.text-web	OTH	adWORDPROCESSOR
ODF_Chart_Template_Fmt	1058	997	ODF Chart Template	application/vnd.oasis.opendocument.chart-template	OTC	adVECTORGRAPHIC
ODF_Formula_Template_Fmt	1059	998	ODF Formula Template	application/vnd.oasis.opendocument.formula-template	OTF	adWORDPROCESSOR
ODF_Drawing_Template_Fmt	1060	316	ODF Drawing/Graphics Template	application/vnd.oasis.opendocument.graphics-template	OTG	adVECTORGRAPHIC
ODF_Image_Template_Fmt	1061	999	ODF Image Template	application/vnd.oasis.opendocument.image-template	OTI	adRASTERIMAGE
ODF_Presentation_Template_Fmt	1062	316	ODF Presentation Template	application/vnd.oasis.opendocument.presentation-template	OTP	adPRESENTATION
ODF_Spreadsheet_Template_Fmt	1063	315	ODF Spreadsheet Template	application/vnd.oasis.opendocument.spreadsheet-template	OTS	adSPREADSHEET

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ODF_Text_Template_Fmt	1064	314	ODF Text Template	application/vnd.oasis.opendocument.text-template	OTT	adWORDPROCESSOR
ODF_Chart_XML_Fmt	1065	1000	ODF Chart flat XML format	application/vnd.oasis.opendocument.chart.xml	FODC	adVECTORGRAPHIC
ODF_Drawing_XML_Fmt	1066	1001	ODF Drawing/Graphics flat XML format	application/vnd.oasis.opendocument.formula.xml	FODG	adWORDPROCESSOR
ODF_Formula_XML_Fmt	1067	1002	ODF Formula flat XML format	application/vnd.oasis.opendocument.graphics.xml	FODF	adVECTORGRAPHIC
ODF_Image_XML_Fmt	1068	1003	ODF Image flat XML format	application/vnd.oasis.opendocument.image.xml	FODI	adRASTERIMAGE
ODF_Presentation_XML_Fmt	1069	1004	ODF Presentation flat XML format	application/vnd.oasis.opendocument.presentation.xml	FODP	adPRESENTATION
ODF_Spreadsheet_XML_Fmt	1070	1005	ODF Spreadsheet flat XML format	application/vnd.oasis.opendocument.spreadsheet.xml	FODS	adSPREADSHEET
ODF_Text_XML_Fmt	1071	1006	ODF Text flat XML format	application/vnd.oasis.opendocument.text.xml	FODT	adWORDPROCESSOR
ODF_Extension_Fmt	1072	1007	ODF Extension format	application/vnd.openofficeorg.extension	OXT	adMISC
StarView_Metafile_Fmt	1073	1008	OpenOffice StarView MetaFile format	image/x-svm	SVM	adRASTERIMAGE
BBeB_LRF_eBook_Fmt	1074	1009	Broad Band eBook (BBeB) in LRF format		LRF	adWORDPROCESSOR
GPG_Trust_DB_Fmt	1075	1010	GPG trust database format		GPG	adMISC
VICE_Emulator_Fmt	1076	1011	VICE (Versatile Commodore Emulator) format		VSF	adMISC
Portable_Game_Notation_Fmt	1077	1012	Portable Game Notation chess format	application/vnd.chess-pgn	PGN	adWORDPROCESSOR
Doom_WAD_Fmt	1078	1013	Doom IWAD/PWAD format	application/x-doom	WAD	adMISC
Device_Tree_Blob_Fmt	1079	1014	Linux Device Tree Blob format		DTB	adMISC
BDF_Font_Fmt	1080	1015	Glyph Bitmap Distribution Format	application/x-font-bdf	BDF	adFONT
PC_Screen_Font_Fmt	1081	1016	PC Screen Font format	application/x-font-psf	PSF	adFONT
JNLP_Fmt	1082	1017	Java Network Launching Protocol	application/x-java-jnlp-file	JNLP	adWORDPROCESSOR
XAML_Browser_Application_Fmt	1083	1018	XAML Browser Application (XBAP) format	application/x-ms-xbap	XBAP	adWORDPROCESSOR
MS_Binder_Fmt	1084	1019	Microsoft Office Binder format	application/x-msbinder	OBP	adENCAPSULATION
XAP_Fmt	1085	1020	Microsoft Silverlight application (XAP) format	application/x-silverlight-app	XAP	adENCAPSULATION
Stuftit_X_Fmt	1086	1021	Stuftit X (SITX) archive format	application/x-stuftitx	SITX	adENCAPSULATION
FIG_Fmt	1087	1022	Facility for Interactive Generation of figures (FIG) image format	application/x-fig	FIG	adVECTORGRAPHIC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
XPIInstall_Fmt	1088	1023	XPIInstall Cross-Platform Installer Module (XPI) format	application/x-xpinstall	XPI	adENCAPSULATION
XDF_Fmt	1089	1024	Extensible Data Format (XDF) XML format		XDF	adWORDPROCESSOR
MXML_Fmt	1090	1025	MXML UI markup language XML format		MXML	adWORDPROCESSOR
MusicXML_Fmt	1091	1026	MusicXML format	application/vnd.recordare.musicxml	MXL	adENCAPSULATION
Finale_Fmt	1092	1027	Finale audio format		MUS	adSOUND
Spotfire_DXP_Fmt	1093	1028	TIBCO Spotfire DXP data format	application/vnd.spotfire.dxp	DXP	adANALYTICS
MS_Office_Theme_2007_Fmt	1094	1029	Microsoft Office theme format	application/vnd.ms-officetheme	THMX	adMISC
Adobe_AIR_Installer_Fmt	1095	1030	Adobe AIR application installer package	application/vnd.adobe.air-application-installer-package+zip	AIR	adENCAPSULATION
Flex_Project_Fmt	1096	1031	Adobe Flash Flex project file format	application/vnd.adobe.fxp	FXP	adENCAPSULATION
FoxPro_Fmt	1097	1032	FoxPro compiled source format		FXP	adLIBRARY
VST_Preset_Fmt	1098	1033	Virtual Studio Technology (VST) preset format		FXP	adSOUND
Mischief_Image_Fmt	1099	1034	Mischief vector graphics image format		ART	adVECTORGRAPHIC
FreeArc_Fmt	1100	1035	FreeArc archive format	application/x-freearc	ARC	adENCAPSULATION
Autodesk_3ds_Fmt	1101	1036	Autodesk 3ds format	application/x-3ds	3DS	adCAD
Monkeys_Audio_Fmt	1102	1037	Monkey's Audio format		APE	adSOUND
CALS_Fmt	1103	1038	CALS raster image format		CAL	adRASTERIMAGE
Dr_Halo_PAL_Fmt	1104	1039	Dr Halo raster image PAL file format		PAL	adRASTERIMAGE
DPG_Fmt	1105	1040	Nintendo DS DPG video format		DPG	adMOVIE
JPEG_XR_Fmt	1106	1041	JPEG XR (extended range) image format	image/vnd.ms-photo	JXR, HDP	adRASTERIMAGE
TCR_eBook_Fmt	1107	1042	TCR (Text Compression for Reader) eBook format		TCR	adWORDPROCESSOR
IHEX_Fmt	1108	1043	Intel Hex format		IHEX	adENCAPSULATION
QCOW_Fmt	1109	1044	QEMU Copy On Write		QCOW	adENCAPSULATION
VDI_Fmt	1110	1045	VirtualBox Disk Image		VDI	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
OneNote_Alternate_Fmt	1111	1046	OneNote Alternative Packaging Format			adWORDPROCESSOR
RMS_Protected_Fmt	1112	1047	Rights Management Services (RMS)-protected format		PFILE, PPDF, PJPNG, PTXT	adWORDPROCESSOR
Portfolio_PDF_Fmt	1113	1048	Portfolio PDF File	application/pdf	PDF	adWORDPROCESSOR
Crystal_Reports_Fmt	1114	1049	SAP Crystal Reports format	application/x-rpt	RPT	adANALYTICS
Thumbs_db_Fmt	1115	1050	Microsoft Windows thumbs.db format		DB	adENCAPSULATION
PagePlus_Fmt	1116	1051	Serif PagePlus format		PPP	adDESKTOPPUBLISH
MS_Project_Exchange_Fmt	1117	1052	Microsoft Project Exchange format		MPX	adSCHEDULE
MS_Management_Pack_MPX_Fmt	1118	1053	Microsoft Systems Center Operation Manager (SCOM) management pack MPX format		MPX	adMISC
AutoCAD_VBA_Project_Fmt	1119	1054	AutoCAD VBA project format		DVB	adMISC
PLY_ASCII_Fmt	1120	1055	Polygon File Format (PLY) ASCII format		PLY	adCAD
PLY_Binary_Fmt	1121	1056	Polygon File Format (PLY) binary format		PLY	adCAD
JavaView_JVX_Fmt	1122	1057	JavaView XML (JVX) format		JVX	adCAD
X3D_Fmt	1123	1058	Extensible 3d Graphics (X3D) XML format	model/x3d+xml	X3D	adCAD
ZBrush_Project_Fmt	1124	1059	ZBrush ZProject (ZPR) format		ZPR	adCAD
ZBrush_Tool_Fmt	1125	1060	ZBrush ZTtool (ZTL) format		ZTL	adCAD
Windows_Installer_Patch_Fmt	1126	1061	Microsoft Windows Installer Patch Package (MSP) format		MSP	adENCAPSULATION
Windows_Installer_Transform_Fmt	1127	1062	Microsoft Windows Installer Transform (MST) format		MST	adENCAPSULATION
Lotus_Approach_Fmt	1128	1063	Lotus Approach format	application/vnd.lotus-approach	APR, MPR	adDATABASE
Outlook_SendRcv_Settings_Fmt	1129	1064	Microsoft Outlook 2002 Send-Receive Settings		SRS	adMISC
MS_Publisher_Scheme_Fmt	1130	1065	Microsoft Publisher colour scheme		SCM	adMISC
SO_Chart_Fmt	1131	1066	Star Office 4,5 Chart	application/vnd.stardivision.chart	SDS	adVECTORGRAPHIC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
SO_Database_Fmt	1132	1067	Star Office 4,5 Database	application/vnd.stardivision.base	SDB	adDATABASE
SO_Library_Fmt	1133	1068	Star Office 4,5 Library		SBL	adLIBRARY
PageMaker_Document_Fmt	1134	1069	Adobe PageMaker document	application/pagemaker	PMD	adDESKTOPPUBLSH
MS_DTS_Fmt	1135	1070	Microsoft Data Transformation Services (DTS) package file		DTS	adMISC
Cognos_PowerPlay_PPR_Fmt	1136	1071	Cognos PowerPlay up to version 7 (PPR) format		PPR	adANALYTICS
Visual_Studio_SUO_Fmt	1137	1072	Microsoft Visual Studio solution user options (suo) file		SUO	adMISC
MS_GraphEdit_Fmt	1138	1073	Microsoft GraphEdit File format		GRF	adMISC
ArcGIS_Graph_Fmt	1139	1074	ArcGIS Graph format		GRF	adGIS
SID_Audio_Fmt	1140	1075	SID Audio format	audio/prs.sid	SID	adSOUND
MrSID_Fmt	1141	1076	LizardTech MrSID image format	image/x-mrsid	SID	adRASTERIMAGE
Cardfile_Fmt	1142	1077	Microsoft Windows Cardfile address book format	application/x-mscardfile	CRD	adWORDPROCESSOR
MS_Word_Mac_4_Fmt	1143	205	Microsoft Word for Macintosh (version 4,5)	application/msword	DOC	adWORDPROCESSOR
WordPerfect_5_Fmt	1144	80	WordPerfect (version 5)	application/x-corel-wordperfect	WOP, DOC	adWORDPROCESSOR
WordPerfect_6_Fmt	1145	178	WordPerfect (version 6 and higher)	application/x-corel-wordperfect	WPD	adWORDPROCESSOR
WordPerfect_Graphics_1_Fmt	1146	85	WordPerfect Graphics (version 1)	application/vnd.wordperfect	WPG, QPG	AutoDetNoFormat
Organization_Chart_Fmt	1147	1078	OrgPlus Organization Chart	application/orgplus	OPX	adDATABASE
Lotus_Organizer_Fmt	1148	1079	Lotus Organizer documents	application/vnd.lotus-organizer	OR2, OR3, OR4, OR5, OR6	adSCHEDULE
MS_DBML_Fmt	1149	1080	Microsoft Database Markup Language XML document		DBML	adWORDPROCESSOR
XMind_Fmt	1150	1081	XMind document	application/xmind	XMIND	adPRESENTATION
MSI_Cerius_Fmt	1151	1082	MSI Cerius chemical formula document	chemical/x-cerius	MSI	adSCIENTIFIC
GenBank_Fmt	1152	1083	GenBank DNA character sequence document	chemical/x-genbank	GB	adSCIENTIFIC
GIS_World_File_Fmt	1153	1084	ESRI GIS World file		BPW, GFW, JGW, J2W,	adGIS

Format Name	Number	Category	Description	MIME Type	Extension	File Class
					PGW, SDW, TFW, WLD	
GIS_Projection_Metadata_Fmt	1154	1085	ESRI Projection Metadata (PRJ) file		PRJ	adGIS
PowerWorld_Binary_Fmt	1155	1086	PowerWorld Binary (PWB) file		PWB	adCAD
PowerWorld_Display_Fmt	1156	1087	PowerWorld Display (PWD) file		PWD	adCAD
ArcXML_Fmt	1157	1088	ESRI ArcIMS project XML file (ArcXML)		AXL	adGIS
GAMS_GDX_Fmt	1158	1089	General Algebraic Modeling System (GAMS) Data Exchange (GDX) format		GDX	adSCIENTIFIC
ArcMap_MXD_Fmt	1159	1090	ArcMap Map Exchange Document project (MXD)		MXD	adGIS
RRDtool_Fmt	1160	1091	RRDtool (Round Robin Database) data file		RRD	adDATABASE
HWPX_Fmt	1161	1092	Hangul HWPX document	application/hwp+zip	HWPX	adWORDPROCESSOR
SolidWorks_2015_Fmt	1162	1093	SolidWorks (2015 onwards) file		SLDPRT, SLDDRW, SLDASM	adCAD
MS_Photo_Editor_Fmt	1163	1094	Microsoft Photo Editor 'embedded GIF' file	application/vnd.ms-photo-editor		adRASTERIMAGE
MS_Word_HTML_Fmt	1164	1095	Microsoft Word HTML format		DOC, HTM	adWORDPROCESSOR
MS_Excel_HTML_Fmt	1165	1096	Microsoft Excel HTML format		XLS, HTM	adWORDPROCESSOR
Portable_FloatMap_Fmt	1166	1097	Portable FloatMap (PFM) image	image/x-portable-floatmap	PFM	adRASTERIMAGE
RGBE_Fmt	1167	1098	Radiance RGBE (HDR) image	image/vnd.radiance	HDR, PIC, RGBE, XYZE	adRASTERIMAGE
APNG_Fmt	1168	1099	Animated Portable Network Graphics (Animated-PNG)	image/apng	APNG, PNG	adANIMATION
Enhanced_Compressed_Wavelet_Fmt	1169	1100	Enhanced Compressed Wavelet image	image/ecw	ECW	adRASTERIMAGE
Ensoniq_Waveset_Fmt	1170	1101	Ensoniq Waveset audio data file		ECW	adSOUND
Corel_Photo_Paint_Fmt	1171	1102	Corel Photo Paint (version 7 and higher)	image/x-corelphotopaint	CPT	adRASTERIMAGE
OpenRaster_Fmt	1172	1103	OpenRaster image	image/openraster	ORA	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Krita_Fmt	1173	1104	Krita image	application/x-krita	KRA	adRASTERIMAGE
Gerber_Fmt	1174	1105	Gerber image format	application/vnd_gerber	GBR	adVECTORGRAPHIC
PGML_Fmt	1175	1106	Precision Graphics Markup Language		PGML	adVECTORGRAPHIC
Away3D_Fmt	1176	1107	Away3D scene file		AWD	adCAD
CAD_3MF_Fmt	1177	1108	3D Manufacturing Format document	application/vnd.ms-package.3dmanufacturing-3dmodel+xml	3MF	adCAD
AMF_Fmt	1178	1109	Additive manufacturing file format (AMF) document	application/x-amf	AMF	adCAD
C3D_Fmt	1179	1110	Coordinate 3D (C3D) format		C3D	adCAD
CAD_3DSystems_BFF_Fmt	1180	1111	3D Sprint (3D Systems) SLA Build file		BFF	adCAD
NRRD_Fmt	1181	1112	NRRD (nearly raw raster data) image format		NRRD	adRASTERIMAGE
Cinema_4D_Fmt	1182	1113	Cinema 4D model		C4D	adCAD
FBX_ASCII_Fmt	1183	1114	Kaydara FBX project (ASCII)		FBX	adCAD
FBX_Binary_Fmt	1184	1115	Kaydara FBX project (binary)		FBX	adCAD
Wavefront_OBJ_Fmt	1185	1116	Wavefront OBJ geometry definition file		OBJ	adCAD
Wavefront_MTL_Fmt	1186	1117	Wavefront Material Template Library (MTL)		MTL	adCAD
MS_Power_BI_Template_Fmt	1187	1118	Microsoft Power BI Desktop template format		PBIT	adANALYTICS
Windows_Sticky_Notes_Fmt	1188	1119	Microsoft Windows Sticky Notes format		SNT	adWORDPROCESSOR
BlakHole_Fmt	1189	1120	BlakHole compression format		BH	adENCAPSULATION
PowerArchiver_Fmt	1190	1121	PowerArchiver PA compression format		PA	adENCAPSULATION
PageMagic_Fmt	1191	1122	NEBS PageMagic format		DTP	adDESKTOPPUBLSH
PIM_Archiver_Fmt	1192	1123	PIM Archiver format		PIM	adENCAPSULATION
Softdisk_Text_Compressor_Fmt	1193	1124	Softdisk Text Compressor format		CTX	adENCAPSULATION
Ability_PhotoPaint_Fmt	1194	1125	Ability Office PhotoPaint image		APX	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Softlib_Fmt	1195	1126	Softdisk Softlib compression format		SLB	adENCAPSULATION
Timeworks_Publisher_Fmt	1196	1127	Timeworks Publisher (Publish It) format		DTP	adDESKTOPPUBLSH
Scribe_Fmt	1197	1128	Scribe markup language and word processing system		MSS	adWORDPROCESSOR
SQLite_Write_Ahead_Log_Fmt	1198	1129	SQLite Write-Ahead Log file		WAL	adDATABASE
SQLite_WAL_Index_Fmt	1199	1130	SQLite WAL-index (shm) file		SHM	adDATABASE
AutoForm_Design_Fmt	1200	1131	AutoForm Design file		AFD	adCAD
TSV_Fmt	1201	1132	Tab-separated values (TSV) file	text/tab-separated-values	TSV, TAB	adWORDPROCESSOR
OpenStreetMap_XML_Fmt	1202	1133	OpenStreetMap XML data		OSM	adGIS
OpenStreetMap_PBF_Fmt	1203	1134	OpenStreetMap Protocolbuffer Binary Format data file (.osm.pbf)		PBF	adGIS
Nero_Audio_Compilation_Fmt	1204	1135	Nero Audio-CD compilation file		NRA	adMISC
Nero_ISO_Compilation_Fmt	1205	1136	Nero ISO compilation file		NRI	adMISC
WordStar_for_Windows_Fmt	1206	1137	WordStar for Windows file		WSD	adWORDPROCESSOR
MS_Outlook_PAB_Fmt	1207	1138	Microsoft Outlook Personal Address Book (PAB)		PAB	adMISC
HLSL_FXO_Fmt	1208	1139	DirectX High-Level Shader Language (HLSL) pre-compiled shader		FXO	adCAD
HLSL_CSO_Fmt	1209	1140	DirectX High-Level Shader Language (HLSL) compiled shader object		CSO	adCAD
Oberon_Document_Fmt	1210	1141	Component Pascal / Oberon Document file ⁴		ODC	adSOURCECODE
Oberon_Symbol_Fmt	1211	1142	Component Pascal / Oberon Symbol file		OSF	adOBJECTMODULE
Oberon_Code_Fmt	1212	1143	Component Pascal / Oberon Code (executable and loadable object) file		OCF	adEXECUTABLE
Python_Bytecode_Fmt	1213	1144	Python compiled bytecode	application/x-bytecode.python	PYC	adEXECUTABLE
PCPaint_Fmt	1214	1145	PCPaint / Pictor Paint image format		PIC	adRASTERIMAGE
PCRaster_Map_Fmt	1215	1146	PCRaster Map / Cross System		MAP, CSF	adGIS

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			Format geographical data			
COM_Type_Library_Fmt	1216	1147	Microsoft Component Object Model (COM) Type library		TLB	adLIBRARY
MS_Visual_C_Export_Fmt	1217	1148	Microsoft Visual C++ Export file		EXP	adLIBRARY
Lotus_Organizer_Report_Fmt	1218	1149	Lotus Organizer report document		REP	adSCHEDULE
Audible_Audiobook_AA_Fmt	1219	1150	Audible Audiobook (AA) file	audio/audible	AA	adSOUND
DOS_RED_Fmt	1220	1151	MS-DOS RED installer library format		RED	adLIBRARY
CA_ZIPXP_Fmt	1221	1152	CA Technologies ZIPXP compressed document		CAZ	adENCAPSULATION
Kindle_Topaz_Fmt	1222	1153	Amazon Kindle Topaz eBook		AZW, AZW1, TPZ	adWORDPROCESSOR
Windows_Shim_Database_Fmt	1223	1154	Microsoft Windows Shim Database file		SDB	adDATABASE
MS_Incremental_Linker_Fmt	1224	1155	Microsoft Visual Studio incremental linker file		ILK	adMISC
Lotus_Smart_Icon_Fmt	1225	1156	Lotus Smart Icon image file		SMI	adRASTERIMAGE
Lotus_Organizer_Layout_Fmt	1226	1157	Lotus Organizer print/paper layout file		PLT	adSCHEDULE
CMZ_Fmt	1227	1158	CMZ compression format		CMZ	adENCAPSULATION
RFFlow_Fmt	1228	1159	RFFlow flowchart document		FLO	adPRESENTATION
InstallShield_Script_Fmt	1229	1160	InstallShield script document		INS	adENCAPSULATION
InstallShield_Rules_Fmt	1230	1161	InstallShield Compiled Rules file		INX	adENCAPSULATION
Windows_FTS_Fmt	1231	1162	Microsoft Windows 95/NT help full-text-search file		FTS	adDATABASE
DVD_Info_Fmt	1232	1163	DVD Information (IFO) file	content/dvd	IFO	adDATABASE
Emacs_Lisp_Bytecode_Fmt	1233	1164	Byte-compiled Lisp (Emacs/XEmacs)	application/x-bytecode.elisp	ELC	adEXECUTABLE
Windows_Resource_Fmt	1234	1165	Microsoft Windows binary resource file		RES	adMISC
MS_Precompiled_Header_Fmt	1235	1166	Microsoft Visual C/C++ binary pre-compiled header		PCH	adMISC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Borland_Turbo_Project_Fmt	1236	1167	Borland Turbo C project file		PRJ	adMISC
PS_Font_Descriptor_Fmt	1237	1168	PostScript binary Font Descriptor file		NTF	adFONT
MySQL_Index_Fmt	1238	1169	MySQL MyISAM Table index		MYI	adDATABASE
MS_SQL_Fmt	1239	1170	Microsoft SQL Server primary database file		MDF	adDATABASE
DNL_eBook_Fmt	1240	1171	DNAML DNL eBook		DNL	adWORDPROCESSOR
GD_Image_Fmt	1241	1172	GD Library image		GD, GD2	adRASTERIMAGE
iTunes_Library_Fmt	1242	1173	Apple iTunes music library		ITL	adDATABASE
MS_SQM_Fmt	1243	1174	Microsoft Windows Live Messenger/Mail log file		SQM	adMISC
VIFF_Fmt	1244	1175	Khoros Visualization Image File Format (VIFF)	image/x-viff	XV, VIF, VIFF	adRASTERIMAGE
JBIG_Fmt	1245	1176	JBIG (JBIG1) image	image/jbig	JBG, JBIG, BIE	adRASTERIMAGE
CodeWarrior_Project_Fmt	1246	1177	CodeWarrior C/C++ project		MCP	adMISC
PaintShop_Pro_JBF_Fmt	1247	1178	PaintShop Pro JBF image cache file	image/jbf	JBF	adMISC
Delphi_Diagram_Portfolio_Fmt	1248	1179	Delphi Diagram Portfolio file		DDP	adMISC

¹MHT, EML, and MBX files might return either format 2, 233, or 395, depending on the text in the file. In general, files that contain fields such as **To**, **From**, **Date**, or **Subject** are considered to be email messages; files that contain fields such as **content-type** and **mime-version** are considered to be MHT files; and files that do not contain any of those fields are considered to be text files.

²All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.

³This format is returned only if you enable source code identification. See [Source Code Identification, on page 86](#).

⁴This format is returned only if you enable extended source code identification. See [Source Code Identification, on page 86](#).

Appendix C: Character Sets

This section provides information on the handling of character sets in the KeyView suite of products, which includes KeyView Filter SDK, KeyView Export SDK, and KeyView Viewing SDK.

- [Multibyte and Bidirectional Support](#) 275
- [Coded Character Sets](#) 283

Multibyte and Bidirectional Support

The KeyView SDKs can process files that contain multibyte characters. A multibyte character encoding represents a single character with consecutive bytes. KeyView can also process text from files that contain bidirectional text. Bidirectional text contains both Latin-based text which is read from left to right, and text that is read from right to left (Hebrew and Arabic).

The following table indicates which character encodings are supported by KeyView for each format.

Multibyte and bidirectional support

Format	Single-byte	Multibyte	Bidirectional
Archive			
7-Zip (7Z)	n/a	n/a	n/a
AD1 Evidence file	n/a	n/a	n/a
ADJ	n/a	n/a	n/a
B1	n/a	n/a	n/a
BinHex (Hqx)	n/a	n/a	n/a
Bzip2 (BZ2)	n/a	n/a	n/a
EnCase – Expert Witness Compression Format (E01)	n/a	n/a	n/a
GZIP (GZ)	n/a	n/a	n/a
ISO (ISO)	n/a	n/a	n/a
Java Archive (JAR)	n/a	n/a	n/a
Legato EMailXtender Archive (EMX)	n/a	n/a	n/a
MacBinary (BIN)	n/a	n/a	n/a
Mac Disk Copy Disk Image (DMG)	n/a	n/a	n/a
Microsoft Backup File (BKF)	n/a	n/a	n/a

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Microsoft Cabinet format (CAB)	n/a	n/a	n/a
Microsoft Compiled HTML Help (CHM)	n/a	n/a	n/a
Microsoft Compressed Folder (LZH)	n/a	n/a	n/a
PKZip (ZIP)	n/a	n/a	n/a
Microsoft Outlook DBX (DBX)	Y	Y	Y
Microsoft Outlook Offline Storage File (OST)	Y	Y	Y
RAR Archive (RAR)	n/a	n/a	n/a
Tape Archive (TAR)	n/a	n/a	n/a
UNIX Compress (Z)	n/a	n/a	n/a
UUEncoding (UUE)	n/a	n/a	n/a
Windows Scrap File (SHS)	n/a	n/a	n/a
WinZip (ZIP)	n/a	n/a	n/a
Binary			
Executable (EXE)	n/a	n/a	n/a
Link Library (DLL)	n/a	n/a	n/a
Computer-aided Design			
AutoCAD Drawing (DWG)	Y	Y	Y
AutoCAD Drawing Exchange (DXF)	Y	Y	Y
CATIA formats (CAT)	Y	N	N
Microsoft Visio (VSD)	Y	Y	Y
Database			
dBase Database	Y	N	N
Microsoft Access (MDB)	Y	Y	N
Microsoft Project (MPP)	Y	Y	N
Desktop Publishing			
Microsoft Publisher	N	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Display			
Adobe Portable Document Format (PDF)	Y	Y ¹	Y
Graphics			
Computer Graphics Metafile (CGM)	Y	N	N
Corel DRAW (CDR)	n/a	n/a	n/a
DCX Fax System (DCX)	Y	N	N
DICOM – Digital Imaging and Communications in Medicine (DCM)	n/a	n/a	n/a
Encapsulated PostScript (EPS)	Y	N	N
Enhanced Metafile (EMF)	Y	Y	N
Graphic Interchange Format (GIF)	n/a	n/a	n/a
JBIG2	n/a	n/a	n/a
JPEG	n/a	n/a	n/a
JPEG 2000	n/a	n/a	n/a
Lotus AMIDraw Graphics (SDW)	n/a	n/a	n/a
Lotus Pic (PIC)	n/a	n/a	n/a
Macintosh Raster (PICT/PCT)	n/a	n/a	n/a
MacPaint (PNTG)	n/a	n/a	n/a
Microsoft Office Drawing (MSO)	n/a	n/a	n/a
Omni Graffle (GRAFFLE)	Y	N	N
PC PaintBrush (PCX)	n/a	n/a	n/a

¹Multibyte PDFs are supported, provided the PDF document is created by using either Character ID-keyed (CID) fonts, predefined CJK CMap files, or ToUnicode font encodings, and does not contain embedded fonts. See the Adobe website and the Adobe Acrobat documentation for more information. Any multibyte characters that are not supported are displayed using the replacement character. By default, the replacement character is a question mark (?).

To determine the type of font encodings that are used in a PDF, open the PDF in Adobe Acrobat, and select File > Document Info > Fonts. If the Encoding column lists Custom or Embedded encodings, you might encounter problems converting the PDF.

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Portable Network Graphics (PNG)	n/a	n/a	n/a
SGI RGB Image (RGB)	n/a	n/a	n/a
Sun Raster Image (RS)	n/a	n/a	n/a
Tagged Image File (TIFF)	Y	N	N
Truevision Targa (TGA)	n/a	n/a	n/a
Windows Animated Cursor (ANI)	n/a	n/a	n/a
Windows Bitmap (BMP)	n/a	n/a	n/a
Windows Icon Cursor (ICO)	n/a	n/a	n/a
Windows Metafile (WMF)	Y	Y	N
WordPerfect Graphics 1 (WPG)	Y	N	N
WordPerfect Graphics 2 (WPG)	Y	N	N
Mail			
Documentum EMC MF Format	Y	Y	Y
Domino XML Language (DXL)	Y	Y	N
GroupWise FileSurf	Y	N	N
Legato Extender (ONM)	Y	Y	N
Lotus Notes database (NSF)	Y	Y	Y
Mailbox (MBX)	Y	Y	Y
Microsoft Entourage Database	Y	Y	Y
Microsoft Outlook (MSG)	Y	Y	Y
Microsoft Outlook Express (EML)	Y	Y	Y
Microsoft Outlook iCalendar	Y	Y	Y
Microsoft Outlook for Macintosh	Y	Y	Y
Microsoft Outlook Offline Storage File	Y	Y	Y
Microsoft Outlook Personal File Folders (PST)	Y	Y	Y
Microsoft Outlook vCard Contact			
Text Mail (MIME)	Y	Y	Y

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Transport Neutral Encapsulation Format	Y	Y	Y
Multimedia			
Advanced Systems Format (ASF)	n/a	n/a	n/a
Audio Interchange File Format (AIFF)	n/a	n/a	n/a
Microsoft Wave Sound (WAV)	n/a	n/a	n/a
MIDI (MID)	n/a	n/a	n/a
MPEG 1 Audio Layer 3 (MP3)	n/a	n/a	n/a
MPEG 1 Video (MPG)	n/a	n/a	n/a
MPEG 2 Audio (MPEGA)	n/a	n/a	n/a
MPEG 4 Audio (MP4)	n/a	n/a	n/a
NeXT/Sun Audio (AU)	n/a	n/a	n/a
QuickTime Movie (QT/MOV)	n/a	n/a	n/a
Windows Video (AVI)	n/a	n/a	n/a
Presentations			
Apple iWork Keynote (GZ)	Y	Y	N
Applix Presents (AG)	character set 1252 only	N	N
Corel Presentations (SHW)	character set 1252 only	N	N
Extensible Forms Description Language (XFD)	Y	Y	N
Lotus Freelance Graphics 2 (PRE)	character set 850 only	N	N
Lotus Freelance Graphics (PRZ)	Y	Japanese, Simple Chinese, Traditional Chinese, Thai only	N
Macromedia Flash (SWF)	Y	Y	N
Microsoft OneNote	Y	Y	N
Microsoft PowerPoint PC (PPT)	character set 1252 only	Traditional Chinese only	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Microsoft PowerPoint Windows (PPT)	Y	Japanese, Simple Chinese, Traditional Chinese, Korean only	Hebrew only
Microsoft PowerPoint Macintosh (PPT)	Y	N	N
Microsoft PowerPoint Windows XML 2007 and 2010 (PPTX)	Y	Y	Y
OASIS Open Document (ODP)	Y	Y	N
OpenOffice Impress (ODP)	Y	Y	N
StarOffice Impress (ODP)	Y	Y	N
Spreadsheets			
Apple iWork Numbers (GZ)	Y	Y	N
Applix Spreadsheets (AS)	character set 1252 only	N	N
Comma Separated Values (CSV)	character set 1252 only	N	N
Corel Quattro Pro (QPW/WB3)	Y	N	N
Data Interchange Format (DIF)	Y	Y	Y ¹
Lotus 1-2-3 (123)	Y	Y	Y
Lotus 1-2-3 (WK4)	Y	Y	N
Lotus 123 Charts (123)	Y	Y	N
Microsoft Excel Charts (XLS)	Y	Y	N
Microsoft Excel Macintosh (XLS)	Y	N	N
Microsoft Excel Windows (XLS)	Y	Y	Y ²
Microsoft Excel Windows XML 2007 (XLSX)	Y	Y	N
Microsoft Office Excel Binary Format (XLSB)	Y	Y	N
Microsoft Works Spreadsheet (S30/S40)	Y	N	N
OASIS Open Document (ODS)	Y	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
OpenOffice Calc (ODS)	Y	Y	N
StarOffice Calc (ODS)	Y	Y	N
Text and Markup			
ANSI (TXT)	Y	Y	Y2
ASCII (TXT)	Y	Y	Y2
HTML (HTM)	Y	Y	Y2, 2
Microsoft Excel Windows XML 2003	Y	Y	Y
Microsoft Word for Windows XML 2003	Y	Y	Y
Microsoft Visio XML 2003	Y	Y	Y
Rich Text Format (RTF)	Y	Y	Y 3
Unicode HTML	Y	Y	Y 2,3
Unicode Text (TXT)	Y	Y	Y2
XHTML	Y	Y	Y3
XML	Y	Y	Y
Word Processing			
Adobe Maker Interchange Format (MIF)	character set 1252 only	N	N
Apple iChat Log (ICHAT)	Y	Y	N
Apple iWork Pages (GZ)	Y	Y	N
Applix Words (AW)	character set 1252 only	N	N
DisplayWrite (IP)	character set 500, 1026 only	N	N
Folio Flat File (FFF)	character set 1252 only	N	N
Founder Chinese E-paper Basic (CEB)	Y	Y	N
Fujitsu Oasys (OA2)	Y	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Hangul (HWP)	Y	Y	N
Health level7 (HL7)	Y	Y	Y
IBM DCA/RTF (DC)	character sets 500, 1026 only	N	N
JustSystems Ichitaro (JTD)	Y	Y	N
Lotus AMI Pro (SAM)	Y	Simple Chinese, Traditional Chinese, Japanese, Thai only	Y
Lotus AMI Professional Write Plus (AMI)	Y	Simple Chinese, Traditional Chinese, Japanese, Thai only	N
Lotus Word Pro (LWP)	Y	Y	Y ³
Lotus SmartMaster (MWP)	Y	Y	N
Microsoft Word PC (DOC)	character set 1252 only	N	N
Microsoft Word Windows V1-2 (DOC)	Y	N	N
Microsoft Word Windows V6, 7, 8, 95 (DOC)	Y	Y	Hebrew only ³
Microsoft Word Windows V97 through 2003 (DOC)	Y	Y	Y ³
Microsoft Word Windows XML 2007 and 2010 (DOCX)	Y	Y	Y ³
Microsoft Word Macintosh (DOC)	Y	N	Y ³
Microsoft Works (WPS)	Y	Japanese only	N
Microsoft Write (WRI)	Y	Japanese only	N
OASIS Open Document (ODT)	Y	Y	N
Omni Outliner (OO3)	Y	Y	N
OpenOffice Writer (ODT)	Y	Y	N
Open Publication Structure eBook (EPUB)	Y	Y	Y
StarOffice Writer (ODT)	Y	Y	N
Skype Log (DBB)	Y	Y (null-terminated charsets)	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
WordPad (RTF)	Y	Y	Y
WordPerfect Linux (WPS)	Y	N	N
WordPerfect Macintosh (WPS)	Y	N	N
WordPerfect Windows (WO)	Y	N	N
XML Paper Specification (XPS)	Y	Y	N
XYWrite Windows (XY4)	character set 1252 only	N	N
Yahoo! Instant Messenger (DAT)	Y	Y (null-terminated charsets)	N

¹The text direction in the output file might not be correct.

²In Export SDK, a bidirectional right-to-left (RTL) tag is extracted from this format and included in the direction element (<dir=RTL>) of the output.

Coded Character Sets

This section lists which character set you can use to specify the target character set. The coded character sets are enumerated in `kvtypes.h` and defined in the Filter class.

Code Character Sets

Coded Character Set	Description	Can be set as target charset?
KVCS_UNKNOWN	Unknown character set	N
KVCS_SJIS	Japanese (uses multibyte encoding), cp932	Y
KVCS_GB	Simplified Chinese (China, Singapore, Malaysia) cp936	Y
KVCS_BIG5	Traditional Chinese (Taiwan, Hong Kong, Macaw) cp950	Y
KVCS_KSC	Korean, cp949	Y
KVCS_1250	Windows Latin 2 (Central Europe)	Y
KVCS_1251	Windows Cyrillic (Slavic)	Y

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_1252	Windows Latin 1 (ANSI)	Y
KVCS_1253	Windows Greek	Y
KVCS_1254	Windows Latin 5 (Turkish)	Y
KVCS_1255	Windows Hebrew	Y
KVCS_1256	Windows Arabic	Y
KVCS_1257	Windows Baltic Rim	Y
KVCS_1258	Windows Vietnamese	Y
KVCS_8859_1	ISO 8859-1 Latin 1 (Western Europe, Latin America)	Y
KVCS_8859_2	ISO 8859-2 Latin 2 (Central Eastern Europe)	Y
KVCS_8859_3	ISO 8859-3 Latin 3 (S.E. Europe)	Y
KVCS_8859_4	ISO 8859-4 Latin 4 (Scandinavia/Baltic)	Y
KVCS_8859_5	ISO 8859-5 Latin/Cyrillic	Y
KVCS_8859_6	ISO 8859-6 Latin/Arabic	Y
KVCS_8859_7	ISO 8859-7 Latin/Greek	Y
KVCS_8859_8	ISO 8859-8 Latin/Hebrew	Y
KVCS_8859_9	ISO 8859-9 Latin/Turkish	Y
KVCS_8859_14	ISO 8859-14	Y
KVCS_8859_15	ISO 8859-15	Y
KVCS_437	DOS Latin US	Y
KVCS_737	DOS Greek	Y
KVCS_775	DOS Baltic Rim	Y
KVCS_850	DOS Latin 1	Y
KVCS_851	DOS Greek	Y
KVCS_852	DOS Latin 2	Y
KVCS_855	DOS Cyrillic	Y

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_857	DOS Turkish	Y
KVCS_860	DOS Portuguese	Y
KVCS_861	DOS Icelandic	Y
KVCS_862	DOS Hebrew	Y
KVCS_863	DOS Canadian French	Y
KVCS_864	DOS Arabic	Y
KVCS_865	DOS Nordic	Y
KVCS_866	DOS Cyrillic Russian	Y
KVCS_869	DOS Greek 2	Y
KVCS_874	Thai	Y
KVCS_PDFMACDOC	PDF MAC DOC	N
KVCS_PDFWINDOC	PDF WIN DOC	N
KVCS_STDENC	Adobe Standard Encoding	N
KVCS_PDFDOC	Adobe standard PDF character set	N
KVCS_037	EBCDIC code page 037	Y
KVCS_1026	EBCDIC code page 1026	Y
KVCS_500	EBCDIC code page 500	Y
KVCS_875	EBCDIC code page 875	Y
KVCS_LMBCS	Lotus multibyte character set Group 1 and Group 2	N
KVCS_UNICODE	Unicode, UCS-2	Y
KVCS_UTF16	16-bit Unicode transformation format	Y
KVCS_UTF8	8-bit Unicode transformation format	Y
KVCS_UTF7	7-bit Unicode transformation format	Y
KVCS_2022_JP	ISO 2022-JP, Japanese mail and news safe encoding (JIS-7)	N

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_2022_CN	ISO 2022-CN, Chinese mail and news safe encoding	N
KVCS_2022_KR	ISO 2022-KR, Korean mail and news safe encoding	N
KVCS_WP6X	Word Perfect 6.x and higher character mapping	N
KVCS_10000	Western European (Macintosh)	Y
KVCS_KSC5601	Unified Hangul	Y
KVCS_GB2312	Simplified Chinese (China, Singapore, Hong Kong)	Y
KVCS_GB12345	Traditional Chinese (China) - analogue of GB2312	Y
KVCS_CNS11643	Traditional Chinese - Taiwan. Supplement to Big5	Y
KVCS_JIS0201	Japanese - contains ASCII character set (JIS-Roman)	N
KVCS_JIS0212	Japanese. Supplement to JIS0208.	Y
KVCS_EUC_JP	Japanese Extended UNIX Code	Y
KVCS_EUC_GB	Simplified Chinese Extended UNIX Code	Y
KVCS_EUC_BIG5	Traditional Chinese Extended UNIX Code	N
KVCS_EUC_KSC	Korean Extended UNIX Code	N
KVCS_424	EBCDIC Hebrew	N
KVCS_856	PC Hebrew (old)	N
KVCS_1006	IBM AIX Pakistan (Urdu)	N
KVCS_KOI8R	Cyrillic (Russian)	Y
KVCS_PDF_JAPAN1	Adobe-Japan1-2 character collection	N
KVCS_PDF_KOREA1	Adobe-Korea1-0 character collection	N
KVCS_PDF_GB1	Adobe-GB1-3 character collection	N
KVCS_PDF_	Adobe-CNS1-2 character collection	N

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
CNS1		
KVCS_2022_JP_8	ISO 2022-JP, Japanese mail and news safe encoding (JIS8)	N
KVCS_720	Arabic DOS-720	Y
KVCS_VISCII	Vietnamese VISCII	Y
KVCS_8859_10	ISO 8859-10 (Latin 6 Nordic)	Y ¹
KVCS_8859_13	ISO 8859-13 (Latin 7 Baltic)	Y ¹
KVCS_57002	ISCII Devanagari (x-iscii-de)	Y ¹
KVCS_57003	ISCII Bengali (x-iscii-be)	Y ¹
KVCS_57004	ISCII Tamil (x-iscii-ta)	Y ¹
KVCS_57005	ISCII Telugu (x-iscii-te)	Y ¹
KVCS_57006	ISCII Assamese (x-iscii-as)	Y ¹
KVCS_57007	ISCII Oriya (x-iscii-or)	Y ¹
KVCS_57008	ISCII Kannada (x-iscii-ka)	Y ¹
KVCS_57009	ISCII Malayalam (x-iscii-ma)	Y ¹
KVCS_57010	ISCII Gujarathi (x-iscii-gu)	Y ¹
KVCS_57011	ISCII Panjabi (x-iscii-pa)	Y ¹
KVCS_GB18030b2	Reserved for internal use	n/a
KVCS_GB18030	GB18030 (Chinese 4-byte character set)	Y
KVCS_8859_11	ISO 8859-11 (Thai)	Y
KVCS_8859_16	ISO 8859-16 (Latin-10 South-Eastern Europe)	Y
KVCS_ARABICMAC	Arabic Mac (x-mac-arabic)	Y
KVCS_KOI8U	Cyrillic (KOI8U Ukrainian)	Y
KVCS_HZGB2312	The 7-bit representation of GB 2312 / RFC 1842	n/a

¹The character set cannot be forced as output in Export SDK and Viewing SDK because the character

set is not supported by the major browsers.

Appendix D: Extract and Format Lotus Notes Subfiles

This section describes how to create XML templates to alter the appearance of extracted Lotus mail note subfiles so that they maintain the look and feel of the original notes.

- [Overview](#)289
- [Customize XML Templates](#)289
- [Template Elements and Attributes](#)291
- [Date and Time Formats](#)296

Overview

KeyView uses the NSF reader, nsfsr, to extract Lotus database files, and places Lotus mail notes in subfiles. The NSF reader uses a set of default XML templates to extract the notes and apply formatting, thereby approximating the look and feel of the original notes.

In some cases, you might need to customize the XML templates, for instance if your notes contain custom data. In such cases, you can modify the existing XML templates or create your own.

During extraction, the NSF reader loads all XML files in the NSFtemplates directory and its subdirectories (except for the NSFtemplates\images directory, which is reserved for images). During initialization, the KeyView XML parser verifies the XML templates. If the templates contain any invalid XML, elements, or attributes, initialization fails and errors are recorded in the nsfsr.log file.

Customize XML Templates

XML templates are enabled by default. In most cases, the default templates should be sufficient; however, you can customize them or create your own as required.

To customize XML templates for Lotus note extraction

1. Modify the template files in the following directory.

```
install\OS\bin\NSFtemplates
```

The main.xml file must exist in the NSFtemplates directory. It is the top-level template file that extracts all subfiles, usually by calling other templates.

2. Make sure that any modifications or additional XML files conform to the supported elements and attributes described in [Template Elements and Attributes, on page 291](#).
3. Extract the Lotus database file.

Use Demo Templates

For testing purposes, you can extract notes by using a set of demo templates, which are provided to demonstrate the proper usage of all the XML elements and attributes, because the default templates do not use all the XML elements.

The demo templates are available at:

install\OS\bin\NSFtemplates

To use the demo XML templates

1. In the `formats.ini` file, set the following parameter.

```
[nsfsr]
UseDemoTemplate=1
```

2. In the `main.xml` file, uncomment the following section.

```
<ifini name="UseDemoTemplate" text="1">
  <call file="demo.xml"/>
  <quit/>
</ifini>
```

Use Old Templates

For testing purposes, you can extract notes by using legacy templates, which produce MHTML output. You can generate similar output by disabling the XML templates, but using the old templates enables you to see the XML code and compare it to the standard and demo templates.

To use the old XML templates

1. In the `formats.ini` file, set the following parameter.

```
[nsfsr]
UseOldTemplate=1
```

2. In the `main.xml` file, uncomment the following section.

```
<ifini name="UseOldTemplate" text="1">
  <call file="default_old.xml">
  <quit>
</ifini>
```

Disable XML Templates

For testing purposes, you can disable XML templates; KeyView extracts the notes in MHTML format. You can compare the MHTML output directly by the NSF reader with the MHTML output indirectly by the NSF reader through the XML templates.

To disable XML templates

1. In the `formats.ini` file, set the following parameter.

```
[nsfsr]
ExtractByTemplate=0
```

Template Elements and Attributes

This section lists the valid XML elements and attributes that you can use when creating or modifying templates. See the demo templates for examples.

Conditional Elements

The following table lists the valid conditional elements.

Conditional elements

Element	Description
<keyview>	The KeyView XML template container ("root") element
<if*>	<p>If the condition from the comparison is true, process the XML. Conditions can be nested up to 25 levels deep.</p> <p>Attributes</p> <ul style="list-style-type: none"> • <code>name</code>. (Required) The name of the main item to compare to <code>item</code> or <code>text</code>. • <code>item</code>. (Required if no <code>text</code>) The name of the item to compare to the item specified by <code>name</code>. • <code>text</code>. (Required if no <code>item</code>) The text to compare to the item specified by <code>name</code>.
<ifex>, <ifnx>	<p>If <code>name</code> item exists and has a <code>text</code> value or not.</p> <p>The Notes item might have a value that cannot be converted to text, such as an image.</p>
<ifeq>, <ifne>, <iflt>, <ifle>, <ifgt>, <ifge>	<p>Respectively, if <code>text</code> ==, !=, <, >, <=, >, >=.</p> <p>Text comparison uses a case-insensitive string compare.</p>
<iftdeq>, <iftdne>, <iftdlt>, <iftdle>, <iftdgt>, <iftdge>	<p>Respectively, if time/date ==, !=, <, >, <=, >, >=.</p> <p>Time/date comparison converts dates to text in local time using the Notes default, <code>TZ_FMT_NEVER</code>, because Notes also sometimes converts fields to text internally. For example:</p> <pre>text="06/30/2005 02:52:04 PM"</pre>

Conditional elements, continued

Element	Description
<iftzeq>, <iftzne>	Respectively, if the time zone equals or does not equal the comparison text, for example CDT, EST, and so on.
<ifini>	If the value of the INI option specified in name equals the text value.
<else>	If the condition from the last <if> or <switch> was false, process XML.
<switch>	If a name value exists, process XML. Attributes <ul style="list-style-type: none"> name. (Required) The name of the main item to compare in <case> subelements.
<case>	If the comparison condition is true, process XML, then stop processing the rest of <switch>. Attributes <ul style="list-style-type: none"> text. (Required) The text to compare to the name item of <switch>.
<default>	If all <case> conditions were false, process XML. This element must be the last element in <switch>, after all the <case> elements. Any <case> elements after the <default> element are ignored.
<for>	If a name value exists, process XML. Process for each part of the name item. Attributes <ul style="list-style-type: none"> name. (Required) The name of the main item. max. (Optional) The maximum index to process. By default, all are processed.
<index>	Output <for> loop index (1-based). <index> is only valid within a <for> element.

Control Elements

The following table lists the valid control elements.

Control Elements

Element	Description
<call>	Call another XML template. You can nest templates up to 10 levels deep. Attributes

Control Elements, continued

Element	Description
	<ul style="list-style-type: none"> file. (Required) The template file name. This name must be unique.
<log>	<p>Log message to the NSF log file.</p> <p>Attributes</p> <ul style="list-style-type: none"> text. (Required) The text to log. type. (Optional) The type of log message. The following values are valid: <ul style="list-style-type: none"> ERROR WARN INFO DIAG (the default option) DEBUG DUMP
<quit>	<p>Stop processing the template. Exits without error.</p> <p>Attributes</p> <ul style="list-style-type: none"> text. (Optional) The text to log. type. (Optional) The type of log message. See <log>, above.
<stop>	<p>Stop processing the template. Exits with an ERROR log message.</p> <p>Attributes</p> <ul style="list-style-type: none"> text. (Required) The text to log.

Data Elements

The following table lists the valid data elements.

Data elements

Element	Description
<text>	<p>Output text.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.
<rich>	<p>Output rich text (MHTML). Images are output in the next part or parts of the MHTML, after the first <HTML> part.</p>

Data elements, continued

Element	Description
	<p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.
<body>	Output the message body in rich text (MHTML). As with <rich>, on the previous page, images are output in the next part or parts of the MHTML.
<form>	Output the message form (usually \$Body field) in rich text (MHTML). <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.
<addr>	Output an address. <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output. type. (Optional) The type of address to output. Set this attribute to CN (Common Name), which is the only supported type.
<name>	Output the name of the last name item, or in other words the current main item. The item must exist.
<format>	Set the default format for <date> and <date_kv>. This element does not set the <text> format. See Date and Time Formats, on page 296 for a list of all Notes and KeyView date and time formats and integer values. <p>Attributes</p> <ul style="list-style-type: none"> format. (Optional. Omit to reset to defaults) The Notes and KeyView date and time format. You can set the following formats: <ul style="list-style-type: none"> TD=int. The Time Date format (TDFMT_*) TS=int. The Time Show format (TSFMT_*) TT=int. The Time Time format (TTFMT_*) TZ=int. The Time Zone format (TZFMT_*) KV=int. The KeyView date and time format <p>where int is an integer value that corresponds to the desired format.</p> <p>Separate multiple formats with commas. For example:</p> <pre>format="TD=0,TS=2,TT=1,TZ=1,KV=55"</pre>
<date>	Output a Notes date. <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.

Data elements, continued

Element	Description
	<ul style="list-style-type: none"> format. (Optional) See <format>, on the previous page. You can set the following values: <ul style="list-style-type: none"> TD TS TT TZ
<date_kv>	<p>Output a KeyView date.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output. format. (Optional) See <format>, on the previous page. You can set the following values: <ul style="list-style-type: none"> TZ KV
<time>	<p>Output a time range, for example 1 hour, 30 minutes.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The item name of the start date or time. item. (Required) The item name of the end date or time.
<zone>	<p>Output a Notes time zone mnemonic, for example MST.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of date item to output.
<zone_utc>	<p>Output a time zone as UTC, for example (UTC-06:00).</p>
<logo>	<p>Output the mail header logo.</p> <p>The image link is included in the output; the actual image is output to a different part of the MHTML subfile.</p>
<image>	<p>Output an image.</p> <p>The image link is included in the output; the actual image is output to the MHTML next part, as with <rich>, on page 293 and <body>, on the previous page.</p>
<image_uri>	<p>Output an image URI, in quotation marks. The actual image is output to a different part of the MHTML subfile.</p> <p>Attributes</p>

Data elements, continued

Element	Description
	<ul style="list-style-type: none"> • <code>link</code>. (Required if there is no <code>file</code>) The image link, such as a form or title name. For example: <ul style="list-style-type: none"> • <code>link="StdNotesLtr0"</code> • <code>file</code>. (Required if there is no <code>link</code>) The name of the image file. The file must exist in the <code>../../templates/images</code> directory. For example: <ul style="list-style-type: none"> • <code>file="boxcheck.gif"</code>

Date and Time Formats

This section lists the supported Notes and KeyView date and time formats for use with `<format>`, `<date>`, and `<date_kv>`.

Lotus Notes Date and Time Formats

This section lists supported Lotus Notes date and time formats, and the integer values that specify each one.

Lotus Notes date and time formats

Format	Integer Value	Description
TDFMT_FULL	0	(The Notes default) Year, month, and day
TDFMT_CPARTIAL	1	Month and day, year if not this year
TDFMT_PARTIAL	2	Month and day
TDFMT_DPARTIAL	3	Year and month
TDFMT_FULL4	4	Four-digit year, month, and day
TDFMT_CPARTIAL4	5	Month and day, four-digit year if not this year
TDFMT_DPARTIAL4	6	Four-digit year and month
TTFMT_FULL	0	(Notes default) Hour, minute, and second
TTFMT_PARTIAL	1	Hour and minute
TTFMT_HOUR	2	Hour

Lotus Notes date and time formats, continued

Format	Integer Value	Description
TZFMT_NEVER	0	(Notes default) All time zones are converted to the current time zone
TZFMT_SOMETIMES	1	Show only when outside the current time zone
TZFMT_ALWAYS	2	Show for all time zones
TSFMT_DATE	0	Date
TSFMT_TIME	1	Time
TSFMT_DATETIME	2	(The Notes default) Date and time
TSFMT_CDATETIME	4	Date and time, or time today or time yesterday

KeyView Date and Time Formats

This section lists KeyView date and time formats. The KeyView formats use the following syntax:

- Month Month = full month name
 Mon = abbreviated month name
 m = month (number)
 mm = two-digit month (leading 0)
- Weekday Weekday = full weekday name
 Wday = abbreviated weekday name
- Year yy = two-digit year
 yyyy = four-digit year
- >Day d = day (number)
 dd = two-digit day (leading 0)
- Time h = 12-hour
 H = 24-hour
 m = minutes
 s = seconds
 P = AM/PM
 p = am/pm

Separators _ = space
 c = comma
 s = slash
 a = dash
 o = dot

KeyView date and time formats

Format	Output	Integer Value
12-Hour and 24-Hour Time Formats		
KVDTF_P	P	1
KVDTF_P_hmm	P h:mm	2
KVDTF_hmm_P	h:mm P	3
KVDTF_P_hhmm	P hh:mm	4
KVDTF_hhmm_P	hh:mm P	5
KVDTF_P_hmmss	P h:mm:ss	6
KVDTF_hmmss_P	h:mm:ss P	7
KVDTF_P_hhmmss	P hh:mm:ss	8
KVDTF_hhmmss_P	hh:mm:ss P	9
KVDTF_Hmm	H:mm	10
KVDTF_HHmm	HH:mm	11
KVDTF_mmss	mm:ss	12
KVDTF_Hmmss	H:mm:ss	13
KVDTF_HHmss	HH:mm:ss	14
Numerical Date Formats with Slashes		
KVDTF_mmsdd	mm/dd	15
KVDTF_msdsyy	m/d/yy	16
KVDTF_mmsddsyy	mm/dd/yy	17
KVDTF_mmsddsyyyy	mm/dd/yyyy	18
KVDTF_ddsmm	dd/mm	19

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_ddsmsyy	dd/mm/yy	20
KVDTF_ddsmsyy_Hmm	dd/mm/yy H:mm	21
KVDTF_ddsmm_P_hmm	dd/mm P h:mm	22
KVDTF_ddsmm_hmm_P	dd/mm h:mm P	23
KVDTF_ddsmm_P_hhmm	dd/mm P hh:mm	24
KVDTF_ddsmm_hhmm_P	dd/mm hh:mm P	25
KVDTF_ddsmsyy_P_hmm	dd/mm/yy P h:mm	26
KVDTF_ddsmsyy_hmm_P	dd/mm/yy h:mm P	27
KVDTF_ddsmsyy_P_hmmss	dd/mm/yy P h:mm:ss	28
KVDTF_ddsmsyy_hmmss_P	dd/mm/yy h:mm:ss P	29
KVDTF_ddsmsyy_P_hhmmss	dd/mm/yy P hh:mm:ss	30
KVDTF_ddsmsyy_hhmmss_P	dd/mm/yy hh:mm:ss P	31
KVDTF_yysmmsdd_P_hhmmss	yy/mm/dd P hh:mm:ss	32
KVDTF_yysmmsdd_hhmmss_P	yy/mm/dd hh:mm:ss P	33
KVDTF_msdsyy_Hmm	m/d/yy H:mm	34
KVDTF_mmsddsyy_Hmm	mm/dd/yy H:mm	35
KVDTF_msdsyy_P_hmm	m/d/yy P h:mm	36
KVDTF_msdsyy_hmm_P	m/d/yy h:mm P	37
KVDTF_mmsddsyy_hmm_P	mm/dd/yy h:mm P	38
KVDTF_mmsdd_P_hhmm	mm/dd P hh:mm	39
KVDTF_mmsdd_hhmm_P	mm/dd hh:mm P	40
KVDTF_mmsddsyy_P_hhmmss	mm/dd/yy P hh:mm:ss	41
KVDTF_mmsddsyy_hhmmss_P	mm/dd/yy hh:mm:ss P	42
KVDTF_msd	m/d	43
KVDTF_yysm	yy/m	44
KVDTF_yysmm	yy/mm	45

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_ysmsd	yy/m/d	46
KVDTF_ysmmsdd	yy/mm/dd	47
KVDTF_yyyysmmsdd	yyyy/mm/dd	48
Numerical Date Formats with Dashes		
KVDTF_ddammayy	dd-mm-yy	49
KVDTF_mmadd	mm-dd	50
KVDTF_mmayy	mm-yy	51
KVDTF_yyammadd	yy-mm-dd	52
KVDTF_yyyymmadd	yyyy-mm-dd	53
KVDTF_yyyymmaddaHHmss	yyyy-mm-dd-HH:mm:ss	54
Numerical Date Formats with Dots		
KVDTF_yyomod	yy.m.d	55
KVDTF_yyommodd	yy.mm.dd	56
KVDTF_mod	m.d	57
KVDTF_mmodd	mm.dd	58
Numerical and String Date Formats with Dashes, Commas, and Spaces		
KVDTF_ddaMon	dd-Mon	59
KVDTF_daMonayy	d-Mon-yy	60
KVDTF_ddaMonayy	dd-Mon-yy	61
KVDTF_ddaMonayyyy	dd-Mon-yyyy	62
KVDTF_Mon	Mon	63
KVDTF_Monayy	Mon-yy	64
KVDTF_Monayyyy	Mon-yyyy	65
KVDTF_Monaddayy	Mon-dd-yy	66
KVDTF_yyammadd_P_hhmmss	yy-mm-dd P hh:mm:ss	67
KVDTF_mmadd_P_hhmm	mm-dd P hh:mm	68

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_Mon_yy	Mon yy	69
KVDTF_Monc_yy	Mon, yy	70
KVDTF_Month	Month	71
KVDTF_Monthyy	Month-yy	72
KVDTF_Month_yy	Month yy	73
KVDTF_Monthc_yy	Month, yy	74
KVDTF_Monthayyyy	Month-yyyy	75
KVDTF_Month_yyyy	Month yyyy	76
KVDTF_Monthc_yyyy	Month, yyyy	77
KVDTF_Mon_dc_yyyy	Mon d, yyyy	78
KVDTF_d_Monc_yyyy	d Mon, yyyy	79
KVDTF_yyyy_Mon_d	yyyy Mon d	80
KVDTF_Month_dc_yyyy	Month d, yyyy	81
KVDTF_d_Monthc_yyyy	d Month, yyyy	82
KVDTF_yyyy_Month_d	yyyy Month d	83
Weekday Date Formats		
KVDTF_wday	wday	84
KVDTF_Weekday	Weekday	85
KVDTF_wdayc_Mon_dc_yyyy	wday, Mon d, yyyy	86
KVDTF_Weekdayc_Month_dc_yyyy	Weekday, Month d, yyyy	87
KVDTF_Weekdayc_d_Monthc_yyyy	Weekday, d Month, yyyy	88

Appendix E: File Format Detection

This section describes how file formats are detected in Filter SDK.

- [Introduction](#) 302
- [Extract Format Information](#) 302
- [Determine Format Support](#) 302
- [Translate Format Information](#) 305
- [Determine a Document Reader](#) 306
- [Category Values in formats.ini](#) 306

Introduction

The KeyView format detection module (`kwad`) detects a file's format, and reports the information to the API, which in turn reports the information to the developer's application. If the detected format is supported by the KeyView SDK, the detection module also loads the appropriate structured access layer and document reader for further processing. For a list of supported formats, see [Supported Formats, on page 191](#).

Extract Format Information

You can extract format information from a document by using either the `fpGetDocInfoStream()` or `fpGetDocInfoFile()` functions. If required, you can then report this information to the developer's application.

The `fpGetDocInfoStream()` and `fpGetDocInfoFile()` functions extract the major format, file class, version, and document attributes, and populate the `ADDOCINFO` structure. This structure and values are defined in the header file `adinfo.h`. See [Filter API Functions, on page 121](#) for more information.

For information on how to translate the extracted format information, see [Translate Format Information, on page 305](#).

Determine Format Support

After the file format is extracted, the detection module uses the `formats.ini` file to determine whether the format is supported by KeyView, and the appropriate structured access layer and reader to load.

The `formats.ini` file is in the directory `install\OS\bin`, where `install` is the path name of the Filter installation directory and `OS` is the name of the operating system. It contains the following information:

- Coded format information. To translate this information, see [Translate Format Information, on page 305](#).
- The reader associated with each format. See [Determine a Document Reader, on page 306](#).

- Configuration parameters.
- Locale settings for internal use.

Example formats.ini file entries

```
123=mw
152=xyw
178=wp6
189=mw6
2=af
200=pdf
205=mb
210=htm
251=htm
```

NOTE: The `formats.ini` file applies to all formats except graphics. Detection of graphics formats is handled by an internal module named KeyView Picture Interchange Format (KPIF).

Refine Detection of Text Files

During text detection, KeyView analyses the first 1 kB and last 1 kB of data in a document. If less than 10% of that data consists of non-ASCII characters, KeyView detects the document as a text file.

However, depending on the type of documents you are working with, the default settings might not provide the desired level of accuracy. Configuration flags enable you to change the amount of data to read at the end of a file, the percentage of non-ASCII characters permitted in a text file, and whether to use or ignore the file extension to determine the document format.

Change the Amount of File Data to Read

During file detection, KeyView reads characters from the beginning and end of a file—by default, it reads the first and last 1,024 bytes of data. Large text files might contain many irrelevant characters at the end of a file, so KeyView might not accurately detect the file format. You can set a configuration flag to increase the amount of data to read from the end of a file during detection.

To change the amount of data to read during detection

- In the `formats.ini` file, set the following flag in the `detection_flags` section:

```
[detection_flags]
non_ascii_chars_end_block_size=kB
```

where *kB* is the number of kilobytes to read from the end of the file, from 0 to 10. The default value is 1.

NOTE: The file size must be greater than the value specified in the flag. If the flag value is greater than the file size, KeyView does not use the flag.

Change the Percentage of Allowed Non-ASCII Characters

By default, if less than 10% of the analyzed data in a document consists of non-ASCII characters, it is detected as a text file. Depending on the type of files that you are working with, changing the default percentage might increase detection accuracy.

To change the percentage of non-ASCII characters allowed in text files

- In the `formats.ini` file, set the following flag in the `detection_flags` section:

```
[detection_flags]
non_ascii_chars_in_text=N
```

where *N* is the percentage of non-ASCII characters to allow in text files. Files that contain a lower percentage of non-ASCII characters than *N* are detected as text files. The default value is 10.

Allow Consecutive NULL Bytes in a Text File

By default, if a document contains consecutive NULL bytes, it is not detected as text. Depending on the type of files that you are working with, changing the default might increase detection accuracy.

To allow consecutive NULL bytes of ASCII characters in text files

In the `formats.ini` file, set the following flag in the `detection_flags` section:

```
[detection_flags]
ascii_allow_null_bytes=I
```

The default value is 0 (do not allow consecutive NULL bytes).

Use the File Extension for Detection

Sometimes KeyView detects certain file formats, such as CSV, as ASCII because of the content of the documents. In such cases, you can configure KeyView to use the file extension to determine the document format. Using the file extension can improve detection of formats such as CSV, but might not detect text files successfully if they have incorrect file extensions.

To use the file extension for ASCII files during detection

- In the `formats.ini` file, set the following flag in the `detection_flags` section:

```
[detection_flags]
use_extension_for_ascii=1
```

The default is 0 (do not use the file extension).

Translate Format Information

Format information can include file attributes in the following categories:

- Major format
- File class
- Minor format
- Major version
- Minor version

Not all categories are required. Many formats only include major format and file class, or major format only.

The format information has the following structure:

MajorFormat.FileClass.MinorFormat.MajorVersion.MinorVersion

For example:

81.2.0.9.0

Each number in the format information represents a file attribute. The entry 81.2.0.9.0 represents a Lotus 1-2-3 Spreadsheet file version 9.0, where

81= Lotus 1-2-3 Spreadsheet (major format)

2 = Spreadsheet (file class)

0 = not defined (minor format)

9 = 9 (major version)

0 = 0 (minor version)

This example applies to the `formats.ini` file. When extracting format information using the `fpDocInfoFile()` or `fpDocInfoStream()` functions, the same format is represented as 294.2.9.0.

NOTE: The format values returned from `fpDocInfoFile()` or `fpDocInfoStream()` differ from those in `formats.ini` because the former defines a unique ID for each major format, while the latter uses a major version, minor version, and minor format to distinguish between formats.

Distinguish Between Formats

The `ADDDOCINFO` structure provides a unique ID for each major format. For example, a call to `fpGetDocInfoFile()` or `fpGetDocInfoStream()` would return 351.1.0 for a Microsoft Word XML format. The major format 351 is unique to this format.

Unlike `ADDDOCINFO`, the `formats.ini` file distinguishes between formats by using the major version number. For example, in the `formats.ini` file, a Microsoft Word 2003 XML format is defined as 285.1.0.100.0. The major format 285 and file class 1 are the same values for generic XML. The major version 100 distinguishes the format as Microsoft Word 2003 XML.

The major version is used to specify the following formats:

- Microsoft Office 2003 XML. This format has the same major format and file class as generic XML (285.1). It is distinguished from generic XML by using the following major versions:
 - Word: 100
 - Excel: 101
 - Visio: 110
- The XHTML format has the same major format and file class as HTML (210.1). It is distinguished from HTML by using the major version 100.

Determine a Document Reader

The format detection module uses the `formats.ini` file to determine whether a format is supported, and to determine the reader to use to parse a format. The entries in the `formats.ini` file list each format's coded value, and an abbreviation for the format's reader.

The reader abbreviation is a truncated version of the reader's library name. Adding "sr" to the end of an abbreviation creates the name of the reader. For example, this example entry specifies that a Lotus 1-2-3 Spreadsheet file version 9.0 is parsed by the Lotus 1-2-3 filter, 1123sr:

```
81.2.0.9.0=1123
```

[List of Required Files for Redistribution, on page 310](#) lists the readers provided with KeyView.

Category Values in formats.ini

The [Detected Formats](#) section lists all of the file formats that can be detected by KeyView, with associated category values for use in the `formats.ini` file. The following tables provide the list of possible file classes and minor formats.

- [File Classes](#)
- [Minor Formats](#)

File Classes

Attribute Number	Description	File class
0	No file class	AutoDetNoFormat
01	Word processor	adWORDPROCESSOR
02	Spreadsheet	adSPREADSHEET
03	Database	adDATABASE
04	Raster image	adRASTERIMAGE

File Classes, continued

Attribute Number	Description	File class
05	Vector graphic	adVECTORGRAPHIC
06	Presentation	adPRESENTATION
07	Executable	adEXECUTABLE
08	Encapsulation	adENCAPSULATION
09	Sound	adSOUND
10	Desktop publishing	adDESKTOPPUBLSH
11	Outline/planning	adOUTLINE
12	Miscellaneous	adMISC
13	Mixed format	adMIXED
14	Font	adFONT
15	Time scheduling	adSCHEDULE
16	Communications	adCOMMUNICATION
17	Object module	adOBJECTMODULE
18	Library module	adLIBRARY
19	Fax	adFAXFORMAT
20	Movie	adMOVIE
21	Animation	adANIMATION
22	Source Code	adSOURCECODE
23	Computer-Aided Design	adCAD
24	BI and analysis tools	adANALYTICS
25	Scientific data	adSCIENTIFIC
26	Geographic Info System	adGIS

Minor Formats

Attribute Number	Minor Format
00	Minor format not defined

Minor Formats, continued

Attribute Number	Minor Format
01	Standard
02	Book
03	Chart
04	Macro
05	Text
06	Binary
07	PC
08	Windows
09	DOS
10	Macintosh
11	RGB
12	TIFF
13	IFF
14	Experimental
15	Format Information
16	RLE
17	Symbol
18	Old
19	Footnote
20	Style
21	Palette
22	Configuration
23	Activity
24	Resource
25	Calculation
26	Glossary

Minor Formats, continued

Attribute Number	Minor Format
27	Spelling
28	Thesaurus
29	Hyphenation
30	Miscellaneous
31	UNIX
32	VAX
33	Driver
34	Archive

Appendix F: List of Required Files for Redistribution

This section lists the Filter files that can be redistributed in your applications under the licensing agreement. Unless noted, these files are in the directory *install\OS\bin*, where *install* is the path of the Filter installation directory and *OS* is the operating system platform.

NOTE: On Windows systems, the libraries are .dll files. On UNIX systems, the libraries are .so, .a, or .sl files.

Core Files

The following core files can be redistributed with your application.

File	Description
formats.ini	Initialization file. For more information on this file, see Determine Format Support, on page 302 .
FilterDotNet.*	The .NET API.
filterfordotnet.dll	Required by the .NET API.
KeyView.jar	The Java API. NOTE: This file can be found at the path <i>install/javaapi/KeyView.jar</i> where <i>install</i> is the Filter SDK installation directory.
KeyViewFilter.	Required by the Java API.
kpifcvt.*	For presentation graphics, converts from one picture format to another.
kpifutil.*	Utility for handling the internal picture interchange format for presentation graphics.
kvfilter_nsl.*	(AIX platforms only.) Alternative Filter API implementation using POSIX standards for starting new processes. See The Filter Process Model, on page 28 .
kvxtract.*	File Extraction API.
kvfilter.*	Filter API.
kvolefio.*	Embedded OLE object writer.
kvutil.*	Internal KeyView utility functions.
kvxpgsa.*	Interface between presentation readers and kvfilter. Required to extract

File	Description
	metadata from AutoCAD files.
kvxssa.*	Interface between spreadsheet readers and kvfilter.
kvxwpsa.*	Interface between word processing readers and kvfilter.
kvzip.*	Zip writer.
kwad.*	File auto-recognition module.
txtcnv.*	Converter for document token stream.
vcredist\	(Windows platforms only) Microsoft Visual C++ Redistributable Packages. NOTE: This folder can be found in the Filter SDK installation directory.

Support Files

The following support files can be redistributed with your application.

File	Description
datafiles\	(Folder) Required by kvlangdetect
NSFtemplates\	(Folder) Templates used by nsfsr to format Lotus mail notes
7z.*	Required by z7zsr and multiarcsr
bentofio.*	Required by 1123sr and kppzrdr.
cbmap.map	Character mappings for Adobe Portable Document Format (PDF).
CEBDLL.dll	Required by cebsr
chartbls.ux	Character mappings.
chmdll.*	Required by chmsr.
codeidentifierplugin.*	Required for source code identification
DFECore.dll	Required by cebsr
Filter.dll	Required by cebsr
kpbmpwrt.*	Required for processing bmp files
kppng.*	Required for ZLIB decompression.
kvdecrypt.*	Decryption utility functions
kvlangdetect.*	Utility functions for language and character set detection.

File	Description
kvxconfig.ini	Contains element extraction settings for XML files.
kvoop.*	Required for out-of-process filtering.
kvthread.*	Required for multithreaded out-of-process filtering.
kv.lic	Contains license information for KeyView products. This file is opened and validated when a KeyView API is used.
langdetecttext.	Required by kvlangdetect.*
libeay32.dll	(Windows platforms only) SSL utility functions used by KeyView mail format readers
libpff.*	Required by pffsr
lib/libstlport.so.1	(Solaris platforms only) Solaris Studio Redistributable
tabledata.dat	Required for table detection
unzipjpg.*	Required for JPEG decompression.
wpmap.*	Extended character mapping for WordPerfect and Corel Presentation.
xmlsh.*	Contains a library of content handlers for each XML file type. Required by the Expat XML parser.

Document Readers

The following readers can be redistributed with your application.

File	Description
ad1sr.*	AD1 Evidence file reader
afsr.*	ASCII reader
aiffsr.*	Audio Interchange Format File (AIFF) reader
asfsr.*	Advanced Systems Format reader
assr.*	Applix Spreadsheet reader
awsr.*	Applix Word reader
b1sr.*	B1 archive reader
bkfsr.*	Microsoft Backup File reader
bmpsr.*	Windows bitmap (BMP) reader

File	Description
bzip2sr.*	Bzip2 reader
cabsr.*	Microsoft Cabinet format reader
cebsr.*	Founder Chinese E-paper Basic reader
chmsr.*	Microsoft Compiled HTML Help reader
csvsr.*	Comma-Separated Values reader
dbfsr.*	dBase Database reader
dbxsr.*	Microsoft Outlook Express DBX reader
dcasr.*	Document Content Architecture/Revisable Form Text (DCA/RFT) reader
dcmsr.*	Digital Imaging and Communications in Medicine (DICOM) reader
difsr.*	Data Interchange Format reader
dmgsr.*	Mac Disk Copy Disk Image File reader
dw4sr.*	DisplayWrite reader
dx1sr.*	Domino XML Language reader
em1sr.*	Microsoft Outlook Express (EML) reader. This is used to filter EML files when the MBX reader is not licensed.
emxsr.*	Legato EMailXtender (EMX) reader
encasesr.*	Expert Witness Compression Format (EnCase) v6 reader
encase2sr.*	Expert Witness Compression Format (EnCase) v7 reader
entsr.*	Microsoft Entourage Database Format reader
epubsr.*	Open Publication Structure eBook reader
foliosr.*	Folio Flat File reader
gifsr.*	Graphics Interchange Format (GIF) reader
gwfssr.*	GroupWise FileSurf reader
h17sr.*	Health level7 reader (metadata only)
htmsr.*	HTML and XHTML reader
hwpsr.*	Hangul 97 reader
hwposr.*	Hangul 2002, 2005, 2007 reader
ichatsr.*	Apple iChat Log reader

File	Description
icssr.*	Microsoft Outlook iCalendar reader
isosr.*	ISO-9660 CD Disc Image Format reader
iwss13sr.*	iWork 13 Numbers reader
iwwp13sr.*	iWork 13 Pages reader
iwwpsr.*	Apple iWork Pages reader
iwsssr.*	Apple iWork Numbers reader
jp2000sr.*	JPEG 2000 metadata reader
jpgsr.*	JPEG metadata reader
jtdsr.*	JustSystems Ichitaro reader
kpagrdr.*	Applix Presentations reader
kpCATrdr.*	CATIA format reader
kpcgmrdr.*	Computer Graphics Metafile reader
kpDWGrdr.*	AutoCAD Drawing format reader
kpDXFrdr.*	AutoCAD Drawing Exchange format reader
kpemfrdr.*	Enhanced Metafile reader
kpGFLrdr.*	Omni Graffle reader
kpgifrdr.*	Graphic Interchange Format (GIF) reader
kpiwpg13rdr.*	iWork 13 keynote reader
kpIWPGrdr.*	Apple iWork Keynote reader
kpjp2000rdr.*	JPEG 2000 reader
kpmsordr.*	Microsoft Office Drawing Objects (office 97, 2000, and XP) reader
kpnbmprdr.*	Notes Bitmap reader (for embedded images in DXL files)
kp0DArdr.*	AutoCAD reader (Windows only)
kpodfrdr.*	Oasis Open Document Format presentation (ODP) reader
kp0NErdr.*	Microsoft OneNote reader
kpoxdrdr.*	Open Office XML Diagram Graphics reader.
kpp40rdr.*	Microsoft PowerPoint PC 4.0 and PowerPoint Mac reader

File	Description
kpp95rdr.*	Microsoft PowerPoint 95 reader
kpp97rdr.*	Microsoft PowerPoint 97 and higher reader
kppctrdr.*	Macintosh Quick Draw Picture (PICT) reader
kppicrdr.*	Pictor PC Paint (PIC) reader
kpppxrdr.*	Microsoft PowerPoint XML reader 2007
kpprerdr.*	Lotus Freelance Graphics for Windows V2.0 reader
kpprzrdr.*	Lotus Freelance Graphics 96/97/98 reader
kpsddrdr.*	StarOffice Impress reader
kpsdwrdr.*	Lotus Ami Pro Graphics reader
kpshwrdr.*	Corel Presentations reader
kpugrdr.*	Unigraphics (UG) NX reader
kpvsd2rdr.*	Microsoft Visio reader
kpvsdxrdr.*	Microsoft Visio 2013 reader
kpwg2rdr.*	WordPerfect Graphics 2 reader
kpwmfrdr.*	Windows Metafile reader
kpwpgrdr.*	WordPerfect Graphics 1 reader
kpXFDLrdr.*	Extensible Forms Description Language reader
kvgzsr.*	GZIP reader
kvhqxsr.*	BinHex reader
kvzeesr.*	UNIX Compress reader
l123sr.*	Lotus 123 v96/97/98 reader
lasr.*	Lotus AMI Pro reader
1tbenn30.d11	Lotus Word Pro support (supported on Windows x86 platform only)
1tscsn10.d11	Lotus Word Pro support (supported on Windows x86 platform only)
1wpapin.d11	Lotus Word Pro support (supported on Windows x86 platform only)
1wppann.d11	Lotus Word Pro support (supported on Windows x86 platform only)
1wpsr.d11	Lotus Word Pro reader (supported on Windows x86 platform only)

File	Description
lzhsr.*	Microsoft Compression Folder reader
macbinsr.*	MacBinary reader
mbsr.*	Microsoft Word Macintosh reader
mbxsr.*	Mailbox (MBX) and Microsoft Outlook Express (EML) reader ¹
mdbsr.*	Microsoft Access reader
mhtsr.*	MIME HTML reader
mifsr.*	Adobe Maker Interchange reader
misr.*	Microsoft Word 2 reader
mp3sr.*	MP3 reader for metadata extraction reader
mpeg4sr.*	MPEG-4 Audio file reader
mppsр.*	Microsoft Project reader
msgsr.*	Microsoft Outlook (MSG) reader
mspubsr.*	Microsoft Publisher reader
msw6sr.*	Microsoft Works 6 and 2000 reader
mswsr.*	Microsoft Works V1 and 2 reader
multiarcsr	ARJ Reader
mw6sr.*	Microsoft Word 95 reader
mw8sr.*	Microsoft Word 97, 2000, and XP reader
mwsr.*	Microsoft Word for DOS and Microsoft Write reader
mwssr.*	Microsoft Works Spreadsheet reader
mwxsr.*	Microsoft Word 2007 XML reader
nsfsr.*	Lotus Notes database reader 1
oa2sr.*	Fujitsu Oasys reader
odfssr.*	Oasis Open Document Format spreadsheets (ODS) reader
odfwpsr.*	Oasis Open Document Format word processing (ODS) reader

¹This reader is an advanced feature and is sold and licensed separately from KeyView Filter SDK. See [License Information, on page 18](#)

File	Description
olesr.*	Embedded OLE object reader
olmsr.*	Microsoft Outlook for Macintosh reader
onmsr.*	Legato EMailXtender Native Message reader
oo3sr.*	Omni Outliner reader
pdf2sr.*	Alternative Adobe Portable Document Format file (PDF) reader
pdfsr.*	Adobe Portable Document Format file (PDF) reader
pffsr.*	Microsoft Outlook Offline Storage File reader
pngsr.*	Portable Network Graphics (PNG) reader
pstsr.dll	Microsoft Outlook Personal Folders file MAPI-based reader (supported on Windows platform only) ¹
pstnsr.*	Microsoft Outlook Personal Folders file native reader ¹
pstxsr.*	Microsoft Outlook Personal Folders file native reader ¹
qpssr.*	Corel Quattro Pro spreadsheet reader
qpwsr.*	Corel Quattro Pro version X4 spreadsheet reader
rarsr.*	RAR Archive reader
riffsr.*	Microsoft WAVE reader
rtfsr.*	Microsoft Rich Text reader
skypesr.*	Skype log file reader
sosr.*	StarOffice/OpenOffice reader
starcsr.*	StarOffice Calc reader
starwsr.*	StarOffice Writer reader
sunadsr.*	Sun Audio Data reader
swfsr.*	Macromedia Flash reader
tarsr.*	Tape archive reader
tifsr.*	TIFF reader (metadata only)
tnefsr.*	Transfer Neutral Encapsulation Format
unihtmsr.*	Unicode HTML reader
unisr.*	Unicode reader

File	Description
unzip.*	Zip file reader
utf8sr.*	UTF-8 reader
uudsr.*	UUEncoding reader
vcfsr.*	Microsoft Outlook vCard Contact reader
vsdsr.*	Microsoft Visio reader
wkssr.*	Lotus 123 v2.0 through 5.0 reader
wosr.*	WordPerfect 5.x reader
wp6sr.*	WordPerfect 6.0 through 10.0 reader
wpmsr.*	WordPerfect for Macintosh reader
xlsbsr.*	Microsoft Office 2007 Excel Binary Format reader
xlssr.*	Microsoft Excel reader
xlsxsr.*	Microsoft Excel 2007 XML reader
xmlsr.*	Generic XML reader
xpssr.*	XML Paper Specification reader
xywsr.*	XYWrite reader
yimsr.*	Yahoo! Instant Messenger reader
z7zsr.*	7-Zip reader

Appendix G: Develop a Custom Reader

This section describes how to develop a reader for a format not supported by KeyView.

• Introduction	319
• How to Write a Custom Reader	320
• Development Tips	330
• Functions	331

Introduction

The Filter SDK enables you to write custom readers for formats not directly supported by KeyView. A reader is required to parse the file format and generate a KeyView token stream, which represents the content and format of the document. Filter can then use this token stream to generate a text version of the original document. The readers interact with a structured access layer and a writer to generate a text file in Filter, an HTML file in HTML Export, an XML file in XML Export, and a near-to-original view of the document in the Viewing SDK.

The complexity of a custom reader depends on the file format used by the source document type. A simple reader extracts only the textual content, but ignores formatting and all other non-textual content. Readers of increasing complexity must address one or more of the following:

- formatting (including fonts, foreground and background colors, paragraph borders and shading, character and paragraph styles)
- tables
- lists
- headers
- footers
- footnotes
- endnotes
- graphics
- bookmarks to internal links
- hyperlinks to external documents or webpages
- other structures, such as a table of contents or index

Even a simple reader might have to parse the following components of a document:

- word processing commands or tags
- encrypted or encoded text
- multiple character sets

- text modified, but retained within the file
- text displayed in an order other than its physical occurrence within the source file

It is very important to fully understand the file specification for the file format used by the document. This is essential in determining how to parse the source file and generate a token stream that accurately and effectively represents the original document.

Within Filter, the custom reader must interact with a structured access layer and the format detection API, which in turn interacts with the top-level API. For a description of the Filter architecture, see [Architectural Overview, on page 21](#).

The custom reader must have a module definition file (*.def) that defines the exported API function calls. In addition, the `formats.ini` file must be modified to identify the custom reader and its associated format detection function.

See the source code for the sample custom reader (`utf8sr`), which parses plain text files encoded in UTF-8. The source code is in the directory `install/samples/utf8sr`, where `install` is the path name of the Filter installation directory.

How to Write a Custom Reader

Two include files define the requirements for a custom reader: `kvcfsr.h` and `kvtoken.h`. The definitions of the KeyView tokens are in `kvtoken.h`. For more information on tokens, see [Token Buffer, on the next page](#). The file `kvcfsr.h` defines two structures: `TPReaderInterface` and `adTPDocInfo`.

The `TPReaderInterface` structure defines the API functions implemented by the custom reader. For basic readers, only the first four functions must be implemented. These functions are called by the structured access layer to parse the source file and generate the token stream.

All readers must be threadsafe. This means that global variables must not be used. To pass information between functions, it is necessary to define a "global" context structure that stores all information required throughout the life of the DLL. The initial parameter of all but one of the `TPReaderInterface` functions is a pointer to a global context structure defined for the custom reader.

The `adTPDocInfo` structure defines the information required for the format detection API, which associates the custom reader with the required file format.

Naming Conventions

Use the following naming conventions for functions and files:

- The initial letters of the custom reader file name should identify the file format being parsed. For example, `pdf` for Adobe PDF files, `rtf` for RTF files, and `xls` for Microsoft Excel files. In the examples in this appendix, this is represented by `xxx`.
- The name of the shared library must end with the letters `sr`.
- The name of the exported functions in the module definition file must be `xxxGetReaderInterface` and `xxxsrAutoDet`.

NOTE: The letters `sr` are excluded from `xxxGetReaderInterface`, but are included in `xxxsrAutoDet`.

Basic Steps

The basic steps for developing a custom reader are as follows.

To develop a custom reader

1. Design the global context structure.
2. Write the basic API functions:
 - `xxxAllocateContext()`
 - `xxxInitDoc()`
 - `xxxFillBuffer()`
 - `xxxFreeContext()`
 - `xxxCharSet()`
 - `xxxsrAutoDet()`

From within the `xxxFillBuffer()` function, it is necessary to call other functions that repeatedly read a chunk of a source file, parse the chunk, and generate a token stream until the entire source file is processed.

3. Map all but the last function to the `TPReaderInterface` structure.
4. Write the module definition file (`*.def`), exporting the reader interface and format detection functions.
5. Modify the `formats.ini` file to identify the custom reader and its associated format detection function. See `xxxsrAutoDet()`, on page 331. For example, the following lines would be added to the `[Formats]` section of the `formats.ini` file for the UTF-8 reader:

```
456.1.0.0=utf8
[CustomFilters]
1=utf8sr
```

Token Buffer

Filter technology parses the native file structure to generate an intermediate stream called a *token buffer*. The token buffer consists of multiple sequences of tokens, which are defined in `kvtoken.h` and listed below.

```
#define KVT_TEXT          0x00 /* PutText() */
#define KVT_PARAINFO     0x01 /* SetParaInfo() */
#define KVT_SETTABS      0x02 /* SetTabs() */
#define KVT_TAB          0x03 /* Tab() */
#define KVT_MODE         0x04 /* SetMode() */
```

```
#define KVT_PARASPACE      0x05 /* SetParaSpace() */
#define KVT_ROWDEFN       0x06 /* DefineRow(), EndTable() */
#define KVT_COLUMNS       0x07 /* StartColumns(), etc. */
#define KVT_CELLSTART     0x08 /* NextCell() */
#define KVT_BITMAP        0x09 /* Reserved for annotations. */
#define KVT_PAGEOBJ       0x0A /* PutHeader(), PrintPage(), etc.*/
#define KVT_NOOP          0x0B /* Just skip a BYTE. */
#define KVT_PAGE_BREAK    0x0C /* PageBreak() */
#define KVT_PARA_BREAK    0x0D /* ParaEnd() */
#define KVT_LINE_BREAK    0x0E /* LineBreak() */
#define KVT_SET_FONT      0x0F /* SetFont() */
#define KVT_PAGE          0x10 /* SetPageInfo() */
#define KVT_HOTSPOT       0x11 /* StartHotSpot() */
#define KVT_LINESPACE     0x12 /* SetLineSpacing() */
#define KVT_COLOR         0x13 /* VESetTextColor(),VESetBkColor()*/
#define KVT_PICTURE       0x14 /* PutPicture() */
#define KVT_CELLMERGE     0x15 /* MergeCells() */
#define KVT_RULE          0x16 /* HorzRule() */
#define KVT_PATTERN       0x17 /* StartPattern(), etc. */
#define KVT_BORDER        0x18 /* StartParaBorder(), etc. */
#define KVT_HEADING       0x19 /* PutParaHeading() */
#define KVT_LISTING       0x1A /* StartList(), etc. */
#define KVT_CHARSET       0x1B /* SetCharSet() */
#define KVT_STYLE         0x1C /* PutCharStyle(), PutParaStyle()*/
#define KVT_BIDI          0x1D /* Set Bidirectional text */
#define KVT_LOCALE        0x1E /* Set locale of a document */
#define KVT_ZONE          0x1F /* StartZone(), EndZone() */
#define KVT_POSITION      0x20 /* SetPosition(), etc. */
#define KVT_AUTOREC       0x21 /* Reserved for Internal Use */
#define KVT_METADATA      0x22 /* Rsserved for Internal Use */
#define KVT_BYTEORDER     0x23 /* SetByteOrder() */
#define KVT_PARASPACEAUTO 0x24 /* SetParaSpaceAuto() */
#define KVT_ATTACH        0x25 /* PutAttachment() */
#define KVT_TOCPrintIMAGE 0x26 /* StartTOCPrintImage(), etc. */
#define KVT_STREAM        0x27 /* PutStream(),Reserved */
#define KVT_REVISIONMARK  0x28 /* StartRevisionMark(),
EndRevisionMark(), SetRMAuthor(), SetRMDateTime() */
#define KVT_DOCXTRINFO    0x29 /* SetDocXtrInfo() */
#define KVT_PCTEMDFT      0x30 /* SetPctEmdFt() */
```

A token is a single-byte identifier that corresponds to attributes in a document. Each token has one or more associated macros that provide detailed information about an attribute. Many of these tokens define components of the document, such as page margins, line indentation, and foreground and background color. Collectively, these are referred to as the *state* of the document. This state changes as the document is parsed.

Macros

Some of the macros are simple while others are complicated. An example of a simple macro is `ParaEnd` (`pcBuf`) which terminates the current paragraph.

```
#define ParaEnd(pcBuf) \
{ \
    *pcBuf++ = KVT_PARA_BREAK; \
    KVT_PUTINT(pcBuf, KVT_SIZE_PARA_BREAK); \
}
```

In Filter SDK, this generates an `0x0d, 0x0a` pair of bytes on a Windows machine. In HTML Export this can generate a `<p style="...">` element, depending on the value of other paragraph attributes.

One of the more complicated macros is `PutPictureEx()`.

```
#define PutPictureEx(pcBuf, lpszKey, cx, cy, flags, \
    scaleHeight, scaleWidth, \
    cropFromL, cropFromT, cropFromR, cropFromB, \
    anchorHorizontal, anchorVertical, offsetX, offsetY)\
{\
    PutPic(pcBuf, lpszKey, cx, cy, flags, \
    scaleHeight, scaleWidth, \
    cropFromL, cropFromT, cropFromR, cropFromB, \
    anchorHorizontal, anchorVertical, offsetX, offsetY, \
    180, 0, 180, 0, -1, 0, 0, 0, 0) \
}
```

You can generate a representation of the token stream by running `filtertest.exe` with the `-d` command-line option. This stream does not include the tokens generated for headers or footers. The `filtertest.exe` is in the directory `install\samples\utf8\bin`, where `install` is the path name of the Filter installation directory.

Reader Interface

All custom readers use the reader interface defined in `kvcfsr.h`. The members of this structure are:

```
fpAllocateContext()  
fpInitDoc()  
fpFillBuffer()  
fpFreeContext()  
fpHotSpothit()  
fpGetSummaryInfo()  
fpOpenStream()  
fpCloseStream()  
fpGetURL()  
fpGetCharSet()
```

NOTE: `fpHotSpothit()` and `fpGetURL()` are currently reserved and must be `NULL`.

Function Flow

The structured access layer calls the functions as follows:

1. `fpAllocateContext()` is called and returns a pointer to the global context structure.
2. After further processing within the structured access layer, `fpInitDoc()` is called. This function performs all required initialization for the global context structure and then returns control to the structured access layer.
3. After further processing within the structured access layer, the `fpFillBuffer()` function is called repeatedly until the document is completely parsed.
4. Finally, `fpFreeContext()` is called. This function frees all memory allocated within the custom reader and then returns control to the structured access layer.

Related Topics

- [Functions, on page 331](#)

Example Development of `fffFillBuffer()`

The following is an example of how the `fpFillBuffer()` function in `foliosr` could be developed. The example demonstrates how the code changes as limitations of the implementation are identified. With each implementation, code revisions are shown in bold.

Implementation 1—`fpFillBuffer()` Function

```
/******  
*Function: fffFillBuffer()  
*Summary: Read fff input from stream and parse into kvtoken.h codes  
*****/  
int pascal _export fffFillBuffer(  
    void    *pCFContext,  
    BYTE    *pcBuf,  
    UINT    *pnBufOut,  
    int     *pnPercentDone,  
    UINT    cbBufOutMax )  
{  
    BOOL bRetVal;  
    TPfffGlobals *pContext = (TPfffGlobals *)pCFContext;  
    pContext->pcBufOut = pcBuf;  
    fffReadSourceFile(pContext);  
    bRetVal = fffProcessBuffer(pContext, pcBuf);  
    *pnPercentDone = (int)(pContext->unTotalBytesProcessed *  
        (UINT)100 / pContext->unFileSize);  
    *pnBufOut = (UINT)(pContext->pcBufOut - pcBuf);  
    return (bRetVal ? KVERR_Success : KVERR_General);  
}
```

The parameters in `fffFillBuffer()` are as follows:

Parameter	In/Out	Description
pCFContext	In	A pointer to the context structure of the custom reader.
pcBuf	In/Out	A pointer to the token output buffer.
pnBufOut	Out	A pointer to the number of bytes written to the output buffer.
pnPercentDone	Out	A pointer to the percentage complete.
cbBufOutMax	In	The maximum number of bytes that the token output buffer can hold.

Structure of Implementation 1

1. The local variable pContext is set to the address of the pCFContext void pointer, cast to a pointer to the global context structure for the reader. This provides access to all members of this structure.
2. After setting the pContext variable, a call is made to read the source file.
3. Next, a call is made to fffProcessBuffer(). The second parameter in the call is a pointer to the token output buffer. If this call fails, usually because of memory allocation errors, it returns FALSE.
4. The percentage complete is calculated.
5. The number of BYTES written to the token output buffer is calculated. This is based on the value of pContext->pcBufOut, which is increased each time a token is written to the buffer.
6. The function returns to the structured access layer.
7. Subsequent calls to fffFillBuffer() are made by the structured access layer until the percentage complete is 100.

Problems with Implementation 1

- There is a limit to the size of the token output buffer, typically 4 KB. If fffProcessBuffer() generates a token stream larger than this, there is a memory overflow. If fffProcessBuffer() generates a small token stream and the entire file has not been read, the output token buffer is underutilized.
- It might not be possible to process the entire input buffer from the source file because of boundary conditions. An example of a "boundary condition" is when the input buffer terminates part way through a control sequence in the original document. Another file read operation is required before the complete control sequence can be parsed.
- This function might be interrupted by other calls from the structured access layer to process headers, footers, footnotes, and endnotes, or to retrieve the document summary information. This can cause values of variables in the global context to change, and the source file to be repositioned.

Implementation 2—Processing a Large Token Stream

Implementation 2 addresses the problem of processing a token stream that is larger than the output buffer size limit.

```

/*****
* Function:  fffFillBuffer()
* Summary:   Read fff input from stream and parse into kvtoken.h codes
*****/
int pascal _export fffFillBuffer(
    void    *pCFContext,
    BYTE    *pcBuf,
    UINT    *pnBufOut,
    int     *pnPercentDone,
    UINT    cbBufOutMax )
{
    BOOL bRetVal = TRUE;
    TPfffGlobals *pContext = (TPfffGlobals *)pCFContext;
    pContext->pcBufOut = pcBuf;
    pContext->cbBufOutMax = 9 * cbBufOutMax / 10; /* Process the portion of the
fff file that is in the input buffer but do * not return from the fffFillBuffer()
function unless the output buffer is * at least 90% full. If any of the memory
allocations fail during the * execution of fffProcessBuffer(), bRetVal will be
set to FALSE, resulting * in this conversion failing "gracefully".
    */
    do
    {
        if( pContext->bBufOutFull )
        {
            pContext->bBufOutFull = FALSE;
        }
        else
        {
            fffReadSourceFile(pContext);
        }
        bRetVal = fffProcessBuffer(pContext, pcBuf);
        *pnPercentDone = (int)(pContext->unTotalBytesProcessed *
            (UINT)100 / pContext->unFileSize);
    }while( bRetVal && !pContext->bBufOutFull && *pnPercentDone < 100 );
    *pnBufOut = (UINT)(pContext->pcBufOut - pcBuf);
    return (bRetVal ? KVERR_Success : KVERR_General);
}

```

Structure of Implementation 2

1. `cbBufOutMax` is used to set `pContext->cbBufOutMax`. This is used in `fffProcessBuffer()` to monitor how full the token output buffer becomes as the source file is processed.
2. When the source file input buffer has been processed, `fffProcessBuffer()` returns, and the percentage complete is calculated.
3. If the token output buffer is not filled to a value greater than `pContext->cbBufOutMax`, `pContext->bBufOutFull` remains set to `FALSE`, and if the percentage complete is less than 100, the `do-while` loop is re-entered without returning from this function to the structured access layer. There is another call to `fffReadSourceFile()`, followed by `fffProcessBuffer()`.
4. When the token output buffer is filled to a value greater than `pContext->cbBufOutMax`, `pContext->bBufOutFull` is set to `TRUE`. In this case, the `do-while` loop ends, the number of bytes written to the token output buffer is calculated, and control returns to the structured access layer.
5. The structured access layer continues to make calls to `fffFillBuffer()` until the entire source file is processed.
6. Each time the structured access layer calls `fffFillBuffer()`, another empty token output buffer is provided for the custom reader to use.
7. If the previous call to `fffFillBuffer()` exited because the previous token output buffer exceeded allowable capacity, `pContext->bBufOutFull` is reset to `FALSE` and no call is made to read the next buffer from the input source file.

Problems with Implementation 2

- It might not be possible to process the entire input buffer from the source file because of boundary conditions.
- This function might be interrupted by other calls from the structured access layer to process headers, footers, footnotes, or endnotes, or to retrieve the document summary information. This can cause values of variables in the global context to change, and the source file to be repositioned.

Boundary Conditions

A boundary condition can result from many situations arising from input file processing. For example, the input buffer might end with an incomplete command. In Folio flat files, this could be an incomplete element. In other word processing documents, a boundary condition might result from an incomplete control sequence, a split double-byte character, or a partial UTF-7 or UTF-8 sequence. These can be handled jointly by `fffProcessBuffer()`, which must detect the boundary condition, and `fffReadSourceFile()`.

The following example shows partial code used in `fffReadSourceFile()`:

```
/*  
*  
* Function:    fffReadSourceFile()  
*/
```

```

*
*****/
int pascal fffReadSourceFile(TPfffGlobals *pContext)
{
    int nBytes;
    /* Transfer remaining data to beginning of buffer prior to next read */
    if( pContext->nResidualBytes )
    {
        memcpy(pContext->cInputBuf, pContext->pcBufIn, pContext->nResidualBytes);
    }
    /* Read from file, without over-writing any text from the previous buffer */
    nBytes = (*pContext->pIO->kwReadFunc)(pContext->pIO,
        pContext->cInputBuf + pContext->nResidualBytes,
        BUFFERSIZE - pContext->nResidualBytes);
    /* Update input buffer control parameters */
    pContext->unTotalBytesRead += (UINT)nBytes;
    pContext->pcBufIn = pContext->cInputBuf;
    pContext->pcBufInMax = pContext->pcBufIn + pContext->nResidualBytes + nBytes;
    pContext->nResidualBytes = 0;
    return nBytes;
}

```

If `fffProcessBuffer()` is unable to process the entire input source file buffer, it sets the value for `pContext->nResidualBytes`. When the next call to `fffReadSourceFile()` is made, any residual bytes are copied to the beginning of the input source file buffer, and the number of bytes to be read is reduced to make sure that this buffer does not overflow.

A good way to test the code for boundary conditions is to vary the size of `BUFFERSIZE` and make sure that the results remain consistent.

NOTE: With `ReadSourceFile()`, the source file can be read by calls to retrieve header or footer information. If this occurs, the value for `pContext->unTotalBytesRead` is incorrect.

Implementation 3—Interrupting Structured Access Layer Calls

Implementation 3 addresses the problem of boundary conditions and interrupting calls from the structured access layer.

```

/*****
* Function:   fffFillBuffer()
* Summary:   Read fff input from stream and parse into kvtoken.h codes
*****/
int pascal _export fffFillBuffer(
    void    *pCFContext,
    BYTE    *pcBuf,
    UINT    *pnBufOut,
    int     *pnPercentDone,
    UINT    cbBufOutMax )
{
    double dTotalBytesProcessed, dFileSize;

```



```
    BOOL bRetVal = TRUE;
    TPfffGlobals *pContext = (TPfffGlobals *)pCFContext;
    pContext->pcBufOut = pcBuf;
    pContext->cbBufOutMax = 9 * cbBufOutMax / 10;
/* Process the portion of the fff file that is in the input buffer but do
* not return from the fffFillBuffer() function unless the output buffer is
* at least 90% full. If any of the memory allocations fail during the
* execution of fffProcessBuffer(), bRetVal will be set to FALSE, resulting
* in this conversion failing "gracefully". */
    do
    {
        if( pContext->bBufOutFull )
        {
            pContext->bBufOutFull = FALSE;
        }
        else
        {
            fffReadSourceFile(pContext);
        }
        bRetVal = fffProcessBuffer(pContext, pcBuf);
        if( pContext->bHeaderCompleted )

    {
        *pnPercentDone = 100;
        pContext->bHeaderCompleted = FALSE;
    }
    else if( pContext->bFooterCompleted )

    {
        *pnPercentDone = 100;
        pContext->bFooterCompleted = FALSE;
    }
    else

    {
        if( pContext->unTotalBytesProcessed >= pContext->unFileSize )
        {
            *pnPercentDone = 100;
        }
        else if( pContext->unFileSize < FFF_MAX_ULONG )
        {
            *pnPercentDone = (int)(pContext->unTotalBytesProcessed *
(UINT)100 / pContext->unFileSize);
        }
        else

    {
        dTotalBytesProcessed = pContext->unTotalBytesProcessed;
        dFileSize = pContext->unFileSize;
```

```
        *pnPercentDone = (int)(dTotalBytesProcessed * 100 / dFileSize);
    }
}
}while( bRetVal && !pContext->bBufOutFull && *pnPercentDone < 100 );
*pnBufOut = (UINT)(pContext->pcBufOut - pcBuf);
return (bRetVal ? KVERR_Success : KVERR_General);
}
```

Structure of Implementation 3

- The most significant change in Implementation 3 is the addition of the code that checks whether the processing of the header or footer is complete. The variables for `pContext->bHeaderCompleted` and `pContext->bFooterCompleted` are set to **TRUE** in `fffProcessBuffer()` when a header or footer is processed and the end of that portion of the document is reached.
- The other piece of code added in Implementation 3 is unique to `foliosr`. Folio files can be 50 MB or larger. Therefore, an unsigned integer is too small to accurately calculate the percentage complete. If the file size exceeds `FFF_MAX_ULONG`, which is defined as `(UINT)(0xFFFFFFFF / 0x64)`, the doubles are used for that calculation.
- Prior to returning, the token output buffer is as full as possible and never overflows. The minimum number of calls is made.

Development Tips

- Avoid unnecessary initialization.

The context variable is allocated in `fpAllocateContext()`. This structure must be immediately `memset()` to zero. This sets all `BOOL` values to **FALSE**, all pointers to **NULL**, and all integers to **0**. Only non-zero, non-NULL and `BOOLs` that must be **TRUE** need to be initialized. This is best done in `fpInitDoc()`.

- Know where you are in the input source file.

If you are processing headers, footers, notes, or (in the case of `rtfsr`) tables, you must be able to reposition the file pointer as required.

- Check buffer boundaries continuously.

Whenever you advance through the buffer, you need to know whether there is enough of the input stream to completely process the current command. If not, you need to append the next section of the input file before continuing.

- Strive for a "clean" token stream.

Use `filtertest` with the `-d` command-line option to generate a *token* version of the document. If there are redundant tokens, the reader is producing an inefficient token stream. You can keep the token stream free from redundancies by storing the state of the document and then applying the changes only when content is encountered. Content can be text, tabs, or picture objects. The `filtertest.exe` is in the directory `install\samples\utf8\bin`, where `install` is the path name of the Filter installation directory.

- Avoid large `switch()` statements whenever possible. They make both development and debugging more complicated than necessary. If there is a fixed set of commands, consider using a hash table that enables you to quickly identify a pointer to the function that handles that command.
- Filtering document metadata is a separate process.

Remember that `fpGetSummaryInfo()` is a completely separate process from the rest of your code. It creates its own context variable structure. It does not have to call `fpFillBuffer()`.

- Use caution when processing headers, footers, and notes.

If you need to process these items, the structured access layer calls `fpOpenStream()` and `fpCloseStream()`. It is critical that you save the state of your document and the file pointer position prior to returning from `fpOpenStream()`. Prior to returning from `fpCloseStream()`, you must restore the file pointer and the previous state of your document.

- Test your code.

The structured access layer for each SDK is unique. Test your code in Filter SDK, Export SDK, and Viewing SDK.

Functions

This section describes the functions used by custom readers to manage the source file and generate token streams required to convert a document.

xxxxsrAutoDet()

This function analyzes the source document and determines whether the detected file format requires the custom reader. It is called only when the `[CustomFilters]` section of the `formats.ini` file contains an entry identifying the complete file name of the custom reader. For more information on the `formats.ini` file, see [File Format Detection, on page 302](#).

Syntax

```
Bool pascal _export xxxxsrAutoDet(  
    adTPDocInfo *pTPDocInfo,  
    KPTPIOobj *pIO)
```

Arguments

`pTPDocInfo` A pointer to the `adTPDocInfo` structure provided by the structured access layer.

`pIO` A pointer to the I/O stream object for the document processed.

Returns

- TRUE if the file format matches that of the custom reader.
- FALSE if the file format does not match that of the custom reader.

Discussion

- Typically, only the first 1 KB of the file is read into a buffer and analyzed to determine if it matches the file format of the custom reader. If a match is determined, the following four members of the `adTPDocInfo` structure must be assigned before returning TRUE:

<code>adClass</code>	Must be set to 1 .
<code>adFormat</code>	A numerical value assigned to this reader in the <code>[Formats]</code> section of the <code>formats.ini</code> file.
<code>descStr</code>	A string describing the file format.
<code>mMnmemStr</code>	The initial part of the custom reader file name with the "sr" excluded.

- If the return value is TRUE, the custom reader is used to parse the file and generate the token stream.
- If the return value is FALSE, all other readers in the `[CustomFilters]` section of the `formats.ini` file are tried. If no match is found, the file detection process continues checking for the formats supported by Filter SDK.
- The entry in the `[Formats]` section of the `formats.ini` file should be of the form `aaa.bbb.ccc.ddd`, where `aaa` is the value used for the `adFormat` parameter, `bbb` is the value of the file class, `ccc` is the value of the minor format, and `ddd` is the value of the major version.

xxxAllocateContext()

This function allocates a global memory block for a data context. A handle to this memory is returned to the structured access layer. The structured access layer passes this handle back to all reader entry points.

Syntax

```
void * pascal _export xxxAllocateContext(  
    void *pSALContext,  
    LPARAM (pascal *fp)(void *,  
    UINT LPARAM),  
    Bool *pbOpenDoc,  
    TPVAPIServices *pVapi,  
    DWORD dwFlags)
```

Arguments

<code>pSALContext</code>	A pointer to the global data context structure of the structured access layer.
<code>fp</code>	A pointer to a structure of callback functions supported by the structured access layer.
<code>pbOpenDoc</code>	You must set this <code>BOOL</code> value to <code>TRUE</code> if the allocation of memory for the global data context structure is successful.
<code>pVapi</code>	A pointer to a structure providing memory management and character conversion functions. Because this functionality is proprietary to Micro Focus, <code>TPVAPIServices</code> is redefined as <code>void</code> in <code>kvcfsr.h</code> .
<code>dwFlags</code>	Run-time flags controlled by the structured access layer.

Returns

- Upon success, a pointer to the global data context structure for the custom reader. This pointer is passed back to all other custom reader entry points.
- Upon error, a `NULL` pointer. This causes the structured access layer to shut down the process.

Discussion

The global context structure should be `memset()` to zero in this function.

`xxxFreeContext()`

This function terminates an instance of the custom reader.

Syntax

```
int pascal _export xxxFreeContext(void *pCFContext)
```

Arguments

`pCFContext` A pointer to the global context structure for the custom reader.

Returns

- Upon success, `KVERR_Success`.
- Upon error, a non-zero error code.

Discussion

All memory that still remains allocated within the custom reader must be freed within this function.

xxxInitDoc()

This function initializes non-zero, non-null members of `pContext`.

Syntax

```
int pascal _export xxxInitDoc(  
    void          *pCfContext,  
    adDocDesc     *pAutoInfo,  
    long          lcbFileSize,  
    KPTPIOobj     *pIO )
```

Arguments

`pCfContext` A pointer to the global context structure for the custom reader.

`pAutoInfo` A pointer to an `adDocDesc` structure defined in `kwautdef`.

`lcbFileSize` The length of the source file in bytes.

`pIo` A pointer to a `KPTPIOobj` structure defined in `kvioobj.h`.

Returns

- Upon success, `KVERR_Success`.
- Upon error, a non-zero error code. This causes the structured access layer to shut down the process.

Discussion

- For custom readers, the `pAutoInfo` variable can be ignored.
- If the structured access layer has determined the length of the source file, that value is provided by the `lcbFileSize` parameter. If it is zero, the file size must be determined in this function.
- The pointer `pIO` provides access to file management functions defined in `kvioobj.h`.
- In this function, all non-zero, non-NULL members of the global context structure should be initialized.

xxxFillBuffer()

This function controls parsing of the source file and generation of tokens defined in `kvtoken.h`.

Syntax

```
int pascal _export xxxFillBuffer(  
    void *pCFCContext,  
    BYTE *pcBuf,  
    UINT *pnBufOut,  
    int *pnPercentDone,  
    UINT cbBufOutMax)
```

Arguments

pCFCContext	A pointer to the global context structure for the custom reader.
pcBuf	A pointer to a memory buffer to which the tokens are written.
pnBufOut	A pointer to a variable that specifies the actual number of bytes written to the token buffer.
pnPercentDone	A pointer to a variable that specifies the percentage completed of the file parsing.
cbBufOutMax	A pointer to a variable that specifies the maximum number of bytes written to the token buffer.

Returns

- Upon success, `KVERR_Success`.
- Upon error, a non-zero error code. This causes the structured access layer to shut down the process.

Discussion

- Calls are made to read and parse the source file within this function.
- This function is called repeatedly by the structured access layer until either the return value is `FALSE` or the percentage complete is 100.
- The actual number of bytes written to the token buffer must not exceed the value of `cbBufOutMax`.

xxxGetSummaryInfo()

This function is required to extract document summary information.

Syntax

```
int pascal _export xxxGetSummaryInfo(  
    void *pCFCContext,
```

```
KVSummaryInfoEx    *pInfo,  
BOOL               bFreeInfo)
```

Arguments

pCFContext A pointer to the global context structure for the custom reader.

pInfo A pointer to a `KVSummaryInfoEx` structure defined in `kvtypes.h`.

bFreeInfo A `BOOL` value indicating whether to free memory allocated for summary information.

Returns

- Upon success, `KVERR_Success`.
- Upon error, a non-zero error code.

Discussion

This function uses an instance of the global context structure that is different from the one used by all other reader interface functions.

This function can call the same functions used by `xxxFillBuffer()` or can be completely independent.

For more information, see [Extract Metadata, on page 61](#).

xxxOpenStream()

This function is required when initiating processing of peripheral elements such as document headers, footers, footnotes, and endnotes.

Syntax

```
int pascal _export xxxOpenStream(  
    void    *pCFContext,  
    int     type,  
    int     nOrdinal)
```

Arguments

pCFContext A pointer to the global context structure for the custom reader.

type An integer identifying a specific header, footer, footnote, or endnote. Options are defined in `kvcfsr.h`.

nOrdinal An integer identifying a specific header, footer, footnote, or endnote. See the associated macros in `kvtoken.h`.

Returns

- Upon success, `KVERR_Success`.
- Upon error, a non-zero error code.

Discussion

A call to this function results in a call to `xxxFillBuffer()`. The function `xxxFillBuffer()` provides a new empty output buffer and a new token stream input buffer to process the alternate stream for peripheral elements. In this alternate stream, paragraph and character style properties are likely different from the main body. Therefore, as the document is parsed, the existing values from the main body must be saved. When the processing of the alternate stream is completed and processing of the main body resumes, these values must be restored in `xxxCloseStream()`.

xxxCloseStream()

This function is required when terminating processing for document headers, footers, footnotes, and endnotes.

Syntax

```
int pascal _export xxxCloseStream(  
    void *pCFContext,  
    int type)
```

Arguments

`pCFContext` A pointer to the global context structure for the custom reader.

`type` An integer identifying a specific header, footer, footnote, or endnote. Options are defined in `kvcfsr.h`.

Returns

- Upon success, `KVERR_Success`.
- Upon error, a non-zero error code.

Discussion

Prior to exiting this function, the previously saved values in the global context structure must be restored. This ensures that processing of the main body resumes with the correct document state.

xxxCharSet()

This function identifies the character encoding used within the source document.

Syntax

```
KVCharSet pascal _export xxxCharSet(  
    void *pCFContext,  
    BOOL *bMSBLSB)
```

Arguments

`pCFContext` A pointer to the global context structure for the custom reader.

`bMSBLSB` The `BOOL` value required for Unicode text. Set this argument to `TRUE` for Big Endian and `FALSE` for Little Endian.

Returns

One of the enumerated values defined in the `KVCharSet` structure of `kvtypes.h`.

Discussion

If the custom reader can determine the character encoding of the document, the corresponding enumerated value is returned. If the character encoding cannot be determined, `KVCS_UNKNOWN` is returned.

Appendix H: Password Protected Files

This section lists supported password-protected container and non-container files and describes how to open them.

- [Supported Password Protected File Types](#) 339
- [Open Password Protected Container Files](#) 340
- [Filter Password Protected Files](#) 340

Supported Password Protected File Types

The following table lists the password-protected file types that KeyView supports.

Key to support table

Symbol	Description
Y	Format is supported.
N	Format is not supported.
S	Support for viewing subfiles.
V	Support for viewing content.
P	Password required.
C	Password and certificate or User ID file required.

Supported password-protected file types

File Type	Version	Filter	Export	Extract	View	Credentials
PST (Windows)	n/a	N	N	Y	S	P
PST (non-Windows) ¹	n/a	N	N	Y	S	N
ZIP	n/a	N	N	Y	S	P
7-Zip	n/a	N	N	Y	S	P
RAR	n/a	N	N	Y	S	P
SMIME in MSG, EML, MBX	n/a	N	N	Y	N	C

¹The native PST readers, pstxsr and pstnsr, do not require credentials to open password-protected PST files that use compressible encryption.

Supported password-protected file types, continued

File Type	Version	Filter	Export	Extract	View	Credentials
Lotus Notes NSF	n/a	N	N	Y	N	C
Adobe PDF	n/a	Y	Y	Y	V	P
Microsoft Office	97-2003 2007 2010	Y	Y	Y	V	P

Open Password Protected Container Files

This section describes how to extract password-protected container files by using the C API. The following guidelines apply to specific file types.

- **Lotus Notes NSF files.** If you are running a Notes client with an active user connected to a Domino server, you must specify the user's password as a credential regardless of whether the NSF files you are opening are protected. This enables KeyView to access the Notes client and the Lotus Notes API. If the Notes client is not running with an active user, KeyView does not require credentials to access the client.
- **PST files.** To open password-protected PST files that use high encryption (Microsoft Outlook 2003 only), you must use the MAPI-based PST reader (`pstsr`). The native PST readers (`pstxsr` and `pstnsr`) do not support files that use high encryption and return the error message `KVERR_PasswordProtected` if a PST file is encrypted with high encryption.

To open container files

1. Define the credential information in the `KVOpenFileArg` data structure.
2. Pass `KVOpenFileArg` to the `fpOpenFile()` function.
3. Call `fpCloseFile()`.

Filter Password Protected Files

This section describes how to filter password-protected non-container files with the C API.

To filter password-protected files

1. Call the `fpInIt()` or `fpInItWithLicenseData()` function.
2. Call the `fpFilterConfig()` function with the following arguments:

Argument	Parameter
nType	KVFLT_SETSRCPASSWORD
nValue	TRUE
pData	The source file password. The password is a null-terminated string with a maximum length of 255 characters (the final byte is null).

For example:

```
(*fpFilterConfig)(pKVFilter, KVFLT_SETSRCPASSWORD, TRUE, password);
```

where *password* is a null-terminated string of 255 or fewer characters.

3. Call the [fpFilterFile\(\)](#) or [fpFilterStream\(\)](#) function.

Appendix I: Microsoft Rights Management Service Protected Files

This section contains information about KeyView support for Microsoft Rights Management Service (RMS).

- [Microsoft Rights Management Service](#)342
- [Supported Formats](#)342

Microsoft Rights Management Service

The Microsoft Rights Management Service (RMS) allows you to classify and optionally encrypt documents. This service forms the rights management part of Microsoft Azure Information Protection (AIP).

For many of the files that RMS can classify and encrypt, KeyView can identify whether they have been encrypted with RMS encryption. It can also extract metadata (including the RMS classification) and XrML associated with the document.

Supported Formats

KeyView support for RMS files depends on the encryption method that RMS uses for each file type, and on whether the file is classified or protected. In RMS, classified files have additional labels to inform users of their sensitivity, while protected files are encrypted so that only authorized users can view them.

In some cases, KeyView format detection returns a different file type depending on whether the file is classified or protected.

The following sections provide information about the RMS support for different file types, and metadata support.

For more information about XrML extraction, see the `subFileType` member of the [KVSubFileInfo](#), on [page 116](#) structure.

Microsoft Office Files

The following table describes KeyView detected formats for Microsoft Office files that RMS encrypts by creating an OLE container.

For these files:

- KeyView can get classification metadata.
- KeyView can detect whether the file is RMS encrypted (the `kWindowsRMSEncrypted` flag).

In most cases, KeyView can also extract the XrML file for these files when they are protected. However, it does not identify the XrML files as `KVSubFileType_XrML`.

File extensions	Format detected when file is classified but not protected	Format detected when file is protected	XrML extraction
docx, dotx	MS_Word_2007_Fmt	MS_Office_2007_Fmt	Yes
docm, dotm	MS_Word_Macro_2007_Fmt	MS_Office_2007_Fmt	Yes
pptx, potx, ppsx	MS_PPT_2007_Fmt	MS_Office_2007_Fmt	Yes
pptm, potm, ppsm	MS_PPT_Macro_2007_Fmt	MS_Office_2007_Fmt	Yes
vsdx	MS_Visio_2013_Fmt	MS_Office_2007_Fmt	Yes
vsdm, vssm, vssx, vstm, vstx	MS_Visio_2013_Macro_Fmt MS_Visio_2013_Stencil_Fmt MS_Visio_2013_Stencil_Macro_Fmt MS_Visio_2013_Template_Fmt MS_Visio_2013_Template_Macro_Fmt	MS_Office_2007_Fmt	Yes
xlsx, xltx	MS_Excel_2007_Fmt	MS_Office_2007_Fmt	Yes
xlsm, xlsb, xltm	MS_Excel_Macro_2007_Fmt MS_Excel_Binary_2007_Fmt	MS_Office_2007_Fmt	Yes
xps	MS_XPS_Fmt	MS_Office_2007_Fmt	Yes
doc, dot	MS_Word_95_Fmt MS_Word_97_Fmt MS_Word_2000_Fmt	MS_Word_95_Fmt MS_Word_97_Fmt MS_Word_2000_Fmt	Yes
ppt, pot, pps	PowerPoint_95_Fmt PowerPoint_97_Fmt	PowerPoint_95_Fmt PowerPoint_97_Fmt	No
xls, xla, xlam, xlt	Excel_Fmt Excel_Macro_Fmt Excel_95_Fmt Excel_97_Fmt Excel_2000_Fmt	Excel_Fmt Excel_Macro_Fmt Excel_95_Fmt Excel_97_Fmt Excel_2000_Fmt	Yes

Implemented as pFile

The following table describes the KeyView detected formats for files that RMS encrypts by creating a pFile around the document.

For these files:

- KeyView can get classification metadata.
- KeyView can detect whether the file is RMS encrypted (the `kWindowsRMSEncrypted` flag).

- KeyView can extract the XrML if the file is protected.

File extensions	Format detected when file is classified but not protected	Format detected when file is protected	Notes
pfile	n/a	RMS_Protected_Fmt	
vsd	MS_Visio_Fmt	RMS_Protected_Fmt	
vdw, vss, vst	MS_Visio_Fmt	RMS_Protected_Fmt	
mpp, mpt	MS_Project_4_Fmt MS_Project_41_Fmt MS_Project_98_Fmt MS_Project_2000_Fmt MS_Project_2007_Fmt	RMS_Protected_Fmt	
pub	MS_Publisher_98_Fmt	RMS_Protected_Fmt	
jpg	JPEG_File_Interchange_Fmt	RMS_Protected_Fmt	Protected format has extension pjpg. When classified but not protected, the classification metadata is XMP.
png	PNG_Fmt	RMS_Protected_Fmt	Protected format has extension ppng.
tif	TIFF_Fmt	RMS_Protected_Fmt	Protected format has extension ptif. When classified but not protected, the classification metadata is XMP.
dng	TIFF_Fmt	RMS_Protected_Fmt	When classified but not protected, the classification metadata is XMP.
dwx	MS_XPS_Fmt	RMS_Protected_Fmt	

PDF Files

The following table describes the KeyView detected formats for PDF documents, which RMS encrypts by creating an encrypted PDF (in which each stream and metadata value is encrypted), wrapped in a

container PDF. KeyView allows you to extract the encrypted PDF from the container, and then for the extracted file:

- KeyView can detect whether the file is RMS encrypted (the `kWindowsRMSEncrypted` flag).
- KeyView can extract the XrML if the file is protected.

File extensions	Format detected when file is classified but not protected	Format detected when file is protected
pdf	PDF_Fmt PDF_Portfolio_Fmt	PDF_Fmt PDF_Portfolio_Fmt

Send documentation feedback

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Filter SDK C Programming Guide (Micro Focus KeyView 12.4)

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to swpdl.idoldocsfeedback@microfocus.com.

We appreciate your feedback!