Hewlett Packard
Enterprise

KeyView

Software Version: 11.5

Filter SDK C Programming Guide

Document Release Date: October 2017

Software Release Date: October 2017

Legal notices

Warranty

Restricted rights legend

Copyright notice

Trademark notices

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 recent software updates, go to https://downloads.autonomy.com/productDownloads.jsp.

To verify that you are using the most recent edition of a document, go to https://softwaresupport.hpe.com/group/softwaresupport/search-result?doctype=online help.

This site requires that you register for an HPE Passport and sign in. To register for an HPE Passport ID, go to https://hpp12.passport.hpe.com/hppcf/login.do.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Support

Visit the HPE Software Support Online web site at https://softwaresupport.hpe.com.

This web site provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software online support provides customer self-solve capabilities. It provides 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 support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Access product documentation
- Manage support contracts
- Look up HPE support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HPE Passport user and sign in. Many also require a support contract.

To register for an HPE Passport ID, go to https://hpp12.passport.hpe.com/hppcf/login.do.

To find more information about access levels, go to https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

To check for recent software updates, go to https://downloads.autonomy.com/productDownloads.jsp.

Contents

Part I: Overview of Filter SDK

This section provides an overview of the 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

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 300 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
- 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 Vista Business Edition x86 and x64. Other editions of Vista have not been tested, but are likely supported.
- Microsoft Windows 2008 Server Enterprise Edition x86 and x64
- Microsoft Windows 2008 Server R2
- Microsoft Windows 7 x86 and x64
- Microsoft Windows 8 x86 and x64
- Oracle Solaris 10 SPARC
- Oracle Solaris 10 x86 and x64
- Red Hat Enterprise Linux 5.0 x86 and x64
- Red Hat Enterprise Linux 6.0 x86 and x64
- SuSE Linux Enterprise Server 10, 10.1, 11, x86 and 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	Sun C 5.8 Patch 121015-06 2007/10/03

Platform	Architecture	Compiler Name	Compiler Version
		11	
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	xlC_r / cc_r	IBM XL C/C++ Enterprise Edition V8.0
Mac OSX	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 and .NET Components

Component	Compiler
Java components	Java 1.5
.NET components	Microsoft Visual J# 2005 Compiler 8.00.50727.42

Software Dependencies

Some KeyView components require specific third-party software:

- Java Runtime Environment (JRE) or Java Software Developer Kit (JDK) version 1.5 is required for Java API and graphics conversion in Export SDK.
- Outlook 2002 client or later versions is required when processing Microsoft Outlook Personal Folders (PST) files using the MAPI-based reader (`pstsr`). The native PST reader (`pstnsr`) does not require an Outlook client.

> **NOTE:**
> If you are using 32-bit KeyView, you must install 32-bit Outlook. If you are using 64-bit KeyView, you must install 64-bit Outlook.
>
> If the bit editions do not match, 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.
> ```
>
> Additionally, KeyView displays the following return code:
>
> ```
> Error 32: KVError_PSTAccessFailed.
> ```

- 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.

- Microsoft .NET Framework SDK version 2.0, Microsoft .NET Framework version 2.0 Redistributable Package is required if you are programming in a .NET environment.
- 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, `KeyView`*`ProductName`*`SDK_`*`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_11.5_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 19.

   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:

  `adapi.h`
  `adinfo.h`
  `kvcfsr.h`
  `kvxtract.h`
  `kvfilt.h`
  `kvfilt2.h`
  `kvfilter.h`
  `kvioobj.h`
  `kvtoken.h`
  `kvtypes.h`
  `kvxtract.h`
  `kwautdef.h`

- 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 297.

License Information

During installation, the installation program validates the organization name and license key that you enter, and 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
XXXXXXX-XXXXXXX-XXXXXXX-XXXXXXX
```

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) reader (`pstsr` and `pstnsr`)
  - 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 HPE 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=eml` 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
   XXXXXXX-XXXXXXX-XXXXXXX-XXXXXXX
   ```

2. Replace the text *`COMPANY NAME`* with the company name that appears at the top of the License Key Sheet provided by HPE. Enter the text exactly as it appears in the document.
3. Replace the characters *`XXXXXX-XXXXXXX-XXXXXXX-XXXXXXX`* with the appropriate license key from the License Key Sheet provided by HPE. 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
install`\`*OS*`\bin`	Contains the libraries, the format detection file `formats.ini`, the license key file `kv.lic`, and other supporting files.
install`\`*OS*`\lib`	(Solaris installations only) Contains the redistributable `libstlport.so.1` library, which is required to run KeyView on Solaris platforms.
install`\dotnetapi`	Contains the source files for the .NET API.
install`\dotnetapi\dotnethelp`	Contains the help for the .NET API.
install`\dotnetapi\sample`	Contains the sample programs for the .NET API.
install`\cppapi`	Contains the source files for the C++ API.
install`\cppapi\sample`	Contains the sample programs for the C++ API.
install`\guide`	Contains the KeyView Filter SDK programming guides in PDF and HTML format.
install`\include`	Contains the header files required for Filter.
install`\javaapi\javadoc`	Contains the Javadoc for the Java API.
install`\javaapi\sample`	Contains the source files and sample programs for the Java API.
install`\rel_notes`	Contains the *KeyView Filter SDK Release Notes* in PDF format.
install`\samples\filter`	Contains a sample program demonstrating the Filter interface for the C API.

Installed directory structure, continued

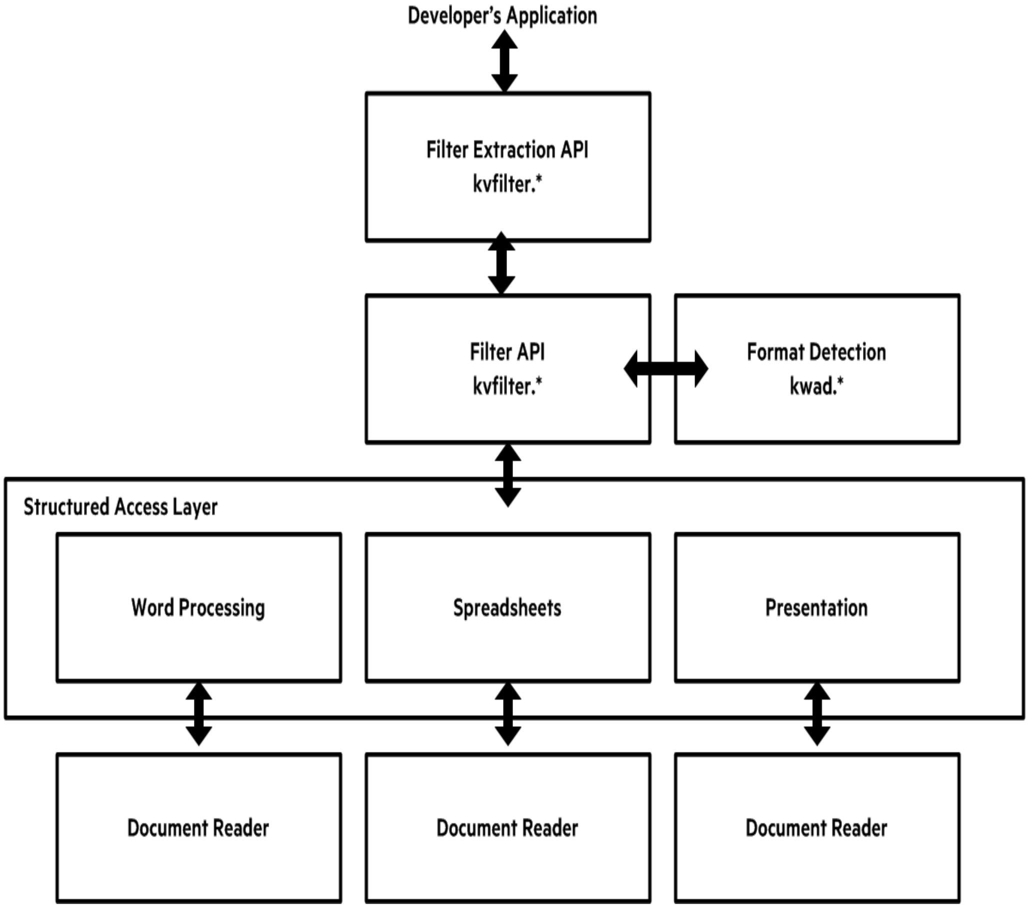
Directory	Description
`install` `\samples\filterca`	Contains a C sample program demonstrating extraction of a content access stream.
`install` `\samples\pdfini`	Contains the initialization file used to extract custom metadata from PDF documents.
`install` `\samples\tstxtract`	Contains a C sample program demonstrating the File Extraction interface.
`install` `\samples\utf8sr`	Contains the source for the sample document reader for UTF-8 files. You can use this to create your own custom document readers.
`install` `\samples\utf8sr\bin`	Contains the C program `filtertest`. You can use this program to test your custom document readers. See Develop a Custom Reader, on page 297.

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

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 36.
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 266 for more information on format detection.
Structured	There are three modules that reside in the structured access layer—one each for word

Architectural Components , continued

Component	Description
Access Layer	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 290 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 143). 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 36.

- 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 266.
- 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 36 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(). 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 `fpInit()` 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 fpInit(). This function’s return value, `pContext`, is passed as the first argument to all other Filter functions.
3. Pass the context pointer from `fpInit()` and the address of a structure that contains pointers to the File Extraction API functions in the call to `KVGetExtractInterface()`. See KVGetExtractInterface(), on page 91.
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.
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 Filtering in Stream Mode, on the previous page through Filtering in Stream Mode, on the previous page 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()`. 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, HPE 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

`fpCanFilterFile()`	`fpCanFilterStream()`
`fpFilterFile()`	`fpFilterStream()`
`fpGetDocInfoFile()`	`fpGetDocInfoStream()`
`fpGetOLESummaryInfo()`	`fpGetOLESummaryInfoFile()`
`fpGetDocInfoFile()`	`fpGetDocInfoStream()`

File Extraction API

`fpCloseFile()`	`fpExtractSubFile()`
`fpFreeStruct()`	`fpGetMainFileInfo()`
`fpGetSubFileInfo()`	`fpGetSubFileMetaData()`
`fpOpenFile()`	`KVGetExtractInterface()`

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 no 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 fpInit() 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 117.
4. Optionally, call a metadata extraction function if a filter function was called in the previous step. See fpGetDocInfoFile(), on page 134 or fpGetDocInfoStream(), on page 135.

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 `fpInit()` 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 `fpInit()` 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. HPE 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:**
> HPE 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 page 34.

> **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, below). 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 `KVOOP_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 `KVOOP` 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 `autnhelper.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 `KVOOP_KEEP_LOGS` environment variable.

On UNIX, set the variable as follows:

```
setenv KVOOP_KEEP_LOGS 1
```

On Windows, set the variable as follows:

```
set KVOOP_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 fpInit() 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

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.

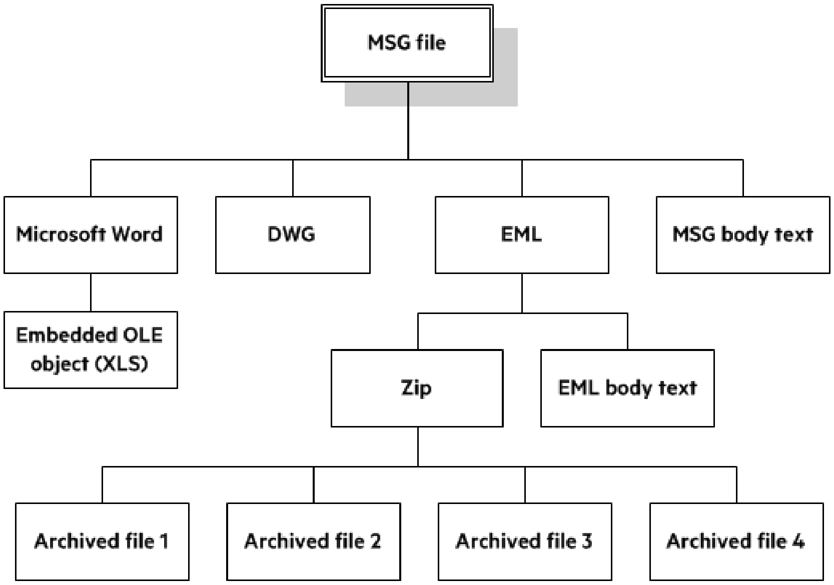
The following are examples of container files:

- 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 184 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.



> **NOTE:** 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()` 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.

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.

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.

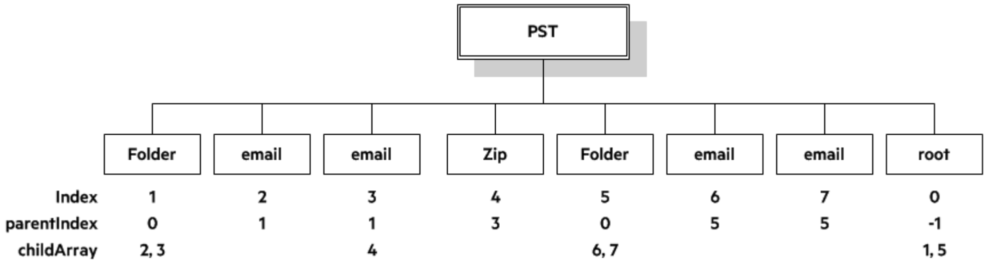
Create a Root Node

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.

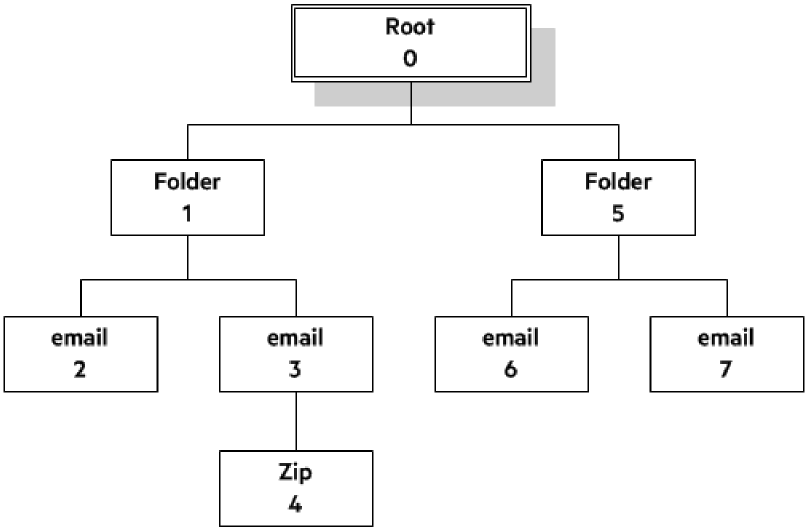
To create the root node, set `openFlag` to **`KVOpenFileFlag_CreateRootNode`** in the call to fpOpenFile(). When you create a root node, the value of `numSubFiles` in KVMainFileInfo includes the root node. For example, when you call `fpGetMainFileInfo()` on a Microsoft Word document with three embedded 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 MSG, EML, MBX, PST, and NSF files, by calling the fpGetSubFileMetaData() function. You can extract a predefined set of metadata fields, individual fields, or both, that are unique to a file format.

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. When you retrieve *all* metadata for a file—that is, pass `NULL` for the array of metadata—the complete set of default metadata, not all available metadata in the file, is returned.

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.

> **NOTE:** 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 44.

Extract the Default Metadata Set

To extract the default metadata set, call the fpGetSubFileMetaData() function, and pass `0` for `metaNameCount` and `NULL` for `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;
```

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.
`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 the previous page), depending on which MAPI properties exist in the MSG file.

MSG-specific Metadata List, continued

Field Name (string to specify)	Description
`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`".

> **NOTE:** 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).

> **NOTE:** 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_I4`
`PT_BINARY`
`PT_BOOLEAN`
`PT_DOUBLE`
`PT_FLOAT`
`PT_LONG`
`PT_SHORT`
`PT_STRING8`
`PT_TSTRING`
`PT_SYSTIME`
`PT_UNICODE`

> **NOTE:** 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;
```

> **NOTE:** 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, above.) 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, above.) 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.

> **NOTE:** 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 (`emlsr`) 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 HPE. See Update License Information, on page 20 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 the previous page.)

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.

> **NOTE:** The Microsoft Outlook Personal Folders (PST) 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 HPE. See Update License Information, on page 20 for information on adding a new license key to an existing installation.

Use the Native or MAPI-based Reader

KeyView accesses PST files in one of two ways:

- indirectly using the Microsoft Messaging Application Programming Interface (MAPI) reader named `pstsr`.
- directly using the native PST reader named `pstnsr`.

On UNIX platforms, the native reader is always used to process PST files because the MAPI-based reader only runs on Windows x86 and x64. On Windows, you can specify either reader; however, the MAPI-based reader is used by default.

The differences between the two readers are summarized in the following table:

Feature/Requirement	Native Reader (pstnsr)	MAPI-based Reader (pstsr)
All platforms supported	Yes	Windows x86 and x64 only
Outlook client required	No	Yes
MAPI properties supported	Yes All properties defined in `mapitags.h`. Object properties are not supported.	Yes All properties defined in `mapitags.h`. Object properties are not supported.
Password protection supported	Yes	Yes (using `KVCredential` structure)
Compressible encryption supported	Yes	Yes
High encryption supported	No	Yes

To use the MAPI-based reader for PST files, change the PST entry in the `formats.ini` file as follows:

```
297=pst
```

To use the native reader for PST files, change the PST entry in the `formats.ini` file as follows:

```
297=pstn
```

> **NOTE:** 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.

Use the Native PST Reader (pstnsr)

The native PST reader accesses PST files directly without relying on the Microsoft interface to the PST format. It runs on both Windows and UNIX, and does not require an Outlook client on the system processing the PST files. However, the native reader does not support password-protected PST files that use high encryption.

Use the MAPI Reader (pstsr)

The `pstsr` reader accesses PST files indirectly by using Microsoft’s Messaging Application Programming Interface (MAPI). MAPI is a standard Windows message interface that enables different mail programs and other mail-aware applications (such as word processors and spreadsheets) to exchange messages and attachments with each other. MAPI allows KeyView to open a PST file, traverse the folders and Outlook items, and extract the items inside the PST file.

> **NOTE:** 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 (`pstsr`) only.

System Requirements

Because MAPI is supported on Windows platforms only, you can filter PST files on Windows only. Because MAPI relies on functionality in Microsoft Outlook, a Microsoft Outlook client must be installed on the same machine as the application filtering PST files, and must be the default email application. KeyView supports the following PST formats and Outlook clients:

- Outlook 97 or higher PST files
- Outlook 2002 or later clients

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

> **NOTE:** The bit edition of Microsoft Outlook must match that of the KeyView software. 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.
>
> If the bit editions do not match, 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.
> ```
>
> Additionally, KeyView displays the following return code:
>
> ```
> Error 32: KVError_PSTAccessFailed.
> ```

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

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`.

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`.

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.

> **NOTE:** To open password-protected PST files that use high encryption, you must use the MAPI-based PST reader (`pstsr`).
> The native PST reader (`pstnsr`) returns the error message `KVERR_PasswordProtected` if a PST 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
  ```

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

  > **NOTE:** 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.

> **NOTE:** 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]
  LNDParser=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.)

> **NOTE:** 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 HPE. See Update License Information, on page 20 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).

HPE 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_LIBARY_PATH
   ```

   where *`keyview_bin`* is the location of the KeyView `bin` directory. HPE 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. HPE 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. HPE 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 253.

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.

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 37). 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 316 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 `_kv`*n*, where *n* is an integer incremented from 0 for each extracted item.
- One of the following extensions:

Type	File Extension
email message	`.mail`
calendar appointment	`.cal`
contact entry	`.cont`
task entry	`.task`
note	`.note`
journal entry	`.jrnl`
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_kv`*n*, 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

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:

```
-KVOOPE 12 # Time: 11:14:32 # File Len = 68140
-KVOOPE 13 # Time: 11:23:05 # H:\files\WP\Word97\fnldmsa.doc
-KVOOPE  5 # Time: 12:15:54 # H:\files\SS\XL2000\corporate.xsl
-KVOOPE  5 # Time: 12:45:19 # H:\files\WP\WPerf5\wp501.doc
-KVOOPE 12 # Time: 14:25:33 # H:\files\PG\PPoint95\95.ppt
-KVOOPE 26 # Time: 16:26:04 # File Len = 19117568
-KVOOPE 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_General`). 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 `kvoop.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 fpInit() or fpOpenStreamEx2() to either **KVF_OOPLOGON** or **KVF_OOPLOGOFF**.

Use Environment Variables

To enable logging, add the `KVOOPLOGON` environment variable, and set the variable value to **1**. To disable logging, do not set the `KVOOPLOGON` 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\kvoop.log` on Windows and `/tmp/kvoop.log` on UNIX.

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

```
[kvooplog]
KvoopLogName=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 `kvtypes.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() 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
`nType`	`KVFLT_SETOOPSRCFILE`
`nValue`	`TRUE`
`pData`	`input_filename`

For example:

```
char      inputfile[250];
(*fpFilterConfig)(pKVFilter, KVFLT_SETOOPSRCFILE, 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 `kvtypes.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 39.

> **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 184 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 86.

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()`.
`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[n].isValid`	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[n].type`	Specifies the data type of the metadata element. The types are defined in the structure KVSumInfoType in `kvtypes.h`.
`si.pElem[n].data`	A pointer to the content of the element. 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[n].pcType`	The name of the metadata field.

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 184 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, HPE 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
Yes	KVCS_936	No	OS code page
Yes	No	UNICODE	UNICODE

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	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(), 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()` 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 223 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(). 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() 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 reading order enables KeyView to output 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 `kvtypes.h` (see LPDF_DIRECTION, on page 182): • **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=lpdf
   ```

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_`	Left-to-right paragraph direction.

Flag	Description
`LTR`	
`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
`total item number`	The total number of metadata tags that are listed.
`metadata_tag_name`	The metadata tag name used in the PDF files.
`datatype`	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() 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() 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_KEEPSOFTHYPHEN`
`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 82 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 82 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 Extract Microsoft Excel Formulas, on the previous page 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()`	`=MDETERM()`	`=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`

Value	Date format	KeyView output
`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, iyou can enable this option programatically 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, below). 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 266 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() 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 86.

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`	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 266 for more information on format ID values. If you are adding configuration settings for a custom XML document type, this option is not defined.
`szRoot`	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. To further qualify the element, specify its namespace. See Specify an Element's Namespace and Attribute, on the next page.
`szInMetaElement`	The elements extracted from the file as metadata. All other elements are extracted as text. 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.

Configuration Option	**Description**
`szExMetaElement`	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. You cannot exclude any metadata elements from the output for StarOffice files. All metadata is extracted regardless of this setting. 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.
`szInContentElement`	The elements extracted from the file as content text. Enter an asterisk (*) to extract all elements including child elements. 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.
`szExContentElement`	The child elements in the included content elements that are not extracted from the file as content text. 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`	The attribute values extracted from the file. If attributes are not defined here, attribute values are not extracted.
	Enter the namespace (if used), element name, and attribute name in the following format:
	`namespace:elementname@attributename`
	For example:
	`keyview:division@name`
	Separate multiple entries with commas.

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:elemname@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 `[configN]`, 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]
TabDelimitedOutput=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.

HPE 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`.

Chapter 5: Sample Programs

This section describes the sample programs provided with Filter SDK.

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 223 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`	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. If this option is not set, only first-level subfiles are extracted. In this case, only the Word file would be extracted.
`-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 filter, above.

`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 31.
`-e`	Run Filter in stream mode. See Filtering in Stream Mode, on page 28.
`-h`	Extract headers and footers, as well as the body text. See fpInit(), on page 143.
`-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.
`-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

Options for the Filter Sample Program , continued

Option	Description
	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 xmlconfigfile`	Filter an XML file by using customized extraction settings defined in the `kvxconfig.ini` 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 tempdirectory`	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 password`	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 fpGetXmpInfoFile(), on page 141 and fpGetXmpInfo(), on page 140.
`-xmpr`	Return XMP metadata as a raw XMP packet. See fpGetXmpInfoFile(), on page 141 and fpGetXmpInfo(), on page 140.
`-embeddedfont`	If you use this option, text that contains embedded fonts is not filtered from PDF documents. See fpFilterConfig(), on page 125.

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()

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()`.
`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 112.

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 Recreate a File's Hierarchy, 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 113.
- 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 93.

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 in an NSF file. See Extract Mail Metadata, on page 39.

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

- When you pass in `0` for `metaArg->metaNameCount`, and `NULL` for `metaArg->metaNameArray`, a set of default metadata is retrieved. See Extract Mail Metadata, on page 39.
- 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_TO;

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()`.
`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()`.
`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 151.

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 `kvxtract.h`.

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 `kvxtract.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 `kvxtract.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 91. When you call the KVGetExtractInterface() function, this structure assigns pointers to the functions. The structure is defined in `kvxtract.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 91.

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 93. The structure is defined in `kvxtract.h`.

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

Member Descriptions

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 161.
`index`	The index number of the subfile to be extracted.
`srcCharset`	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 `KVCharSet` of `kvtypes.h`. See KVExtractSubFileArg, above.
`trgCharset`	If the file type is `KVFileType_Main`, this is the target character set of the extracted file. Otherwise, this is ignored. The character sets are enumerated in `KVCharSet` in `kvtypes.h`. See KVExtractSubFileArg, above.

`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. **NOTE:** 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. **NOTE:** 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.
`filePath`	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 `extractDir` to create the full output path. See KVExtractSubFileArg, on the previous page.
`extractDir`	A pointer to the directory to which subfiles are extracted. This directory must exist. If you set this flag, the path specified in `KVOpenFileArg->extractDir` is ignored. You can use this in conjunction with `filePath` to create the full output

	path.
`stream`	A pointer to an output stream defined by KVOutputStream. See KVExtractSubFileArg, on page 104 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 184 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 full path, `filePath` is the output path, and the path in `extractDir` is ignored.
  - 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 110.
  - 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 `kvxtract.h`.

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

Member Descriptions

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 161.
`index`	The index number of the subfile for which metadata is extracted.
`metaNameCount`	The number of metadata fields to be extracted.
`metaNameArray`	A pointer to the KVMetaName structure that contains an array of metadata tags whose values are retrieved.
`srcCharset`	Specifies the source character set of the metadata when the format's reader cannot determine the character set. The character sets are enumerated in `KVCharSet` of `kvtypes.h`. See KVGetSubFileMetaArg, on the previous page.
`trgCharset`	The target character set of the extracted metadata. The character sets are enumerated in `KVCharSet` in `kvtypes.h`.
`isMSBLSB`	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 184 lists the formats for which the source character set can be determined.
- To retrieve a predefined list of metadata, pass `0` for `metaNameCount` and `NULL` for `metaNameArray`. The metadata in Extract Mail Metadata, on page 39 is extracted.

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 `kvxtract.h`.

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

Member Descriptions

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 161.
`numSubFiles`	The number of subfiles in the main file.
`docInfo`	The file's major format (such as Microsoft Word or Corel Presentation), as defined by the structure `ADDOCINFO`. See ADDOCINFO, on page 156.
`charset`	The character set of the main file.
`isMSBLSB`	This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).
`infoFlag`	A bitwise flag that provides additional information about the main file. The following flag is available: `KVMainFileInfoFlag_HasContent`—The main file contains text that can be filtered. Below are some examples of how this flag is used: For an MSG file without attachments, `numSubFiles` is `1` (message body text), and this flag is `FALSE` because the MSG file itself does not contain text. For a Zip file with three files, `numSubFiles` is `3`, and this flag is `FALSE` 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 117.
- 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 110.

KVMetadataElem

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

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

Member Descriptions

`isDataValid`	Specifies whether the metadata returned from the API is valid data.
`dataID`	The integer name of the extracted metadata field.
`dataType`	The data type of the metadata field. The types are defined in KVMetadataType in `kvtypes.h`.
`strType`	A pointer to the string name of the metadata field.
`data`	The contents of the metadata field. If the `type` member is `KVMetadata_Int4` or `KVMetadata_Bool`, 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 `kvxtract.h`.

```
typedef struct tag_KVMetaName
{
    KVMetaNameType    type;
    union
    {
        void          *pname;
        int            iname;
        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.
`iname`	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 `kvxtract.h`.

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

Member Descriptions

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 161.
`cred`	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.
`stream`	A pointer to the developer-assigned instance of `KVInputStream`. The

	`KVInputStream` structure defines the input stream that contains the source. See KVInputStream, on page 159. If you are using a file as input, this is `NULL`.
`filePath`	A pointer to the full file path to the source file. If you are using a stream as input, this is `NULL`.
`extractDir`	A pointer to the default directory to which subfiles are extracted. This directory must exist. You can use this in conjunction with `KVExtractSubFileArg->filePath` to create the full output path. See KVExtractSubFileArg, on page 104.
`openFlag`	A bitwise flag that defines additional parameters for opening the file. The following flag is available: `KVOpenFileFlag_CreateRootNode`—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

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 161.
`filePath`	The full path to which the subfile was extracted. If the subfile is embedded in the main file as a link, this is the external path to the subfile. If you output the data to a stream, the extraction path is not returned.
`fileName`	The original path, file name, or path and file name of the subfile. If the subfile is embedded in the main file as a link, this is the external path to the subfile.
`infoFlag`	A bitwise flag that provides additional information about the extracted subfile. The following flags are available: • `KVSubFileExtractInfoFlag_NeedsExtraction`—The file might contain subfiles and should be extracted further. • `KVSubFileExtractInfoFlag_FileCreated`—The file was created on disk. • `KVSubFileExtractInfoFlag_CharsetConverted`—The subfile’s character set was converted. • `KVSubFileExtractInfoFlag_External`—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 `filePath` or `fileName`.

	• `KVSubFileExtractInfoFlag_FolderCreated`—A folder was created. • `KVSubFileExtractInfoFlag_NonFormattedBodyExtracted`—Indicates that a plain text version of the message was extracted due to an error extracting the formatted version of the message.
`docInfo`	The file's major format (such as Microsoft Word or Corel Presentation), as defined by the structure `ADDOCINFO`. See ADDOCINFO, on page 156. If you output the data to a stream, the file format is not returned.

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 161.
`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 55.
`subFileType`	The subfile's position in the container file's hierarchy. The following options are available: `KVSubFileType_Main`—The subfile is at the top level of the main file. This is the default subfile type. See Discussion, on page 115. `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 Recreate a File’s Hierarchy, on page 38).
`subFileSize`	The size of the subfile in bytes. This information might be useful if you do not want to extract very large files. This value is approximate and is the maximum size of the subfile. The subfile is usually smaller than this value when it is extracted.
`infoFlag`	A bitwise flag that provides additional information about the subfile. The following flags are available: `KVSubFileInfoFlag_NeedsExtraction`—The subfile might contain subfiles. It must be extracted further to conclusively determine whether it contains subfiles. `KVSubFileInfoFlag_Secure`—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. `KVSubFileInfoFlag_SMIME`—The subfile is S/MIME-encrypted and credentials are required to extract it. This applies to .eml and .pst files only. `KVSubFileInfoFlag_External`—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 `subFileName`. `KVSubFileInfoFlag_MailItem`—When the subfile type is `KVSubFileType_Attachment`, this indicates that the attachment is a mail item. This flag applies to PST, MSG, and NSF files only.
`charset`	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 `KVCS_UNKNOWN`.
`isMSBLSB`	This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).
`fileTime`	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: • EML attachments • OLE objects in a Microsoft Office document • Embedded images
`parentIndex`	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 `parentIndex` is `-1`.
`childCount`	The number of first-level children in the subfile.
`childArray`	A pointer to an array of first-level children in the subfile.

Discussion

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

File format	KVSubFileType_Main 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 99 and fpExtractSubFile(), on page 93.
- 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 `kvxtract.h`.

```
typedef struct  tag_KVSubFileMetaData
{
    KVStructHeader;
    int                 nElem;
    KVMetadataElem**    ppElem;
    unsigned long       infoFlag;
}
KVSubFileMetaDataRec, *KVSubFileMetaData;
```

Member Descriptions

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 161.
`nElem`	The number of metadata fields contained in the array.
`ppElem`	A pointer to an array of pointers that are the memory addresses of metadata field values in the KVMetadataElem structure.

`infoFlag`	A bitwise flag that defines additional properties of the extracted metadata. The following flag is available: `KVSubFileMetaInfoFlag_CharsetConverted`—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()

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 `kvfilt.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, HPE 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 = (KVErrorCode (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 184.

If `KVERR_ General` is returned, you can retrieve the extended error code by using fpGetKvErrorCodeEx (), on page 136.

Syntax

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

Arguments

`pContext`	A pointer returned from `fpInit()`.
`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 170.

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()`.
`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 170.

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

```
BOOL pascal fpFilterConfig(
    void   *pContext,
    int     nType,
    int     nValue,
    void   *pData );
```

Arguments

`pContext`	A pointer returned from `fpInit()`.
`nType`	The configuration flag. This is a symbolic constant defined in `kvtypes.h`. The available options are described in the Filter Configuration Flags 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()` and before the call to `fpFilterStream()` or `fpFilterFile()`.
- 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
`KVFLT_SETOOPSRCFILE`	If you set this flag to **`TRUE`**, the input file name is reported in the out-of-process error log when the file generates an error in stream mode. See Report the File Name in Stream Mode, on page 60. The default is `FALSE`.

Filter Configuration Flags, continued

Flag	Description
	`nValue` is `TRUE` or `FALSE`. `pData` is the name of the input file generating errors.
`KVFLT_SETTEMPDIRECTORY`	This flag enables you to specify the directory where temporary files created during filtering processes are stored. `nValue` is set to **`0`**. `pData` is the path name of the directory where temporary files are stored.
`KVFLT_LOGICALPDF`	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. `nValue` is one of the paragraph direction options defined in the LPDF_DIRECTION enumerated type in `kvtypes.h`. `pData` is `NULL`.
`KVFLT_SETXMLCONFIGINFO`	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. The settings are defined in the KVXConfigInfo structure. To set custom settings for more than one document type, call the `fpFilterConfig()` function once for each type. You can also modify element extraction settings by using the `kvxconfig.ini` file. See Configure Element Extraction for XML Documents, on page 77. `nValue` is set to **`0`**. `pData` is a pointer to the KVXConfigInfo structure.
`KVFLT_INCLREVISIONMARK`	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. 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. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_SETSRCPASSWORD`	This flag enables you to define a password used to open a password-protected file for filtering. See Filter Password Protected Files, on

Filter Configuration Flags, continued

Flag	Description
	page 317. `nValue` is `TRUE`. `pData` is the source file password, which can have a maximum length of 255 characters (the final byte is null).
`KVFLT_NOEMBEDDEDOBJECT`	If you set this flag to **`TRUE`**, embedded objects in Microsoft Word documents are not extracted. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_SHOWHIDDENTEXT`	If you set this flag to **`TRUE`**, hidden text from Microsoft Word, Excel, and PowerPoint documents is extracted. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_NOCOMMENTS`	If you set this flag to **`TRUE`**, comments from Microsoft Word, PowerPoint, or Excel documents are not extracted. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_SKIPEMBEDDEDFONT`	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. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_SHOWDATEFIELDCODE`	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. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_ SHOWFILENAMEFIELDCODE`	If you set this flag to **`TRUE`**, file name field codes are extracted from Microsoft Word documents. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_KEEPSOFTHYPHEN`	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. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.

Filter Configuration Flags, continued

Flag	Description
`KVFLT_EXPORTALLMETADATA`	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. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_EXPORTTAGGEDCONTENT`	If you set this flag to **`TRUE`**, tagged PDF content is filtered from PDF documents. See Filter Tagged PDF Content, on page 70. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_SetConfigurableArguments`	If you set this flag to **`TRUE`**, the `pData` is a variable of configurable arguments. `nValue` is `TRUE` or `FALSE`. `pData` is a variable of configurable arguments.
`KVFLT_SETOUTPUTCHARSET`	This flag enables the output character set to be changed. `pData` is one of the character encodings defined in the `KVCharSet` enumerated type in `kvtypes.h`.
`KVFLT_SHOWHIDDENTEXT`	If you set this flag to **`TRUE`**, hidden text from Microsoft Word, Excel, PowerPoint, and PDF documents is extracted. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_EXTRACTIMAGES`	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. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_TABDELIMITED`	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 84 for more details. `nValue` is `TRUE` or `FALSE`. `pData` is `NULL`.
`KVFLT_STANDARDIZECELLFORMATS`	If you set this flag to **`TRUE`**, standardization of cell formats in Microsoft Excel files is enabled. See Standardize Cell Formats, on page 76. `nValue` is `TRUE` or `FALSE`.

Filter Configuration Flags, continued

Flag	Description
	`pData` is `NULL`.

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 136.

Syntax

```
KVErrorCode pascal fpFilterFile(
    void              *pContext,
    char              *szInputFile,
    char              *szOutputFile,
    KVSummaryInfoEx   *pSummaryInfo );
```

Arguments

`pContext`	A pointer returned from `fpInit()`.
`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 170.

Discussion

This function runs in process or out of process. See The Filter Process Model, on page 29.

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()`.
`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 170.

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-> KVFilterOutput == 0`

- This function runs in process or out of process. See The Filter Process Model, on page 29.

Example

```
error = (int)(*pFilterInterface->fpFilterStream)( pFilter, pStream, &filterOut,
NULL );
```

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()`.
`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()`.
`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 136.

Syntax

```
BOOL pascal fpGetDocInfoFile(
    void       *pContext,
    char       *szFile,
    ADDOCINFO  *pADDocInfo );
```

Arguments

`pContext`	A pointer returned from `fpInit()`.
`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 136.

Discussion

This function runs in process or out of process. See The Filter Process Model, on page 29.

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()`.
`pInput`	A pointer to the input stream.
`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`.

Discussion

This function runs in process or out of process. See The Filter Process Model, on page 29.

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 170.

Syntax

```
KVErrorCodeEx pascal fpGetKvErrorCodeEx ( void *pContext )
```

Arguments

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

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()`.
`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 170.

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 29.

fpGetOLESummaryInfoFile()

This function extracts document metadata from a file.

Syntax

```
KVErrorCode pascal fpGetOLESummaryInfoFile(
    void                *pContext,
    char                *szFile,
    KVSummaryInfoEx     *pSummaryInfo);
```

Arguments

`pContext`	A pointer returned from `fpInit()`.
`szFile`	The name of the input file.
`pSummaryInfo`	A pointer to the KVSummaryInfoEx structure. 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 KVSumInfoElemEx structure.

Returns

The return value is an error code. See KVErrorCode, on page 170.

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 29.

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()`.

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

`pContext`	The pointer returned by fpInit(), on page 143.
`pInput`	A pointer to the input stream.
`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 170.

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 29.
- XMP extraction is supported only for PDF, JPG, TIFF, and XML files.
- XMP extraction is supported on the Windows, Linux, AIX, FreeBSD, and OSX 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 170.

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 29.
- XMP extraction is only supported for the PDF, JPG, TIFF, and XML files.
- XMP extraction is supported for the following platforms:
  - Windows x86 32-bit and 64-bit
  - Linux x86 32-bit and 64-bit
  - Linux x86 32-bit and 64-bit using libc6 library
  - Linux x86 32-bit and 64-bit for Redhat 4
  - Linux Itanium 64-bit
  - AIX Risk 32-bit and 64-bit

    - FreeBSD 32-bit
    - OSX 32-bit Universal

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 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 29.
`KVF_INPROCESS`	Enables in-process filtering. See The Filter Process Model, on page 29.
`KVF_HEADERFOOTERTAGS`	Puts tags around header and footer data.
`KVF_HEADERFOOTER`	Extracts headers and footers.

`KVF_UNICODEMSBLSB`	Uses the byte order for Big Endian systems (MSBLSB) for Unicode text. MSBLSB is the "Most Significant Byte Least Significant Byte."
`KVF_UNICODELSBMSB`	Uses the byte order for Little Endian systems (LSBMSB) for Unicode text. LSBMSB 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` `KVF_DETECT_INPROCESS`	Set these flags in `fpInit()` or `fpOpenStreamEx2()` 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`. 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 34.

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.

fpOpenStream()

This function opens a stream for filtering.

Syntax

```
void * pascal fpOpenStream(
    void             *pContext,
    KVInputStream    *pInput );
```

Arguments

`pContext`	A pointer returned from `fpInit()`.
`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 26 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()`.
`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.

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 26 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()`.

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()`.
`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()`.
`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()`.
`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()`, 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()`.

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`.

KVFltInterfaceEx

The members of this structure are pointers to the functions described in Filter API Functions, on page 117. 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 * );
  KVErrorCode (pascal *fpCanFilterFile)( void *, char * );
  KVErrorCode (pascal *fpCanFilterStream) (void *, void *);
  KVErrorCode (pascal *fpFilterStream)( void *, void *, KVFilterOutput *,
KVSummaryInfoEx * );
  KVErrorCode (pascal *fpFilterFile)( void *, char *, char *, KVSummaryInfoEx * );
  KVErrorCode (pascal *fpGetOLESummaryInfo)( void *, KVInputStream *, KVSummaryInfoEx *
);
  KVErrorCode (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 * );
  KVErrorCode (pascal *fpGetArchiveFileInfo)( void *, int, TPArchiveFileInfo * );
  KVErrorCode (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 *);
  KVErrorCode (pascal *fpCanFilterAsContainer)(void *, KVInputStream *);
  void *      (pascal *fpOpenContainerStream)(void *, KVInputStream *);
  BOOL        (pascal *fpCloseContainerStream)( void *, void *);
  int         (pascal *fpGetNumFilesInContainer)( void *, void *);
  KVErrorCode (pascal *fpGetContainerSubFileInfo)( void *, void *, int,
TPContainerSubFileInfo *);
  BOOL        (pascal *fpSetExtractionPath)(void *, void *, char *, BOOL);
  void        (pascal *fpSetExtractionOverwrite)( void *, void *, BOOL);
  KVErrorCode (pascal *fpExtractContainerSubFile)( void *, void *, int,
TPContainerSubFileInfo *);
  KVErrorCode (pascal *fpGetContainerContent)( void *, void *, KVFilterOutput *,
BOOL * );
  KVErrorCodeEx (pascal *fpGetKvErrorCodeEx)( void *pContext );
  BOOL        (pascal *fpFilterConfig)( void *pContext, int nType, int nValue, void
*p );
/* Revision 2 of Filter Interface API starts here (#define KVFLTINTERFACE_REVISION)
*/
  KVErrorCode (pascal *fpGetSubFileMetadada)( void *, void *, int, int *, int,
KVSummaryInfoEx *, int );
  KVErrorCode (pascal *fpFreeSubFileMetadada)( void *, void *, KVSummaryInfoEx * );
}
KVFltInterfaceEx;
KVErrorCode pascal KV_GetFilterInterfaceEx( KVFltInterfaceEx *pInterfaceEx, int
version );
```

Member Descriptions

The member functions are described in Filter API Functions, on page 117.

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

`eClass`	The file class of the source document (for example, spreadsheet, word processor, or encapsulation format), as defined by the enumerated type `ENDocClass` in `adinfo.h`.
`eFormat`	The major format of the source document (for example Microsoft Word XML format or Corel Presentation), as defined by the enumerated type `ENdocFmt` in `adinfo.h`. The `ENdocFmt` type provides a unique ID for each major format.
`lVersion`	The version number of the file format. The number is multiplied by 1,000 (for example, 1.02 is represented by 1020).
`ulAttributes`	Other attributes of the document, as defined by the enumerated type `ENdocAttributes` in `adinfo.h`.

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 82.

```
typedef struct _KV_CONFIG_ARG_TAG
{
    unsigned int       keyID;
    int                keyType;
    KV_CONFIG_DATA     keyData;
    unsigned int       keyDataSize;
}
KV_CONFIG_Arg;
```

Member Descriptions

`keyID`	Determines the kind of configuration flags that you can use as values of `keyData`. If you use the same `keyID` more than once, the most recent setting overrides the previous setting.
`keyType`	The type of data for the `keyData` element. Set to **`KV_INT32ARG`**.
`keyData`	`KV_CONFIG_DATA` is a union defined in `kvtypes.h`. Only `intArg` is supported, where the value of `intArg` is one of the flags in the corresponding `keyID`.
`keyDataSize`	The size of `keyData`. 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 HPE 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 `kvxtract.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_Bool`, 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 `fpInit()`.
`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`	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 `ENdocFmt`. See File Format Detection, on page 266 for more information on format ID values. If you are adding configuration settings for a custom XML document type, this is not defined.
`pszRoot`	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. To further qualify the element, specify its namespace. See Specify an Element’s Namespace and Attribute, on page 80.
`pszInMeta`	The elements extracted from the file as metadata. All other elements are extracted as text. 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 page 80.
`pszExMeta`	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 `pszExMeta` because it is binary data and should not be extracted. You cannot exclude any metadata elements from the output for StarOffice files. All metadata is extracted regardless of this setting.

	To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element’s Namespace and Attribute, on page 80.
`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 80.
`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 80.
`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: *namespace*`:`*elementname*`@`*attributename* For example: `hpe: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

`encoding`	The type of encoding.
`bIsLittleEndian`	Indicates whether little-endian byte ordering is used.
`nNoOfElements`	The total number of elements.
`pXmpInfoElems`	A pointer to the KVXmpInfoElems structure.
`usXpacketData`	A copy of the XMP data.
`pExtension`	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

The enumerated types are in `adinfo.h`, `kvtypes.h`, `kv.h`, and `kvxtract.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
`KVCS_UNKNOWN`	0
`KVCS_SJIS`	1
`KVCS_GB`	2
`KVCS_BIG5`	3
`KVCS_KSC`	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 103. This enumerated type is defined in `kvxtract.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_`	The credential in `KVCredentialComponent` is a password.

```
 Password
```

KVErrorCode

This enumerated type defines the type of error generated if Filter fails. This enumerated type is defined in `kvtypes.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_ADSNotFound,         /* 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 `fpContinue()` returns `FALSE`.

`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 errror 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 `kvtypes.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`	The PST file could not be converted. This error might be returned when a call to `fpOpenFile()` returns `NULL` for one of the following reasons: • 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. 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.
`KVError_PasswordRequired`	To open the file, you must provide credentials. This error might be returned when a call to `fpOpenFile()` returns `NULL`.
`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 `fpOpenFile()` returns `NULL`. Some document readers are considered advanced features and are licensed separately from the KeyView SDK (for example, the PST and MBX readers). Contact your HPE sales representative to get an updated license key.
`KVError_OopBadConfig`	Information in the `kvxconfig.ini` 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 `fpClose()`, 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 169.

- If an extended error code is called for a format to which the error does not apply, the `KVError_Last` code is returned.

`VectorPictureAnchor`	An anchor for embedded vector graphics.
`RasterPictureAnchor`	An anchor for embedded raster graphics.
`H1Anchor`	An anchor for level 1 heading blocks (H1).
`H2Anchor`	An anchor for level 2 heading blocks (H2).
`H3Anchor`	An anchor for level 3 heading blocks (H3).
`H4Anchor`	An anchor for level 4 heading blocks (H4).
`H5Anchor`	An anchor for level 5 heading blocks (H5).
`H6Anchor`	An anchor for level 6 heading blocks (H6).
`XAnchor`	An anchor for an external file.
`AnimatedGIFAnchor`	An anchor for embedded animated GIF graphics.
`CSSAnchor`	An anchor for an external CSS file.
`GeneralAnchor`	Reserved for future use.
`DBAnchor`	Used internally.
`JPEGAnchor`	An anchor for an embedded JPEG graphic.

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 `isDataValid` 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

`KVMetadata_Unknown`	The value in the property is of an unknown type.
`KVMetadata_Bool`	The value in the property is a Boolean value. The corresponding MAPI type is `PT_BOOLEAN`.
`KVMetadata_Binary`	The value in the property is a byte array. The corresponding MAPI type is `PT_BINARY`.
`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 169.

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 109. This enumerated type is defined in `kvxtract.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_Bool`	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 Draft, Reviewed, or Final.
`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 information about the file formats that can be detected and processed (either filtered, converted, or displayed) by the KeyView suite of products. The KeyView suite includes KeyView Filter SDK, KeyView Export SDK, and KeyView Viewing SDK.

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.

  The formats listed in this section can also be detected by the KeyView format detection module (`kwad`). The Supported Formats (Detected) section lists formats that can be detected, but cannot be filtered, converted, or displayed.

- 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

Key to Support Tables, continued

Symbol	Description
	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
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
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	n/a	emxsr	EMX	N	N	Y	Y	N	n/a	N

---

[1] 7zip 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
Archive										
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
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
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
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

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

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	kpODArdr kpDWGrdr[1]	DWG	Y	Y[2]	Y[3]	N	Y	Y	N

---

[1] On Windows platforms, kpODArdr is used for all versions up to 2007 and graphic rendering is supported; for later versions, only text extraction is supported through the kpDWGrdr or kpDXFrdr reader.

[2] On non-Windows platforms, graphic rendering is supported through the kpDWGrdr reader for versions R13, R14, R15, and R18 (2004); for other versions, only text extraction is supported.

[3] On non-Windows platforms, graphic rendering is supported through the kpDWGrdr reader for versions R13, R14, R15, and R18 (2004); for other versions, only text extraction is supported.

Supported CAD Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
AutoCAD Drawing Exchange	R13, R14, R15/2000, 2004, 2007, 2010, 2013	kpODArdr kpDXFrdr[1]	DXF	Y	Y[2]	Y[3]	N	Y	Y	N
CATIA formats	5	kpCATrdr	CAT[4]	Y	N	N	N	Y	N	N
Microsoft Visio	4, 5, 2000, 2002, 2003, 2007, 2010[5]	vsdsr	VSD	Y	Y	Y	Y[6]	Y	Y	N
		kpVSD2rdr	VSD, VSS VST	Y	Y	Y	N	Y	Y	N
	2013	ActiveX components	VSDM VSSM VSTM VSDX VSSX VSTX	N	N	Y[7]	N	Y	N	N
		kpVSDXrdr	VSDM	Y	Y	Y[4]	Y	Y	Y	N

---

[1] On Windows platforms, kpODArdr is used for all versions up to 2007 and graphic rendering is supported; for later versions, only text extraction is supported through the kpDWGrdr or kpDXFrdr reader.

[2] On non-Windows platforms, graphic rendering is supported through the kpDXFrdr reader for versions R13, R14, R15, and R18 (2004); for other versions, only text extraction is supported.

[3] On Windows platforms, kpODArdr is used for all versions up to 2007 and graphic rendering is supported; for later versions, only text extraction is supported through the kpDWGrdr or kpDXFrdr reader.

[4] All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.

[5] Viewing 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.

[6] Extraction of embedded OLE objects is supported for Filter on Windows platforms only.

[7] Visio 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 CAD Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
			VSSM VSTM VSDX VSSX VSTX							

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
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	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

---

[1] Charset is not supported for Microsoft Access 95 or 97.

Display Formats

Supported Display Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe PDF	1.1 to 1.7	pdfsr	PDF	Y	Y	N	Y[1]	Y	Y	N
		pdf2sr	PDF	N	Y	N	N	N	N	N
		kppdfrdr	PDF	N	Y	Y	N	N	N	N
		kppdf2rdr[2]	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[3]	CGM	Y	Y	Y	N	N	N	N
CorelDRAW[4]	through 9.0 10, 11, 12, X3	kpcdrrdr	CDR	N	Y	Y	N	N	N	N

---

[1] Includes support for extraction of subfiles from PDF Portfolio documents.
[2] kppdf2rdr is an alternate graphic-based reader that produces high-fidelity output but does not support other features such as highlighting or text searching.
[3] Files with non-partitioned data are supported.

[4] CDR/CDR with TIFF header.

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
DCX Fax System	n/a	kpdcxrdr	DCX	N	Y	Y	N	N	N	N
Digital Imaging & Communications in Medicine (DICOM)	n/a	dcmsr	DCM	M	N	N	N	Y	N	N
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	kpgifrdr	GIF	N	Y	Y	N	N	N	N
		gifsr		M	M	N	N	Y	N	N
JBIG2	n/a	kpJBIG2rdr	JBIG2	N	Y	Y	N	N	N	N
JPEG	n/a	kpjpgrdr	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
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

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Office Drawing	n/a	kpmsordr	MSO	N	Y	Y	N	N	N	N
Omni Graffle	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
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
Tagged Image File	through 6.0[1]	tifsr	TIFF	M	M	N	N	Y	N	N
		kptifrdr	TIFF	N	Y	Y	N	N	N	N
Truevision Targa	2	kptrardr	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	Y	Y	N	N	N	N
WordPerfect Graphics 1	1	kpwpgrdr	WPG	N	Y	Y	N	N	N	N

---

[1] The 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.

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
WordPerfect Graphics 2	2, 7	kpwg2rdr	WPG	N	Y	Y	N	N	N	N

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,	mbxsr[3]	MBX	N	N	T	Y	Y	Y	N

---

[1] Supports non-encrypted embedded files only.

[2] KeyView supports MBX files created by Eudora Email and Mozilla Thunderbird. MBX files created by other common mail applications are typically filtered, converted, and displayed.

[3] This 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
	Eudora 6.2									
Microsoft Entourage Database	2004	entsr	various	N	N	Y	Y	Y	Y	N
Microsoft Outlook	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	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,	pffsr[5]	OST	N	N	Y	Y	Y	Y	N

---

[1] This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[2] Returns "Unicode" character set for version 2003 and up, and "Unknown" character set for previous versions.

[3] This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[4] This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[5] The 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
	2013									
Microsoft Outlook Personal Folder	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	pstsr[1][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
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	tnefsr	various	N	N	Y	Y	Y	Y	N

---

[1] This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[2] Uses Microsoft Messaging Application Programming Interface (MAPI). Note that the native PST reader (`pstsr`) works only on Windows, and requires that you have Microsoft Outlook installed. As an alternative, the MAPI reader (`pstnsr`) runs on all platforms, and does not require Microsoft Outlook. For more information on using the native PST reader or the MAPI reader, see the sections 'Use the Native PST Reader (pstnsr) ' and 'Use the MAPI Reader (pstsr)' in Chapter 3.

[3] This reader supports both clear signed and encrypted S/MIME. KeyView supports S/MIME for PST, EML, MBX, and MSG files.

[4] This 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
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
		aiffsr	AIFF	M	N	N	N	Y	N	N
Microsoft Wave Sound	n/a	MCI	WAV	N	N	Y	N	N	N	N
		riffsr	WAV	M	N	N	N	Y	N	N
MIDI	n/a	MCI	MID	N	N	Y	N	N	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
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-4 Audio	n/a	mpeg4sr	MP4 3GP	M	N	N	N	Y	N	N
NeXT/Sun Audio	n/a	MCI	AU	N	N	Y	N	N	N	N
QuickTime Movie	2, 3, 4	MCI	QT MOV	N	N	Y	N	N	N	N

Supported Multimedia Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
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	kpIWPGrdr	GZ	Y	Y	Y	N	Y	Y	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	kpprzrdr	PRZ	Y	Y	Y	N	N	N	N
Lotus Freelance Graphics 2	2	kpprerdr	PRE	Y	Y	Y	N	N	N	N

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 OneNote	2007, 2010, 2013, 2016	kpONErdr	ONE ONETOC2	Y	Y	Y	Y	N	Y	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	kpppxrdr	PPTX PPTM POTX POTM PPSX PPSM PPAM	Y	Y	Y	Y	Y	Y	Y

---

[1] The character set cannot be determined for versions 5.x and lower.

[2] Slide 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	6, 7, 8, 9	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N

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	iwss13sr	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

---

[1] Generated by OpenOffice Impress 2.0, StarOffice 8 Impress, and IBM Lotus Symphony Presentation 3.0.

[2] Supported using the `olesr` embedded objects reader.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
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
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	xlsxsr	XLSX XLTX XLSM XLTM XLAM	Y	Y	Y	Y	Y	Y	Y
Microsoft Excel Binary	2007, 2010,	xlsbsr	XLSB	Y	Y	Y	N	N	N	N

---

[1] Supported using the embedded objects reader `olesr`.

[2] Supported for versions 97 and higher using the embedded objects reader `olesr`.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Format	2013, 2016									
Microsoft Works Spreadsheet	2, 3, 4	mwssr	S30 S40	Y	Y	Y	N	N	Y	N
OASIS Open Document Format	1, 2[1]	odfsssr	ODS SXC STC	Y	Y	Y	Y[2]	Y	Y	N
OpenOffice Calc, LibreOffice Calc	1 to 5	sosr	SXC ODS OTS	Y	T	T	N	Y	Y	N
StarOffice Calc	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

[1] Generated by OpenOffice Calc 2.0, StarOffice 8 Calc, and IBM Lotus Symphony Spreadsheet 3.0.

[2] Supported using the embedded objects reader `olesr`.

Supported Text and Markup Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
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 1.7	rtfsr	RTF	Y	Y	Y	N	P	Y	Y
Unicode HTML	n/a	unihtmsr	HTM	Y	Y	Y	N	Y	Y	N
Unicode Text	3, 4	unisr	TXT	Y	Y	Y	N	N	Y	N
XHTML	1.0	htmsr	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	iwwp13sr	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-paper Basic	3.2.1	cebsr[1]	CEB	Y	N	N	N	N	N	N
Fujitsu Oasys	7	oa2sr	OA2	Y	Y	Y	N	P	N	N

---

[1] This 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
Haansoft Hangul	97	hwpsr	HWP	Y	N	N	N	N	Y	N
	2002, 2005, 2007, 2010	hwposr	HWP	Y	T	T	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 through 2013	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 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 and 2.0	misr	DOC	Y	Y	Y	N	N	N	Y

---

[1] Supported using the embedded objects reader `olesr`.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
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[1]	Y	Y	Y
Microsoft Word Windows XML	2007, 2010, 2013, 2016	mwxsr	DOCM DOCX DOTX DOTM	Y	Y	Y	Y	Y	Y	Y
Microsoft Works	1, 2, 3, 4	mswsr	WPS	Y	Y	Y	N	N	N	Y
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[2]	odfwpsr	ODT SXW STW	Y	Y	Y	Y[3]	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

---

[1] Supported using the embedded objects reader `olesr`.

[2] Generated by OpenOffice Writer 2.0, StarOffice 8 Writer, and IBM Lotus Symphony Documents 3.0.

[3] Supported using the embedded objects reader `olesr`.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Open Publication Structure eBook	2.0, 3.0	epubsr	EPUB	Y	Y	Y	N	Y	Y	N
StarOffice Writer	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	xpssr	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

---

[1] To successfully use this reader, you must set the `KV_YAHOO_ID` environment variable to the Yahoo user ID. You can optionally set the `KV_OTHER_YAHOO_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.

Supported Formats (Detected)

The file formats listed in this section can be detected by the KeyView format detection module (`kwad`), but cannot be filtered, converted, or displayed. The detection module determines a file's format and reports the information to the developer's application.

The formats listed in Supported Formats, on page 184 can be detected as well as filtered, exported, and viewed.

- Ability Office (SS, DB, GR, WP, COM)
- AC3 audio
- ACT
- Adobe FrameMaker
- Adobe FrameMaker Markup Language
- AES Multiplus Comm
- Aldus Freehand (Macintosh)
- Aldus PageMaker (DOS)
- Aldus PageMaker (Macintosh)
- Amiga IFF-8SVX sound
- Amiga MOD sound
- Apple Binary Property List
- Apple Double
- Apple iWork
- Apple Photoshop Document
- Apple Single
- Apple XML Property List
- Appleworks
- Applix Alis
- Applix Asterix
- Applix Graphics
- ARC/PAK Archive
- ASCII-armored PGP encoded
- ASCII-armored PGP Public Keyring
- ASCII-armored PGP signed
- AutoDesk Animator FLIC Animation
- AutoDesk Animator Pro FLIC Animation
- AutoDesk WHIP
- AutoShade Rendering
- B1 Archive
- BlackBerry Activation File

- CADAM Drawing
- CADAM Drawing Overlay
- CCITT Group 3 1-Dimensional (G31D)
- COMET TOP Word
- Confifer Software WavPack
- Convergent Tech DEF Comm.
- Corel Draw CMX
- cpio Archive (UNIX/VAX/SUN)
- CPT Communication
- Creative Voice (VOC) sound
- Curses Screen Image (UNIX/VAX/SUN)
- Data Point VISTAWORD
- DCX Fax
- DEC WPS PLUS
- DECdx
- Desktop Color Separation (DCS)
- Device Independent file (DVI)
- DG CEOwrite
- DG Common Data Stream (CDS)
- DIF Spreadsheet
- Digital Document Interchange Format (DDIF)
- Digital Imaging and Communications in Medicine (DICOM)
- Disk Doubler Compression
- EBCDIC Text
- eFax
- ENABLE
- ENABLE Spreadsheet (SSF)
- Envoy (EVY)
- Executable UNIX/VAX/SUN
- FileMaker (Macintosh)
- FPX format
- Framework
- Framework II
- Freehand 11
- FTP Session Data
- GEM Bit Image
- Ghost Disk Image
- Google SketchUp

- Graphics Environment Manager (GEM VDI)
- Harvard Graphics
- Hewlett Packard
- Honey Bull DSA101
- HP Graphics Language (HP-GL)
- HP Graphics Language (Plotter)
- HP PCL and PJL Languages
- HP Word PC
- IBM 1403 Line Printer
- IBM DCA-FFT
- IBM DCF Script
- Informix SmartWare II
- Informix SmartWare II Communication File
- Informix SmartWare II Database
- Informix SmartWare Spreadsheet
- Interleaf
- Java Class file
- JPEG File Interchange Format (JFIF)
- Keyhole Markup Language
- KW ODA G4 (G4)
- KW ODA G31D (G31)
- KW ODA Internal G32D (G32)
- KW ODA Internal Raw Bitmap (RBM)
- Lasergraphics Language
- Link Library UNIX/VAX/SUN
- Lotus Notes Bitmap
- Lotus Notes CDF
- Lotus Screen Cam
- Lyrix
- Macromedia Director
- MacWrite
- MacWrite II
- MASS-11
- MATLAB MAT Format
- Micrografx Designer
- Microsoft Access 2007
- Microsoft Access 2007 Template
- Microsoft Common Object File Format (COFF)

- Microsoft Compiled HTML Help
- Microsoft Device Independent Bitmap
- Microsoft Document Imaging (MDI)
- Microsoft Excel 2007 Macro-Enabled Spreadsheet Template
- Microsoft Excel 2007 Spreadsheet Template
- Microsoft Exchange Server Database File
- Microsoft Object File Library
- Microsoft Office Drawing
- Microsoft Office Groove
- Microsoft Outlook Restricted Permission Message File
- Microsoft Windows Cursor (CUR) Graphics
- Microsoft Windows Group File
- Microsoft Windows Help File
- Microsoft Windows Icon (ICO)
- Microsoft Windows NT Event Log
- Microsoft Windows OLE 2 Encapsulation
- Microsoft Windows Vista Event Log
- Microsoft Word (UNIX)
- Microsoft Works (Macintosh)
- Microsoft Works Communication (Macintosh)
- Microsoft Works Communication (Windows)
- Microsoft Works Database (Macintosh)
- Microsoft Works Database (PC)
- Microsoft Works Database (Windows)
- Microsoft Works Spreadsheet (Macintosh)
- Microstation
- Milestone Document
- MORE Database Outliner (Macintosh)
- MPEG4 (ISO IEC MPEG4)
- MPEG-PS container with CDXA stream
- MS DOS Batch File format
- MS DOS Device Driver
- MultiMate 4.0
- Multiplan Spreadsheet
- Navy DIF
- NBI Async Archive Format
- NBI Net Archive Format
- Nero Encrypted File

- Netscape Bookmark file
- NeWS font file (SUN)
- NIOS TOP
- Nota Bene
- NURSTOR Drawing
- Object Module UNIX/VAX/SUN
- ODA/ODIF
- ODA/ODIF (FOD 26)
- Office Writer
- OLE DIB object
- OLIDIF
- Open PGP (new format packets)
- OS/2 PM Metafile Graphics
- PaperPort image file
- Paradox (PC) Database
- PC COM executable (detected in file mode only)
- PC Library Module
- PC Object Module
- PC True Type Font
- PCD Image
- PeachCalc Spreadsheet
- Persuasion Presentation
- PEX Binary Archive (SUN)
- PGP Compressed Data
- PGP Encrypted Data
- PGP Public Keyring
- PGP Secret Keyring
- PGP Signature Certificate
- PGP Signed and Encrypted Data
- PGP Signed Data
- Philips Script
- PKCS #12 (p12) Format
- Plan Perfect
- Portable Bitmap Utilities (PBM)
- Portable Greymap Utilities (PGM)
- Portable Pixmap Utilities (PPM)
- PostScript File
- PostScript Type 1 Font File

- PRIMEWORD
- Program Information File
- PTC Creo
- Q & A for DOS
- Q & A for Windows
- Quadratron Q-One (V1.93J)
- Quadratron Q-One (V2.0)
- Quark Xpress (Macintosh)
- QuickDraw 3D Metafile (3DMF)
- Real Audio
- RealLegal E-Transcript
- Reflex Database (R2D)
- RIFF Device Independent Bitmap
- RIFF MIDI
- RIFF Multimedia Movie
- SAMNA Word IV
- Samsung Electronics JungUm Global format
- SEG-Y Seismic Data format
- Serialized Object Format (SOF) Encapsulation
- SGML
- Simple Vector Format (SVF)
- SMTP document
- SolidWorks
- Sony WAVE64 format
- Star Office Calc Spreadsheet (versions 3-5)
- Star Office Impress Presentation (versions 3-5)
- Star Office Math (versions 3-5)
- Star Office Writer Text (versions 3-5)
- StuffIt Archive (Macintosh)
- SUN vfont definition
- SYLK Spreadsheet
- Symphony Spreadsheet
- Targon Word (V 2.0)
- Unigraphics NX
- Uniplex (V6.01)
- UNIX SHAR Encapsulation
- Usenet format
- Volkswriter

- Vorbis OGG format
- VRML
- VRML 2.0
- WANG PC
- Wang WITA
- WANG WPS Comm.
- Web ARChive (WARC)
- Windows C++ Object Storage
- Windows Journal
- Windows Micrografx Draw (DRW)
- Windows Palette
- Windows scrap file (SHS)
- Wireless Markup Language
- Word Connection
- WordMARC word processor
- WordPerfect General File
- WordStar
- WordStar 6.0
- WordStar 2000
- WriteNow
- Writing Assistant word processor
- X Bitmap (XBM)
- X Image
- X Pixmap (XPM)
- Xerox 860 Comm.
- Xerox DocuWorks
- Xerox Writer word processor
- Yahoo! Messenger chat log
- Zipped Keyhole Markup Language

Appendix B: 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

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).

Multibyte and bidirectional support, below 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
Microsoft Cabinet format (CAB)	n/a	n/a	n/a

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
Display			
Adobe Portable Document Format (PDF)	Y	Y[1]	Y
Graphics			
Computer Graphics Metafile (CGM)	Y	N	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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 EMCMF Format	Y	Y	Y

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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[2]

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
OpenOffice Calc (ODS)	Y	Y	N
StarOffice Calc (ODS)	Y	Y	N
Text and Markup			
ANSI (TXT)	Y	Y	Y[2]
ASCII (TXT)	Y	Y	Y[2]
HTML (HTM)	Y	Y	Y[2, 3]
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	Y[2]
XHTML	Y	Y	Y[3]
XML	Y	Y	Y
Word Processing			
Adobe Maker Interchange Format (MIF)	character set 1252 only	N	N
Apple iChat Log (ICHAT)	Y	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

1

Multibyte 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.

2

The text direction in the output file might not be correct.

3

In 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
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

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
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
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

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
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
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

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
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_CNS1	Adobe-CNS1-2 character collection	N
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

1

The 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 C: File Formats and Extensions

This section lists the KeyView file format numbers and their associated file extensions.

File Format and Extension Table

This section lists the KeyView file format codes and the file extensions that they are most commonly associated with.

> **NOTE:** This is not a complete list of file extensions. KeyView returns format codes based on file content, which cannot always be predicted from the file extension. Some file extensions might also be associated with multiple format numbers.

KeyView file formats and extensions

Format Name	Format Number	Format Description	Associated File Extension
AES_Multiplus_Comm_Fmt	1	Multiplus (AES)	PTF
ASCII_Text_Fmt	2	Text	
MSDOS_Batch_File_Fmt	3	MS-DOS Batch File	BAT
Applix_Alis_Fmt	4	APPLIX ASTERIX	AX
BMP_Fmt	5	Windows Bitmap	BMP
CT_DEF_Fmt	6	Convergent Technologies DEF Comm. Format	
Corel_Draw_Fmt	7	Corel Draw	CDR
CGM_ClearText_Fmt	8	Computer Graphics Metafile (CGM)	CGM[1]
CGM_Binary_Fmt	9	Computer Graphics Metafile (CGM)	CGM [1]
CGM_Character_Fmt	10	Computer Graphics Metafile (CGM)	CGM [1]
Word_Connection_Fmt	11	Word Connection	CN
COMET_TOP_Word_Fmt	12	COMET TOP	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
CEOwrite_Fmt	13	CEOwrite	CW
DSA101_Fmt	14	DSA101 (Honeywell Bull)	
DCA_RFT_Fmt	15	DCA-RFT (IBM Revisable Form)	RFT
CDA_DDIF_Fmt	16	CDA / DDIF	
DG_CDS_Fmt	17	DG Common Data Stream (CDS)	CDS
Micrografx_Draw_Fmt	18	Windows Draw (Micrografx)	DRW
Data_Point_VistaWord_Fmt	19	Vistaword	
DECdx_Fmt	20	DECdx	DX
Enable_WP_Fmt	21	Enable Word Processing	WPF
EPSF_Fmt	22	Encapsulated PostScript	EPS [1]
Preview_EPSF_Fmt	23	Encapsulated PostScript	EPS [1]
MS_Executable_Fmt	24	MSDOS/Windows Program	EXE
G31D_Fmt	25	CCITT G3 1D	
GIF_87a_Fmt	26	Graphics Interchange Format (GIF87a)	GIF [1]
GIF_89a_Fmt	27	Graphics Interchange Format (GIF89a)	GIF [1]
HP_Word_PC_Fmt	28	HP Word PC	HW
IBM_1403_LinePrinter_Fmt	29	IBM 1403 Line Printer	I4
IBM_DCF_Script_Fmt	30	DCF Script	IC
IBM_DCA_FFT_Fmt	31	DCA-FFT (IBM Final Form)	IF
Interleaf_Fmt	32	Interleaf	
GEM_Image_Fmt	33	GEM Bit Image	IMG
IBM_Display_Write_Fmt	34	Display Write	IP

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Sun_Raster_Fmt	35	Sun Raster	RAS
Ami_Pro_Fmt	36	Lotus Ami Pro	SAM
Ami_Pro_StyleSheet_Fmt	37	Lotus Ami Pro Style Sheet	
MORE_Fmt	38	MORE Database MAC	
Lyrix_Fmt	39	Lyrix Word Processing	
MASS_11_Fmt	40	MASS-11	M1
MacPaint_Fmt	41	MacPaint	PNTG
MS_Word_Mac_Fmt	42	Microsoft Word for Macintosh	DOC [1]
SmartWare_II_Comm_Fmt	43	SmartWare II	
MS_Word_Win_Fmt	44	Microsoft Word for Windows	DOC [1]
Multimate_Fmt	45	MultiMate	MM [1]
Multimate_Fnote_Fmt	46	MultiMate Footnote File	FNX [1]
Multimate_Adv_Fmt	47	MultiMate Advantage	
Multimate_Adv_Fnote_Fmt	48	MultiMate Advantage Footnote File	
Multimate_Adv_II_Fmt	49	MultiMate Advantage II	MM[1]
Multimate_Adv_II_Fnote_Fmt	50	MultiMate Advantage II Footnote File	FNX [1]
Multiplan_PC_Fmt	51	Multiplan (PC)	
Multiplan_Mac_Fmt	52	Multiplan (Mac)	
MS_RTF_Fmt	53	Rich Text Format (RTF)	RTF
MS_Word_PC_Fmt	54	Microsoft Word for PC	DOC [1]
MS_Word_PC_StyleSheet_Fmt	55	Microsoft Word for PC Style Sheet	DOC [1]
MS_Word_PC_Glossary_Fmt	56	Microsoft Word for PC Glossary	DOC [1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
MS_Word_PC_Driver_Fmt	57	Microsoft Word for PC Driver	DOC [1]
MS_Word_PC_Misc_Fmt	58	Microsoft Word for PC Miscellaneous File	DOC [1]
NBI_Async_Archive_Fmt	59	NBI Async Archive Format	
Navy_DIF_Fmt	60	Navy DIF	ND
NBI_Net_Archive_Fmt	61	NBI Net Archive Format	NN
NIOS_TOP_Fmt	62	NIOS TOP	
FileMaker_Mac_Fmt	63	Filemaker MAC	FP5, FP7
ODA_Q1_11_Fmt	64	ODA / ODIF	OD[1]
ODA_Q1_12_Fmt	65	ODA / ODIF	OD [1]
OLIDIF_Fmt	66	OLIDIF (Olivetti)	
Office_Writer_Fmt	67	Office Writer	OW
PC_Paintbrush_Fmt	68	PC Paintbrush Graphics (PCX)	PCX
CPT_Comm_Fmt	69	CPT	
Lotus_PIC_Fmt	70	Lotus PIC	PIC
Mac_PICT_Fmt	71	QuickDraw Picture	PCT
Philips_Script_Word_Fmt	72	Philips Script	
PostScript_Fmt	73	PostScript	PS
PRIMEWORD_Fmt	74	PRIMEWORD	
Quadratron_Q_One_v1_Fmt	75	Q-One V1.93J	Q1 [1], QX [1]
Quadratron_Q_One_v2_Fmt	76	Q-One V2.0	Q1 [1], QX [1]
SAMNA_Word_IV_Fmt	77	SAMNA Word	SAM

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Ami_Pro_Draw_Fmt	78	Lotus Ami Pro Draw	SDW
SYLK_Spreadsheet_Fmt	79	SYLK	
SmartWare_II_WP_Fmt	80	SmartWare II	
Symphony_Fmt	81	Symphony	WR1
Targa_Fmt	82	Targa	TGA
TIFF_Fmt	83	TIFF	TIF, TIFF
Targon_Word_Fmt	84	Targon Word	TW
Uniplex_Ucalc_Fmt	85	Uniplex Ucalc	SS
Uniplex_WP_Fmt	86	Uniplex	UP
MS_Word_UNIX_Fmt	87	Microsoft Word UNIX	DOC[1]
WANG_PC_Fmt	88	WANG PC	
WordERA_Fmt	89	WordERA	
WANG_WPS_Comm_Fmt	90	WANG WPS	WF
WordPerfect_Mac_Fmt	91	WordPerfect MAC	WPM, WPD[1]
WordPerfect_Fmt	92	WordPerfect	WO, WPD[1]
WordPerfect_VAX_Fmt	93	WordPerfect VAX	WPD[1]
WordPerfect_Macro_Fmt	94	WordPerfect Macro	
WordPerfect_Dictionary_Fmt	95	WordPerfect Spelling Dictionary	
WordPerfect_Thesaurus_Fmt	96	WordPerfect Thesaurus	
WordPerfect_Resource_Fmt	97	WordPerfect Resource File	
WordPerfect_Driver_	98	WordPerfect Driver	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Fmt			
WordPerfect_Cfg_Fmt	99	WordPerfect Configuration File	
WordPerfect_Hyphenation_Fmt	100	WordPerfect Hyphenation Dictionary	
WordPerfect_Misc_Fmt	101	WordPerfect Miscellaneous File	WPD[1]
WordMARC_Fmt	102	WordMARC	WM, PW
Windows_Metafile_Fmt	103	Windows Metafile	WMF[1]
Windows_Metafile_NoHdr_Fmt	104	Windows Metafile (no header)	WMF[1]
SmartWare_II_DB_Fmt	105	SmartWare II	
WordPerfect_Graphics_Fmt	106	WordPerfect Graphics	WPG, QPG
WordStar_Fmt	107	WordStar	WS
WANG_WITA_Fmt	108	WANG WITA	WT
Xerox_860_Comm_Fmt	109	Xerox 860	
Xerox_Writer_Fmt	110	Xerox Writer	
DIF_SpreadSheet_Fmt	111	Data Interchange Format (DIF)	DIF
Enable_Spreadsheet_Fmt	112	Enable Spreadsheet	SSF
SuperCalc_Fmt	113	Supercalc	CAL
UltraCalc_Fmt	114	UltraCalc	
SmartWare_II_SS_Fmt	115	SmartWare II	
SOF_Encapsulation_Fmt	116	Serialized Object Format (SOF)	SOF

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
PowerPoint_Win_Fmt	117	PowerPoint PC	PPT[1]
PowerPoint_Mac_Fmt	118	PowerPoint MAC	PPT[1]
PowerPoint_95_Fmt	119	PowerPoint 95	PPT[1]
PowerPoint_97_Fmt	120	PowerPoint 97	PPT[1]
PageMaker_Mac_Fmt	121	PageMaker for Macintosh	
PageMaker_Win_Fmt	122	PageMaker for Windows	
MS_Works_Mac_WP_Fmt	123	Microsoft Works for MAC	
MS_Works_Mac_DB_Fmt	124	Microsoft Works for MAC	
MS_Works_Mac_SS_Fmt	125	Microsoft Works for MAC	
MS_Works_Mac_Comm_Fmt	126	Microsoft Works for MAC	
MS_Works_DOS_WP_Fmt	127	Microsoft Works for DOS	WPS[1]
MS_Works_DOS_DB_Fmt	128	Microsoft Works for DOS	WDB[1]
MS_Works_DOS_SS_Fmt	129	Microsoft Works for DOS	
MS_Works_Win_WP_Fmt	130	Microsoft Works for Windows	WPS[1]
MS_Works_Win_DB_Fmt	131	Microsoft Works for Windows	WDB[1]
MS_Works_Win_SS_Fmt	132	Microsoft Works for Windows	S30, S40
PC_Library_Fmt	133	DOS/Windows Object Library	
MacWrite_Fmt	134	MacWrite	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
MacWrite_II_Fmt	135	MacWrite II	
Freehand_Fmt	136	Freehand MAC	
Disk_Doubler_Fmt	137	Disk Doubler	
HP_GL_Fmt	138	HP Graphics Language	HPGL
FrameMaker_Fmt	139	FrameMaker	FM, FRM
FrameMaker_Book_Fmt	140	FrameMaker	BOOK
Maker_Markup_Language_Fmt	141	Maker Markup Language	
Maker_Interchange_Fmt	142	Maker Interchange Format (MIF)	MIF
JPEG_File_Interchange_Fmt	143	Interchange Format	JPG, JPEG
Reflex_Fmt	144	Reflex	
Framework_Fmt	145	Framework	
Framework_II_Fmt	146	Framework II	FW3
Paradox_Fmt	147	Paradox	DB
MS_Windows_Write_Fmt	148	Windows Write	WRI
Quattro_Pro_DOS_Fmt	149	Quattro Pro for DOS	
Quattro_Pro_Win_Fmt	150	Quattro Pro for Windows	WB2, WB3
Persuasion_Fmt	151	Persuasion	
Windows_Icon_Fmt	152	Windows Icon Format	ICO
Windows_Cursor_Fmt	153	Windows Cursor	CUR
MS_Project_Activity_Fmt	154	Microsoft Project	MPP[1]
MS_Project_Resource_Fmt	155	Microsoft Project	MPP[1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
MS_Project_Calc_Fmt	156	Microsoft Project	MPP[1]
PKZIP_Fmt	157	ZIP Archive	ZIP
Quark_Xpress_Fmt	158	Quark Xpress MAC	
ARC_PAK_Archive_Fmt	159	PAK/ARC Archive	ARC, PAK
MS_Publisher_Fmt	160	Microsoft Publisher	PUB[1]
PlanPerfect_Fmt	161	PlanPerfect	
WordPerfect_Auxiliary_Fmt	162	WordPerfect auxiliary file	WPW
MS_WAVE_Audio_Fmt	163	Microsoft Wave	WAV
MIDI_Audio_Fmt	164	MIDI	MID, MIDI
AutoCAD_DXF_Binary_Fmt	165	AutoCAD DXF	DXF[1]
AutoCAD_DXF_Text_Fmt	166	AutoCAD DXF	DXF[1]
dBase_Fmt	167	dBase	DBF
OS_2_PM_Metafile_Fmt	168	OS/2 PM Metafile	MET
Lasergraphics_Language_Fmt	169	Lasergraphics Language	
AutoShade_Rendering_Fmt	170	AutoShade Rendering	
GEM_VDI_Fmt	171	GEM VDI	VDI
Windows_Help_Fmt	172	Windows Help File	HLP
Volkswriter_Fmt	173	Volkswriter	VW4
Ability_WP_Fmt	174	Ability	
Ability_DB_Fmt	175	Ability	
Ability_SS_Fmt	176	Ability	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Ability_Comm_Fmt	177	Ability	
Ability_Image_Fmt	178	Ability	
XyWrite_Fmt	179	XYWrite / Nota Bene	XY4
CSV_Fmt	180	CSV (Comma Separated Values)	CSV
IBM_Writing_Assistant_Fmt	181	IBM Writing Assistant	IWA
WordStar_2000_Fmt	182	WordStar 2000	WS2
HP_PCL_Fmt	183	HP Printer Control Language	PCL
UNIX_Exe_PreSysV_VAX_Fmt	184	Unix Executable (PDP-11/pre-System V VAX)	
UNIX_Exe_Basic_16_Fmt	185	Unix Executable (Basic-16)	
UNIX_Exe_x86_Fmt	186	Unix Executable (x86)	
UNIX_Exe_iAPX_286_Fmt	187	Unix Executable (iAPX 286)	
UNIX_Exe_MC68k_Fmt	188	Unix Executable (MC680x0)	
UNIX_Exe_3B20_Fmt	189	Unix Executable (3B20)	
UNIX_Exe_WE32000_Fmt	190	Unix Executable (WE32000)	
UNIX_Exe_VAX_Fmt	191	Unix Executable (VAX)	
UNIX_Exe_Bell_5_Fmt	192	Unix Executable (Bell 5.0)	
UNIX_Obj_VAX_Demand_Fmt	193	Unix Object Module (VAX Demand)	
UNIX_Obj_MS8086_Fmt	194	Unix Object Module (old MS 8086)	
UNIX_Obj_Z8000_Fmt	195	Unix Object Module (Z8000)	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
AU_Audio_Fmt	196	NeXT/Sun Audio Data	AU
NeWS_Font_Fmt	197	NeWS bitmap font	
cpio_Archive_CRChdr_Fmt	198	cpio archive (CRC Header)	
cpio_Archive_CHRhdr_Fmt	199	cpio archive (CHR Header)	
PEX_Binary_Archive_Fmt	200	SUN PEX Binary Archive	
Sun_vfont_Fmt	201	SUN vfont Definition	
Curses_Screen_Fmt	202	Curses Screen Image	
UUEncoded_Fmt	203	UU encoded	UUE
WriteNow_Fmt	204	WriteNow MAC	
PC_Obj_Fmt	205	DOS/Windows Object Module	
Windows_Group_Fmt	206	Windows Group	
TrueType_Font_Fmt	207	TrueType Font	TTF
Windows_PIF_Fmt	208	Program Information File (PIF)	PIF
MS_COM_Executable_Fmt	209	PC (.COM)	COM
StuffIt_Fmt	210	StuffIt (MAC)	HQX
PeachCalc_Fmt	211	PeachCalc	
Wang_GDL_Fmt	212	WANG Office GDL Header	
Q_A_DOS_Fmt	213	Q & A for DOS	
Q_A_Win_Fmt	214	Q & A for Windows	JW
WPS_PLUS_Fmt	215	WPS-PLUS	WPL
DCX_Fmt	216	DCX FAX Format(PCX images	DCX
OLE_Fmt	217	OLE Compound Document	OLE
EBCDIC_Fmt	218	EBCDIC Text	
DCS_Fmt	219	DCS	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
UNIX_SHAR_Fmt	220	SHAR	SHAR
Lotus_Notes_BitMap_Fmt	221	Lotus Notes Bitmap	
Lotus_Notes_CDF_Fmt	222	Lotus Notes CDF	CDF
Compress_Fmt	223	Unix Compress	Z
GZ_Compress_Fmt	224	GZ Compress	GZ[1]
TAR_Fmt	225	TAR	TAR
ODIF_FOD26_Fmt	226	ODA / ODIF	F26
ODIF_FOD36_Fmt	227	ODA / ODIF	F36
ALIS_Fmt	228	ALIS	
Envoy_Fmt	229	Envoy	EVY
PDF_Fmt	230	Portable Document Format	PDF
BinHex_Fmt	231	BinHex	HQX
SMTP_Fmt	232	SMTP	SMTP
MIME_Fmt	233	MIME[2]	EML, MBX
USENET_Fmt	234	USENET	
SGML_Fmt	235	SGML	SGML
HTML_Fmt	236	HTML	HTM[1], HTML [1]
ACT_Fmt	237	ACT	ACT
PNG_Fmt	238	Portable Network Graphics (PNG)	PNG
MS_Video_Fmt	239	Video for Windows (AVI)	AVI
Windows_Animated_Cursor_Fmt	240	Windows Animated Cursor	ANI
Windows_CPP_Obj_Storage_Fmt	241	Windows C++ Object Storage	
Windows_Palette_Fmt	242	Windows Palette	PAL
RIFF_DIB_Fmt	243	RIFF Device Independent Bitmap	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
RIFF_MIDI_Fmt	244	RIFF MIDI	RMI
RIFF_Multimedia_Movie_Fmt	245	RIFF Multimedia Movie	
MPEG_Fmt	246	MPEG Movie	MPG, MPEG[1]
QuickTime_Fmt	247	QuickTime Movie, MPEG-4 Audio	MOV, QT, MP4
AIFF_Fmt	248	Audio Interchange File Format (AIFF)	AIF, AIFF
Amiga_MOD_Fmt	249	Amiga MOD	MOD
Amiga_IFF_8SVX_Fmt	250	Amiga IFF (8SVX) Sound	IFF
Creative_Voice_Audio_Fmt	251	Creative Voice (VOC)	VOC
AutoDesk_Animator_FLI_Fmt	252	AutoDesk Animator FLIC	FLI
AutoDesk_AnimatorPro_FLC_Fmt	253	AutoDesk Animator Pro FLIC	FLC
Compactor_Archive_Fmt	254	Compactor / Compact Pro	
VRML_Fmt	255	VRML	WRL
QuickDraw_3D_Metafile_Fmt	256	QuickDraw 3D Metafile	
PGP_Secret_Keyring_Fmt	257	PGP Secret Keyring	
PGP_Public_Keyring_Fmt	258	PGP Public Keyring	
PGP_Encrypted_Data_Fmt	259	PGP Encrypted Data	
PGP_Signed_Data_Fmt	260	PGP Signed Data	
PGP_SignedEncrypted_Data_Fmt	261	PGP Signed and Encrypted Data	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
PGP_Sign_Certificate_Fmt	262	PGP Signature Certificate	
PGP_Compressed_Data_Fmt	263	PGP Compressed Data	
PGP_ASCII_Public_Keyring_Fmt	264	ASCII-armored PGP Public Keyring	
PGP_ASCII_Encoded_Fmt	265	ASCII-armored PGP encoded	PGP[1]
PGP_ASCII_Signed_Fmt	266	ASCII-armored PGP encoded	PGP[1]
OLE_DIB_Fmt	267	OLE DIB object	
SGI_Image_Fmt	268	SGI Image	RGB
Lotus_ScreenCam_Fmt	269	Lotus ScreenCam	
MPEG_Audio_Fmt	270	MPEG Audio	MPEGA
FTP_Software_Session_Fmt	271	FTP Session Data	STE
Netscape_Bookmark_File_Fmt	272	Netscape Bookmark File	HTM[1]
Corel_Draw_CMX_Fmt	273	Corel CMX	CMX
AutoDesk_DWG_Fmt	274	AutoDesk Drawing (DWG)	DWG
AutoDesk_WHIP_Fmt	275	AutoDesk WHIP	WHP
Macromedia_Director_Fmt	276	Macromedia Director	DCR
Real_Audio_Fmt	277	Real Audio	RM
MSDOS_Device_Driver_Fmt	278	MSDOS Device Driver	SYS
Micrografx_Designer_Fmt	279	Micrografx Designer	DSF

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
SVF_Fmt	280	Simple Vector Format (SVF)	SVF
Applix_Words_Fmt	281	Applix Words	AW
Applix_Graphics_Fmt	282	Applix Graphics	AG
MS_Access_Fmt	283	Microsoft Access	MDB[1]
MS_Access_95_Fmt	284	Microsoft Access 95	MDB[1]
MS_Access_97_Fmt	285	Microsoft Access 97	MDB[1]
MacBinary_Fmt	286	MacBinary	BIN
Apple_Single_Fmt	287	Apple Single	
Apple_Double_Fmt	288	Apple Double	
Enhanced_Metafile_Fmt	289	Enhanced Metafile	EMF
MS_Office_Drawing_Fmt	290	Microsoft Office Drawing	
XML_Fmt	291	XML	XML[1]
DeVice_Independent_Fmt	292	DeVice Independent file (DVI)	DVI
Unicode_Fmt	293	Unicode	UNI
Lotus_123_Worksheet_Fmt	294	Lotus 1-2-3	WK1[1]
Lotus_123_Format_Fmt	295	Lotus 1-2-3 Formatting	FM3
Lotus_123_97_Fmt	296	Lotus 1-2-3 97	WK1[1]
Lotus_Word_Pro_96_Fmt	297	Lotus Word Pro 96	LWP[1]
Lotus_Word_Pro_97_Fmt	298	Lotus Word Pro 97	LWP[1]
Freelance_DOS_Fmt	299	Lotus Freelance for DOS	
Freelance_Win_Fmt	300	Lotus Freelance for Windows	PRE

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Freelance_OS2_Fmt	301	Lotus Freelance for OS/2	PRS
Freelance_96_Fmt	302	Lotus Freelance 96	PRZ[1]
Freelance_97_Fmt	303	Lotus Freelance 97	PRZ[1]
MS_Word_95_Fmt	304	Microsoft Word 95	DOC[1]
MS_Word_97_Fmt	305	Microsoft Word 97	DOC[1]
Excel_Fmt	306	Microsoft Excel	XLS[1]
Excel_Chart_Fmt	307	Microsoft Excel	XLS[1]
Excel_Macro_Fmt	308	Microsoft Excel	XLS[1]
Excel_95_Fmt	309	Microsoft Excel 95	XLS[1]
Excel_97_Fmt	310	Microsoft Excel 97	XLS[1]
Corel_ Presentations_Fmt	311	Corel Presentations	XFD, XFDL
Harvard_Graphics_ Fmt	312	Harvard Graphics	
Harvard_Graphics_ Chart_Fmt	313	Harvard Graphics Chart	CH3, CHT
Harvard_Graphics_ Symbol_Fmt	314	Harvard Graphics Symbol File	SY3
Harvard_Graphics_ Cfg_Fmt	315	Harvard Graphics Configuration File	
Harvard_Graphics_ Palette_Fmt	316	Harvard Graphics Palette	
Lotus_123_R9_Fmt	317	Lotus 1-2-3 Release 9	
Applix_ Spreadsheets_Fmt	318	Applix Spreadsheets	AS
MS_Pocket_Word_ Fmt	319	Microsoft Pocket Word	PWD, DOC[1]
MS_DIB_Fmt	320	MS Windows Device Independent Bitmap	
MS_Word_2000_Fmt	321	Microsoft Word 2000	DOC[1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Excel_2000_Fmt	322	Microsoft Excel 2000	XLS[1]
PowerPoint_2000_Fmt	323	Microsoft PowerPoint 2000	PPT
MS_Access_2000_Fmt	324	Microsoft Access 2000	MDB[1], MPP[1]
MS_Project_4_Fmt	325	Microsoft Project 4	MPP[1]
MS_Project_41_Fmt	326	Microsoft Project 4.1	MPP[1]
MS_Project_98_Fmt	327	Microsoft Project 98	MPP[1]
Folio_Flat_Fmt	328	Folio Flat File	FFF
HWP_Fmt	329	HWP(Arae-Ah Hangul)	HWP
ICHITARO_Fmt	330	ICHITARO V4-10	
IS_XML_Fmt	331	Extended or Custom XML	XML[1]
Oasys_Fmt	332	Oasys format	OA2, OA3
PBM_ASC_Fmt	333	Portable Bitmap Utilities ASCII Format	
PBM_BIN_Fmt	334	Portable Bitmap Utilities Binary Format	
PGM_ASC_Fmt	335	Portable Greymap Utilities ASCII Format	
PGM_BIN_Fmt	336	Portable Greymap Utilities Binary Format	PGM
PPM_ASC_Fmt	337	Portable Pixmap Utilities ASCII Format	
PPM_BIN_Fmt	338	Portable Pixmap Utilities Binary Format	
XBM_Fmt	339	X Bitmap Format	XBM
XPM_Fmt	340	X Pixmap Format	XPM
FPX_Fmt	341	FPX Format	FPX
PCD_Fmt	342	PCD Format	PCD

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
MS_Visio_Fmt	343	Microsoft Visio	VSD
MS_Project_2000_Fmt	344	Microsoft Project 2000	MPP[1]
MS_Outlook_Fmt	345	Microsoft Outlook	MSG, OFT
ELF_Relocatable_Fmt	346	ELF Relocatable	O
ELF_Executable_Fmt	347	ELF Executable	
ELF_Dynamic_Lib_Fmt	348	ELF Dynamic Library	SO
MS_Word_XML_Fmt	349	Microsoft Word 2003 XML	XML[1]
MS_Excel_XML_Fmt	350	Microsoft Excel 2003 XML	XML[1]
MS_Visio_XML_Fmt	351	Microsoft Visio 2003 XML	VDX
SO_Text_XML_Fmt	352	StarOffice Text XML	SXW[1], ODT[1]
SO_Spreadsheet_XML_Fmt	353	StarOffice Spreadsheet XML	SXC[1], ODS[1]
SO_Presentation_XML_Fmt	354	StarOffice Presentation XML	SXI[1], SXP[1], ODP[1]
XHTML_Fmt	355	XHTML	XML[1]
MS_OutlookPST_Fmt	356	Microsoft Outlook PST	PST
RAR_Fmt	357	RAR	RAR
Lotus_Notes_NSF_Fmt	358	IBM Lotus Notes Database NSF/NTF	NSF
Macromedia_Flash_Fmt	359	SWF	SWF
MS_Word_2007_Fmt	360	Microsoft Word 2007 XML	DOCX, DOTX
MS_Excel_2007_Fmt	361	Microsoft Excel 2007 XML	XLSX, XLTX
MS_PPT_2007_Fmt	362	Microsoft PPT 2007 XML	PPTX, POTX, PPSX
OpenPGP_Fmt	363	OpenPGP Message Format (with new	PGP

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
		packet format)	
Intergraph_V7_ DGN_Fmt	364	Intergraph Standard File Format (ISFF) V7 DGN (non-OLE)	DGN[1]
MicroStation_V8_ DGN_Fmt	365	MicroStation V8 DGN (OLE)	DGN[1]
MS_Word_Macro_ 2007_Fmt	366	Microsoft Word Macro 2007 XML	DOCM, DOTM
MS_Excel_Macro_ 2007_Fmt	367	Microsoft Excel Macro 2007 XML	XLSM, XLTM, XLAM
MS_PPT_Macro_ 2007_Fmt	368	Microsoft PPT Macro 2007 XML	PPTM, POTM, PPSM, PPAM
LZH_Fmt	369	LHA Archive	LZH, LHA
Office_2007_Fmt	370	Office 2007 document	XLSB
MS_XPS_Fmt	371	Microsoft XML Paper Specification (XPS)	XPS
Lotus_Domino_DXL_ Fmt	372	IBM Lotus representation of Domino design elements in XML format	DXL
ODF_Text_Fmt	373	ODF Text	ODT[1], SXW[1], STW
ODF_Spreadsheet_ Fmt	374	ODF Spreadsheet	ODS[1], SXC[1], STC
ODF_Presentation_ Fmt	375	ODF Presentation	SXD[1], SXI[1], ODG[1], , ODP[1]
Legato_Extender_ ONM_Fmt	376	Legato Extender Native Message ONM	ONM
bin_Unknown_Fmt	377	n/a	
TNEF_Fmt	378	Transport Neutral Encapsulation Format (TNEF)	various
CADAM_Drawing_ Fmt	379	CADAM Drawing	CDD
CADAM_Drawing_ Overlay_Fmt	380	CADAM Drawing Overlay	CDO
NURSTOR_	381	NURSTOR Drawing	NUR

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
Drawing_Fmt			
HP_GLP_Fmt	382	HP Graphics Language (Plotter)	HPG
ASF_Fmt	383	Advanced Systems Format (ASF)	ASF
WMA_Fmt	384	Window Media Audio Format (WMA)	WMA
WMV_Fmt	385	Window Media Video Format (WMV)	WMV
EMX_Fmt	386	Legato EMailXtender Archives Format (EMX)	EMX
Z7Z_Fmt	387	7 Zip Format(7z)	7Z
MS_Excel_Binary_2007_Fmt	388	Microsoft Excel Binary 2007	XLSB
CAB_Fmt	389	Microsoft Cabinet File (CAB)	CAB
CATIA_Fmt	390	CATIA Formats (CAT*)	CAT[3]
YIM_Fmt	391	Yahoo Instant Messenger History	DAT[1]
ODF_Drawing_Fmt	392	ODF Drawing	SXD[1], SX[1], ODG[1]
Founder_CEB_Fmt	393	Founder Chinese E-paper Basic (ceb)	CEB
QPW_Fmt	394	Quattro Pro 9+ for Windows	QPW
MHT_Fmt	395	MHT format[2]	MHT
MDI_Fmt	396	Microsoft Document Imaging Format	MDI
GRV_Fmt	397	Microsoft Office Groove Format	GRV
IWWP_Fmt	398	Apple iWork Pages format	PAGES, GZ[1]
IWSS_Fmt	399	Apple iWork Numbers format	NUMBERS, GZ[1]
IWPG_Fmt	400	Apple iWork Keynote format	KEY, GZ[1]
BKF_Fmt	401	Windows Backup File	BKF
MS_Access_2007_Fmt	402	Microsoft Access 2007	ACCDB
ENT_Fmt	403	Microsoft Entourage Database Format	
DMG_Fmt	404	Mac Disk Copy Disk Image File	

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
CWK_Fmt	405	AppleWorks File	
OO3_Fmt	406	Omni Outliner File	OO3
OPML_Fmt	407	Omni Outliner File	OPML
Omni_Graffle_XML_File	408	Omni Graffle XML File	GRAFFLE
PSD_Fmt	409	Photoshop Document	PSD
Apple_Binary_PList_Fmt	410	Apple Binary Property List format	
Apple_iChat_Fmt	411	Apple iChat format	
OOUTLINE_Fmt	412	OOutliner File	OOUTLINE
BZIP2_Fmt	413	Bzip 2 Compressed File	BZ2
ISO_Fmt	414	ISO-9660 CD Disc Image Format	ISO
DocuWorks_Fmt	415	DocuWorks Format	XDW
RealMedia_Fmt	416	RealMedia Streaming Media	RM, RA
AC3Audio_Fmt	417	AC3 Audio File Format	AC3
NEF_Fmt	418	Nero Encrypted File	NEF
SolidWorks_Fmt	419	SolidWorks Format Files	SLDASM, SLDPRT, SLDDRW
XFDL_Fmt	420	Extensible Forms Description Language	XFDL, XFD
Apple_XML_PList_Fmt	421	Apple XML Property List format	
OneNote_Fmt	422	OneNote Note Format	ONE
Dicom_Fmt	424	Digital Imaging and Communications in Medicine	DCM
EnCase_Fmt	425	Expert Witness Compression Format (EnCase)	E01, L01, Lx01
Scrap_Fmt	426	Shell Scrap Object File	SHS
MS_Project_2007_Fmt	427	Microsoft Project 2007	MPP[1]

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
MS_Publisher_98_Fmt	428	Microsoft Publisher 98/2000/2002/2003/2007/	PUB[1]
Skype_Fmt	429	Skype Log File	DBB
Hl7_Fmt	430	Health level7 message	HL7
MS_OutlookOST_Fmt	431	Microsoft Outlook OST	OST
Epub_Fmt	432	Electronic Publication	EPUB
MS_OEDBX_Fmt	433	Microsoft Outlook Express DBX	DBX
BB_Activ_Fmt	434	BlackBerry Activation File	DAT[1]
DiskImage_Fmt	435	Disk Image	
Milestone_Fmt	436	Milestone Document	MLS, ML3, ML4, ML5, ML6, ML7, ML8, ML9
E_Transcript_Fmt	437	RealLegal E-Transcript File	PTX
PostScript_Font_Fmt	438	PostScript Type 1 Font	PFB
Ghost_DiskImage_Fmt	439	Ghost Disk Image File	GHO, GHS
JPEG_2000_JP2_File_Fmt	440	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	JP2, JPF, J2K, JPWL, JPX, PGX
Unicode_HTML_Fmt	441	Unicode HTML	HTM[1], HTML[1]
CHM_Fmt	442	Microsoft Compiled HTML Help	CHM
EMCMF_Fmt	443	Documentum EMCMF format	EMCMF
MS_Access_2007_Tmpl_Fmt	444	Microsoft Access 2007 Template	ACCDT
Jungum_Fmt	445	Samsung Electronics Jungum Global document	GUL
JBIG2_Fmt	446	JBIG2 File Format	JB2, JBIG2
EFax_Fmt	447	eFax file	EFX
AD1_Fmt	448	AD1 Evidence file	AD1
SketchUp_Fmt	449	Google SketchUp	SKP

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
GWFS_Email_Fmt	450	Group Wise File Surf email	GWFS
JNT_Fmt	451	Windows Journal format	JNT
Yahoo_yChat_Fmt	452	Yahoo! Messenger chat log	YCHAT
PaperPort_MAX_File_Fmt	453	PaperPort image file	MAX
ARJ_Fmt	454	ARJ (Archive by Robert Jung) file format	ARJ
RPMSG_Fmt	455	Microsoft Outlook Restricted Permission Message	RPMSG
MAT_Fmt	456	MATLAB file format	MAT, FIG
SGY_Fmt	457	SEG-Y Seismic Data format	SGY, SEGY
CDXA_MPEG_PS_Fmt	458	MPEG-PS container with CDXA stream	MPG[1]
EVT_Fmt	459	Microsoft Windows NT Event Log	EVT
EVTX_Fmt	460	Microsoft Windows Vista Event Log	EVTX
MS_OutlookOLM_Fmt	461	Microsoft Outlook for Macintosh format	OLM
WARC_Fmt	462	Web ARChive	WARC
JAVACLASS_Fmt	463	Java Class format	CLASS
VCF_Fmt	464	Microsoft Outlook vCard file format	VCF
EDB_Fmt	465	Microsoft Exchange Server Database file format	EDB
ICS_Fmt	466	Microsoft Outlook iCalendar file format	ICS, VCS
MS_Visio_2013_Fmt	467	Microsoft Visio 2013	VSDX, VSTX, VSSX
MS_Visio_2013_Macro_Fmt	468	Microsoft Visio 2013 macro	VSDM, VSTM, VSSM
ICHITARO_Compr_Fmt	469	ICHITARO Compressed format	JTDC
IWWP13_Fmt	470	Apple iWork 2013 Pages format	IWA

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
IWSS13_Fmt	471	Apple iWork 2013 Numbers format	IWA
IWPG13_Fmt	472	Apple iWork 2013 Keynote format	IWA
XZ_Fmt	473	XZ archive format	XZ
Sony_WAVE64_Fmt	474	Sony Wave64 format	W64
Conifer_WAVPACK_Fmt	475	Conifer Wavpack format	WV
Xiph_OGG_VORBIS_Fmt	476	Xiph Ogg Vorbis format	OGG
MS_Visio_2013_Stencil_Fmt	477	MS Visio 2013 stencil format	VSSX
MS_Visio_2013_Stencil_Macro_Fmt	478	MS Visio 2013 stencil Macro format	VSSM
MS_Visio_2013_Template_Fmt	479	MS Visio 2013 template format	VSTX
MS_Visio_2013_Template_Macro_Fmt	480	MS Visio 2013 template Macro format	VSTM
Borland_Reflex_2_Fmt	481	Borland Reflex 2 format	R2D
PKCS_12_Fmt	482	PKCS #12 (p12) format	P12, PFX
B1_Fmt	483	B1 format	B1
ISO_IEC_MPEG_4_Fmt	484	ISO/IEC MPEG-4 format	MP4
RAR5_Fmt	485	RAR5 Format	RAR5
Unigraphics_NX_Fmt	486	Unigraphics (UG) NX CAD Format	PRT
PTC_Creo_Fmt	487	PTC Creo CAD Format	ASM, PRT
KML_Fmt	488	Keyhole Markup Language	KML
KMZ_Fmt	489	Zipped Keyhole Markup Language	KMZ
WML_Fmt	490	Wireless Markup Language	WML

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
SO_Text_Fmt	492	Star Office Writer Text	SDW, SGL, VOR
SO_Spreadsheet_Fmt	493	Star Office Calc Spreadsheet	SDC
SO_Presentation_Fmt	494	Star Office Impress Presentation	SDD, SDA
SO_Math_Fmt	495	Star Office Math	SMF

1

This file extension can return more than one format number.

2

MHT, 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.

3

All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.

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

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 255.
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*>`	If the condition from the comparison is true, process the XML. Conditions can be nested up to 25 levels deep. **Attributes** • `name`. (Required) The name of the main item to compare to `item` or `text`. • `item`. (Required if no `text`) The name of the item to compare to the item specified by `name`. • `text`. (Required if no `item`) The text to compare to the item specified by `name`.
`<ifex>, <ifnx>`	If `name` item exists and has a `text` value or not. The Notes item might have a value that cannot be converted to text, such as an image.

`<ifeq>, <ifne>,`
`<iflt>, <ifle>,`
`<ifgt>, <ifge>`
	Respectively, if `text` `==`, `!=`, `<`, `>`, `<=`, `>`, `>=`.
	Text comparison uses a case-insensitive string compare.

`<iftdeq>, <iftdne>,`
`<iftdlt>, <iftdle>,`
`<iftdgt>, <iftdge>`
	Respectively, if time/date `==`, `!=`, `<`, `>`, `<=`, `>`, `>=`.
	Time/date comparison converts dates to text in local time using the Notes default, `TZFMT_NEVER`, because Notes also sometimes converts fields to text internally. For example:
	`text="06/30/2005 02:52:04 PM"`

`<iftzeq>, <iftzne>`	Respectively, if the time zone equals or does not equal the comparison `text`, for example `CDT`, `EST`, and so on.

Conditional elements, continued

Element	Description
`<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** • `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** • `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** • `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** • `file`. (Required) The template file name. This name must be unique.
`<log>`	Log message to the NSF log file. **Attributes** • `text`. (Required) The text to log.

Control Elements, continued

Element	Description
	• `type`. (Optional) The type of log message. The following values are valid: ◦ `ERROR` ◦ `WARN` ◦ `INFO` ◦ `DIAG` (the default option) ◦ `DEBUG` ◦ `DUMP`
`<quit>`	Stop processing the template. Exits without error. **Attributes** • `text`. (Optional) The text to log. • `type`. (Optional) The type of log message. See <log>, on the previous page.
`<stop>`	Stop processing the template. Exits with an `ERROR` log message. **Attributes** • `text`. (Required) The text to log.

Data Elements

The following table lists the valid data elements.

Data elements

Element	Description
`<text>`	Output text. **Attributes** • `name`. (Required if there is no parent) The name of the item to output.
`<rich>`	Output rich text (MHTML). Images are output in the next part or parts of the MHTML, after the first `<HTML>` part. **Attributes** • `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>, above, images are output in the next part or parts of the MHTML.
`<form>`	Output the message form (usually `$Body` field) in rich text (MHTML). **Attributes** • `name`. (Required if there is no parent) The name of the item to output.

Data elements, continued

Element	Description
`<addr>`	Output an address. **Attributes** - `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 the next page for a list of all Notes and KeyView date and time formats and integer values. **Attributes** - `format`. (Optional. Omit to reset to defaults) The Notes and KeyView date and time format. You can set the following formats: ◦ `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 where `int` is an integer value that corresponds to the desired format. Separate multiple formats with commas. For example: `format="TD=0,TS=2,TT=1,TZ=1,KV=55"`

`<date>`	Output a Notes date.
	Attributes
	- `name`. (Required if there is no parent) The name of the item to output.
	- `format`. (Optional) See <format>, above. You can set the following values:
	  ◦ `TD`
	  ◦ `TS`
	  ◦ `TT`
	  ◦ `TZ`

`<date_kv>`	Output a KeyView date. **Attributes** - `name`. (Required if there is no parent) The name of the item to output. - `format`. (Optional) See <format>, above. You can set the following values: ◦ `TZ`

Data elements, continued

Element	Description
	○ `KV`
`<time>`	Output a time range, for example 1 hour, 30 minutes. **Attributes** • `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>`	Output a Notes time zone mnemonic, for example `MST`. **Attributes** • `name`. (Required if there is no parent) The name of date item to output.
`<zone_utc>`	Output a time zone as UTC, for example `(UTC-06:00)`.
`<logo>`	Output the mail header logo. The image link is included in the output; the actual image is output to a different part of the MHTML subfile.
`<image>`	Output an image. The image link is included in the output; the actual image is output to the MHTML next part, as with <rich>, on page 257 and <body>, on page 257.
`<image_uri>`	Output an image URI, in quotation marks. The actual image is output to a different part of the MHTML subfile. **Attributes** • `link`. (Required if there is no `file`) The image link, such as a form or title name. For example: • `link="StdNotesLtr0"` • `file`. (Required if there is no `link`) The name of the image file. The file must exist in the `../../templates/images` directory. For example: • `file="boxcheck.gif"`

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
`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

KeyView date and time formats, continued

Format	Output	Integer Value
`KVDTF_mmss`	`mm:ss`	12
`KVDTF_Hmmss`	`H:mm:ss`	13
`KVDTF_HHmmss`	`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
`KVDTF_ddsmmsyy`	`dd/mm/yy`	20
`KVDTF_ddsmmsyy_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_ddsmmsyy_P_hmm`	`dd/mm/yy P h:mm`	26
`KVDTF_ddsmmsyy_hmm_P`	`dd/mm/yy h:mm P`	27
`KVDTF_ddsmmsyy_P_hmmss`	`dd/mm/yy P h:mm:ss`	28
`KVDTF_ddsmmsyy_hmmss_P`	`dd/mm/yy h:mm:ss P`	29
`KVDTF_ddsmmsyy_P_hhmmss`	`dd/mm/yy P hh:mm:ss`	30
`KVDTF_ddsmmsyy_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

KeyView date and time formats, continued

Format	Output	Integer Value
`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
`KVDTF_yysmsd`	`yy/m/d`	46
`KVDTF_yysmmsdd`	`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_yyyyammadd`	`yyyy-mm-dd`	53
`KVDTF_yyyyammaddaHHmmss`	`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

KeyView date and time formats, continued

Format	Output	Integer Value
`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
`KVDTF_Mon_yy`	`Mon yy`	69
`KVDTF_Monc_yy`	`Mon, yy`	70
`KVDTF_Month`	`Month`	71
`KVDTF_Monthayy`	`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

KeyView date and time formats, continued

Format	Output	Integer Value
`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

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 184.

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 117 for more information.

For information on how to translate the extracted format information, see Translate Format Information, on page 269.

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 269.
- The reader associated with each format. See Determine a Document Reader, on page 270.
- 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=1
```

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 `ADDOCINFO` 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 `ADDOCINFO`, 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, `l123sr`:

```
81.2.0.9.0=l123
```

List of Required Files for Redistribution, on page 289 lists the readers provided with KeyView.

Category Values in formats.ini

This section lists the possible category values for format information in the `formats.ini` file. The corresponding values for format information extracted by a call to `fpGetDocInfoFile()` or `fpGetDocInfoStream()` are listed in the header file `adinfo.h`.

- Major Formats
- File Classes
- Minor Formats

Major Formats

Number	Format	File Class
1	AES Multiplus Comm Format	Word processor
2	ASCII File word processor/MS DOS Batch File format	Word processor
3	Applix Asterix	Word processor
4	Microsoft Windows Bitmap image (BMP)	Raster image
5	Convergent Tech DEF Comm. format	Word processor
6	Corel Draw (CDR)	Vector graphic

Major Formats, continued

Number	Format	File Class
7	Keyword COM.FILE (KSIF)	
8	Computer Graphics Metafile (CGM)	Vector graphic
9	Word Connection	Word processor
10	COMET TOP Word	Word processor
11	DG CEOwrite	Word processor
12	Honey Bull DSA101	Word processor
13	IBM DCA-RFT	Word processor
14	DDIF	Word processor
15	Dummy File (Internal)	
16	DG Common Data Stream (CDS)	Word processor
17	Dummy Print File (Internal)	
18	Windows Micrografx Draw (DRW)	Vector graphic
19	Data Point VISTAWORD	Word processor
20	DECdx	Word processor
21	Enable	Word processor
22	Encapsulated PostScript (EPS)	Raster image
23	DOS/Windows Executable (EXE, DLL)	Executable
24	CCITT Group 3 1-Dimensional (G31D)	Raster image
25	Graphics Interchange format (GIF)	Raster image
26	Hewlett Packard	Word processor
27	IBM 1403 Line Printer	Word processor
28	IBM DCF Script	Word processor
29	IBM DCA-FFT	Word processor
30	Interleaf	Word processor
31	GEM Bit Image	Raster image
32	IBM Display Write 4	Word processor
33	Raster Graphics	Raster image

Major Formats, continued

Number	Format	File Class
34	Keywords PICL	
35	Lotus AMI Pro	Word processor
36	MORE Database Outliner (Mac)	Outline/planning
37	Lyrix	Word processor
38	MASS-11	Word processor
39	MacPaint	Raster image
40	Microsoft Word Mac	Word processor
41	Informix SmartWare II Communication File	Communications
42	Microsoft Word for Windows	Word processor
43	MultiMate 4.0	Word processor
44	Multiplan Spreadsheet	Spreadsheet
45	Microsoft Rich Text Format (RTF)	Word processor
46	Microsoft Word 5.0 (PC)	Word processor
47	NBI Async Archive Format	Word processor
48	Navy DIF	Word processor
49	NBI Net Archive Format	Word processor
50	NIOS TOP	Word processor
51	FileMaker (Mac)	Database
52	ODA/ODIF	Word processor
53	OLIDIF	Word processor
54	Keyword OSM	
55	Office Writer	Word processor
56	PC Paint Brush Graphics (PCX)	Raster image
57	CPT Communication Format	Word processor
58	Lotus PIC	Vector graphic
59	Macintosh Quick Draw Picture Format (PICT)	Raster image
60	Philips Script	Word processor

Major Formats, continued

Number	Format	File Class
61	PostScript File	Vector graphic
62	PRIMEWORD	Word processor
63	Quadratron Q-One (V1.93J)	Word processor
64	Quadratron Q-One (V2.0)	Word processor
65	SAMNA Word IV	Word processor
66	Lotus AMI Pro Draw (SDW)	Raster image
67	SYLK Spreadsheet	Spreadsheet
68	Informix SmartWare II	Word processor
69	Symphony Spreadsheet	Spreadsheet
70	Truevision Targa	Raster image
71	Tagged Image File (TIFF)	Raster image
72	Targon Word (V 2.0)	Word processor
73	Uniplex Ucalc Spreadsheet	Spreadsheet
74	Uniplex (V6.01)	Word processor
75	Microsoft Word (UNIX)	Word processor
76	WANG PC	Word processor
77	WordERA (V 1.0)	Word processor
78	WANG WPS Comm. format	Word processor
79	WordPerfect Mac	Word processor
80	WordPerfect 5.2	Word processor
81	Lotus 1-2-3 Spreadsheet	Spreadsheet
82	WordMARC word processor	Word processor
83	Microsoft Windows Metafile (WMF) Graphics	Raster image
84	Informix SmartWare II Database	Database
85	WordPerfect Graphics V1.0 (WPG)	Raster image
86	WordPerfect	Word processor
87	WordStar	Word processor

Major Formats, continued

Number	Format	File Class
88	Wang WITA	Word processor
89	Xerox 860 Comm. format	Word processor
90	Microsoft Excel Spreadsheet	Spreadsheet
91	Xerox Writer word processor	Word processor
92	DIF Spreadsheet	Spreadsheet
93	ENABLE Spreadsheet	Spreadsheet
94	Supercalc Spreadsheet	Spreadsheet
95	Ultracalc Spreadsheet	Spreadsheet
96	Informix SmartWare Spreadsheet	Spreadsheet
97	Serialized Object Format (SOF) Encapsulation format	Encapsulation
98	Microsoft PowerPoint (PC)	Presentation
99	Microsoft PowerPoint (Mac)	Presentation
100	Aldus PageMaker (Mac)	Desktop Publishing
101	Aldus PageMaker (DOS)	Desktop Publishing
103	Microsoft Works (Mac)	Word processor
104	Microsoft Works Database (Mac)	Database
105	Microsoft Works Spreadsheet (Mac)	Spreadsheet
106	Microsoft Works Communication (Mac)	Communication
107	Microsoft Works (PC)	Word processor
108	Microsoft Works Database (PC)	Database
109	Microsoft Works Spreadsheet (PC)	Spreadsheet
111	PC Library Module	Library module
112	MacWrite	Word processor
113	MacWrite II	Word processor
114	Aldus Freehand Mac	Vector graphic
115	Disk Doubler Compression format	Encapsulation
116	HP Graphics Language (HP-GL)	Vector graphic

Major Formats, continued

Number	Format	File Class
117	Adobe Maker Interchange Format (MIF)	Desktop Publishing
118	JPEG File Interchange Format (JFIF)	Raster image
119	Reflex Database	Database
120	Framework II	Mixed format
121	Paradox (PC) Database	Database
123	Microsoft Windows Write	Word processor
124	Quattro Pro Spreadsheet (DOS)	Spreadsheet
126	Persuasion Presentation	Presentation
127	Corel Presentation	Presentation
128	Microsoft Windows Icon Format (ICO) Graphics	Raster image
129	Microsoft Project	Time scheduling
131	Harvard Graphics	Desktop publishing
132	Zip Archive Format	Encapsulation
133	Microsoft Windows Cursor (CUR) Graphics	Raster image
134	Quark Express (Mac)	Desktop publishing
135	ARC/PAK Archive format	Encapsulation
136	Adobe FrameMaker	Desktop publishing
137	Microsoft Publisher	Desktop publishing
138	Plan Perfect	Time scheduling
139	WordPerfect General File Format	Miscellaneous
140	Lotus Freelance	Presentation
141	Microsoft Wave Sound File	Sound
142	MIDI Sound File	Sound
143	AutoCAD DXF Graphics	Vector graphic
144	dBase Database	Database
145	OS/2 PM Metafile Graphics	Vector graphic
146	Lasergraphics Language	Vector graphic

Major Formats, continued

Number	Format	File Class
147	AutoShade Rendering File Format	Vector graphic
148	Graphics Environment Manager (GEM VDI)	Vector graphic
149	Microsoft Windows Help File	Miscellaneous
150	Volkswriter	Word processor
151	Ability Office (SS, DB, GR, WP, COM)	
152	XyWrite/Nota Bene	Word processor
153	Comma Separated Values (CSV)	Spreadsheet
154	Writing Assistant word processor	Word processor
155	WordStar 2000	Word processor
156	WordStar 6.0	Word processor
157	HP Printer Control Language (PCL)	Vector graphic
158	(UNIX/VAX/SUN) Executable	Executable
159	(UNIX/VAX/SUN) Object Module	Object module
160	(UNIX/VAX/SUN) Link Library	Library module
161	NeXT SUN Audio Data	Sound
162	NeWS font file (SUN)	Font
163	cpio Archive Format (UNIX/VAX/SUN)	Encapsulation
164	PEX Binary Archive (SUN)	Encapsulation
165	SUN vfont definition	Font
166	Curses Screen Image (UNIX/VAX/SUN)	Raster image
167	UU Encoded Encryption File	Encapsulation
168	WriteNow	Word processor
169	PC Object Module	Object module
170	Microsoft Windows Group File	Miscellaneous
171	PC True Type Font	Font
172	Program Information File	Miscellaneous
173	PC COM executable file	Executable

Major Formats, continued

Number	Format	File Class
174	Adobe FrameMaker Markup Language	Desktop publishing
175	Stuff It Archive (Mac)	Encapsulation
176	PeachCalc Spreadsheet	Spreadsheet
177	Wang Office GDL Header Encapsulation	Encapsulation
178	WordPerfect 6.0	Word processor
179	Q & A for DOS	Word processor
180	Q & A for Windows	Word processor
181	DEC WPS PLUS	Word processor
182	DCX Fax format	Fax
183	Microsoft Windows OLE 2 Encapsulation	Encapsulation
184	Quattro Pro for Windows	Spreadsheet
185	Keyword Viewer Markup Format	
186	EBCDIC Text	Word processor
187	DCS	Word processor
188	Microsoft Excel Spreadsheet 95, 2000	Spreadsheet
189	Microsoft Word for Windows 95	Word processor
190	UNIX SHAR Encapsulation	Encapsulation
191	Lotus Notes Bitmap	Raster image
192	UNIX Compress Encapsulation	Encapsulation
193	Lotus Notes CDF	Word processor
194	UNIX TAR Encapsulation	Encapsulation
195	WordPerfect Graphics V2.0 (WPG2)	Raster image Vector graphic
196	ODA/ODIF (FOD 26)	Word processor
197	ALIS	Word processor
198	GZ Compress Encapsulation	Encapsulation
199	Envoy (EVY)	Word processor

Major Formats, continued

Number	Format	File Class
200	Adobe Portable Document Format (PDF)	Word processor
201	KW ODA Internal Raw Bitmap (RBM)	Raster image
202	KW ODA G4 (G4)	Raster image
203	KW ODA G31D (G31)	Raster image
204	KW ODA Internal G32D (G32)	Raster image
205	Microsoft Word for Mac V 4.x/5.x	Word processor
206	BinHex 4.0 encoded file	Encapsulation
207	SMTP document	Encapsulation
208	MIME format - Microsoft Outlook Express (EML)/Mailbox (MBX)	Encapsulation
209	SGML document	Word processor
210	HTML document XHTML [1]	Word processor
211	ACT Format	Word processor
212	Microsoft PowerPoint 95	Presentation
213	Portable Network Graphics (PNG)	Raster image
214	Video for Windows	Movie
215	Windows Animated Cursor	Raster image
216	Windows C++ Object Storage	Mixed format
217	Windows Palette	Raster image
218	RIFF Device Independent Bitmap	Raster image
219	RIFF MIDI	Sound
220	RIFF Multimedia Movie	Movie
221	MPEG Movie	Movie
222	QuickTime Movie	Movie
223	Audio Interchange File Format (AIFF) Sound	Sound
224	Amiga MOD Sound	Sound
225	Amiga IFF (8SVX) Sound	Sound

Major Formats, continued

Number	Format	File Class
226	Creative Voice (VOC) Sound	Sound
227	Microsoft Works (Windows)	Word processor
228	Microsoft Works Spreadsheet (Windows)	Spreadsheet
229	AutoDesk Animator FLIC Animation	Animation
230	AutoDesk Animator Pro FLIC Animation	Animation
231	Microsoft Works Database (Windows)	Database
232	Microsoft Works Communication (Windows)	Communications
233	Compactor / Compact Pro Archive	Encapsulation
234	VRML	Vector graphic
235	QuickDraw 3D Metafile (3DMF)	Vector graphic
236	PGP Secret Keyring	Encapsulation
237	PGP Public Keyring	Encapsulation
238	PGP Encrypted Data	Encapsulation
239	PGP Signed Data	Encapsulation
240	PGP Signed and Encrypted Data	Encapsulation
241	PGP Signature Certificate	Encapsulation
242	ASCII-armored PGP Public Keyring	Encapsulation
243	ASCII-armored PGP encoded	Encapsulation
244	ASCII-armored PGP signed	Encapsulation
245	OLE DIB object	Raster image
246	PGP Compressed Data	Encapsulation
247	SGI Image	Raster image
248	Lotus Screen Cam	Animation
249	MPEG Audio	Sound
250	FTP Session Data	Communications
251	Netscape Bookmark file	Word processor
252	Corel Draw CMX	Vector image

Major Formats, continued

Number	Format	File Class
253	AutoCAD Drawing (DWG)	Vector graphic
254	AutoDesk WHIP	Vector graphic
255	Macromedia Director	Animation
256	Real Audio	Sound
257	MS DOS Device Driver	Executable
258	Micrografx Designer	Vector graphic
259	Simple Vector format (SVF)	Vector graphic
260	WordPerfect Office document (WPD)	
261	Applix Words	Word processor
262	Applix Graphics	Presentation
263	Microsoft Access	Database
264	Usenet format	Word processor
265	MacBinary	Encapsulation
266	Apple Single	Encapsulation
267	Apple Double	Encapsulation
268	Lotus Word Pro	Word processor
269	Microsoft Word 97, 2000	Word processor
270	Enhanced Window Metafile	Vector graphic
271	Microsoft Office Drawing	Vector graphic
272	Microsoft PowerPoint 97, 2000	Presentation
273	Extended or Custom XML	Word processor
274	Device Independent file (DVI)	Vector graphic
275	Unicode	Word processor
276	Framework	Mixed
277	KPIF Chart Stream	
278	Applix Spreadsheet	Spreadsheet
279	Microsoft Device Independent Bitmap	Raster image

Major Formats, continued

Number	Format	File Class
280	KeyView GPF Filter	
281	Microsoft Project 98, 2000, 2002	Time scheduling
282	Folio Flat file	Word processor
283	HWP (Arae-Ah Hangul)	Word processor
284	JustSystems Ichitaro	Word processor
285	Generic XML format Microsoft Office 2003 XML format [2]	Word processor
286	Fujitsu Oasys	Word processor
287	Portable Bitmap Utilities (PBM)	Raster image
288	Portable Greymap Utilities (PGM)	Raster image
289	Portable Pixmap Utilities (PPM)	Raster image
290	X Bitmap (XBM)	Raster image
291	X Pixmap (XPM)	Raster image
292	X Image	Raster image
293	PCD Image	Raster image
294	Microsoft Visio	Presentation
295	Microsoft Outlook (MSG)	Encapsulation
296	XHTML document	Word processor
297	Microsoft Outlook Personal Folders file (PST)	Encapsulation
298	WinRAR Compressed Archive format (RAR)	Encapsulation
299	Lotus Notes Database (NSF) Legato Extender ONM	Encapsulation
300	Macromedia Flash	Word processor
301	Microsoft Word 2007 (XML format)	Word processor
302	Microsoft Excel 2007 (XML format)	Spreadsheet
303	Microsoft PowerPoint 2007 (XML format)	Presentation
304	Open PGP (new format packets only)	Encapsulation

Major Formats, continued

Number	Format	File Class
305	Intergraph version 7 DGN	Vector graphic
306	Microstation version 8 DGN	Vector graphic
307	Microsoft Word 2007 Macro	Word processor
308	Microsoft Excel 2007 Macro	Spreadsheet
309	Microsoft PowerPoint Macro	Presentation
310	Microsoft Compression folder (LZH)	Encapsulation
311	Office 2007 Document	Miscellaneous
312	XML Paper Specification	Word processor
313	Lotus Domino Extensible Language	Encapsulation
314	OASIS Open Document (ODT)	Word processor
315	OASIS Open Document (ODS)	Spreadsheet
316	OASIS Open Document (ODP)	Presentation
317	Legato EMailXtender Native Message	Word Processor
319	Transfer Neutral Encapsulation Format (TNEF)	Encapsulation
320	CADAM Drawing	Vector graphic
321	CADAM Drawing Overlay	Vector graphic
322	NURSTOR Drawing	Vector graphic
323	HP Graphics Language (Plotter)	Vector graphic
324	Advanced Systems Format	Miscellaneous
325	Windows Media Audio Format	Sound
326	Windows Media Video Format	Movie
327	Legato EMailXtender Archive	Encapsulation
328	7-Zip	Encapsulation
329	Microsoft Office 2007 Excel Binary Format	Spreadsheet
330	Microsoft Cabinet File	Encapsulation
331	CATIA formats	Vector graphic
332	Yahoo! Instant Messenger	Word processor

Major Formats, continued

Number	Format	File Class
333	Founder Chinese E-paper Basic	Word processor
334	Corel Quattro Pro X4	Spreadsheet
335	MIME HTML	Word processor
336	Microsoft Document Imaging Format	Raster image
337	Microsoft Office Groove File Format	Word processor
338	Apple iWorks Pages	Word processor
339	Apple iWorks Numbers	Spreadsheet
340	Apple iWorks Keynote	Presentation
341	Microsoft Backup File	Encapsulation
342	Microsoft Access 2007	Database
343	Microsoft Entourage Database	Encapsulation
344	Mac Disk Copy Disk Image File	Encapsularion
345	Appleworks File	Word processor
346	Omni Outliner (OO3) File	Word processor
347	Omni Outliner (OPML) File	Word processor
348	Omni Graffle XML File	Vector graphic
349	Apple Photoshop Document	Raster image
350	Apple Binary Property List	Miscellaneous
351	Apple iChat Format	Word processor
352	Omni Outliner (OOUTLINE) File	Word processor
353	Bzip 2 Compressed File	Encapsulation
354	ISO-9660 CD Disc Image Format	Encapsulation
355	Xerox DocuWorks	Word processor
356	RealMedia Streaming Media	Movie
357	AC3 Audio File Format	Sound
358	Nero Encrypted File	Encapsulation
359	SolidWorks	Vector graphic

Major Formats, continued

Number	Format	File Class
362	UniGraphics NX	Vector graphic
366	Extensible Forms Description Language	Presentation
367	Apple XML Property List	Miscellaneous
368	OneNote Note Format	Presentation
370	Digital Imaging and Communications in Medicine (DICOM)	Raster image
371	Expert Witness Compression Format	Encapsulation
372	Shell Scrap Object File	Encapsulation
373	Microsoft Project 2007	Time scheduling
374	Microsoft Publisher 98–	Desktop publishing
375	Skype Log File	Word processor
376	Lotus Notes Bitmap Format (DXL embedded images)	Raster image
377	Health level7 message	Word processor
378	Microsoft Outlook Offline Storage File	Encapsulation
379	Open Publication Structure eBook	Word processor
380	Microsoft Outlook Express DBX	Encapsulation
381	BlackBerry Activation File	Word processor
382	Disk Image	Encapsulation
383	Milestone	Raster Image
384	RealLegal E-Transcript File	Word processor
385	PostScript Type 1 Font	Font
386	Ghost Disk Image File	Encapsulation
387	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	Raster Image
388	Unicode HTML	Word processor
389	Microsoft Compiled HTML Help	Encapsulation
390	Documentum EMCMF	Encapsulation
393	JBIG2 File	Raster image
395	AD1 Evidence file	Encapsulation

Major Formats, continued

Number	Format	File Class
397	Group Wise File Surf email	Encapsulation
402	ARJ	Encapsulation
409	Microsoft Outlook for Macintosh	Encapsulation
412	Microsoft Outlook vCard Contact	Word processor
414	Microsoft Outlook iCalendar	Encapsulation
418	Apple iWork 2013 Pages	Word processor
419	Apple iWork 2013 Numbers	Spreadsheet
420	Apple iWork 2013 Keynote	Presentation
421	XZ	Encapsulation
427	B1	Encapsulation
428	MP4	Movie
429	Rar5	Encapsulation
430	PTC Creo	Vector graphic
431	Keyhole Markup Language	
432	Zipped Keyhole Markup Language	
433	Wireless Markup Language	
435	Star Office Writer Text	
436	Star Office Calc Spreadsheet	
437	Star Office Impress Presentation	
438	Star Office Math	

1 If the major version is 100, the file format is XHTML.

2 The major version determines whether the Microsoft Office XML file is a Word, Excel or Visio document. The major version for each format is as follows:
Word: 100
Excel: 101
Visio: 110

File Classes

Attribute Number	File Class
0	No file class

File Classes, continued

Attribute Number	File Class
01	Word processor
02	Spreadsheet
03	Database
04	Raster image
05	Vector graphic
06	Presentation
07	Executable
08	Encapsulation
09	Sound
10	Desktop publishing
11	Outline/planning
12	Miscellaneous
13	Mixed format
14	Font
15	Time scheduling
16	Communications
17	Object module
18	Library module
19	Fax
20	Movie
21	Animation

Minor Formats

Attribute Number	Minor Format
00	Minor format not defined
01	Standard
02	Book

Minor Formats, continued

Attribute Number	Minor Format
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
27	Spelling
28	Thesaurus
29	Hyphenation

Minor Formats, continued

Attribute Number	Minor Format
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. These files are 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.

> **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 266.
`FilterDotNet.*`	Required by .NET API.
`KeyViewFilter.*`	Required by the Java API.
`kpifcnvt.*`	For presentation graphics, converts from one picture format to another.
`kpifutil.*`	Utility for handling the internal picture interchange format for presentation graphics.
`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 metadata from AutoCAD files.
`kvxsssa.*`	Interface between spreadsheet readers and `kvfilter`.
`kvxwpsa.*`	Interface between word processing readers and `kvfilter`.
`kwad.*`	File auto-recognition module.
`txtcnv.*`	Converter for document token stream.

Support Files

The following support files can be redistributed with your application.

File	**Description**
`bentofio.*`	Required by `l123sr` and `kpprzrdr`.
`cbmap.map`	Character mappings for Adobe Portable Document Format (PDF).
`chartbls.ux`	Character mappings.
`chmdll.*`	Required by `chmsr`.
`kppng.*`	Required for ZLIB decompression.
`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.
`MSVCP60.*`	Microsoft Visual C++ Runtime library V6.0.
`msvcrt.*`	Microsoft Visual C Runtime library.
`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

File	Description
`bl1sr.*`	B1 archive reader
`bkfsr.*`	Microsoft Backup File reader
`bmpsr.*`	Windows bitmap (BMP) reader
`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
`dxlsr.*`	Domino XML Language reader
`emlsr.*`	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
`hl7sr.*`	Health level7 reader (metadata only)
`htmsr.*`	HTML and XHTML reader

File	Description
`hwpsr.*`	Hangul 97 reader
`hwposr.*`	Hangul 2002, 2005, 2007 reader
`ichatsr.*`	Apple iChat Log reader
`icssr.*`	Microsoft Outlook iCalendar reader
`isosr.*`	ISO-9660 CD Disc Image Format 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
`kpIWPGrdr.*`	Apple iWork Keynote reader
`kpmsordr.*`	Microsoft Office Drawing Objects (office 97, 2000, and XP) reader
`kpODArdr.*`	AutoCAD reader (Windows only)
`kpodfrdr.*`	Oasis Open Document Format presentation (ODP) reader
`kpONErdr.*`	Microsoft OneNote reader
`kpp40rdr.*`	Microsoft PowerPoint PC 4.0 and PowerPoint Mac reader
`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

File	Description
`kpppxrdr.*`	Microsoft PowerPoint XML reader 2007
`kpprerdr.*`	Lotus Freelance Graphics for Windows V2.0 reader
`kpprzrdr.*`	Lotus Freelance Graphics 96/97/98 reader
`kpshwrdr.*`	Corel Presentations reader
`kpugrdr.*`	Unigraphics (UG) NX 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
`ltbenn30.dll`	Lotus Word Pro support (supported on Windows x86 platform only)
`ltscsn10.dll`	Lotus Word Pro support (supported on Windows x86 platform only)
`lwpapin.dll`	Lotus Word Pro support (supported on Windows x86 platform only)
`lwppann.dll`	Lotus Word Pro support (supported on Windows x86 platform only)
`lwpsr.dll`	Lotus Word Pro reader (supported on Windows x86 platform only)
`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

---

[1] This reader is an advanced feature and is sold and licensed separately from KeyView Filter SDK. See License Information, on page 19

File	Description
`misr.*`	Microsoft Word 2 reader
`mp3sr.*`	MP3 reader for metadata extraction reader
`mpeg4sr.*`	MPEG-4 Audio file reader
`mppsr.*`	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
`odfsssr.*`	Oasis Open Document Format spreadsheets (ODS) reader
`odfwpsr.*`	Oasis Open Document Format word processing (ODS) reader
`olesr.*`	Embedded OLE object reader
`olmsr.*`	Microsoft Outlook for Macintosh reader
`onmsr.*`	Legato EMailXtender Native Message reader
`oo3sr.*`	Omni Outliner 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]

File	Description
`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
`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
`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

File	Description
`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

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 23.

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 309. 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, KVTSIZE_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 309

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.

Parameter	In/Out	Description
`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 `BOOL`s 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.

xxxsrAutoDet()

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 266.

Syntax

```
Bool pascal _export xxxsrAutoDet(
    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`:

`adClass`	Must be set to `1`.
`adFormat`	A numerical value assigned to this reader in the `[Formats]` section of the `formats.ini` file.
`descStr`	A string describing the file format.
`mMnmemStr`	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

`pSALContext`	A pointer to the global data context structure of the structured access layer.
`fp`	A pointer to a structure of callback functions supported by the structured access layer.
`pbOpenDoc`	You must set this `BOOL` value to **`TRUE`** if the allocation of memory for the global data context structure is successful.
`pVapi`	A pointer to a structure providing memory management and character conversion functions. Because this functionality is proprietary to HPE, `TPVAPIServices` is redefined as `void` in `kvcfsr.h`.
`dwFlags`	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    *pCFContext,
    BYTE    *pcBuf,
    UINT    *pnBufOut,
    int     *pnPercentDone,
    UINT     cbBufOutMax)
```

Arguments

`pCFContext`	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                 *pCFContext,
    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

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
Lotus Notes NSF	n/a	N	N	Y	N	C

---

[1] The native PST reader, `pstnsr`, does 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
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 reader (`pstnsr`) returns the error message `KVERR_PasswordProtected` if a PST 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() function.

   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:
2. `(*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.

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 (KeyView 11.5)

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 autonomytpfeedback@hpe.com.

We appreciate your feedback!