KeyView

Software Version 12.2

HTML Export SDK C and COM Programming Guide

MICRO FOCUS®

Legal notices

Copyright notice

Documentation updates

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

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

You can check for more recent versions of a document through the MySupport portal. Many areas of the portal, including the one for documentation, require you to sign in with a Software Passport. If you need a Passport, you can create one when prompted to sign in.

Additionally, if you subscribe to the appropriate product support service, you will receive new or updated editions of documentation. Contact your Micro Focus sales representative for details.

Support

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

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

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

Many areas of the portal require you to sign in with a Software Passport. If you need a Passport, you can create one when prompted to sign in. To learn about the different access levels the portal uses, see the Access Levels descriptions.

Contents

Part I: Overview of HTML Export

This section provides an overview of the Micro Focus IDOL KeyView Export SDK and describes how to use the C and COM implementations of the API.

Chapter 1: Introducing HTML Export

This guide is for developers who incorporate the Micro Focus KeyView HTML conversion technology into their custom web applications using a C or COM development environment. It is intended for readers who are familiar with HTML, C, and/or COM.

This section describes the KeyView Export SDK package.

Overview

HTML Export is part of the KeyView Export SDK. It enables you to convert virtually any document, spreadsheet, presentation, or graphic into high-fidelity HTML. Incorporating this technology into your web-based applications enables your end-users to access a document even if they do not have the appropriate plug-in or native application. With HTML Export, you control the content, structure, and format of the HTML output using either easily customized templates, or the flexible and robust APIs.

Export SDK supports a number of programming environments, such as Visual Basic, Java,.NET, and Delphi and runs on all popular operating system platforms including Windows, Solaris, HP-UX, IBM AIX, and Linux.

Export 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

- Dynamically convert word processing, spreadsheet, presentation, and graphics files into web-ready, 4.0-compliant HTML.

- Export supports over 300 formats in 70 languages.

- Convert files either in-process or out of process. Out-of-process conversion ensures the stability and robustness of the calling application if a corrupt document causes an exception or causes the conversion process to fail.

- You can extract files embedded within files by using the File Extraction API, and then convert them

by using the Export API.

- Use redirected input/output. You can provide an input stream that is not restricted to file system access.

- Export automatically recognizes the file format being converted and uses the appropriate reader. Your application does not need to rely on file name extensions to determine the file format.

- Create heading levels in the output file either by using the structure in the source document or by allowing Export to automatically generate a structure based on document properties, such as font or font attributes.

- Use callbacks to control aspects of the conversion process, such as file naming and the insertion of scripts.

- Manage memory allocation to optimize speed and performance of application.

- Insert predefined HTML markup at specific points in the output stream.

- Create navigable documents by automatically inserting links into target HTML. You can also break large documents into multiple linked web pages.

- Apply Cascading Style Sheets (CSS) to improve the fidelity of the output.

- Map paragraph and character styles in word processing documents to any markup that you specify in the output.

- Control the resolution of rasterized vector graphics to optimize storage requirements or image quality.

- Select the target format for converted graphics, including GIF, JPEG, CGM, PNG, WMF, and Java on Windows, and Java and JPEG on Unix and Linux.

- Define the background, colors, and fonts used in the final HTML document, or maintain the source document's existing attributes.

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 11	Sun C 5.8 Patch 121015-06 2007/10/03
Linux	x86	gcc / g++	3.4.3 (Redhat 4), 4.1.0 (SuSE Linux 10)
	x64	gcc / g++	4.1.0 (Redhat 4), 4.1.0 (SuSE Linux 10)
IBM AIX	Power	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 Components

Component	Compiler
Java components	Java 1.5

C++ Filter SDK

The C++ Filter SDK is supported on:

- Linux using GCC 5 or later

- Windows using Visual Studio 2015 or later

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 or later is required to process Microsoft Outlook Personal Folders (PST) files using the MAPI-based reader (`pstsr`). The native PST readers (`pstxsr` and `pstnsr`) do not require Outlook.

  > **NOTE:**
  > You must install an edition of Microsoft Outlook (32-bit or 64-bit) that matches the KeyView software. For example, if you use 32-bit KeyView, install 32-bit Outlook. If you use 64-bit KeyView, install 64-bit Outlook.
  >
  > If the editions do not match, KeyView returns `Error 32: KVError_PSTAccessFailed` and an error message from Microsoft Office Outlook is displayed: `Either there is a no default mail client or the current mail client cannot fulfill the messaging request. Please run Microsoft Outlook and set it as the default mail client.`

- Lotus Notes or Lotus Domino is required for Lotus Notes database (NSF) file processing. The minimum requirement is 6.5.1, but version 8.5 is recommended.

- The Microsoft .NET Framework is required if you are using the .NET implementation of the API.

- Microsoft Visual C++ 2013 and Microsoft Visual C++ 2010 Redistributables (Windows only).

Windows Installation

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

To install the SDK

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

   For example:

   `KeyViewExportSDK_12.2_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 22.

   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:

   ```
   ./KeyViewExportSDK_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 Export installation contains:

- Libraries and executable files necessary for converting source documents into high-quality, web-ready HTML (see Files Required for Redistribution, on page 401).

- The include files that define the functions and structures used by the application to establish an interface with Export:

  `adinfo.h`

  `kvhtml.h`

  `kvtypes.h`

```
kvxtract.h
```

- The Java API implemented in the `com.verity.api.export` package contained in the `KeyView.jar` file.

- Several sample programs that demonstrate Export's functionality.

- Sample images that can be used as navigation buttons and background textures in your output.

- Template files that enable you to set conversion options without modifying at the API level. They can be used to generate a wide range of output, from highly-stylized user-defined HTML to stripped-down, text-only output suitable for use with an indexing engine.

- Sample style sheet: `WordStyle.css` (for word processing documents).

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) readers (`pstsr`, `pstnsr`, and `pstxsr`)

  ○ Lotus Notes database (NSF) reader (`nsfsr`)

  ○ Mailbox (MBX) reader (`mbxsr`)

  ○ Character set detection library (`kvlangdetect`)

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

Enable Advanced Document Readers

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

If you are enabling the MBX reader in an existing installation of Export, in addition to updating the license key, change the parameter `208=eml` to `208=mbx` in the `formats_e.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 Micro Focus. 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 Micro Focus. The license key is listed in the **Key** column in the **Standalone Products** table. The key is a string that contains 31 characters, for example, `2TQD22D-2M6FV66-2KPF23S-2GEM5AB`. Enter the characters exactly as they appear in the document, including the dashes, but do not include a leading or trailing space.

4. The finished `kv.lic` file looks similar to the following:

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

5. Save the `kv.lic` file.

Directory Structure

The following table describes the directories created during the HTML Export installation. The variable *install* is the path name of the Export installation directory (for example, `/usr/autonomy/KeyviewExportSDK` on UNIX, or `C:\Program Files\Autonomy\KeyviewExportSDK` on Windows). On UNIX, the HTML Export directory is named `/htmlexpt`.

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\KeyviewExportSDK\WINDOWS\bin`.

HTML Export installed directory structure

Directory	Description
install*OS*\bin	Contains the libraries, executables for sample programs, the Java program (`kvraster.class`), the Java applet (`kvvector.jar`), the format detection file, `formats_e.ini`, the license key file (`kv.lic`), and a number of 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\dotnet\sample	The `HtmlConvFileToFile.cs` C# sample program demonstrating the .NET interface.
install\HTML Export\docs	Contains the converted version of the sample word processing, spreadsheet, and presentations files.
install\HTML Export\guide	Contains the *HTML Export C and COM Programming Guide* and *HTML Export Java Programming Guide* in HTML and PDF format.
install\HTML Export\include	Contains the header files.
install\HTML Export\programs\bin	Contains the executable files for the Visual Basic sample program called Export Demo.
install\HTML Export\programs\callback	Contains the C source code and supporting files for a sample program that demonstrates how user callbacks can dynamically shape the HTML conversion.
install\HTML Export\programs\cnv2html	Contains the C source code for a sample program that creates a single HTML file. The executable for this sample program is in the `bin` directory.
install\HTML Export\programs\cnv2htmloop	Contains the C source code for a sample program that creates a single HTML file out of process.
install\HTML Export\programs\comsamp	Contains the Visual Basic source code and supporting files for a sample program that demonstrates the use of the COM interface.
install\HTML Export\programs\ExportDemo	Contains the source code for a sample Visual Basic program. The executable for this program is in the `bin` directory. The Export Demo is available through the **Start** menu.
install\HTML Export\programs\htmlini	Contains the C source code and supporting files for a sample program that uses template files to set the conversion options.
install\HTML Export\programs\htmloop	Contains a Visual C++ sample program that uses the Microsoft Foundation Classes (MFC) to provide out-of-process HTML

HTML Export installed directory structure, continued

Directory	Description
	conversion using the COM automation server.
install\HTML Export\programs\images	Contains the background graphics and navigation buttons used by the template files.
install\HTML Export\programs\index	Contains the C source code and supporting files for a sample program that produces text-only HTML.
install\HTML Export\programs\ini	Contains the template files used to set the conversion options in the C API.
install\HTML Export\programs\io_samp	Contains the C source code and supporting files for a sample program that demonstrates how to input a stream by providing a simple wrapper around the ANSI C interface `fread()`, `fopen()`, and so on.
install\HTML Export\programs\jstree	Contains the C source code and supporting files for a sample program that employs JavaScript to produce an expandable table of contents in a frame-based HTML output file.
install\HTML Export\programs\jvtree	Contains the C source code and supporting files for a sample program that uses the `JVTree` Java applet in creating an expandable table of contents in a frame-based HTML output file.
install\HTML Export\programs\jvtree_demo	C code sample that creates a frame-based HTML stream using the `JVTree` Java applet to display the table of contents.
install\HTML Export\programs\pdfini	Contains the configuration file used to extract custom metadata from PDF documents.
install\HTML Export\programs\tempout	The default output directory for converted files. Contains a sample style sheet.
install\HTML Export\programs\tstxtract	Contains the C source code and supporting files for a sample program that demonstrates the File Extraction interface.
install\HTML Export\programs\wizard	Contains the source code and supporting files for the Visual Basic program HTML Export Template Wizard.
install\HTML Export\rel_notes	Contains the *HTML Export Release Notes* in HTML and PDF format.
install\javaapi\ini	Contains the template files used with the Java API.
install\javaapi\javadoc	Contains the Javadoc for the Java API.

HTML Export installed directory structure, continued

Directory	Description
install\javaapi\sample	Contains the source files and sample programs for the Java API.
install\testdocs	Contains sample word processing, spreadsheet, and presentation files that you can use to test HTML Export's options. You might also find this directory useful when testing your own applications.

Definition of Terms

The following are specialized terms used throughout the guide.

anchor	HTML markup that defines both anchors and hyperlinks. An anchor is a named place in a document to which other documents can form a link. Anchors use the HTML anchor tags (`<a> </a>`) to facilitate navigation within a document.
block	All source document content (including subheadings) associated with Heading Level 1. Export identifies and/or generates blocks from the input stream for the implementation of the your HTML markup.
block chunk or chunk	All source document content associated with Heading Levels 2 through 6. Chunks are subdivisions of blocks. You can supply specific HTML markup for the different levels of block chunks.
callback	A function optionally supplied by your application and called from the Export API. For example, callbacks allow your application to monitor the progress of the conversion process dynamically.
stream	Transmission of a file's content between memory and disk in a continuous flow.
token	The vehicle for conveying specific types of information to and from the API during the conversion process. Tokens are placeholders for markup that appears in the output. See Export Tokens, on page 390.

Chapter 2: Getting Started

This section provides an overview of the HTML Export SDK and describes how to use the C and COM implementations of the API.

Architectural Overview

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

HTML Export Architecture
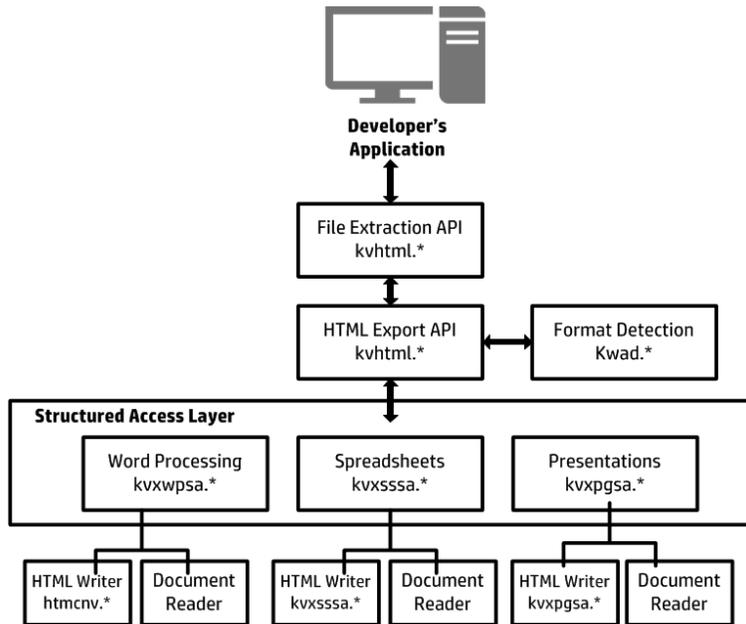
Each component is described in the following table.

Architectural Components

Component	Description
Developer's Application	The developer's application interfaces directly with the HTML Export API through either a C-language, Java, or COM implementation.
File Extraction API	The File Extraction API opens a file and extracts the file's subfiles so that the subfiles are available for conversion. See Use the File Extraction API, on page 53.
HTML Export API	The HTML Export API exposes the functionality of HTML Export and controls all other HTML Export modules during the conversion process.
Format Detection Module	The format detection module determines the file type of the source file, which enables the HTML Export interface to load the appropriate structured access layer module and document reader. See File Format Detection, on page 393.
Structured Access Layer	The structured access layer contains three modules: one for word processing, one for spreadsheets, and one for presentations and graphics. Information from the format detection module determines which access layer module operates at this stage of the conversion. The structured access layer performs the following: 1. Loads the appropriate document reader. 2. Processes the data stream from the document reader. 3. Determines table of contents entries. 4. Sends the stream to the appropriate HTML writer. 5. Accepts the HTML stream from the HTML writer. 6. Generates the HTML output file with a table of contents, metadata, and the document's contents, and sends it to the HTML Export interface.
Document Reader	Each document reader reads a specific file format and sends a text stream of the document to the structured access layer. Word processing readers return a *token stream* to the structured access layer. A token stream contains the document contents and messages (tokens) that precede the content and identify the type of information that follows them. Each reader is loaded as required by the structured access layer. See Document Readers and Writers, on page 404 for a complete list of document readers.
HTML Writers	Each HTML writer accepts a text stream or token stream from the structured access layer and generates an equivalent HTML stream that is sent back to the structured access layer. The structured access layer then generates the output file. See Document Readers and Writers, on page 404 for a list of format writers.

Memory Abstraction

All dynamic memory allocations in Export modules are abstracted through a C interface. This memory allocation interface is defined in the `KVMemoryStream` structure in `kvtypes.h`. KVMemoryStream, on page 202. 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. The `callback` sample program demonstrates Export memory management features.

callback, on page 125.

Enhance Performance

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

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_e.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 (fpInit(), on page 174). 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_e.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.

The `formats_e.ini` file is in the directory *install*`\`*OS*`\bin`, where *install* is the path name of the Export installation directory and *OS* is the name of the operating system.

Convert Files Out of Process

Export can run independently from the calling application. This is called *out of process*. Out-of-process conversions protect the stability of the calling application in the rare case when a malformed document causes Export to fail. You can also run Export in the same process as the calling application. This is

called *in process*. However, it is strongly recommended you convert documents out of process whenever possible.

The Export out-of-process framework uses a client-server architecture. The calling application sends an out-of-process conversion request to the Service Request Broker in the main Export process. The Broker then creates, monitors, and manages a Servant process for the request—each request is handled by one independent Servant process. Data is exchanged between the application thread and the Servant through TCP/IP sockets. The source data is sent to the Servant process as a data stream or file, converted in the Servant, and then returned to the application thread. At that point, the application can either terminate the Servant process or send more data for conversion.

Multiple conversion requests can be sent from multiple threads in the calling application simultaneously. All requests sent from one thread are processed by the Servant mapped to that thread, in other words, each thread can only have one Servant to process its conversion requests.

Any standard conversion errors generated by the Servant are sent to the application.

> **NOTE:** Currently, the main Export process and Servant processes must run on the same host.

The following are requirements for running Export out of process:

- Internet Protocol (TCP/IP) must be installed

- Multithreaded processing must be supported on the operating system platform

- The user application must be built with a multithreaded runtime library

fpConvertStream()	fpGetConvertFileList()
fpGetSummaryInfo()	fpSetStyleMapping()
KVHTMLConfig()	KVHTMLConvertFile()
KVHTMLSetStyleSheet()	

> **NOTE:** When converting in out-of-process mode, these functions must be called after the call to start an out-of-process session and before the call to end an out-of-process session.

Other HTML Export functions, the File Extraction functions, and the COM methods always run in-process.

Configure Out-of-Process Conversions

Although most components of the out-of-process conversion are transparent, the following parameters are configurable:

- File-size threshold/temporary file location

- Conversion time-out

- Listener port numbers and time-out

- Connection time-out and retry

- Servant process name

These parameters are defined internally, but you can override the default by defining the parameter in the `formats_e.ini` file. The `formats_e.ini` file is in the directory `install\OS\bin`, where `install` is the path name of the Export installation directory and `OS` is the name of the operating system.

To set the parameters, add the following section to the `formats_e.ini` file:

```
[KVExportOOPOptions]
TempFileSizeMark=
TempFilePath=
WaitForConvert=
WaitForConnectionTime=
ListenerPortList=
ListenerTimeout=
ConnectRetryInterval=
ConnectRetry=
ServantName=
EnableDebugOutput=
EnableDebugLog=
LogFilePath=
ClientLogFile=
ServerLogFile=
```

Each parameter is described in the following table. The default values for these parameters are set to ensure reasonable performance on most systems. If you are processing a large number of files, or running Export on a slow machine, you might need to increase some of the time-out and retry values.

Parameters for Out-of-Process Conversion

Parameter	Description
TempFileSizeMark unit = megabytes default=10	The *file-size threshold*. If the input file received by the Servant is larger than this value, temporary files are created to store the data. The directory in which the temporary files are stored is defined by the `TempFilePath` parameter. If the file received is smaller than this value, the data is stored in memory in the Servant. This applies only when the input is a stream.
TempFilePath type = file path default = current working directory	The directory in which temporary files are stored. Temporary files are created when you use the `fpConvertStream()` API, and the input file surpasses the file-size threshold (`TempFileSizeMark`). If the Servant cannot access the file path, an error is generated. This applies only when converting in stream mode.
WaitForConvert unit = seconds default = 1800 range = 30~3600	The length of time to wait for a Servant to convert a file. If the conversion is not completed within the specified time, the error code "`Wait for child process failed`" is generated.
WaitForConnectionTime	The length of time to wait for the Servant to connect to the application

Parameters for Out-of-Process Conversion, continued

Parameter	Description
unit = seconds default = 180 range = 15~600	thread after the application has sent a conversion request to the Broker. If the Servant does not connect within the specified time, the error code "`Wait for child process failed`" is generated. If there are many Servant processes running simultaneously, you might need to increase this value.
`ListenerPortList` type = integer default = 9985, 9986, 9987, 9988, 9989	The TCP/IP port number used for communication between the calling application and the Servant. You can specify a single port number, or a series of numbers separated by commas.
`ListenerTimeout` unit = seconds default = 10 range = 5~30	The length of time to wait for the Servant listener thread to get a process ID from the Servant after the connection is established. If the ID is not obtained within the specified time, the error code "`Wait for child process failed`" is generated. During this time, no other Servant can connect with the application.
`ConnectRetryInterval` unit = microseconds default = 0.1 range = 50000~500000	The length of time to wait after a Servant has failed to connect to the application before it retries the connection. A Servant might be unable to connect because the application is waiting for another Servant to send a process ID. To calculate the *total retry interval*, the value set here is added to the platform-specific TCP retry value (on Windows, this is 1 second).
`ConnectRetry` type = integer default = 120 range = 30~600	The number of attempts the Servant makes to connect to the calling application. This value and the total retry interval determine the total delay time. The total delay is calculated as follows: `ConnectRetryInterval` + `platform-specific_TCP_retry_value` * `ConnectRetry` For example, if the `ConnectRetryInterval` is set to 2 seconds, and the Export process is running on Windows (the default TCP retry value on Windows is 1 second), the total delay would be: 2 + 1 * 120 = 360 The Servant would attempt to connect to the application every 3 seconds for 120 attempts for a total of 360 seconds.
`ServantName` type = string default = servant	The name of the Servant process. To move the Servant to another location, enter a fully qualified path.

Run Export Out of Process—Overview

To convert files out of process

1. If required, set parameters for the out-of-process conversion in the `formats_e.ini` file. See Configure Out-of-Process Conversions, on page 30.

2. Initialize an Export session.

3. If you are using streams, create an input stream.

4. Define the conversion options.

5. Initialize an out-of-process session.

6. Convert the input and/or call other functions that can run out of process.

7. Shut down the out-of-process session.

8. Repeat Step 3 through Step 7 for additional files.

9. Terminate the out-of-process session and the Servant process.

10. Shutdown the Export session.

Recommendations

- To ensure that multithreaded conversions 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. Creating a context pointer for every thread does not affect performance because the context pointer uses minimal resources.

- All functions that can run in out-of-process mode must be called within the out-of-process session (that is, after the call to initialize the out-of-process session and before the call to end the out-of-process session).

- When terminating an out-of-process session, persist the Servant process by setting the Boolean flag `bKeepServantAlive` in the `KVHTMLEndOOPSession()` function or `endOOPSession` method. If the Servant process remains active, subsequent conversion requests are processed more quickly because the Servant process is already prepared to receive data. Only terminate the Servant when there are no more out-of-process requests.

- To recover from a failure in the Servant process, start a new out-of-process session. This creates a new Servant process for the next conversion.

Run Export Out of Process

The `cnv2htmloop` sample program demonstrates how to run Export out of process.

To convert files out of process in the C API

1. If required, set parameters for the out-of-process conversion in the `formats_e.ini` file. See Configure Out-of-Process Conversions, on page 30.

2. Declare instances of the following types and assign values to the members as required:

   `KVHTMLTemplateEx`
   `KVHTMLOptionsEx`
   `KVHTMLHeadingInfo`
   `KVHTMLTOCOptions`

   See HTML Export API Structures, on page 200 for more information.

3. Load the `KVHTML` library and obtain the `KVHTMLInterfaceEx` entry point by calling `KVHTMLGetInterfaceEx()`.

   See KVHTMLGetInterfaceEx(), on page 161.

4. Initialize an Export session by calling `fpInit()`. See fpInit(), on page 174.

5. If you are using streams for the input and output source, follow these steps; otherwise, proceed to Step 6:

   a. Create an input stream (`KVInputStream`) by calling `fpFileToInputStreamCreate()`. See fpFileToInputStreamCreate(), on page 166.

   b. Create an output stream (`KVOutputStream`) by calling `fpFileToOutputStreamCreate()`. See fpFileToOutputStreamCreate(), on page 167.

   c. Proceed to Step 6.

6. Set up an out-of-process session by calling `KVHTMLStartOOPSession()`.

   See KVHTMLStartOOPSession(), on page 191. This function performs the following:

   - Initializes the out-of-process session.

   - Specifies the input stream or file. If you are using an input file, set `pFileName` to the file name, and set `pInputStream` to **NULL**. If you are using an input stream, set `pInputStream` to point to `KVInputStream`, and set `pFileName` to **NULL**.

   - Passes conversion options from the `KVHTMLTemplateEx`, `KVHTMLOptionsEx`, and `KVHTMLTOCOptions` data structures.

   - Creates a Servant process.

   - Establishes a communication channel between the application thread and the Servant.

   - Sends the data to the Servant.

     See the sample code in Example—KVHTMLStartOOPSession, on the next page, and KVHTMLStartOOPSession(), on page 191.

7. Convert the input and generate the output files by calling `KVHTMLConvertFile()` or `fpConvertStream()`. The `KVHTMLTemplateEx`, `KVHTMLOptionsEx`, and `KVHTMLTOCOptions` structures are passed in the call to `KVHTMLStartOOPSession()`, and should be `NULL` in the conversion call. A conversion function can be called only once in a single out-of-process session.

8. Terminate the out-of-process session by calling `KVHTMLEndOOPSession()`. The Servant ends the current conversion session, and releases the source data and session resources. See sample code in Example—KVHTMLEndOOPSession, on the next page, and KVHTMLEndOOPSession(), on page 187.

9. If you used streams, free the memory allocated for the input stream and output stream by calling the `fpFileToInputSreamFree()` and `fpFileToOutputStreamFree()` functions. See fpFileToInputStreamFree(), on page 167 and fpFileToOutputStreamFree(), on page 168.

10. Repeat Step 5 to Step 9 for additional files.

11. After all files are converted, terminate the out-of-process session *and* the Servant process by calling `KVHTMLEndOOPSession()` and setting the Boolean to **FALSE**.

12. After the out-of-process session and Servant are terminated, shut down the Export session by calling `fpShutDown()`. See fpShutDown(), on page 176.

Example—KVHTMLStartOOPSession

The following sample code is from the `cnv2htmloop` sample program:

```
/* declare OOP startsession function pointer */
KVHTML_START_OOP_SESSION fpKVHTMLStartOOPSession;
/* assign OOP startsession function pointer */
fpKVHTMLStartOOPSession = (KVHTML_START_OOP_SESSION)mpGetProcAddress
                          (hKVHTML, "KVHTMLStartOOPSession");
   if(!fpKVHTMLStartOOPSession)
   {
      printf("Error assigning KVHTMLStartOOPSession pointer\n");
      (*KVHTMLInt.fpFileToInputStreamFree)(pKVHTML, &Input);
      (*KVHTMLInt.fpFileToOutputStreamFree)(pKVHTML, &Output);
      mpFreeLibrary(hKVHTML);
      return 7;
   }
/********START OOP SESSION ****************/
if(!(*fpKVHTMLStartOOPSession)(pKVHTML,
      &Input,
      NULL,
      &HTMLTemplates,      /* Markup and related variables */
      &HTMLOptions,        /* Options */
      NULL,                /* TOC options */
      &oopServantPID,
      &error,
      0,
      NULL,
      NULL))
{
   printf("Error calling fpKVHTMLStartOOPSession \n");
   (*KVHTMLInt.fpShutDown)(pKVHTML);
   mpFreeLibrary(hKVHTML);
```

```
    return 9;
}
```

Example—KVHTMLEndOOPSession

The following sample code is from the `cnv2htmloop` sample program:

```
/* declare endsession function pointer */
KVHTML_END_OOP_SESSION    fpKVHTMLEndOOPSession;
/* assign OOP endsession function pointer */
fpKVHTMLEndOOPSession = (KVHTML_END_OOP_SESSION)mpGetProcAddress
                        (hKVHTML, "KVHTMLEndOOPSession");
   if(!fpKVHTMLEndOOPSession)
   {
      printf("Error assigning KVHTMLEndOOPSession pointer\n");
      (*KVHTMLInt.fpFileToInputStreamFree)(pKVHTML, &Input);
      (*KVHTMLInt.fpFileToOutputStreamFree)(pKVHTML, &Output);
      mpFreeLibrary(hKVHTML);
      return 8;
   }
/********END OOP SESSION, DO NOT KEEP SERVANT ALIVE *********/
if(!(*fpKVHTMLEndOOPSession)(pKVHTML,
      FALSE,
      &error,
      0,
      NULL,
      NULL))
{
   printf("Error calling fpKVHTMLEndOOPSession \n");
   (*KVHTMLInt.fpShutDown)(pKVHTML);
   mpFreeLibrary(hKVHTML);
   return 10;
}
```

Convert Files

KeyView Export SDK enables you to *convert* many different types of documents to HTML. Converting is the process of extracting the text from a document without the application-specific markup, and applying HTML markup. However, the conversion process can also include the following:

- Extracting subfiles—exposes all subfiles for conversion. See Subfile Extraction, on the next page.

- Setting conversion options—determines the content, structure, and appearance of the HTML output. See Set Conversion Options, on page 38.

- Extracting the file's format—detects a file's format, and reports the information to the API, which in turn reports the information to the developer's application. See Extract File Format Information, on page 79.

- Extracting metadata—extracts selected metadata (document properties) from a file. See Extract Metadata, on page 76.

- Converting character sets—controls the character set of both the input and the output text. See Convert Character Sets, on page 80.

- Implementing callbacks—controls the conversion while it is in progress. See HTML Export API Callback Functions, on page 195.

You can use one of the following methods to convert documents:

- Use the Export Demo sample program. This Visual Basic program demonstrates most Export API functionality and is the easiest way to get started. See Use the Export Demo Program, on page 41.

- Use the C-language implementation of the API from your C or C++ application. See Use the C-Language Implementation of the API, on page 45.

- Use the COM implementation of the API from your Visual Basic, Delphi, or J++ or C application (32-bit Windows platforms only). See Use the COM Implementation of the API, on page 48.

- Use the C or COM sample programs. See Sample Programs, on page 117.

> **NOTE:** Micro Focus strongly recommends that you convert documents *out of process*. During out-of-process conversion, Export runs independently from the calling application. Out-of-process conversions protects the stability of the calling application in the rare case when a malformed document causes Export to fail. Convert Files Out of Process, on page 29.

Subfile Extraction

To convert a file, you must first determine whether the source file contains any subfiles (attachments, embedded objects, and so on). A file that contains subfiles is called a *container* file. Compressed 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 by using the *File Extraction API*. The extraction process is done repeatedly until all subfiles are extracted and exposed for conversion. After a subfile is extracted, you can use the HTML Export API to convert the file.

If a file is not a container, you should pass it directly to the HTML Export API for conversion without extraction.

See Use the File Extraction API, on page 53 for more information.

Convert Outlook Email without Using the Extraction API

Micro Focus strongly recommends that you convert all container files, including Microsoft Outlook files, by using the File Extraction API. However, you can convert Outlook email messages (MSG) directly by using the Export API and the MSG reader (msgsr).

> **NOTE:** The MSG reader only extracts the message body of an MSG file. Attachments are not extracted.

To convert MSG files by using the MSG reader, add the following to the `formats_e.ini` file (`TRUE` is case-sensitive):

```
[ContainerOptions]
bConvertMSG=TRUE
```

Set Conversion Options

Conversion options are parameters that determine the content, structure, and appearance of the HTML output. For example, you can specify the markup inserted at the beginning and end of specific HTML blocks, whether a heading is included in the table of contents, the output character set, or the resolution at which graphics are converted. The conversion options can be set either in the API or in the template files. Regardless of the method used to set the options, the values are ultimately passed to the API and used to populate the following data structures:

- KVHTMLTemplateEx, on page 225
- KVHTMLOptionsEx, on page 216
- KVHTMLHeadingInfo, on page 209
- KVHTMLTOCOptions, on page 230

The conversion options are described in HTML Export API Structures, on page 200.

Set Conversion Options by Using the API

Set conversion options by using any of the following functions:

- fpConvertStream(), on page 163
- KVHTMLConvertFile(), on page 185
- KVHTMLStartOOPSession(), on page 191

Set Conversion Options by Using the Template Files

HTML Export includes templates in the form of initialization files (`.ini`). The templates provide a quick and easy way to modify the conversion options without programming at the API level. However, the template files do not give you complete control of the conversion process. To control some features, you must use the API directly.

You can use a text editor to customize the template files. For example, to change the output character set from the default `KVCS_UNKNOWN` to `KVCS_SJIS` in the `default.ini` template, make the following change shown in bold:

```
[KVHTMLOptionsEx]
OutputCharSet=KVCS_SJIS
bUseDocumentColors=TRUE
```

To create valid HTML, a template file *must* define at least two structures: `KVHTMLTemplateEx` and `KVHTMLOptionsEx`.

> **NOTE:** If you enter markup in the template files that is not compliant with HTML standards, HTML Export inserts the markup into the output file unchanged. This might result in a malformed HTML file.

An application must then read the template file and write the data to the appropriate Export structures. In the sample program `htmlini`, a template file is supplied as a command-line argument (see htmlini, on page 123).

The characteristics of some of the template files are demonstrated in the *HTML Export Getting Started* page. The Getting Started page, named `htmstart.html`, is in the directory `install\htmlexport\docs`, where `install` is the path name of the Export installation directory. It compares the output generated using a set of sample documents and the template files. The source documents used in the page are in the directory `install\testdocs`.

Templates

The template files for the C API implementation are in the directory `install\htmlexport\programs\ini`, where `install` is the path name of the Export installation directory. The following templates are provided:

Template	Description
Arabic (`bidi_arabic.ini`)	• Based on the default template (`default.ini`). • The Arabic character set is defined in the template. • `<dir="rtl">` added to the Body tag to indicate that the text is read from right to left.
Cascading style sheet (`css_ex.ini`)	This template writes style sheet information to an external Cascading Style Sheet (CSS) file or reads the information from an existing CSS file. This makes the HTML output significantly smaller because the information is not stored within the output file. It also allows you to use the same style sheet for many conversions. See Use Style Sheets, on page 87 and Use Style Sheets with htmlini, on page 125 for more information on using an external CSS file.
Default (`default.ini`)	• Segments word processing documents, spreadsheets, and presentations into multiple files according to the document's heading levels. • Creates two frames. The table of contents (based on the source document's heading levels and page breaks) appears in the left frame. The document contents associated with the table of contents entry selected in the left frame appears in the right frame. • Inserts **Previous** and **Next** buttons at the end of each block. • Supports URLs. • Supports headers, footers, footnotes, and endnotes.

Template	Description
	• Converts graphics to JPEG with the original size preserved.
	• Converts presentation slides to HTML as individual JPEG files.
Hebrew (`bidi_hebrew.ini`)	• Based on the default template (`default.ini`).
	• The Hebrew character set is defined in the template.
	• `<dir="rtl">` added to the Body tag to indicate that text is read from right to left.
Low bandwidth (`lowband.ini`)	This template is useful when you need to provide information to a mobile workforce that might not always have access to fast connections.
	• Creates text-only HTML.
	• Suppresses the source document's embedded graphics.
Multiple files with three frames (`logotoc.ini`)	• Segments word processing documents, spreadsheets, and presentations into multiple files according to the document's heading levels.
	• Creates three frames. A corporate logo is displayed in the top left frame. The table of contents (based on source document heading levels and page breaks) appears in the bottom left frame, and the HTML files appear in the right frame.
	• Inserts **Previous** and **Next** links at the end of each block.
	• Supports URLs.
No frills (`nofrills.ini`)	• Creates a single HTML file.
	• Supports URLs.
	• Maintains the source document's fonts and styles.
	• Does not create a table of contents.
	• Does not list the source document's metadata.
PDF bookmarks in a frame (`pdfframe.ini`)	This template is optimized to display PDF bookmarks in a separate frame.
	• Segments a PDF file into two HTML files; one contains the table of contents (based on the bookmarks in the PDF file), and the other contains the document text.
	• Creates two frames. The table of contents appears in the left frame, and the document appears in the right frame.
	• Forces the output character set to UTF-8.
	• Does not insert **Back to Top**, **Previous**, or **Next** links.
	See Convert PDF Files, on page 92 for more information on generating a table of contents from bookmarks in a PDF file.
Single file with	This template is useful when you want to print the document.

Template	Description
)	• Creates a single HTML file.
	• Creates a table of contents at the top of the HTML document.
	• Uses worksheet names to create the table of contents entries for spreadsheets. If worksheet names do not exist in the source document, "Sheet1," "Sheet2," "Sheet3," and so on are used.
	• Uses slide titles to create the table of contents entries for presentations. If slide titles do not exist in the source document, "slide 1," "slide 2," "slide 3," and so on are used.
	• Lists all metadata (Title, Subject, Author, Comments, and so on).
	• Converts graphics to JPEG with the original resolution preserved.
	• Converts presentation slides to HTML as individual JPEG files.
Style mapping (`wordstyle.ini`)	This template demonstrates how to map paragraph and character styles in a word processing document to arbitrary markup (including CSS, XML, or HTML). Using style mapping, you can use external Cascading Style Sheet (CSS) files to define styles used in the HTML, alter the structure of a document, delete content, or replace content with a specified character (`redact`). See Map Styles, on page 84.
UNIX web server (`defunix.ini`)	• Based on the default template (`default.ini`). • Converts embedded graphics or presentations to either JPEG or HTML Export's Java target. See Display Vector Graphics on UNIX and Linux, on page 87.

Use the Export Demo Program

The easiest way to get started with Export is to become familiar with its capabilities through the Visual Basic sample program, Export Demo. The source code for the program is in the directory *install*\htmlexport\programs\ExportDemo, where *install* is the path name of the Export installation directory. Export Demo is for Windows only, and requires Internet Explorer 4.01 with Service Pack 1 or higher.

The output options that control the look of the output files are predefined in Export Demo and cannot be changed in the user interface. Export Demo uses a small sample of the options available in the Export API.

The Template Wizard sample program is an example of a Visual Basic program that does allow the user to control some of the output options with template files. See Template Wizard, on page 127. You can use the sample documents in the directory *install*\testdocs to experiment with converting different file formats.

To launch the sample program, select **Export Demo** from **Start** | **Programs** | **Autonomy** | **Export SDK** | **HTML Export**. The following dialog appears:

Export Demo: Launching
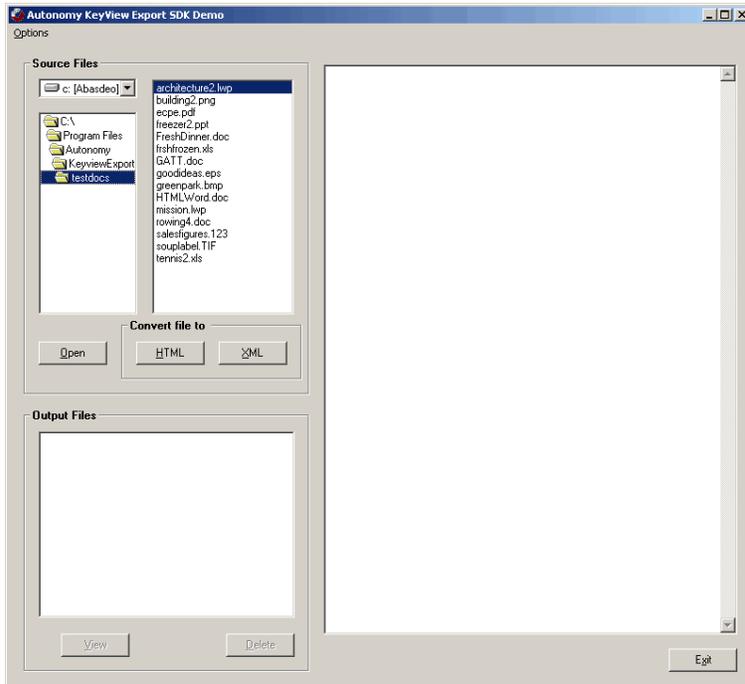
NOTE: XML conversion using XML Export is available in Export Demo if you have XML Export installed. If you do not have XML Export installed, the **XML** button is disabled.

Change Input/Output Directories

If HTML Export is installed in the default directory, the output and input directories are automatically set.

The default location for source files is the directory $install$\testdocs.

The default location for output files is the directory $install$\htmlexport\programs\tempout.

If HTML Export is installed in a directory other than the default, you are prompted to select an output and input directory when you first start Export Demo.

To change the default directories for the source and output files

1. Select **Options** | **Set Directories**. The following dialog appears:

   Export Demo: Setting Directories
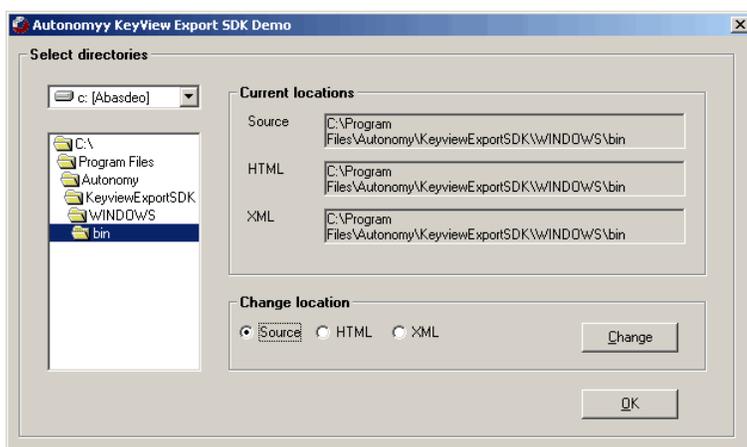
2. From the tree view, select the drive letter and directory for the source or output files.

3. In **Change Location**, select which files are stored in the directory, either **Source** or **HTML**.

4. Click **Change**. The **Current Locations** fields are updated with the new selection.

5. Follow the same procedure for the other file types.

Set Configuration Options

With HTML Export, you can configure options prior to the document conversion by using the `KVHTMLConfig()` function. Export Demo demonstrates this function, and allows you to:

- Specify a PDF reader.
- Specify whether rotated text in a PDF file is displayed in its original position or at the bottom of the page.

Convert PDF Files

In Export Demo, PDF documents can be converted in one of two ways:

- generate HTML output by using the basic PDF reader (`pdfsr`)
- generate a raster image for each page of the PDF file by using a graphic-based PDF reader (`kppdfrdr` or `kppdf2rdr`). See Convert PDF Files to Raster Images, on page 95.

Export Demo provides an option to select the type of reader you want to use to convert PDF documents. By default, the basic reader (`pdfsr`) is used to convert PDF documents.

To specify that the graphic-based reader be used to convert PDF documents

1. Ensure that Export Demo is not running.

2. Set the appropriate configuration file options. See Use a Graphic-Based Reader, on page 94.

3. Start the Export Demo program.

4. Select **Options** | **HTML Config** | **Set Hifi**.

> **NOTE: Note**: PDF documents can also be converted to a logical reading order by using the
> PDF reader `pdfsr`. This feature is demonstrated in the sample program `cnv2html`. See
> Convert PDF Files to a Logical Reading Order, on page 96.

Convert Rotated Text

In HTML Export, you can specify how rotated text is displayed in the HTML output. By default, rotated
text in a file is displayed in its original position, at the original font size, and at 0 degrees rotation.
Because the text is the original size, but might be displayed in a smaller space, the text might overlap
adjacent text in the HTML output. You use the text rotation configuration option to avoid this problem. If
this option is set, rotated text is displayed at the bottom of the page on which it appears. See Convert
Rotated Text, on page 101. Currently, this configuration option applies only to PDF files.

To specify that rotated text be displayed at the bottom of the page on which it appears, select **Options** |
HTML Config | **Set Text Rotate**.

Convert Files

To convert a single file

1. Select **Options** | **Convert** | **Single file**.
2. Select the document from the file list, and then click **HTML** in the **Convert file to** pane.

To convert files in a directory

1. Select **Options** | **Convert** | **Entire directory**.
2. Click **HTML** in the **Convert directory to** pane.

   To view a converted file, double-click the output file in the **Output Files** pane, or select the output
   file, and then click **View**. The converted file is displayed in the view pane:

Export Demo: Converting Files
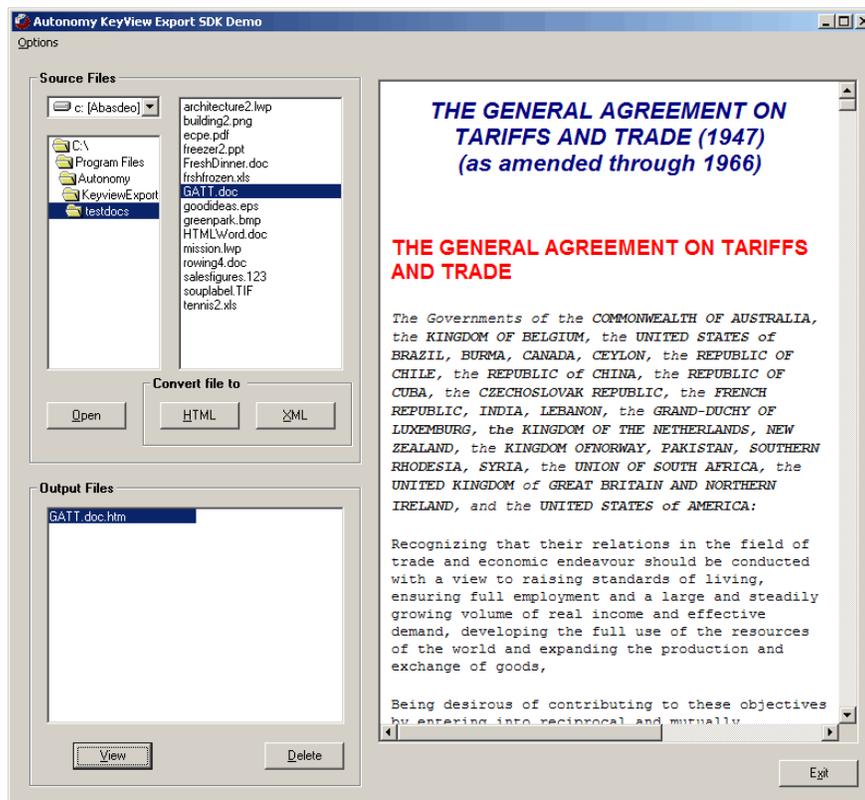
To view the original document, select the document from the file list, and then click **Open**. If you have an application on your system associated with the file, the file is displayed in that application.

To delete output files, select the file in the **Output Files** pane and click **Delete**.

Use the C-Language Implementation of the API

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

- File Extraction API Functions, on page 135—Open and extract subfiles in a container file. These functions also extract metadata and file format information, and control character set conversion on extraction.

- HTML Export API Functions, on page 161— Extract format information (metadata, character set, and format), create an input/output stream from a file, and open, convert, and close the stream.

- HTML Export API Callback Functions, on page 195—Controls the conversion while it is in progress.

Input/Output Operations

In the Export API, the source input and target output can be either a physical file accessed through a file path, or a *stream* created from a data source. A stream is a C structure that contains pointers to I/O functions similar in nature to their standard ANSI C counterparts. This structure is passed to Export

functions in place of the standard input source. The input stream is defined by the structure `KVInputStream` in `kvtypes.h`. The output stream is defined by the structure `KVOutputStream` in `kvtypes.h`. See KVInputStream, on page 201 and KVOutputStream, on page 202.

You can create an input stream either by using the `fpFileToInputStreamCreate()` function, or by using code similar to the example code in the `io_samp` sample program. You can create an output stream by using the `fpFileToOutputStreamCreate()` function. These functions assign C equivalent I/O functions to `fpOpen()`, `fpRead()`, `fpSeek()`, `fpTell()`, and `fpClose()`. See fpFileToInputStreamCreate(), on page 166 and fpFileToOutputStreamCreate(), on page 167.

Convert Files

To use the C-language implementation of the API

1. Develop the HTML markup and tokens to be assigned to the required members of a declared instance of `KVHTMLTemplateEx`.

   If you use markup in the structure that is not compliant with HTML standards, HTML Export inserts the markup into the output file unchanged. This might result in a malformed HTML file.

2. Declare instances of the following types and assign values to the members as required:

   ```
   KVHTMLTemplateEx
   KVHTMLOptionsEx
   KVHTMLHeadingInfo
   KVHTMLTOCOptions
   ```

   See HTML Export API Structures, on page 200 for more information.

3. Load the `KVHTML` library and obtain the `KVHTMLInterfaceEx` entry point by calling `KVHTMLGetInterfaceEx()`. See KVHTMLGetInterfaceEx(), on page 161.

4. Initialize an Export session by calling `fpInit()`. The function's return value, `pContext`, is passed as the first argument to all other Export functions. See fpInit(), on page 174.

5. Pass the context pointer from `fpInit()` and the address of a structure containing pointers to the File Extraction API functions in the call to `KVGetExtractInterface()`. See. KVGetExtractInterface(), on page 135.

6. If you are using streams for the input and output source, follow these steps; otherwise, proceed to Step 7:

   a. Create an input stream (`KVInputStream`) either by calling `fpFileToInputStreamCreate()`, or by using code similar to the example code in the `io_samp` sample program. fpFileToInputStreamCreate(), on page 166.

   b. Create an output stream (`KVOutputStream`) either by calling `fpFileToOutputStreamCreate()`, or by using code similar to the example code in the `io_samp` sample program. fpFileToOutputStreamCreate(), on page 167.

   c. Proceed to Step 7.

7. Declare the input stream or file name in the `KVOpenFileArg` structure. See KVOpenFileArg, on page 154.

8. Open the source file by calling `fpOpenFile()` and passing the `KVOpenFileArg` structure. This call defines the parameters necessary to open a file for extraction. See fpOpenFile(), on page 143.

9. Determine whether the source file is a container file (contains subfiles) by calling `fpGetMainFileInfo()`. See fpGetMainFileInfo(), on page 139.

10. If the call to `fpGetMainFileInfo()` determined the source file is a container file, proceed to Step 11; otherwise, proceed to Step 14.

11. Determine whether the subfile is itself a container (contains subfiles) by calling `fpGetSubFileInfo()`. See fpGetSubFileInfo(), on page 140.

12. Extract the subfile by calling `fpExtractSubFile()`. See fpExtractSubFile(), on page 136.

13. If the call to `fpGetSubFileInfo()` determined the subfile is a container file, repeat Step 6 through Step 12 until all subfiles are extracted; otherwise, proceed to Step 14.

14. Setup an out-of-process session by calling `KVHTMLStartOOPSession()`. See KVHTMLStartOOPSession(), on page 191.

15. Convert the input and generate the output files by calling `KVHTMLConvertFile()` or `fpConvertStream()`. The structures `KVHTMLTemplate`, `KVHTMLOptions`, and `KVHTMLTOCOptions` are defined in the call to `KVHTMLStartOOPSession()`, and should be `NULL` in the conversion call. A conversion function can be called only once in a single out-of-process session. See KVHTMLConvertFile(), on page 185, and fpConvertStream(), on page 163.

16. If you are using callbacks, they are called while the conversion process is underway. If required, you can specify alternate paths and file names for output files, including using the table of content entries for the file names. See HTML Export API Callback Functions, on page 195.

17. If you are converting additional files, terminate the out-of-process session by calling `KVHTMLEndOOPSession()` and setting the Boolean to **TRUE**. The Servant ends the current conversion session, and releases the source data and session resources.

    If you are not converting additional files, terminate the out-of-process session *and* the Servant process by calling `KVHTMLEndOOPSession()` and setting the Boolean to **FALSE**. See KVHTMLEndOOPSession(), on page 187.

18. Close the file by calling `fpCloseFile()`. See fpCloseFile(), on page 136.

19. If you used streams, free the memory allocated for the input stream and output stream by calling the functions `fpFileToInputSreamFree()` and `fpFileToOutputStreamFree()`. See fpFileToInputStreamFree(), on page 167 and fpFileToOutputStreamFree(), on page 168.

20. Repeat Step 6 through Step 18 for additional source files.

21. Shutdown the Export session by calling `fpShutDown()`. See fpShutDown(), on page 176.

Multithreaded Conversions

To ensure that multithreaded conversions are thread-safe, you must create a unique context pointer for every thread by initializing the Export session with `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. Creating a context pointer for every thread does not affect performance because the context pointer uses minimal resources.

For example, your code should have the following logic for one thread:

```
fpInit()
   KVGetExtractInterface()
   fpFileToInputStreamCreate()
   fpFileToOutputStreamCreate()
      fpOpenFile()
      fpGetMainFileInfo()          /* container file */
      fpGetSubFileInfo()
      fpExtractSubFile
      KVHTMLStartOOPSession()
      fpConvertStream()
      KVHTMLEndOOPSession(bKeepServantAlive TRUE)
      fpCloseFile()
   fpFileToInputSreamFree()
   fpFileToOutputStreamFree()
      set input/output file
      fpOpenFile()
      fpGetMainFileInfo()             /* not a container file */
      KVHTMLStartOOPSession()
      KVHTMLConvertFile()
      KVHTMLEndOOPSession(bKeepServantAlive TRUE)
      fpCloseFile()
   ...
fpShutdown()
```

Use the COM Implementation of the API

The COM implementation of HTML Export is only applicable to Win32 environments. It is supported in both out-of-process (`htmserv.exe`) and in-process (`htmserv.dll`) versions. Programming with either interface is identical. The out-of-process version provides a more robust HTML conversion, but at the expense of making out-of-process calls. To use either version of the COM implementation, you must register the COM component. Both components support self-registration and self-unregistration. You can only register one COM component.

To use the COM Automation Server

1. Register the COM server by using one of the following methods:

   - To register the out-of-process COM server, run:

     install*OS*\bin\htmserv.exe -RegServer

   - To register the in-process COM server, run:

     regsvr32.exe *install**OS*\bin\htmserv.dll

   - To unregister the out-of-process COM server, run:

     install*OS*\bin\htmserv.exe -UnRegServer

and then reboot the machine.

- To unregister the in-process COM server, run:

  `regsvr32.exe -u install\OS\bin\htmserv.dll`

  where `install` is the path name of the Export installation directory.

  The `regsvr32.exe` is a Microsoft Windows program used to register in-process COM objects and is stored in the `install\OS\bin` directory.

2. Confirm the following entry is in the Windows registry:

   `\\HKEY_CLASS_ROOT\VerityHtmServ.Application`

3. Create an instance of the COM object. See the `comsamp` sample for an example.

4. Specify the source file by using the `pszInputFile` property.

5. Specify the location of the HTML Export libraries by using the `pszKeyViewDir` property.

6. Use the properties and methods described in COM Interface Methods and Events, on page 255 and COM Interface Properties, on page 260.

Sample Implementation

The following code, which is found in the sample Visual Basic program named `comsamp`, demonstrates how to use the properties, methods, and events of the ActiveX Controls from within Visual Basic.

Define the htmserv Object

The sample code will not function unless you first define the `htmserv` object in Visual Basic.

To define the object for Visual Basic 6

1. Select **References...** from the **Project** menu.

2. Search available references for "HTML Export COM Server Library," and select it.

3. If the HTML Export COM Server Library is unavailable, follow the registering instructions in Use the COM Implementation of the API, on the previous page.

Sample Code

1. Declare the variable `MyRef` as an instance of `htmserv` (the HTML Export COM server):

   `Dim WithEvents MyRef As htmserv`

2. Specify the source file by setting the input file property `pszInputFile`:

   ```
   Private Sub Convert_Click()
   MyRef.pszInputFile = File1.Path & "\" & File1.FileName
   ```

3. Define the `GetSummaryInfo` method and metadata properties:

```
Dim nTotal As Long
Dim nValid As Long
Dim nType As Long
Dim nVal As Long
Dim szVal As String
Dim szUserVal As String
...
On Error GoTo Handler
Call MyRef.GetSummaryInfo(3, nTotal, nValid, nType, nVal, szVal, szUserVal)
MsgBox szUserVal & " = " & szVal
```

4. Call the `ConvertFileToFile` method:

```
Convert:
    nRet = MyRef.ConvertFileToFile("c:\temp\temp.htm")
    WebBrowser1.Navigate ("c:\temp\temp.htm")
    Exit Sub
```

5. The `comsamp` sample program specifies the default directory for source files as the `C:\Program Files\Autonomy\KeyviewExportSDK\testdocs` directory, and the directory in which binaries are stored as the `C:\Program Files\Autonomy\KeyviewExportSDK\>OS\bin` directory, where *OS* is the name of the operating system. To change these directories to match your installation, set the `Path` property to the location of the `testdocs` directory, and set the `pszKeyViewDir` property to the location of the HTML Export binary files:

```
Private Sub Form_Load()
    Set MyRef = New htmserv
    Dir1.Path = "C:\myinstall\testdocs"
    MyRef.pszKeyViewDir = "C:\myinstall\bin"
End Sub
```

6. Implement the `Continue` event that is called by HTML Export:

```
Private Function MyRef_Continue(ByVal PercentDone As Long) As Long
    ProgressBar1.Value = PercentDone
    MyRef_Continue = True
    ProgressBar1.Refresh
End Function
```

See for more information.

7. Implement the `UserCallback` event:

```
Private Function MyRef_UserCallback(ByVal szUserString As String) As String
    MsgBox (szUserString)
    MyRef_UserCallback = "Output this text to HTML"
End Function
```

See for more information.

The code below demonstrates an alternate way to initiate an instance of the COM server:

```
Dim HTM As Object
Set HTM = CreateObject("VerityHtmServ.Application")
```

where `HTM` is the COM Automation Server object.

Part II: Use the Export API

This explains how to perform some basic tasks by using the File Extraction and Export APIs, and describes the sample programs. It contains the following chapters:

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 convert 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 284 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.
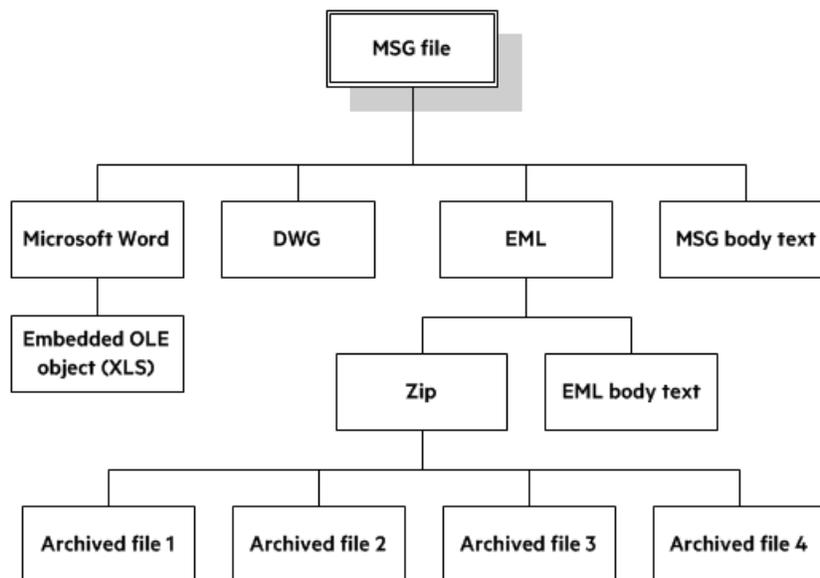
> **NOTE:** The parent MSG file contains four first-level children. The body text of a message file, although not a standalone file in the container, is considered a child of the parent file.

Extract Subfiles

To convert 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 conversion. After a subfile is extracted, you can call Export API functions to convert the file.

If you want to convert a container file and its subfiles to a single file, you must extract all files from the container, convert the files, and then append each converted 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, convert 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, convert the file.

Sanitize Absolute Paths

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

To set the path, you might use the path in the container file that you are extracting from, as returned from the function fpGetSubFileInfo(), on page 140. However, if the path is an absolute path, the file could be created outside the directory you have chosen as the extract directory. Your application might then contain a vulnerability that could be exploited to write files to unexpected locations in the file system. This section discusses some KeyView features that can help you secure your application by sanitizing paths.

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

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

The following table shows the effect on some example paths.

Requested path	Path of extracted file (not sanitized)	Path of extracted file (sanitized)
`file.txt`	*extractDir*/file.txt	*extractDir*/file.txt
`dir/file.txt`	*extractDir*/dir/file.txt	*extractDir*/dir/file.txt
`../file.txt`	*extractDir*/file.txt	*extractDir*/file.txt
`/dir/file.txt`	/dir/file.txt	*extractDir*/dir/file.txt

To sanitize absolute paths

- In the KVExtractSubFileArg struct that you pass in to fpExtractSubFile, set the flag `KVExtractionFlag_SanitizeAbsolutePaths`. When KeyView sanitizes a path and the resulting directory does not exist, extraction fails unless you instruct KeyView to create the directory, so you might also want to set the flag `KVExtractionFlag_CreateDir`. You can find the path that a file was actually extracted to from the KVSubFileExtractInfo structure.

To sanitize absolute paths (through configuration)

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

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

Extract Images

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

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

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

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

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

Recreate a File's Hierarchy

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

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

Create a Root Node

Because of their structure, some container files do not contain a subfile or folder which acts as a root directory on which the hierarchy can be based. For example, subfiles in a Zip archive can be extracted, but none of the subfiles represent the root of the hierarchy. In this case, you must create an artificial *root node* at the top of the file hierarchy as a point of reference for each child, and ultimately to recreate the relationships. This artificial root node is an internal object, and is extracted to disk as a directory called `root`. Its index number is 0.

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

Recreate a File's Hierarchy—Example

For example, you might extract a PST file that contains seven subfiles with a root node enabled. The
call to `fpGetMainFileInfo()` returns the number of subfiles as eight (seven subfiles and one root
node). The following diagram shows the structure and the available hierarchy information after the
subfiles are extracted:
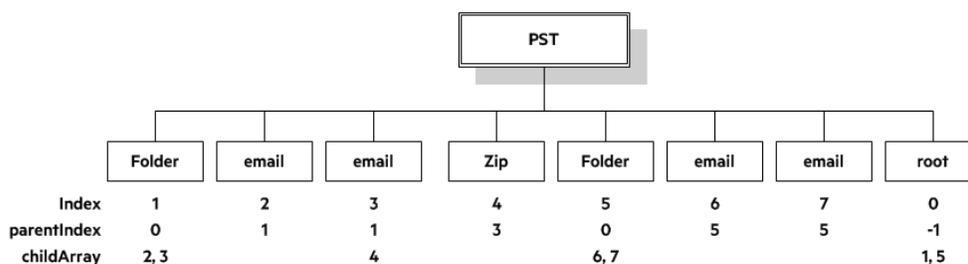
The `parentIndex` specifies the index number of a subfile's parent. The `childArray` specifies an array
of a subfile's children. With this information, you can recreate the hierarchy shown in the following
diagram.
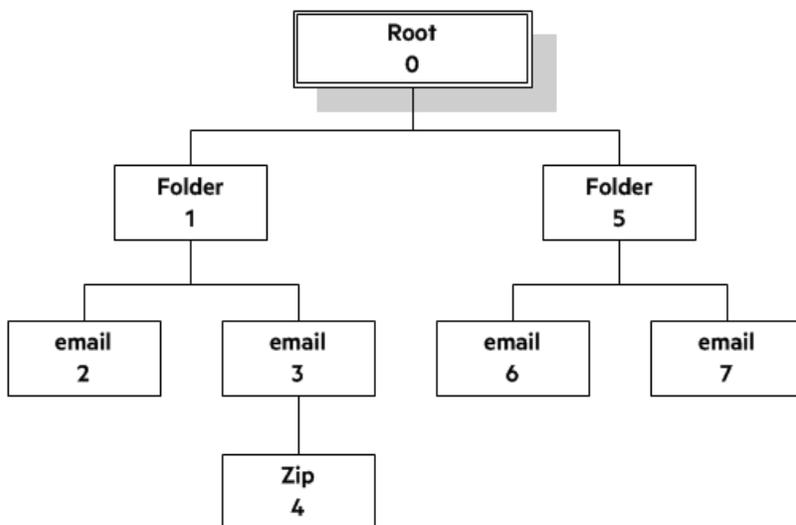
Extract Mail Metadata

You can extract metadata, such as subject, sender, and recipient, from subfiles of mail formats, by
calling the fpGetSubFileMetaData() function. You can extract a predefined set of common metadata

fields, a list of metadata fields by their names or MAPI properties, or, for some subfile types, all the metadata in the file.

Default Metadata Set

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

Default Mail Metadata List

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

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

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

> **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 62.

Extract the Default Metadata Set

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

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

metaArg.index = subFileIndex;
```

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

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

Extract All Metadata

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

Microsoft Outlook (MSG) Metadata

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

MSG-specific Metadata List

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

MSG-specific Metadata List, continued

Field Name (string to specify)	Description
Received	The date and time a message was delivered. This value is displayed in the **Received** field in the message's **Properties** dialog in Outlook.
Sender	The name and email address of the message sender. This value is a concatenation of two MAPI properties in the following format: "PR_SENDER_NAME" <PR_SENDER_EMAIL_ADDRESS> The Sender value might be the same as or different than the default metadata From value (see Default Metadata Set, on page 58), depending on which MAPI properties exist in the MSG file.
Sensitivity	The value indicating the message sender's opinion of the sensitivity of a message. For example, Personal, Private, or Confidential. This value is displayed in the **Sensitivity** field in the message's **Properties** dialog in Outlook.
TransportMsgHeaders	Transport-specific message envelope information. This value corresponds to the MAPI property PR_TRANSPORT_MESSAGE_HEADERS.
StartDate	An appointment start date. This value corresponds to the PR_START_DATE MAPI property.
EndDate	An appointment end date. This value corresponds to the PR_END_DATE MAPI property.

Extract MSG-Specific Metadata

To extract specific metadata fields from an MSG file, call the fpGetSubFileMetaData() function, and pass the field name defined in Default Metadata Set, on page 58 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_DOUBLE  PT_STRING8

PT_I4      PT_FLOAT   PT_TSTRING

PT_BINARY  PT_LONG    PT_SYSTIME

PT_BOOLEAN PT_SHORT   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, on the previous page.) If the Outlook file contains a non-mail attachment, the attachment is extracted in its native format to the same directory as the message text file. If the Outlook file contains a mail attachment, the complete attachment (including message text and attachments), the message text file, and any non-mail attachments are extracted to the same directory as the main message.

> **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 Micro Focus. See Update License Information, on page 23 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) readers are an advanced feature and are sold and licensed separately. To enable these readers in a KeyView SDK, you must obtain an appropriate license key from Micro Focus. For information about adding a new license key to an existing installation, see Update License Information, on page 23.

Choose the Reader to use for PST Files

KeyView provides several ways of processing PST files:

- Indirectly, using the Microsoft Messaging Application Programming Interface (MAPI). MAPI is a Microsoft interface that enables different applications to exchange messages and attachments with each other. MAPI allows KeyView to open a PST file, traverse the folders, and extract items. The `pstsr` reader uses MAPI, but works only on Windows and requires that Microsoft Outlook is installed.

- Directly, without relying on the Microsoft interface to the PST format. Accessing the file directly does not require Microsoft Outlook. The `pstxsr` reader is available for Windows (32-bit and 64-bit) and Linux (64-bit only). The `pstnsr` reader is an alternative native reader, for the platforms not supported by `pstxsr`.

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

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

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

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

297=**pstxsr**

> **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.

> **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

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

- Outlook 97 or later PST files

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

- Outlook 2002 or later clients

  > **NOTE:**
  > You must install an edition of Microsoft Outlook (32-bit or 64-bit) that matches the KeyView software. For example, if you use 32-bit KeyView, install 32-bit Outlook. If you use 64-bit KeyView, install 64-bit Outlook.
  >
  > If the editions do not match, KeyView returns `Error 32: KVError_PSTAccessFailed` and an error message from Microsoft Office Outlook is displayed: `Either there is a no default mail client or the current mail client cannot fulfill the messaging request. Please run Microsoft Outlook and set it as the default mail client.`

MAPI Attachment Methods

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

`ATTACH_BY_VALUE`

`ATTACH_EMBEDDED_MSG`

`ATTACH_OLE`

`ATTACH_BY_REFERENCE`

`ATTACH_BY_REF_ONLY`

`ATTACH_BY_REF_RESOLVE`

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

To extract "attach by reference" attachments

1. Determine whether the attachment uses an `ATTACH_BY_REFERENCE`, `ATTACH_BY_REF_ONLY`, or `ATTACH_BY_REF_RESOLVE` method by retrieving the MAPI property `PR_ATTACH_METHOD`.

2. If the attachment uses one of the "attach by reference" methods, get the fully qualified path to the attachment by retrieving the MAPI properties `PR_ATTACH_LONG_PATHNAME` or `PR_ATTACH_PATHNAME`.

3. You can then either copy the files from their original location to the path where the PST file is extracted, or use the Export API functions to convert 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 readers (`pstxsr` and `pstnsr`) return the error message `KVERR_PasswordProtected` if a PST file is encrypted with high encryption.

Detect PST Files While the Outlook Client is Running

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

- Close Outlook when using the Extraction API.

- Detect PST files by extension only and bypass the format detection module. To enable this option, add the following lines to the `formats_e.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 64.

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_e.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_e.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 64.)

> **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 Micro Focus. See Update License Information, on page 23 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 converting 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 convert 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 conversion:

Windows

1. Install the Lotus Notes client or Lotus Domino server. You do not need to configure the client or server.
2. Make sure that the `notes.ini` file is in the proper location.

- If Lotus Notes is installed, the file should appear in the *install*\lotus\notes directory, where *install* is the installation directory.

- If only Lotus Domino is installed, the file should appear in the *install*\lotus\domino directory, where *install* is the installation directory.

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

  ```
  [Notes]
  ```

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

Solaris

1. Install Lotus Domino server. You do not need to configure the server.

2. Make sure that the notes.ini file is in the *install*/lotus/notes/latest/sunspa directory, where *install* is the directory where Lotus Notes is installed. If the file does not exist, create an ASCII file named notes.ini, and add the following text:

   ```
   [Notes]
   ```

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

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

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

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

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

AIX 5.x

1. Install the bos.iocp.rte file set if it is not already installed, and reboot the machine. See the Lotus Domino server documentation for more information.

2. Install Lotus Domino server. You do not need to configure the server.

3. Make sure that the notes.ini file is in the *install*/lotus/notes/latest/ibmpow directory, where *install* is the directory where Lotus Notes is installed. If the file does not exist, create an ASCII file named notes.ini, and add the following text:

   ```
   [Notes]
   ```

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

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

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

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

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

Linux

1. Install Lotus Domino server. You do not need to configure the server.

2. Make sure that the `notes.ini` file is in the *install*/lotus/notes/latest/linux directory, where *install* is the directory where Lotus Notes is installed. If the file does not exist, create an ASCII file named `notes.ini`, and add the following text:

   ```
   [Notes]
   ```

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

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

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

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

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

Open Secured NSF Files

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

Format Note Subfiles

The KeyView NSF reader uses XML templates to format note subfiles. You can customize the templates to approximate the look and feel of the original notes as closely as possible. For more information, see Extract and Format Lotus Notes Subfiles, on page 377.

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_e.ini file:

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

Extract Embedded OLE Objects

Embedded OLE objects can be converted in two ways:

- Using the File Extraction API, the OLE object is first extracted from the main file and saved to disk. It can then be converted by making a separate conversion call.

- Using the HTML Export API, the main file is converted to HTML and the OLE object is converted to a graphics file that is referenced in the HTML file .

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 54). However, some ZIP files use password protection, in which case you must use a different method to enter the required credentials.

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 HTML Export API

This section describes how to perform some basic tasks by using the HTML Export API.

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 57.

> **NOTE: Note**: KeyView can extract metadata from a document only 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 284 lists the file formats for which metadata can be extracted. KeyView does not generate metadata automatically from the document contents.

Extract Metadata by Using the API

You can extract the metadata at the API level. The API extracts all valid metadata fields that exist in the file.

Use the C API

To extract metadata by using the C API

1. Declare a pointer to the `KVSummaryInfoEx` structure. KVSummaryInfoEx, on page 206.

2. Call the `fpGetSummaryInfo()` function. See fpGetSummaryInfo(), on page 173.

Use the COM interface

To extract metadata by using the COM interface, call the `GetSummaryInfo()` method. See GetSummaryInfo, on page 257.

Extract Metadata by Using a Template File

When using a template file, KeyView recognizes two types of metadata: *standard* and *non-standard*. Standard metadata includes fields, such as Title, Author, and Subject. The standard fields are enumerated from 1 to 41 in `KVSumType` in the header file `kvtypes.h`. Non-standard metadata includes any field not listed from 1 to 41 in `KVSumType`, such as user-defined fields (for example, custom property fields in Microsoft Word documents), or fields that are unique to a particular file type (for example, "Artist" or "Genre" fields in MP3 files). Enumerated types 42 and greater are reserved for non-standard metadata.

To extract metadata by using a template file

1. Insert metadata tokens in a member of the `KVHTMLTemplateEx` structure in the template file. This defines the point at which the metadata appears in the HTML output.

2. If you are using the `$USERSUMMARY` or `$SUMMARY` token, define the `szUserSummary` member of the `KVHTMLTemplateEx` structure in the template file. This determines the markup and tokens generated when these metadata tokens are processed.

3. In your application, read the template file and write the data to the `KVHTMLTemplateEx` structure.

   See htmlini, on page 123.

The following metadata tokens can be used in the template files:

Token	Description
$SUMMARY*NN*	Inserts the data from a *specified* metadata field. *NN* is a number from 00 through 42 enumerated in `KVSumType` in `kvtypes.h`.
$SUMMARY	Inserts the data from valid metadata fields in the range of 0 to 33 using the markup provided in `pszUserSummary`.
$USERSUMMARY	Inserts the data from *every* valid non-standard metadata field using the markup provided in `pszUserSummary`.

Token	Description
$CONTENT	Inserts the content of the metadata field specified by the $NAME token.
$NAME	Inserts the name of a the metadata field, such as "Title," "Author," or "Subject."

Depending on the markup in szUserSummary, the extracted metadata might not appear in the browser when the HTML file is displayed, but might appear in the output file. Most of the KeyView-supplied template files extract standard metadata from a document, and include it in the output HTML. However, they do not display the metadata in a browser.

Examples

$SUMMARY*NN*

The following markup displays the contents of the "Title" field at the top of the main HTML file:

szMainTop=$SUMMARY01

In KVSumType, 01 is the enumerated value for the "Title" metadata field.

$SUMMARY

The following markup extracts all standard fields, and includes them in the first heading level 1 HTML block:

szFirstH1Start=$SUMMARY

szUserSummary=<meta name="$NAME" content="$CONTENT" />

This example extracts the field name ($NAME) and field content ($CONTENT) for standard metadata from a document, and includes it at the beginning of the first Heading level 1 HTML block. However, it does not display the metadata in the browser. The HTML output might look like this:

```
<meta name="CodePage" content="1252" />
<meta name="Title" content="My design document" />
<meta name="Subject" content="design specifications" />
<meta name="Author" content="John Doe" />
<meta name="Keywords" content="" />
<meta name="Comments" content="" />
<meta name="Template" content="Normal.dot" />
<meta name="LastAuthor" content="lchapman" />
<meta name="RevNumber" content="6" />
<meta name="EditTime" content="01/01/1601, 0:08" />
<meta name="LastPrinted" content="14/01/2002, 14:06" />
<meta name="Create_DTM" content="27/08/2003, 10:31" />
<meta name="LastSave_DTM" content="29/08/2003, 14:07" />
<meta name="PageCount" content="1" />
<meta name="WordCount" content="4062" />
<meta name="CharCount" content="23159" />
<meta name="AppName" content="Microsoft Word 9.0" />
```

```
<meta name="Security" content="0" />
<meta name="Category" content="software" />
<meta name="LineCount" content="192" />
<meta name="ParCount" content="46" />
<meta name="ScaleCrop" content="FALSE" />
<meta name="Manager" content="" />
<meta name="Company" content="Autonomy" />
```

To display the metadata in a browser, use the following markup in `szUserSummary`:

```
<hr>name="$NAME" content="$CONTENT" <br>/>
```

$USERSUMMARY

The following markup extracts non-standard fields, and includes them at the bottom of the main HTML file:

```
szMainBottom=$USERSUMMARY

szUserSummary=<meta name="$NAME" content="$CONTENT" />
```

This example extracts the field name (`$NAME`) and field content (`$CONTENT`) for non-standard metadata from a document, and includes it at the bottom of the main HTML file. However, it does not display the metadata in the browser. The HTML output might look like this:

```
meta name="Telephone number" content="444-111-2222"
meta name="Recorded date" content="07/03/2003, 23:00"
meta name="Source" content="TRUE"
meta name="my property" content="reserved"
```

To display the metadata in a browser, use the following markup in `szUserSummary`:

```
<hr>name="$NAME" content="$CONTENT" <br>/>
```

Extract File Format Information

Export can detect a file's format, and report the information to the API, which in turn reports the information to the developer's application. This feature enables you to apply customized conversion settings based on a file's format. See File Format Detection, on page 393 for more information on format detection.

Use the C API

To extract file format information by using the C API

1. Declare a pointer to the `KVStreamInfo` data structure. KVStreamInfo, on page 203.

2. Call the `fpGetStreamInfo()` function. fpGetStreamInfo(), on page 172.

Use the COM interface

To extract file format information by using the COM interface, call the `GetFileInfo()` method. See GetFileInfo, on page 256.

Convert Character Sets

Export allows you to control the character set of both the input and the output text. This is accomplished by either

- setting the source and/or target character set in the API, or

- basing the input/output on the character set of the document (if the document character set is stored in the document and can be determined by the document reader).

The character sets are enumerated in `KVCharSet` of `kvtypes.h`. Not all character sets can be used to specify the target character set. See Code Character Sets, on page 371 for a list of character sets that can be used as a target character set.

Determine the Character Set of the Output Text

To determine the output character set of a converted document, Export considers the following:

- Whether the reader can extract the character set from the document. This depends on whether the file format can provide character set information and whether the document actually contains character set information.

  The section Supported Formats, on page 284 indicates the file formats for which character set information can be extracted. If character set information cannot be determined for your document type, you must set the source, the target character set, or both, in the API.

- Whether a source character set is set in the API.

  > **NOTE:** To set the source character set, you must specify a character set *and* set the `bForceSrcCharSet` member of the `KVHTMLOptionsEx` structure to **TRUE**.

- Whether a target character set is set in the API.

  > **NOTE:** To set the target character set, you must specify a character set *and* set the `bForceOutputCharSet` member of the `KVHTMLOptionsEx` structure to **TRUE**.

Guidelines for Character Set Conversion

The following diagram shows how the output character set is determined when the document character set can be determined:

Document Character Set Can Be Determined
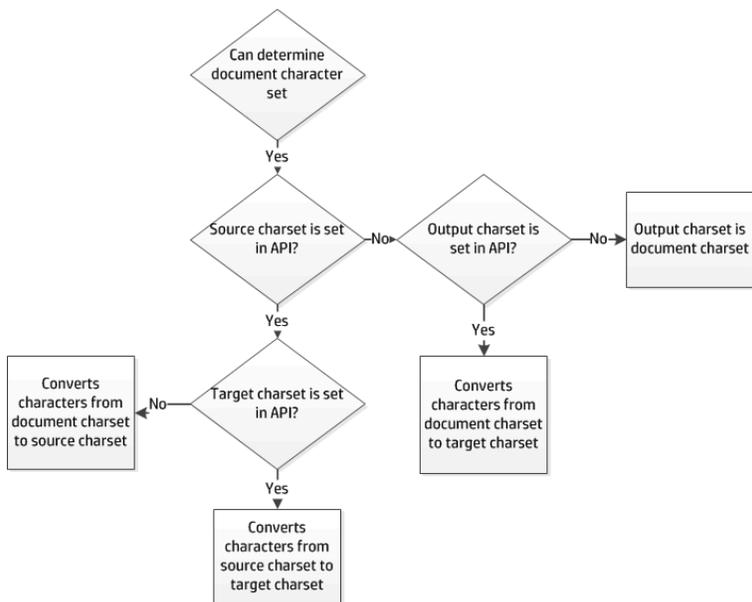
The following diagram shows how the output character set is determined when the document character set cannot be determined:

Document Character Set Cannot Be Determined
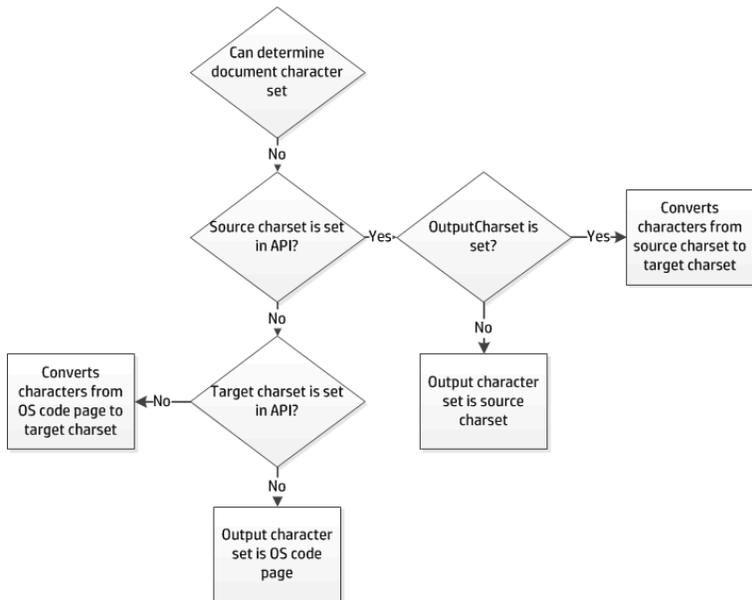
Examples of Character Set Conversion

The examples below demonstrate possible configurations for mapping character sets and the expected output for each scenario.

Document Character Set Can be Determined

For the example in the following table, the document is an RTF file. The section Word Processing Formats, on page 308 indicates that the document character set *can* be obtained from this file type. The document character set is Traditional Chinese (BIG5).

Document character set can be determined

Source charset set	Target charset set	Output charset
KVCS_GB	KVCS_UTF8	KVCS_UTF8 Converts GB (Simplified Chinese) to UTF-8. The output character set is the target character set specified in the API.
KVCS_GB	--	KVCS_GB Converts BIG5 to GB (Simplified Chinese). The output character set is the source character set specified in the API.
--	KVCS_UTF8	KVCS_UTF8 Converts BIG5 to UTF-8. The output character set is the target character set specified in the API.
--	--	KVCS_BIG5 The output character set is the document character set. No conversion.

Document Character Set Cannot be Determined

For the example in the following table, the document is an ASCII file. The section Word Processing Formats, on page 308 indicates that the document character set *cannot* be obtained from this file type. The document character set is KVCS_1251.

Document character set cannot be determined

Source charset set	Target charset set	Output charset
KVCS_1252	KVCS_ UTF8	KVCS_UTF8 Converts KVCS_1252 to KVCS_UTF8. The output character set is the target character set specified in the API.
KVCS_1252	KVCS_ UNKNOWN	KVCS_1252 The output character set is the source character set specified in the API because KVCS_UNKNOWN cannot be used. No conversion.

Document character set cannot be determined, continued

Source charset set	Target charset set	Output charset
KVCS_1252	--	KVCS_1252 The output character set is the source character set specified in the API. No conversion.
--	KVCS_1252	KVCS_1252 Converts OS code page to `KVCS_1252`. The output character set is the target character set specified in the API.
--	--	The output character set is OS code page. No conversion.

Set the Character Set During Conversion

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

To specify the source character set for documents from which the document character set cannot be obtained by the reader

1. Set the `eSrcCharSet` member of the `KVHTMLOptionsEx` structure to one of the character sets enumerated in `KVCharSet` in `kvtypes.h`. See KVHTMLOptionsEx, on page 216.

2. Set the `bForceSrcCharSet` member of the `KVHTMLOptionsEx` structure to **TRUE**. SeeKVHTMLOptionsEx, on page 216.

To specify the target character set

1. Set the `OutputCharSet` member of the `KVHTMLOptionsEx` structure to one of the character sets enumerated in `KVCharSet` in `kvtypes.h`. KVHTMLOptionsEx, on page 216.

2. Set the `bForceOutputCharSet` member of the `KVHTMLOptionsEx` structure to **TRUE**. See KVHTMLOptionsEx, on page 216.

Set the Character Set During File Extraction from a Container

You can convert the character set of a container subfile at the time the subfile is extracted from the container and before it is converted to HTML. This is most often used to set the output character set of a mail message's body text. See Use the HTML Export API, on page 76.

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`. See fpExtractSubFile(), on page 136.

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`.

Map Styles

Export can map paragraph and character styles in any word processing format that contains styles (such as Microsoft Word, RTF, or Folio Flat File) to user-defined markup. This feature is useful for shaping the look of the HTML output, or for generating user-defined metadata (including using XML tags) for indexing, searching, and navigation. With this feature, you can redact (hide) text in the source document, delete content, or change the overall structure of the output. You can also embed style sheet styles in the output defined in the HTML.

To enable style mapping, you must indicate which paragraph and/or character styles are to be mapped, and define the starting and ending markup to be included in the HTML output.

For example, if the source Microsoft Word document contains the character style "Recipe," and the content of the style in Microsoft Word is "Brownies," you can specify that the starting markup be `<recipe>` and the ending markup `</recipe>`. This would result in the output HTML containing: `<recipe>`Brownies`</recipe>`.

You can also use style mapping to control the look of the HTML output either by using a Cascading Style Sheet (CSS) or by defining the style directly in the starting markup. For example, if a Word document contains the paragraph style "Colorful", you can have markup of the form `<p><div class="rainbow">` inserted at the front of the paragraph and markup of the form `</div></p>` inserted at the end of the paragraph. "Rainbow" is a CSS style defined in an externally provided CSS file referenced at the top of the HTML output.

Style mapping is enabled in the `wordstyle.ini` template file. The *HTML Export Getting Started* page demonstrates the output resulting from a conversion with `wordstyle.ini`. The Getting Started page, named `htmstart.html`, is in the directory *install*`\htmlexport\docs`, where *install* is the path name of the Export installation directory. The source documents used in the page are in the directory *install*`\testdocs`.

> **NOTE:** When the user-defined markup in `KVStyle` conflicts with other markup generated by Export, the user-defined markup takes precedence.

Use the C API

To map styles by using the C API

1. Define the `KVStyle` structure. See KVStyle , on page 205. The information in this structure includes:

   - the markup to be added to the beginning and end of a paragraph or character style.

   - the name of the word processing style (for example, "Heading 1") to which style mapping applies. Style names are case sensitive.

   - the flag which defines instructions on how to process the content associated with a paragraph or character style. The flags are defined in `kvtypes.h` and described in Flags for Defining Styles, on page 86.

2. Call the `fpSetStyleMapping()` function. See fpSetStyleMapping(), on page 175.

Use a Template file

To map styles by using a template file

1. Use the `KVStyle` parameter to specify how many styles are being mapped. For example, if there are nine mapped heading levels, add the following:

```
[KVStyle]
NumStyles=9
```

2. For each style, there must be a `[StyleX]` entry that contains the markup that appears at the start and end of the defined style. For example, in the `wordstyle.ini` sample file, the first heading level is defined as follows:

```
[Style1]
StyleName=Colorful
MarkUpStart=<div class="colorful">
MarkUpEnd=<!-- end of colorful --></div>
```

These values are used in `StyleName`, `MarkUpStart`, and `MarkUpEnd` in the `KVStyle` structure. See KVStyle , on page 205.

3. For each style, define the flag that applies. Flags define instructions on how to process the content associated with a paragraph or character style. They are defined in `kvtypes.h` and described in Flags for Defining Styles, on the next page. This value is used in `dwflags` of the `KVStyle` structure. See KVStyle , on page 205. The value associated with each flag is a hexadecimal number. You can set an option by either entering the converted decimal value or entering the flag's text.

```
Flags=0
```

A finished entry in a template file could look like this:

```
[KVStyle]
NumStyles=3

[Style1]
StyleName=Colorful
MarkUpStart=<div class="Colorful">
MarkUpEnd=<!-- End of Colorful --></div>
Flags=0

[Style2]
StyleName=RedactPara
MarkUpStart=<div class="RedactPara">
MarkUpEnd=<!-- End of RedactPara --></div>
Flags=2048

[Style3]
StyleName=Code
MarkUpStart=<pre>
```

```
MarkUpEnd=<!-- End of Code --></pre>
Flags=KVSTYLE_PRE
```

Use the COM interface

To map styles by using the COM interface

1. Call the `AddStyleMapping()`, `GetStyleMapping()`, and `RemoveStyleMapping()` methods. See AddStyleMapping, on page 255, GetStyleMapping, on page 256, and RemoveStyleMapping, on page 258.

2. Define the `NumStyles` property. See NumStyles, on page 270.

Flags for Defining Styles

Flag	Description
`KVSTYLE_PRE`	The `KVSTYLE_PRE` flag specifies that white space should be preserved (treated as characters, not word separators), and that mode changes, such as changes in font size within a paragraph, should be ignored. This allows the tags `<pre>` and `</pre>` to be used.
`KVSTYLE_HEADING[1-6]`	The flags `KVSTYLE_HEADING[1-6]` specify that a given style is to be detected and processed as a heading. Heading flags are exclusive. This means a style cannot be processed as both h1 and h2. By default, Export maps the heading style "Heading 1" to `<h1></h1>`, and so on, for heading levels 1 through 6. If you use style mappings, the default mapping is overridden. Therefore, you must supply markup for *all* heading levels. Export uses heading levels to define the overall structure of the HTML output.
`KVSTYLE_ORDERLIST`	The `KVSTYLE_ORDERLIST` flag specifies that the style should be tagged as an ordered list. **Currently not implemented**.
`KVSTYLE_UNORDEREDLIST`	The `KVSTYLE_UNORDERLIST` flag specifies that the style should be tagged as an unordered list. **Currently not implemented**.
`KVSTYLE_DELETECONTENT`	The `KVSTYLE_DELETECONTENT` flag specifies that the content associated with the style tag should be deleted from the output.
`KVSTYLE_ ONCONSECUTIVEPARAGRAPHS`	The `KVSTYLE_ONCONSECUTIVEPARAGRAPHS` flag specifies that the style should be applied to consecutive paragraphs of the document. If this flag is used, and two or more paragraphs require the same style, the opening and closing tags that normally appear between each paragraph are not generated.

Flags for Defining Styles, continued

Flag	Description
KVSTYLE_REDACT	The KVSTYLE_REDACT flag is used to hide sensitive or confidential information in the source document. It specifies that the text associated with the style tag should be replaced in the HTML output with a selected character. The default replacement character is "X," but you can specify a different replacement character by setting cRedact. SeecRedact, on page 223.

Use Style Sheets

You can use style sheets to define the overall layout and type specifications of the HTML output. Export can write style sheet information to an external Cascading Style Sheet (CSS) file, or read the information from an existing CSS file during the conversion. The formatting data can either be stored within the output HTML file (inline), or externally in a CSS file. Using an external style sheet makes the HTML output significantly smaller, and allows you to use the same style sheet for many conversions. The style sheet options are enumerated in KVHTMLStyleSheetType.

> **NOTE:** Cascading style sheets can be used only with word processing documents.

To enable CSS formatting and output the generated formatting data within the output HTML stream, set eStyleSheetType member to **CSS_INLINE**, either directly in the KVHTMLOptionsEx structure or in the template file.

To enable CSS formatting and output the generated formatting data in an external CSS file that is referenced in the HTML output as a tag

1. Set eStyleSheetType to **CSS_TOFILE**, either directly in the KVHTMLOptionsEx structure or in the template file.

2. In the template file, use the $STYLESHEET token to specify the URL of the style sheet in the HTML output. The external CSS file is referenced in the output HTML by a LINK statement of the form:

   ```
   <LINK rel="STYLESHEET" href="CSS_file" type="text/css">
   ```

3. Call the KVHTMLSetStyleSheet() function to set the path and file name of the external style sheet. See KVHTMLSetStyleSheet(), on page 190.

The sample program htmlini provides an example of using style sheets. htmlini, on page 123.

Display Vector Graphics on UNIX and Linux

Export offers the option of rasterizing vector graphic content from source documents into a variety of graphics formats including JPEG, PNG, WMF, and CGM. This solution is implemented with Windows Graphical Device Interface (GDI) code, and therefore is not portable to other platforms.

The output format of vector graphics is defined by the `OutputVectorGraphicType` member in the `KVHTMLOptionsEx` structure, and the options are enumerated in `KVHTMLGraphicType` in `kvhtml.h`. See and .

To display vector graphics in presentation, word processing, and spreadsheet files on UNIX and Linux, Export converts the files directly to JPEG by using a Java program named `kvraster.class`. This program uses the Java Abstract Windowing Toolkit (AWT). The AWT requires access to an X Server.

> **NOTE:** If you are using KeyView 10.5.0.0 or Java 1.6, you do not have to set up an X Server; however, if you are using a version of KeyView lower than 10.4 with a version of Java lower than 1.6, you must set up an X Server.

To set up an X Server, do one of the following:

- Run a virtual X Server, such as the Xvfb utility. This utility is included in the X11R6 distribution, or you can download it from the following site:

  http://www.x.org/Downloads.html

  For example, to run the Xvfb utility on a 512 Mb, Solaris 2.8 platform, follow these steps:

  1. Start Xvfb at root:

     ```
     /usr/X11R6/bin/Xvfb :1 -screen 0 1152x900x8 &
     ```

  2. Set the display environment variable:

     ```
     setenv DISPLAY:1.0
     ```

- Make an X display available to the Java runtime by using the `DISPLAY` environment variable. No windows appear on the display. For example, set the `DISPLAY` environment variable as follows:

  ```
  setenv DISPLAY computername:0.0
  ```

  or

  ```
  setenv DISPLAY ipaddress:0.0
  ```

After the X Server is set up, convert the file by following these steps:

1. Add the location of the JRE to the `PATH` environment variable.

2. Set `OutputVectorGraphicType` to **KVGFX_JPEG** in the `defunix.ini` template file or directly in the API.

3. Convert the document to HTML. The graphics in the document are converted to JPEG and stored in the output directory.

> **NOTE:**
> `kvvector.jar` must reside in the output directory.

Search and Highlight Terms

KeyView can use the highlighting API to find and highlight specified text strings in the HTML output. Only text strings that exactly match the search term are highlighted. For example, if the term "house" is

specified, the string "house" would be highlighted in **house, house**s, and **house**d, but would not be highlighted in the term "housing." You can define the text attributes used to highlight the text, such as bold, red, or underlined, and the text's target character set.

If a specified term contains HTML code, it is not found. For example, if the phrase "`weekly schedule`" was specified, the following string in the output HTML would not be found:

```
weekly <b>schedule</b>
```

If you specify multiple terms, and some terms are subsets of other terms, Micro Focus recommends that you specify the superset first. For example,

```
["North American car manufacturers" "car manufacturers" "car"]
```

To specify search terms by using the C API

1. Define the `KVHTMLHighlight` data structure specifying the list of terms, the highlighting attributes, and the case-sensitivity Boolean. See KVHTMLHighlight, on page 211.

2. Call the `fpSetHighlight()` function. See KVHTMLSetHighlight(), on page 189.

Include Revision Information

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 changes were made to a document, You can configure Export to convert the deleted text and graphics and include revision information in the HTML output. (The deleted content and revision information is excluded from the HTML output by default.)

Content that was added to the document is identified by `<ins>` tags and is underlined when displayed in a browser. Content that was deleted from the document is identified by `<del>` tags and is displayed with strikethrough formatting. The `<ins>` and `<del>` tags include the following attributes:

style	This is an optional attribute that is not included by default. You can define a unique HTML style (such as `color: red; background: orange`) that is applied to each reviewer's edits. See Configure the Revision Style, on page 91.
title	The `title` attribute can contain a prefix and revision information which is displayed in a browser. By default, the prefix is either the text string "`inserted:`" or "`deleted:`", and the revision information includes the reviewer name, and the date and time the revision was made.

You can exclude the `title` attribute or define different text strings for the `title` attribute (see Configure the Revision Title, on the next page). |
| cite | The `cite` attribute contains the name of the reviewer who made the revision. |
| datetime | The `datetime` attribute contains the date and time the revision was made. The date is in ISO-8601 format: `YYYY-MM-DDThh:mm:ss`. |

For example, the following markup can be generated for inserted text:

```
<ins style="color: red" title="Inserted: JohnD, 2006-04-24Tl4:47:00"
cite="mailto:JohnD" datetime="2006-04-24T14:47:00">This text was added</ins> in a
previous version.
```

This text is displayed in the browser as:

<u>This text was added</u> in a previous version.

When you hover the cursor over the underlined text in the browser, the text "**Inserted: JohnD, 2006-04-24TI4:47:00**" is displayed as a ToolTip.

> **NOTE:** Whether the text is displayed with strikethrough or underline depends on the configuration and capabilities of the browser.

To convert deleted text and graphics and include revision information

1. Call the `fpInit()` function. See fpInit(), on page 174.

2. Define the `KVRevisionMark` structure. See KVRevisionMark, on page 231.

3. Call the `fpHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	Parameter
nType	KVCFG_INCLREVISIONMARK
nValue	TRUE
pData	A pointer to the `KVRevisionMark` structure which defines the information that appears in the `title` attribute and the HTML styles applied to revised content. If you pass `NULL` to this function, defaults are used.

   For example:

   ```
   KVRevisionMark    RMark;
   memset(&RMark, 0, sizeof(KVRevisionMark));
   KVStructInit(&RMark);
   RMark...
   (*fpHTMLConfig)(pKVHTML, KVCFG_INCLREVISIONMARK, TRUE, &RMark))
   ```

   The `htmlini` sample program demonstrates this function. See htmlini, on page 123.

4. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

Configure the Revision Title

The `title` attribute can contain a prefix and revision information which is displayed in a browser. By default, the prefix is either the text string "`inserted:`" or "`deleted:`" and the reviewer name and date/time are included in the title.

- To exclude the `title` attribute from the `<ins>` and `<del>` tags, set the `RM_TITLE_FLAG` in the `KV_RM_Title` structure to `RMT_Off`. See KV_RM_Title, on page 232.

- To define a different text string for the prefix, specify a new text string in the `pPrefix` member and set the `nSize` and `eCharSet` members of the `KV_RM_Title` structure. See KV_RM_Title, on page 232.

- To change the revision information included in the `title` attribute, set the `RM_TITLE_FLAG` in the `KV_RM_Title` structure. See RM_Title_Flag, on page 252.

The following example sets the prefix as "`Added:`" and "`Removed:`" for inserted and deleted text respectively, and only includes the reviewer name in the `title` attribute:

```
KVRevisionMark      RMark;
char                RMInsPre[16] = "Added:";
char                RMDelPre[16] = "Removed:";
    memset(&RMark, 0, sizeof(KVRevisionMark));
    KVStructInit(&RMark);
    RMark.InsTitle.eFlag = RMT_Author;
    RMark.InsTitle.pPrefix = (BYTE *)(&RMInsPre);
    RMark.InsTitle.nSize = strlen(RMInsPre);
    RMark.InsTitle.eCharSet = KVCS_UTF8;
    RMark.DelTitle.eFlag = RMT_Author;
    RMark.DelTitle.pPrefix = (BYTE *)(&RMDelPre);
    RMark.DelTitle.nSize = strlen(RMDelPre);
    RMark.DelTitle.eCharSet = KVCS_UTF8;
(*fpHTMLConfig)(pKVHTML, KVCFG_INCLREVISIONMARK, TRUE, &RMark))
```

Configure the Revision Style

You can define a unique HTML style (such as `color: red; background: orange`) that is applied to each reviewer's modifications. This allows you to easily differentiate between multiple reviewers' edits. For example, changes made by JSmith are highlighted in red, changes made by RBrown are highlighted in blue, and so on.

To define revision styles, set the number of revision styles in the `nAuthorStyles` member of the `KVRevisionMark` structure, and use the `ppAuthorStyles` member for each style to define the contents of the `style` attribute. See KVRevisionMark, on page 231.

The following example defines two revision styles:

```
KVRevisionMark      RMark;
char    RMAuthorStyle0[60] = "color: red; background: yellow";
char    RMAuthorStyle1[60] = "color: green; background: silver";
    memset(&RMark, 0, sizeof(KVRevisionMark));
    KVStructInit(&RMark);
    RMark.nAuthorStyles = 2;
    RMark.ppAuthorStyles = (char **)malloc(sizeof(char *)*2);
    if(!RMark.ppAuthorStyles) return(1);
    RMark.ppAuthorStyles[0] = RMAuthorStyle0;
    RMark.ppAuthorStyles[1] = RMAuthorStyle1;
(*fpHTMLConfig)(pKVHTML, KVCFG_INCLREVISIONMARK, TRUE, &RMark))
```

If there are more reviewers than defined styles, KeyView applies all available styles to the reviewers in the order in which they are encountered in the document, and then applies styles starting from the beginning of the list to the remaining reviewers. This process is repeated until all reviewers' edits are highlighted.

> **NOTE:** KeyView does not validate styles. They are written directly to the HTML output.

Generate a Revision Summary

You can configure Export to summarize the changes made to a document in a revision summary file that is generated during the HTML conversion. The summary file is created in the directory where the HTML output is generated. The default file name is `output_filename.revsum.htm`. You can customize this file name by using the `fpGetAnchor` callback function. See GetAnchor(), on page 196.

To create a revision summary file, set the `bCreateSummary` flag to **TRUE** in the `KVRevisionMark` structure, and use the `pszRevSumStartBlock` and `pszRevSumEndBlock` members to define the markup and tokens inserted at the beginning and end of the revision summary file.

For example:

```
KVRevisionMark      RMark;
char                RMStartBlock[500] = "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0//EN"> <html> <body>";
char                RMEndBlock[30] = "</body> </html>";
    memset(&RMark, 0, sizeof(KVRevisionMark));
    KVStructInit(&RMark);
    RMark.pszRevSumStartBlock = RMStartBlock;
    RMark.pszRevSumEndBlock = RMEndBlock;
    RMark.bCreateSummary = TRUE;
(*fpHTMLConfig)(pKVHTML, KVCFG_INCLREVISIONMARK, TRUE, &RMark))
```

Extract Text from Text Boxes

By default, the contents of Microsoft Word text boxes are converted to graphics and exported. Alternatively, you can convert the contents of text boxes to text. Note, the extracted text box text is not formatted.

To enable text box extraction

- Add the following to the `formats_e.ini` file:

  ```
  [WordTextBoxOptions]
  OutputText=true
  ```

Convert PDF Files

Export has special configuration options that allow greater control over the conversion of PDF files. These options can improve the fidelity and accuracy of the HTML output.

Use the pdf2sr Reader

In KeyView Export SDK 10.24, the `pdf2sr` reader was added. It generates a high fidelity raster image of each page in the PDF and will insert text that has a zero opacity value in the HTML to allow for text searching in a web browser.

The `pdf2sr` reader has the following features:

- supports standard and custom metadata (non-XMP)
- supports basic text extraction
- supports password protected PDFs

The `pdf2sr` reader has the following limitations:

- does not support logical order
- does not support bidi PDFs
- does not extract subfiles
- does not extract bookmarks from PDFs
- does not give estimations on percent embedded fonts match with display glyphs
- Does not support XMP metadata
- Does not support headers or footers
- does not support annotations
- does not support content access stream
- does not support tagged content (PDFs)

To specify the pdf2sr reader

1. Open the `formats_e.ini` file with a text editor.

2. In the `[Formats]` section, set the following parameter to the `pdf2sr` reader:

   `200=pdf2`

When you use the `pdf2sr` reader, the output HTML uses HTML5 syntax that might be disabled when using Internet Explorer to view the output. It might prompt the user for permission to run. To disable this behavior, configure Internet Explorer as follows:

1. In Internet Explorer, select **Tools** from the menu.

2. Select **Internet Options**.

3. Click the **Advanced** tab.

4. In the **Security** area, click **Allow active content to run in files on My Computer.**

Use a Graphic-Based Reader

Two graphic-based PDF readers are available. The readers display PDFs by converting each page of the PDF to an image. If you do not want to redistribute the Acrobat Reader with your application, you can use a graphic-based reader instead.

The two readers support different features. Choose the appropriate reader depending on your requirements:

- The kppdfrdr reader supports highlighting, annotation, and several other features but also has several graphical limitations.

- The kppdf2rdr reader produces high-fidelity raster images but is a viewer only and does not support highlighting or other features.

Use the kppdfrdr Reader

The kppdfrdr graphic-based reader has the following features:

- supports vector images

- supports rotation and scaling

- supports multibyte and bidirectional text

- allows you to search text in the output

The kppdfrdr reader has the following limitations:

- Embedded fonts in a PDF file are not translated correctly. They are usually displayed using the question mark (?) replacement character.

- If an unsupported font is encountered during conversion, the default font, Times New Roman, is substituted.

- Supports 180 degree rotation only for raster images.

- Supports the following color spaces: DeviceRGB, DeviceGray, DeviceCMYK, CalGray, and CalRGB color spaces. Indexed color spaces are supported as long as they are used with a supported basic color space.

- Does not support hyperlinks.

- Does not extract summary information (metadata).

Use the kppdf2rdr Reader

The kppdf2rdr graphic-based reader produces high-fidelity raster images. However, it has the following limitations:

- Does not support anything beyond viewing, such as text searching.

- Does not support PDFs that contain XFA forms content.

Specify the Graphic-based Reader

By default, the Acrobat control is used to convert PDF documents. Use the following procedure to specify that one of the graphic-based readers be used to convert PDF documents.

To specify the graphic-based reader

1. Open the `formats_e.ini` file with a text editor. The file is installed in the root of the Windows directory.

2. In the `[HiFi]` section, set the following parameter to the graphic-based reader you want to use. Set one of the following values:

   - For the `kppdfrdr` reader:

     `200=kppdfrdr.dll`

     This is the default setting.

   - For the `kppdf2rdr` reader:

     `200=kppdf2rdr.dll`

     Set `CFG_SETHIFIPDF` field in the `HtmlExport` class.

3. Set the `KVCFG_SETHIFIPDF` option in the `fpHTMLConfig()` function (see KVHTMLConfig(), on page 177).

Convert PDF Files to Raster Images

Export allows you to convert each page of a PDF document to a raster image, providing a high-fidelity conversion of the document.

The output format depends on the value of `OutputRasterGraphicType` in `KVHTMLOptionsEx`. See KVHTMLOptionsEx, on page 216.

On UNIX and Linux, the conversion of PDFs to JPEG uses the Java program `kvraster.class`. This Java program requires some setup. See Display Vector Graphics on UNIX and Linux, on page 87.

To use a graphic-based reader to convert PDF documents

1. Specify the graphic-based reader that you want to use.

2. Call the `fpInit()` function. See fpInit(), on page 174.

3. Call the `KVHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	Parameter
nType	KVCFG_SETHIFIPDF

Argument	Parameter
`nValue`	`TRUE` (non-zero)
`pData`	`NULL`

For example:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SETHIFIPDF, TRUE, NULL);
```

The `cnv2html` sample program demonstrates this function. See cnv2html, on page 120.

4. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

Convert 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 convert 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 produce PDF files containing 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.

Logical Reading Order and Paragraph Direction

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 the 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 reader determines the paragraph direction for each PDF page, and then sets the direction accordingly. When a paragraph direction is not specified, this option is used.

> **NOTE:** Conversions 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, let us say that a PDF file contains English paragraphs in three columns that read from left to right, but 80% of the second paragraph contains 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 reader, and is output from right to left.

> **NOTE: 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_e.ini` file. Setting the direction in the API overrides the setting in the `formats_e.ini` file.

Use the C API

To enable PDF logical reading order in the C API

1. Call the `fpInit()` function. See fpInit(), on page 174.

2. Call the `KVHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	Parameter
nType	KVCFG_LOGICALPDF
nValue	Set to one of the following flags which are defined in `kvtypes.h`. (see LPDF_DIRECTION, on page 251): **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. When a paragraph direction is not specified, this option is used. **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:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_LOGICALPDF, LPDF_RTL, NULL);
```

The `cnv2html` sample program demonstrates this function. See cnv2html, on page 120.

3. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

Use the formats_e.ini File

The `formats_e.ini` file is in the directory *install**OS*\\bin, where *install* is the path name of the Export installation directory and *OS* is the name of the operating system.

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

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

   ```
   [Formats]
   ```

   ```
   200=lpdf
   ```

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

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

   where *paragraph_direction* is one of the following:

Flag	Description
LPDF_LTR	Left-to-right paragraph direction

Flag	Description
`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. When a paragraph direction is not specified, this option is used.
`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.

Generate a Table of Contents from PDF Bookmarks

When you convert PDF files to HTML by using the basic reader (`pdfsr`), the table of contents is generated from "bookmarks" within the PDF file. The hyperlinked table of contents can appear either at the beginning of the HTML file or in a separate frame.

Micro Focus recommends that you configure the conversion so that the table of contents appears in a separate frame (the template `pdfframe.ini` demonstrates how to do this). Export uses absolute positioning when converting a PDF file, that is, the text appears in the exact position as in the original document. Table of contents entries do not contain absolute positioning information. Therefore, if the main document and the table of contents are generated in the same output file, the table of contents entries might overlap the body text in the document.

> **NOTE:** When PDF bookmarks are converted to a table of contents in HTML, the generated links do not lead to the exact location of the destination marker, but jump to the page on which the destination marker exists. This is similar to the behavior of the Adobe Acrobat Reader.

Disable Bookmark Conversion

By default, Export converts PDF bookmarks to a table of contents in the HTML output. However, you can configure Export not to generate a table of contents based on the PDF bookmarks.

To prevent conversion of PDF bookmarks

1. Call the `fpInit()` function. See fpInit(), on page 174.

2. Call the `KVHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	Parameter
`nType`	`KVCFG_SUPPRESSTOCPRINTIMAGE`
`nValue`	`TRUE` (non-zero)
`pData`	`NULL`

For example:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SUPPRESSTOCPRINTIMAGE, TRUE, NULL);
```

The sample programs Export Demo and `cnv2html` have `KVCFG_SUPPRESSTOCPRINTIMAGE` enabled. When you use these programs to convert a PDF file with bookmarks, the HTML output does not include a table of contents.

3. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

Convert Invisible Text

PDF documents sometimes contain invisible text. You can search this text in Adobe PDF Reader, but you cannot view it in a web browser.

Toggle Invisible Text

You can add a JavaScript button to the upper right corner of the exported page, which you can click to toggle between invisible and regular text. When you turn on invisible text, the invisible text is displayed and the regular content is hidden; when you turn off invisible text, the invisible text is hidden.

Invisible text is hidden by default. The toggle button only appears if invisible text is detected in the PDF document.

To add an invisible text toggle button

1. Call the `fpInit()` function. See fpInit(), on page 174.

2. Call the `KVHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	Parameter
nType	KVCFG_SETPDFINVISTEXTTOGGLE
nValue	0 (not used)
pData	szButtonName

For example:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SETPDFINVISTEXTTOGGLE, 0, szButtonName);
```

The `cnv2html` and `htmlini` sample programs demonstrate this function. See cnv2html, on page 120 and htmlini, on page 123.

3. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

> **NOTE:** If no invisible text is detected in the PDF document, no toggle button appears in the HTML output even if you set `KVCFG_SETPDFINVISTEXTTOGGLE`.

Specify Opacity of Invisible Text

Invisible text often occurs in PDF documents when the PDF software processes rasterized images through optical character recognition and then inserts the text in the PDF. You might want to display both the invisible text as well as the rasterized image. To do so, you can set the invisible text *opacity* as determined by an integer from 0 to 100, where 0 hides the invisible text and 100 displays it fully.

Invisible text opacity is set to 0 by default.

To set invisible text opacity

1. Call the `fpInit()` function. See fpInit(), on page 174.

2. Call the `KVHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	Parameter
nType	KVCFG_SETPDFINVISTEXTOPACITY
nValue	iInvisOpacity
pData	NULL

   For example:

   ```
   (*fpHTMLConfig)(pKVHTML, KVCFG_SETPDFINVISTEXTOPACITY, iInvisOpacity, NULL);
   ```

   The `htmlini` sample program demonstrates this function. See htmlini, on page 123.

3. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

Convert Rotated Text

By default, rotated text is displayed in its original position, at the original font size, and at 0 degrees rotation in the HTML output. The text is not rotated in the HTML output because text rotation is not supported by HTML.

Because the text is the original size, but might be displayed in a smaller space (at 0 degrees), the text might overlap adjacent text in the HTML output. To avoid this problem, you can specify that the rotated text be removed from its original position and displayed at the bottom of the HTML page on which it appears.

To specify that rotated text be displayed at the bottom of the HTML page

1. Call the `fpInit()` function. See fpInit(), on page 174.

2. Call the `KVHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	**Parameter**
nType	KVCFG_SETTEXTROTATE
nValue	TRUE (non-zero)
pData	NULL

For example:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SETTEXTROTATE, TRUE, NULL);
```

The sample programs Export Demo and cnv2html demonstrate how to use this function. See Use the Export Demo Program, on page 41, and htmlini, on page 123.

3. Call the fpConvertStream() or KVHTMLConvertFile() function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

> **NOTE:** When this feature is enabled, white space is added to the bottom of every HTML page to accommodate any rotated text.

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 maintains the source document's soft hyphens in the output HTML to more accurately represent the source document's layout. However, if you are using Export to generate text output for an indexing engine or are not concerned with maintaining the document's layout, Micro Focus recommends that you remove soft hyphens from the HTML output. To remove soft hyphens, you must enable the soft hyphen flag.

> **NOTE:** If the soft hyphen flag is enabled, *every* hyphen at the end of a line is considered a soft hyphen and removed from the HTML output. If a hard hyphen appears at the end of a line, it is also removed. This might result in an intentionally hyphenated word being extracted without a hyphen.

To remove soft hyphens from the HTMLoutput

1. Call the fpInit() function. See fpInit(), on page 174.

2. Call the KVHTMLConfig() function with the following arguments (see KVHTMLConfig(), on page 177):

Argument	Parameter
nType	KVCFG_DELSOFTHYPHEN
nValue	TRUE (non-zero)
pData	NULL

For example:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_DELSOFTHYPHEN, TRUE, NULL);
```

3. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

Extract Custom Metadata from PDF Files

To extract custom metadata from your PDF files, 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`. See KVSumInfoType, on page 247.

For example:

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

> **NOTE:** Metadata cannot be extracted from PDFs when the PDF is converted to JPEG. See Convert PDF Files to Raster Images, on page 95.

Convert Spreadsheet Files

Export has special configuration options that allow greater control over the conversion of spreadsheet files.

Convert Hidden Text in Microsoft Excel Files

Normally, Export does not convert hidden text from a Microsoft Excel spreadsheet because it is assumed that the text should not be exposed. You can change this default behavior and convert text in hidden rows, columns, and sheets by adding the following lines to the `formats_e.ini` file:

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

Convert Headers and Footers in Microsoft Excel 2003 Files

Normally, Export does not convert headers and footers from Microsoft Excel 2003 spreadsheets. You can change this default behavior and convert headers and footers by adding the following lines to the `formats_e.ini` file:

```
[Options]
ShowHeaderFooter=1
```

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.

Convert Very Large Numbers in Spreadsheet Cells to Precision Numbers

Numbers in Microsoft Excel files can now be exported and written to the output without formatting. By default, numbers are exported 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 export numbers without formatting, add the following options in the following lines to the `formats_e.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 Export 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_e.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:** 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 the following table for a list of supported functions.

When formula extraction is enabled, Export can extract Microsoft Excel formulas containing 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()
```

Set Minimum Image Size

You can set a minimum size limit for the images to export from spreadsheet files. This option can improve performance for documents that have lots of very small images.

To set the minimum image size, add the following lines to the `formats_e.ini` file:

```
[ss_flags]
process_images_with_min_width=N
process_images_with_min_height=M
```

where N and M are the minimum image dimensions, in pixels. For example:

```
[ss_flags]
process_images_with_min_width=150
process_images_with_min_height=250
```

Convert Presentation Files

Export has special configuration options that allow greater control over the conversion of presentation files.

Convert Presentation Files to Raster Images

Export allows you to convert each slide in a presentation document to a raster image, providing a high-fidelity conversion of the document.

The output format depends on the value of `bRasterizeFiles` in `KVHTMLOptionsEx`. See KVHTMLOptionsEx, on page 216.

Convert Presentation Files to a Logical Reading Order

Some presentation files do not store 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.

In general, when you convert a presentation slide to a raster image, the output file retains the logical reading order because it uses the correct coordinates for each element in the output. However, if you do not use the `bRasterizeFiles` option in `KVHTMLOptionsEx` to produce a raster image, you might find that the export process generates output for some files that does not match the logical reading order.

When you do not want to rasterize your presentation files, you can use the `formats_e.ini` file to retain the logical reading order in your files.

The `formats_e.ini` file is in the directory *install**OS*\bin, where *install* is the path name of the Export installation directory and *OS* is the name of the operating system.

To enable logical reading order by using the formats_e.ini file

- In the `formats_e.ini` file, find the `[Options]` section, and set `LogicalOrder` to **1**.

  For example:

  ```
  [Options]
  LogicalOrder=1
  ```

Convert XML Files

Export enables you to extract all or selected content from source XML files (see Configure Element Extraction for XML Documents, below). It 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 393 for more information on format detection.

Configure Element Extraction for XML Documents

When you convert 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 XML sample below, `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 converting 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 sample above, the child element `para` is also extracted.

Export 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 converting 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.

> **NOTE:** You can only use customized element extraction settings when converting files in process. When converting out of process, the default extraction settings are used.

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. See fpInit(), on page 174.

2. Define the `KVXConfigInfo` structure. See KVXConfigInfo, on page 207.

3. Call the `KVHTMLConfig()` function with the following arguments (see KVHTMLConfig(), on page 177:

Argument	Parameter
nType	KVCFG_SETXMLCONFIGINFO
nValue	0
pData	address of the `KVXConfigInfo` structure

   For example:

   ```
   KVXConfigInfo  xinfo;  /* populate xinfo */
   (*fpHTMLConfig)(pKVHTML, KVCFG_SETXMLCONFIGINFO, 0, &xinfo);
   ```

4. Repeat steps 2 and 3 until the settings for all the XML document types you want to customize are

defined.

5. Call the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

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 template file when processing the XML file.

   The C sample program `htmlini` demonstrates how to use a template file during the conversion process. See Introduction, on page 117.

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 Export 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 393 for more information on format ID values.
	If you are adding configuration settings for a custom XML document type, this 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 page 112.

Configuration Option	Description
szInMetaElement	The elements extracted from the file as metadata. All other elements are extracted as text.
	Multiple entries must be separated by 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.
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.
	Multiple entries must be separated by 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.
szInContentElement	The elements extracted from the file as content text. Enter an asterisk (*) to extract all elements including child elements.
	Multiple entries must be separated by 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.
szExContentElement	The child elements in the included content elements that are not extracted from the file as content text.
	Multiple entries must be separated by 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.
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
	Multiple entries must be separated by 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*

You must enclose attribute values that contain space in quotation marks.

For example, the following 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 containing the root element `keyviewxml`, you could add the following section to the end of the initialization file:

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

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

The custom extraction settings must be preceded by a section heading named `[config`*N*`]`, where *N* is an integer that starts at 100 and increases 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 `KVHTMLConfig()` function, the default extraction settings for a generic XML document are used.

Show Hidden Data

Microsoft Word, Excel, and PowerPoint documents contain hidden information, some of which is shown by default when exported, and some of which is hidden by default. There are several options that allow you to determine which types of hidden data are shown.

Hidden Data in Microsoft Documents

You can show several types of hidden data from Microsoft Word, Excel, and PowerPoint documents, each of which has a corresponding flag in the KVHTMLConfig(), on page 177 function, which you can toggle to determine whether the hidden data is shown or not. The following table lists each data type, its default behavior, and its corresponding configuration API flag.

Hidden data settings

Hidden Data Type	Default Behavior	Configuration API Flag
Microsoft Word		
Comments[1]	Shown[2]	`KVCFG_WP_NOCOMMENTS`
Hidden text	Hidden	`KVCFG_WP_SHOWHIDDENTEXT`
Date field codes	Calculated date	`KVCFG_WP_SHOWDATEFIELDCODE`
File name field codes	Document file name	`KVCFG_WP_SHOWFILENAMEFIELDCODE`
Microsoft Excel		
Hidden information	Hidden	`KVCFG_SS_SHOWHIDDENINFOR`
Comments	Hidden	`KVCFG_SS_SHOWCOMMENTS`
Formulas	Calculated value	`KVCFG_SS_SHOWFORMULA`
Microsoft PowerPoint		
Hidden slides	Shown	`KVCFG_PG_HIDEHIDDENSLIDE`
Comments	Shown[3]	`KVCFG_PG_HIDECOMMENT`

[1]Word comment settings can also be toggled with a configuration parameter in the `formats_e.ini` file. See Toggle Word Comment Settings in the formats_e.ini File, on the next page.
[2]Shown by default in Microsoft Word 97 to 2003 documents.
[3]Shown by default in Microsoft PowerPoint 97 to 2000 documents.

Hidden data settings, continued

Hidden Data Type	Default Behavior	Configuration API Flag
Comments slide	Hidden	KVCFG_PG_SHOWCOMMENTSSLIDE[1]
Slide notes[2]	Hidden	KVCFG_PG_SHOWSLIDENOTES

To toggle the display of any type of hidden data

- Use the configuration API and set the third parameter to **TRUE** or **FALSE**:

  `(*fpHTMLConfig)(pKVHTML, KVCFG_WP_NOCOMMENTS, TRUE, NULL)`

  In this example, comments will not be exported from Word documents.

  > **NOTE:** The third parameter affects the *default* behavior. To change the default behavior, set it to **TRUE**.

For more information, see KVHTMLConfig(), on page 177.

Toggle Word Comment Settings in the formats_e.ini File

Microsoft Word 97 to 2003 comment settings can also be controlled through a parameter in the `formats_e.ini` file.

The `formats_e.ini` file is in the directory `install\OS\bin`, where `install` is the path name of the Export installation directory and `OS` is the name of the operating system.

To toggle comment output in formats_e.ini

1. Open the `formats_e.ini` file in a text editor.

2. Under `[Options]`, add the `WP_NOCOMMENTS` parameter and set it to `0` to show comments, or `1` to hide comments. For example:

   ```
   [Options]
   WP_NOCOMMENTS=1
   ```

   > **NOTE:** The `KVCFG_WP_NOCOMMENTS` configuration API flag overrides the setting in `formats_e.ini`.

Toggle PowerPoint Slide Note Settings in the formats_e.ini File

Microsoft PowerPoint slide note settings can also be controlled through a parameter in the `formats_e.ini` file.

---

[1]This setting affects PowerPoint 2003 and 2007 only.
[2]PowerPoint slide note settings can also be toggled with a configuration parameter in the `formats_e.ini` file. See Toggle PowerPoint Slide Note Settings in the formats_e.ini File, below.

The `formats_e.ini` file is in the directory *install**OS*\\bin, where *install* is the path name of the Export installation directory and *OS* is the name of the operating system.

To toggle slide note output in formats_e.ini

1. Open the `formats_e.ini` file in a text editor.

2. Under `[Options]`, add the `ShowSlideNotes` parameter and set it to `1` to show slide notes, or `0` to hide slide notes. For example:

   ```
   [Options]
   ShowSlideNotes=1
   ```

   > **NOTE:** The `KVCFG_PG_SHOWSLIDENOTES` configuration API flag overrides the setting in `formats_e.ini`.

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_e.ini` file:

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

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

Source Code Identification

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

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

You can set source code identification to different levels.

Option	Description
`KVSOURCECODE_OFF`	Do not enable source code identification.
`KVSOURCECODE_ENABLED`	Enable source code identification for the most common source code formats.

KVSOURCECODE_ EXTENDED	Enable source code identification for all supported source code formats. This option might lead to false positives in some cases (for example, a C++ file might get identified as a rarer format).

For the complete list of source code formats supported for both options, see Detected Formats, on page 316.

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

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

Chapter 5: Sample Programs

This section describes the sample programs provided with HTML Export.

Introduction

The sample programs demonstrate how to use the C, Visual Basic and COM implementations of HTML Export.

The sample code is intended to provide a starting point for your own applications or to be used for reference purposes.

The source code and makefile for each program are in the directory:

install\htmlexport\programs*program_name*

where *install* is the path name of the Export installation directory, and *program_name* is the name of the sample program.

C Sample Programs

The C sample programs demonstrate how to use the C implementation of HTML Export. The following sample programs are provided:

- tstxtract, on page 119

You can use the `tstxtract`, `cnv2html`, `cnv2htmloop`, and `htmlini` sample programs on Windows and UNIX. All other sample programs are for Windows only.

> **NOTE:** The sample programs do not parse white space in file names. If your file names contain spaces, use quotation marks around the entire path name. Inserting quotation marks around the file name only does not work.

To compile the sample programs, use the makefiles provided in the sample programs' directory. Ensure the HTML Export `include` directory is specified in the include path of the project. After the executables are compiled and built, you must place them in the same directory as the HTML Export libraries.

Compile the Visual Basic Sample Program

To compile Export Demo, use the Visual Studio project file (`demo_vb.vbp`) in the directory *install*`\htmlexport\programs\ExportDemo`, where *install* is the path name of the Export installation directory. The executable is in the same directory.

COM Sample Program

The COM API is a COM interface to HTML Export and is available on 32-bit Windows platforms only. The interface is well suited for Visual Basic and Visual J++ programmers. The interface is an `IDispatch` interface, and can therefore be used with scripting environments such as Active Server Pages.

The following COM sample programs are provided:

- Template Wizard, on page 127

- comsamp, on page 132

- htmloop, on page 132

To compile the Template Wizard, use the Visual Studio project file (`htmvbwzd.vbp`) in the *install*`\htmlexport\programs\wizard` directory, where *install* is the path name of the Export installation directory.

To compile the `comsamp` sample program, use the Visual Studio project file (`comsamp.vbp`) in the *install*`\htmlexport\programs\comsamp` directory.

To compile `htmloop`, use the Visual Studio project file (`htmloop2.dsw`) in the *install*`\htmlexport\programs\htmloop` directory.

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 convert the files by using one of the conversion sample programs.

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 371 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.
`-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. For the example above, 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 option applies to PST files on Windows only.

Option	Description
-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, then it is extracted as plain text. If this flag is not set, the message body is extracted as plain text when possible.
-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 will be extracted.

bin_directory is the path to the Export `bin` directory. This is required if you do `not` run the program from the `install\Export SDK\bin` directory.

cnv2html

The `cnv2html` sample program creates a single, formatted HTML output file. It is called by the Export Demo sample program, but can also be used on its own.

To run `cnv2html`, type the following at the command line:

`cnv2html [options] inputfile outputfile`

where:

options is one or more of the options listed in Options for cnvhtml, below.

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

outputfile is the full path and file name of the HTML output file.

The following options are available.

Options for cnvhtml

Option	Description
-c KVCFG_ SETHIFIPDF	This option specifies the type of reader used to convert PDF documents. In Export, you can convert PDF documents by using either the graphic-based PDF reader, named `kppdfrdr`, or the basic PDF reader, named `pdfsr`. For more information on each reader, see Convert PDF Files to Raster Images, on page 95.
	By default, the basic reader (`pdfsr`) is used to convert PDF documents.
-c KVCFG_ SETTEXTROTATE	This option specifies that rotated text should be displayed at the bottom of a page on which it appears. By default, rotated text in a file is displayed in its original position, at the original font size, and at 0 degrees rotation. The text is not rotated in the output because text rotation is not supported by HTML. See

Options for cnvhtml, continued

Option	Description
	Convert Rotated Text, on page 101 for more information. Currently, this configuration option applies only to PDF files.
`-c KVCFG_ DELSOFTHYPHEN`	This option specifies that soft hyphens in PDF files are deleted from the converted output. See Control Hyphenation, on page 102.
`-c KVCFG_ SETPDFINVISTEXT TOGGLE ButtonName`	y.
`-pdfauto`	This option specifies that PDF files are output in a logical reading order. The PDF reader determines the paragraph direction (left-to-right or right-to-left) for each PDF page, and then sets the direction accordingly. See Convert PDF Files to a Logical Reading Order, on page 96.
`-pdfltr`	This option specifies that PDF files are output in a logical reading order, and the paragraph direction is left to right. See Convert PDF Files to a Logical Reading Order, on page 96.
`-pdfrtl`	This option specifies that PDF files are output in a logical reading order, and the paragraph direction is right to left. See Convert PDF Files to a Logical Reading Order, on page 96.
`-pdfraw`	This option specifies that PDF files are output in an unstructured paragraph flow. This is the default. Set this flag if logical reading order is enabled, and you want to return to an unstructured paragraph flow. See Convert PDF Files to a Logical Reading Order, on page 96.

cnv2htmloop

The `cnv2htmloop` sample program creates a single, formatted HTML output file, but unlike `cnv2html`, it converts the file out of process. See Convert Files Out of Process, on page 29 for more information on out of process conversions.

To run `cnv2htmloop`, type the following at the command line:

```
cnv2htmloop [options] inputfile outputfile
```

where:

options is one or more of the options listed in Options for cnv2htmloop, on the next page.

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

outputfile is the full path and file name of the HTML output file.

The following options are available.

Options for cnv2htmloop

Option	Description
`-f`	Convert by using out-of-process file mode (default)
`-s`	Convert by using out-of-process stream mode
`-c KVCFG_ SETHIFIPDF`	This option specifies the type of reader used to convert PDF documents. In Export, you can convert PDF documents by using either the graphic-based PDF reader, named `kppdfrdr`, or the basic PDF reader, named `pdfsr`. For more information on each reader, see Convert PDF Files to Raster Images, on page 95. By default, the basic reader (`pdfsr`) is used to convert PDF documents.
`-c KVCFG_ SETTEXTROTATE`	This option specifies that rotated text should be displayed at the bottom of a page on which it appears. By default, rotated text in a file is displayed in its original position, at the original font size, and at 0 degrees rotation. The text is not rotated in the output because text rotation is not supported by HTML. See Convert Rotated Text, on page 101 for more information. Currently, this configuration option applies only to PDF files.
`-c KVCFG_ DELSOFTHYPHEN`	This option specifies that soft hyphens in PDF files are deleted from the converted output. See Control Hyphenation, on page 102.
`-pdfauto`	This option specifies that PDF files are output in a logical reading order. The PDF reader determines the paragraph direction (left-to-right or right-to-left) for each PDF page, and then sets the direction accordingly. See Convert PDF Files to a Logical Reading Order, on page 96.
`-pdfltr`	This option specifies that PDF files are output in a logical reading order, and the paragraph direction is left to right. See Convert PDF Files to a Logical Reading Order, on page 96.
`-pdfrtl`	This option specifies that PDF files are output in a logical reading order, and the paragraph direction is right to left. See Convert PDF Files to a Logical Reading Order, on page 96.
`-pdfraw`	This option specifies that PDF files are output in an unstructured paragraph flow. This is the default. Set this flag if logical reading order is enabled, and you want to return to an unstructured paragraph flow. See Convert PDF Files to a Logical Reading Order, on page 96.

onefile

The `onefile` sample program converts a source document into a single, formatted HTML file.

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

```
onefile inputfile outputfile
```

where:

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

outputfile is the full path and file name of the HTML output file.

index

The `index` sample program produces minimal HTML output suitable for use with indexing engines. It converts a source document into a single, largely unformatted HTML file.

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

index *inputfile outputfile*

where:

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

outputfile is the full path and file name of the HTML output file.

io_samp

The `io_samp` sample program demonstrates how to create an input and an output stream by providing a simple wrapper around the ANSI C interface `fOpen()`, `fRead()`, `fSeek()`, `fTell()`, and `fClose()`. It converts a source document into a single, largely unformatted HTML file.

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

io_samp *inputfile outputfile*

where:

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

outputfile is the full path and file name of the HTML output file.

htmlini

The `htmlini` sample program is used in conjunction with template files to produce HTML documents. For more information on template files, see Set Conversion Options by Using the Template Files, on page 38. Sample template files are in the directory *install*`\htmlexport\programs\ini`. You can use this sample program on Windows and UNIX platforms.

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

htmlini [*options*] *inifile inputfile outputfile*

where:

options is one or more of the options listed in Options for htmlini, on the next page.

inifile is the full path and file name of a template file.

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

outputfile is the full path and file name of the first HTML output file.

The following options are available.

Options for htmlini

Option	Description
`-c css_ filename`	This option writes Cascading Style Sheet (CSS) information to an external file. See Use Style Sheets with htmlini, on the next page.
`-x xmlconfig_ path`	This option converts an XML file by using customized element extraction settings defined in the `kvxconfig.ini` file. If you do not enter the full path to the template file, the program looks for the file in the current working directory (`install\OS\bin`, where `install` is the path name of the Export installation directory and `OS` is the name of the operating system). See Convert XML Files, on page 108.
`-hl term term term`	This option specifies the text string or strings that are found and highlighted in the HTML output. You can specify a maximum of three terms. See Search and Highlight Terms, on page 88.
`-hc charset`	This option specifies the character set of the highlighted search terms in the HTML output.
`-hi`	This option specifies that the text search is case insensitive. You can use this option only when the target character set for the highlighted search term is `KVCS_ 1252`.
`-rm`	This opton converts text and graphics that were deleted from a document with revision tracking enabled, and includes revision information in the HTML output. See Include Revision Information, on page 89.
	This option uses the following hard-coded defaults:
	• the revision title includes the text string "`inserted:`" for `<ins>` tags and "`deleted:`" for `<del>` tags.
	• the revision title includes the reviewer name, date, and time.
	• defines two HTML styles used to highlight reviewers' edits.
	• creates a revision summary file.
`-bp`	This option prevents graphics from being converted and generates image tags with empty `src` attributes. This makes the conversion faster, and maintains the text flow of the original document, because placeholders are generated for the graphics.
`-oop`	This option runs the conversion out of process.
`-fl`	This option prints a list of converted files in the console.
`-pi ButtonName`	This option enables a toggle button in exported PDF documents that you can click to show or hide invisible text. `ButtonName` determines the name of the toggle button. See Toggle Invisible Text, on page 100.
`-ov OpacityValue`	This option specifies the opacity of invisible text in exported PDF documents. `OpacityValue` is an integer from 0 (invisible) to 100 (fully visible). The default is 0. See Specify Opacity of Invisible Text, on page 101.

If the HTML file is output to a directory other than `programs\tempout`, you must update the HTML markup so that the browser can find images used by the templates (such as backgrounds or corporate logos) and the style sheet. The markup contains relative references to the image files (`..\images`).

Use Style Sheets with htmlini

The `htmlini` sample program has an option that allows Cascading Style Sheet (CSS) information to be written to an external file. This makes the HTML output document significantly smaller and enables you to use the same style sheet for many conversions. If the style sheet does not exist or if it is empty, it is created.

To write CSS information to an external file:

1. In the template file, set `eStyleSheetType` to `CSS_TOFILE`. This specifies that the formatting data is stored in a CSS file.

2. In the template file, use the `$STYLESHEET` token to specify the URL of the style sheet in the HTML output. The external CSS file is referenced in the output HTML by a `LINK` statement of the form:

    `<LINK rel="STYLESHEET" href="CSS_file" type="text/css">`

3. At the command prompt, type:

    `htmlini -c stylesheetname inifile inputfile outputfile`

    where `stylesheetname` is the path and file name of the CSS file.

callback

The `callback` sample program demonstrates how you can control the conversion to generate specialized output while it is in progress. The program employs developer-defined callbacks and memory management functions during conversion.

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

`callback inputfile outputfile`

where:

`inputfile` is the full path and file name of the source document.

`outputfile` is the full path and file name of the first HTML output file.

jvtree_demo

The C program `jvtree_demo` creates a frame-based HTML stream which uses the `JVTree.jar` to display the table of contents.

To compile the `jvtree_demo` sample program, use the makefile (`kvhtml.mak`) in the directory `install\htmlexport\programs\jvtree_demo`, where `install` is the path name of the Export installation directory.

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

```
jvtree inputfile outputfile.htm
```

where:

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

outputfile.htm is the full path and file name of the first HTML output file.

> **NOTE:** This program requires the Java Runtime Environment (JRE) 1.3 or higher.
>
> The Java applet, `JVTree.jar`, must be in the same directory as the HTML Export libraries and the output HTML files.
>
> Set the `CLASSPATH` environment variable to include the location of the `\lib` directory and `lib\tools.jar` file for the JDK installed on the machine. Multiple path entries should be separated by semicolons. You must also include the current directory in the search path by using "." in the new setting.

jstree

The `jstree` sample program uses JavaScript to produce an expandable tree view of the table of contents in a frame-based HTML output file.

To compile the `jstree` sample program, use the makefile (`kvhtml.mak`) in the directory *install*\htmlexport\programs\jstree, where *install* is the path name of the Export installation directory.

To run `jstree`, type the following at the command line:
```
jstree inputfile outputfile.htm
```
where:

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

outputfile.htm is the full path and file name of the first HTML output file.

> **NOTE:** You must place the following files in the same directory as the output HTML for the table of contents to function:
>
> ```
> false.gif
> true.gif
> resize.js
> list.js
> ```

JVTree

The `JVTree` sample program employs user callbacks and a Java applet to produce an expandable tree view of the table of contents in a frame-based HTML output file.

To compile the `JVTree` sample program, use the makefile (`kvhtml.mak`) in the directory *install*\htmlexport\programs\jvtree, where *install* is the path name of the Export installation directory.

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

`jvtree inputfile outputfile.htm`

where:

`inputfile` is the full path and file name of the source document.

`outputfile.htm` is the full path and file name of the first HTML output file.

> **NOTE:** This program requires the Java Runtime Environment (JRE) 1.5 or higher.
>
> The Java applet, `JVTree.jar`, must be in the same directory as the HTML Export libraries and the output HTML files.
>
> Set the `CLASSPATH` environment variable to include the location of the `\lib` directory and `lib\tools.jar` file for the JDK installed on the machine. Separate multiple path entries with semicolons. You must also include the current directory in the search path by using "." in the new setting.

Export Demo

Export Demo is a Visual Basic program that provides an easy-to-use graphical user interface to the Export technology. It allows you to select files, convert them to HTML, and view the result in a browser object. The output options that control the look of the output files are predefined in Export Demo and cannot be changed in the user interface.

Export Demo accesses the Export functionality by returning to the operating system and running a C program named `cnv2html`. To adapt the sample program to your needs, modify the GUI by using Visual Basic, and modify the `cnv2html` program by using C. For more information on the C program, see htmlini, on page 123.

To launch Export Demo, select **Export Demo** from **Start** | **Programs** | **Autonomy** | **Export SDK** | **HTML Export**.

The source code for the program is in the directory `install\htmlexport\programs\ExportDemo`, where `install` is the path name of the Export installation directory. Export Demo is for Windows only.

See Use the Export Demo Program, on page 41 for more information.

Template Wizard

The Template Wizard is a Visual Basic program that provides an easy-to-use graphical user interface to the Export technology. It is an example of how to use most of the properties, methods, and events available in the COM Automation Server. See COM Interface Methods and Events, on page 255 and COM Interface Properties, on page 260 for more information on the properties and methods.

The Template Wizard converts documents based on the predefined templates. Several templates are supplied with HTML Export, and they can be customized to suit your needs. See Set Conversion Options by Using the Template Files, on page 38 for details on the template files.

> **NOTE:** The Template Wizard requires a COM server to be registered. See Use the COM Implementation of the API, on page 48.

You can use the Template Wizard to modify existing template files to create your own customized files, and to convert documents to HTML. However, the Template Wizard does not allow you to modify all structures in the template files. To control some display options, you might have to modify the template files directly or use the API.

Convert Documents to HTML by Using the Template Wizard

To convert a document to HTML

1. Launch HTML Export Template Wizard.
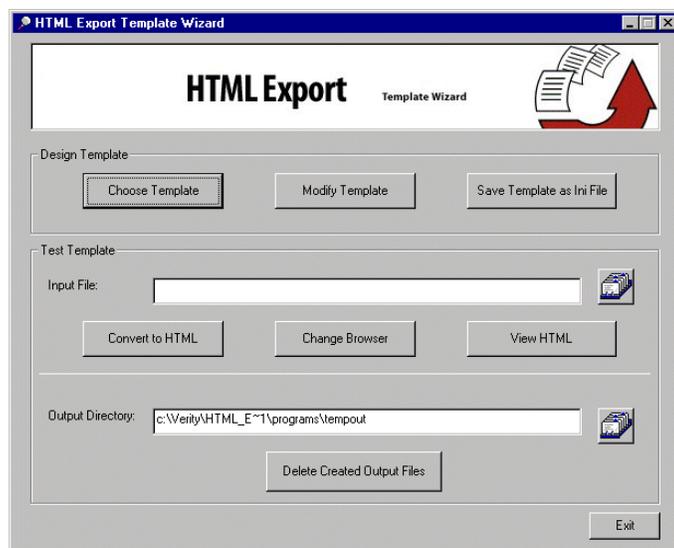
2. Click **Choose Template**.

3. Select one of the templates listed in the Choose Template dialog box, or click the browse button to select another template file. The template files are located in the `programs\ini` directory. Click **OK**.

4. Select an input file by using the browse button next to the **Input File** field.

5. Click **Convert to HTML** to produce an HTML file of the source document.

6. Click **View HTML** to open the converted file in your browser. To specify which browser is used to view the converted HTML files, click **Change Browser**.

   When you change the default browser, the location of the browser's executable file is written to a file named `browser.wiz`. If the browser is not changed, the default browser is the Visual Basic web browser object\OCX.

Change the Output Directory

By default, the output file is stored in the `programs\tempout` directory. To change this directory, click the browse button next to the **Output Directory** field and select a new directory.

If you change the default output directory, you must make the following modifications to the templates:

1. Update the HTML markup so that any images used by the templates (such as backgrounds or corporate logos) are found by the browser. The markup contains relative references to the image files (`..\images`).

2. Update the templates so that the Java applet (`JVTree.jar`) is found.

   > **NOTE:** `JVTree.jar` is used to display a expandable table of contents. JVTree, on page 126 for more information on `JVTree.jar`.

To delete the HTML and image files created in the output directory, click **Delete Created Output Files**.

Modify a Template in the Wizard

The Template Wizard provides limited control over the templates. You might wish to modify the template files directly in a text editor to provide maximum customization of the converted HTML. See Set Conversion Options by Using the Template Files, on page 38.

To modify the default options specified in the Wizard

1. Click **Modify Template**.

2. Modify the properties in the **Options** dialog box. The template parameters associated with each option in the **Options** dialog box are listed below and are described in HTML Export API Structures, on page 200.

Wizard field	Template parameter
All Formats tab	
Use text color in the original document	`bUseDocumentColors`
Use font information in the original document	`bUseDocumentFontInfo`
Use font face attributes in the document	`bSupportFontFace`
Set font size relative to base font size	`bDisplayRelativeFontSize`
Support user font size mapping	`bSupportUserFontSizeMapping`
User Font Size Map	`FontSizeMap`
Emit `<td>$nbsp;</td>` for empty cells	`bNbspEmptyCells`
Support "col=x" HTML parameter	`bSupportRFC1942_cols`

Wizard field	Template parameter
Convert tabs to tables	`bTabsToTables`
Table Border	`SATableBorder`
Table Cell Width	`nTableBorderWidth`
Output Language ID	`OutputLanguageID`
Style Sheet Type	`eStyleSheetType`
Replace Character	`cReplaceChar`
Redact Character	`cRedact`
Word Processing tab	
Minimum length of paragraph	`minParaLen`
Maximum length of paragraph	`maxParaLen`
Minimum font size	`fontSizeMin`
Maximum font size	`fontSizeMax`
Required space before paragraph	`mSpaceBefore`
Required space after paragraph	`mSpaceAfter`
Must be bold	`bMustBeBold`
Must be italic	`bMustBeItalic`
Must be underlined	`bMustBeUnderlined`
Can be non-zero indent	`bNonZeroIndent`
Cannot contain tab	`bNoTabs`
Cannot contain two or more spaces	`bNoMultiSpaces`
Heading Create Options	`headingCreateType`
Force source character set	`bForceSrcCharSet`
Source Character Set	`eSrcCharSet`
Force output character set	`bForceOutputCharSet`
Output Character Set	`OutputCharSet`
Spreadsheet tab	
Support column width	`bSupportColumnWidth`
Generate column headings	`bSupportColumnHeadings`
Generate row headings	`bSupportRowHeadings`

Wizard field	Template parameter
Support cell span	`bSupportCellSpan`
Support row span	`bSupportRowSpan`
Remove empty columns	`bRemoveEmptyColumns`
Enable empty rows	`bEnableEmptyRows`
Specify maximum rows per table	`nRowsBeforeSplit`
Presentation/Image tab	
Output any raster images to the following format	`OutputRasterGraphicType`
Output any vector images to the following format	`OutputVectorGraphicType`
Resolution of output image in x direction	`cxVectorToRasterXRes`
Resolution of output image in y direction	`cyVectorToRasterYRes`
Output presentation graphics as text and images	`bRasterizeFiles`
No pictures	`bNoPictures`
Compression quality	`nCompressionQuality`
Paragraph Styles tab	
Style Name	`StyleName`
MarkUpStart	`MarkUpStart`
MarkUpEnd	`MarkUpEnd`
Heading 1	`Flags=KVSTYLE_HEADING1` (see Flags for Defining Styles, on page 86)
Heading 2	`Flags=KVSTYLE_HEADING2` (see Flags for Defining Styles, on page 86)
Heading 3	`Flags=KVSTYLE_HEADING3` (see Flags for Defining Styles, on page 86)
Heading 4	`Flags=KVSTYLE_HEADING4` (see Flags for Defining Styles, on page 86)
Heading 5	`Flags=KVSTYLE_HEADING5` (see Flags for Defining Styles, on page 86)
Heading 6	`Flags=KVSTYLE_HEADING6` (see Flags for Defining Styles, on page 86)
PRE	`Flags=KVSTYLE_PRE` (see Flags for Defining Styles, on page 86)

Wizard field	Template parameter
Redact	`Flags=KVSTYLE_REDACT` (see Flags for Defining Styles, on page 86)
Order List	`Flags=KVSTYLE_ORDERLIST` (see Flags for Defining Styles, on page 86)
Unordered List	`Flags=KVSTYLE_UNORDERLIST` (see Flags for Defining Styles, on page 86)
Delete Content	`Flags=KVSTYLE_DELETECONTENT` (see Flags for Defining Styles, on page 86)
On Consecutive Paragraphs	`Flags=ONCONSECUTIVEPARAGRAPHS` (see Flags for Defining Styles, on page 86)

3. You can use the **Paragraph Styles** tab to map paragraph and character styles in any word processing document that contains styles (such as Microsoft Word, RTF, or Folio Flat File) to arbitrary markup.

4. To create a new template that preserves any modifications that have been made, click **Save Template as Ini File**.

comsamp

This Visual Basic program illustrates how to use the properties, methods, and events of the ActiveX controls from within Visual Basic. Before you can use this sample you must:

- Register an `htmserv` COM object. See the registration instructions in Use the COM Implementation of the API, on page 48.

- If required, specify the location of the HTML Export binaries and the default directory containing source files.

More detailed information on properties, methods, and events is found in COM Interface Methods and Events, on page 255 and COM Interface Properties, on page 260.

htmloop

The `htmloop` sample program demonstrates how to use the COM interface from within a C++ application. Using the COM interface allows C++ developers to take advantage of out-of-process HTML conversion.

To run `htmloop`, type the following command:

`htmloop install, inifile, inputfile, outputfile.htm`

where:

`install` is the directory where the Export libraries are installed.

`inifile` is the full path and file name of a template file.

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

outputfile.htm is the full path and file name of the HTML output file.

> **NOTE:** You must separate the arguments with commas (*,*). You must **not** enclose directory names that contain spaces in quotation marks. This program does not indicate when the conversion has finished.

Part III: C API Reference

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

- File Extraction API Functions

- File Extraction API Structures

- HTML Export API Functions

- HTML Export API Callback Functions

- HTML Export API Structures

- Enumerated Types

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 conversion. 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(hKVExport,
(char*)"KVGetExtractInterface");
/*Initialize file extraction interface structure using KVStructInit*/
KVStructInit(&extractInterface);
/* Retrieve file extraction interface */
error = (*fpKVGetExtractInterface)(pExport,&extractInterface))
```

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 155.

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 conversion functions.

Example

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

fpGetSubFileInfo()

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

Syntax

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

Arguments

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

Returns

- If the file information is retrieved, the return value is `KVERR_Success`.

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

Discussion

- After the subfile information is retrieved, call fpFreeStruct() to free the memory allocated by this function.

- If the root node is *not* enabled, the first subfile is index `0`. If the root node is enabled, the first subfile is index `1`. The root node is required to recreate a file's hierarchy. See Create a Root Node, on page 56.

- 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 56.

- 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 156.

- 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 136.

Example

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

fpGetSubFileMetaData()

This function extracts metadata from mail stores, mail messages, and non-mail items. See Extract Mail Metadata, on page 57.

Syntax

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

Arguments

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

metaArg   A pointer to the KVGetSubFileMetaArg structure, which defines metadata tags whose values are retrieved.

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

metaData   A pointer to the KVSubFileMetaData structure, which contains the retrieved metadata values.

Returns

- If the metadata is retrieved, the return value is KVERR_Success.

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

Discussion

- KeyView can extract a predefined set of common subfile metadata fields for all mail formats, and can extract all metadata from EML, MBX, MIME, NSF, ICS, and DXL files. To extract the common metadata fields, pass in 0 for `metaArg->metaNameCount`, and `NULL` for `metaArg->metaNameArray`. To extract all metadata, pass in `-1` for `metaArg->metaNameCount` and `NULL` for `metaArg->metaNameArray`. For more information, see Extract Mail Metadata, on page 57.

- 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 conversion.

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(pExport,&openArg,&pFile)))
{
   extractInterface->fpCloseFile(pFile);
   pFile = NULL;
}
```

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 135. 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 135.

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 136. 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;
}
KVExtractSubFileArgRec, *KVExtractSubFileArg;
```

Member Descriptions

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 204.
`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 Discussion, on page 149.
`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 Discussion, on page 149.

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 64.

- 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.

- KVExtractionFlag_SanitizeAbsolutePaths

  If you set this flag, KeyView ensures that the file is extracted to a location

within the extract directory (`extractDir`), even if an absolute path is supplied using `filePath`. When KeyView sanitizes a path and the resulting directory does not exist, extraction fails unless you instruct KeyView to create the directory, so you might also want to set the flag `KVExtractionFlag_CreateDir`. For more information, see Sanitize Absolute Paths, on page 55.

`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 Discussion, below.
`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 Discussion, below.

Discussion

- If the document character set is detected and is also specified in `srcCharset`, the detected character set is overridden by the specified character set. If the source character set is *not* detected and is *not* specified, character set conversion does not occur. The Supported Formats, on page 284 section lists the formats for which the source character set can be determined.

- The `KVSubFileExtractInfoFlag_CharsetConverted` flag in the KVSubFileExtractInfo structure indicates whether the character set of the subfile was converted during extraction.

- The following applies when the output is to a file:

  ○ If `filePath` is a valid absolute path, the file is extracted to the specified path and `extractDir` is ignored. However, if you have set the flag `KVExtractionFlag_SanitizeAbsolutePaths` the output path is modified to ensure it is within the `extractDir`. For more information, see Sanitize Absolute Paths, on page 55.

  ○ 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 154.

  ○ 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 204.
`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 Discussion, below.
`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 284 lists the formats for which the source character set can be determined.

- KeyView can extract a predefined set of common subfile metadata fields for all mail formats, and can extract all metadata from EML, MBX, MIME, NSF, ICS, and DXL files. To extract the common metadata fields, pass in 0 for metaArg->metaNameCount, and NULL for metaArg->metaNameArray. To extract all metadata, pass in -1 for metaArg->metaNameCount and NULL for metaArg->metaNameArray. For more information, see Extract Mail Metadata, on page 57.

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 204.
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 200.
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 converted. 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 converted).

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 conversion functions. See HTML Export API Functions, on page 161.

- 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 154.

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 204.
`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 201. 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 147.
`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 56 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 204.
`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 200.
	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 204.
`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 74.
`subFileType`	The subfile's position in the container file's hierarchy.

- `KVSubFileType_Main`—The subfile is at the top level of the main file. This is the default subfile type. See Discussion, on page 159.

- `KVSubFileType_Attachment`—The subfile is an attachment in a file.

- `KVSubFileType_OLE`—The subfile is an embedded OLE object in a compound document.

- `KVSubFileType_Folder`—The subfile is a folder or the artificial root node (see Create a Root Node, on page 56).

- `KVSubFileType_UncategorisedImage`—An embedded image that has not been categorized by the reader.

- `KVSubFileType_EmbeddedImage`—An embedded image.

- `KVSubFileType_EmbeddedIcon`—An icon used to represent an embedded file.

- `KVSubFileType_EmbeddedContent`—An image used to represent content for an embedded file. This could be an preview image of the actual content, or another representation such as an icon.

- `KVSubFileType_EmbeddedPreview`—A preview of an embedded file. This is usually an image that shows part of the embedded file.

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

`subFileSize`	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 143 and fpExtractSubFile(), on page 136.

- 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 56.

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 204.
`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: HTML Export API Functions

This section describes the functions in the HTML Export API. These functions manage the input and output streams, and perform the document conversion. Each function appears as a function prototype followed by a description of its arguments, its return value, and discussion of its use.

KVHTMLGetInterfaceEx()

> **NOTE:**
> This function has been superseded by KVHTMLGetInterfaceEx2(); `KVHTMLInterfaceEx2()` should be used instead of `KVHTMLInterfaceEx()`.

This is exported by the Export definition file. It supplies function pointers to other Export functions. When `KVHTMLGetInterfaceEx()` is called, it assigns the function pointers in the `KVHTMLInterfaceEx`

structure to other functions described in this chapter. For example, `KVHTMLInterfaceEx.fpInit` is assigned to point to `KVHTMLInitEx()`.

Syntax

```
void pascal KVHTMLGetInterfaceEx (KVHTMLInterfaceEx *pInterface);
```

Arguments

pInterface   A pointer to the structure `KVHTMLInterfaceEx`. See KVHTMLInterfaceEx, on page 212.

Returns

None.

Discussion

- One of the initial steps in using the HTML Export API is to create an instance of a `KVHTMLInterfaceEx` structure and use this function to gain access to other functions.

- You can call the API functions directly. For example, you can call `KVHTMLGetSummaryInfo()` instead of using `fpGetSummaryInfo()` in `KVHTMLInterfaceEx`. However, Micro Focus recommends that you assign the function pointers in `KVHTMLInterfaceEx` to the functions for efficiency.

KVHTMLGetInterfaceEx2()

This function is exported by the Export definition file. It supplies function pointers to other Export functions. When `KVHTMLGetInterfaceEx2()` is called, it assigns the function pointers in the structure `KVHTMLInterfaceEx2` to other functions described in this chapter. For example, `KVHTMLInterfaceEx2.fpInit` is assigned to point to `KVHTMLInitEx()`.

Syntax

```
BOOL pascal KVHTMLGetInterfaceEx2 (KVHTMLInterfaceEx2 *pInterface);
```

Arguments

pInterface   A pointer to the structure `KVHTMLInterfaceEx2`. See KVHTMLInterfaceEx2, on page 214.

Returns

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

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

If the function fails, all function pointers in pInterface are set to NULL.

You must initialize pInterface by calling KVStructInit prior to passing it to
KVHTMLGetInterfaceEx2. If you do not do this, the function fails.

Discussion

- One of the initial steps in using the HTML Export API is to create an instance of a
  KVHTMLInterfaceEx2 structure and use this function to gain access to other functions.

- The API functions can be called directly. For example, you can call KVHTMLGetSummaryInfo()
  instead of using fpGetSummaryInfo() in KVHTMLInterfaceEx2. However, Micro Focus
  recommends that you assign the function pointers in KVHTMLInterfaceEx2 to the functions for
  efficiency.

- You must initialize KVHTMLInterfaceEx2 by calling KVStructInit prior to passing it to
  KVHTMLGetInterfaceEx2, otherwise KVHTMLGetInterfaceEx2 fails.

Example

```
KVHTMLInterfaceEx2 KVHTMLInt;
BOOL (pascal *fpGetInterfaceEx2)(KVHTMLInterfaceEx2 *);
...
KVStructInit(&KVHTMLInt);
(*fpGetInterfaceEx2)(&KVHTMLInt);
```

fpConvertStream()

This function converts either a source stream or file to an output stream.

Syntax

```
BOOL pascal fpConvertStream(
    void                *pContext,
    void                *pCallingContext,
    KVInputStream       *pInput,
    KVOutputStream      *pOutput,
    KVHTMLTemplateEx    *pTemplatesEx,
    KVHTMLOptionsEx     *pOptionsEx,
    KVHTMLCallbacksEx   *pCallbacksEx,
    KVHTMLTOCOptions    *pTOCCreateOptions,
```

```
BOOL                    bIndex,
KVErrorCode             *pError );
```

Arguments

pContext	A pointer returned from `fpInit()`.
pCallingContext	A pointer passed back to the callback functions.
pInput	A pointer to the developer-assigned instance of `KVInputStream`. The `KVInputStream` structure defines the input stream that contains the source for the conversion. See KVInputStream, on page 201.
pOutput	A pointer to the developer-assigned instance of `KVOutputStream`. The `KVOutputStream` structure defines the output stream to which Export writes the generated HTML. See KVOutputStream, on page 202.
pTemplatesEx	A pointer to the `KVHTMLTemplateEx` data structure. It defines the overall structure of the output. Individual elements within the structure define the markup written at specific points in the output stream. See KVHTMLTemplateEx, on page 225.
	If this pointer is `NULL`, the default values for the structure are used.
pOptionsEx	A pointer to the `KVHTMLOptionsEx` data structure. It defines the options that control the markup written in response to the general style and attributes (font, color, and so on) of the document. See KVHTMLOptionsEx, on page 216.
	If this pointer is `NULL`, the default values for the structure are used.
pCallbacksEx	A pointer to the `KVHTMLCallbacksEx` data structure. It is a structure of functions that Export calls for specific, user-defined purposes. See KVHTMLCallbacksEx, on page 208.
	If callbacks are not used, this can be `NULL`.
pTOCCreateOptions	A pointer to the `KVHTMLTOCOptions` data structure. It specifies whether a heading is included in the table of contents. See KVHTMLTOCOptions, on page 230.
	If this pointer is `NULL`, the default values for the structure are used.
bIndex	Set `bIndex` to **TRUE** to generate output with minimal markup and without images. Because the generated output is minimized to textual content, it is suitable for an indexing engine. If you set `bIndex` to **FALSE**, embedded images in a document are regenerated as separate files and stored in the output directory.
	You can also set this option through the bNoPictures, on page 223 member in the template files.
pError	A pointer to an error code if the call to `fpConvertStream()` fails.

Returns

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

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

Discussion

- Only pContext, pInput, pOutput, and bIndex are required. All other pointers should be NULL when they are not set.

- If pCallbacksEx is NULL, pOptionsEx->pszDefaultOutputDirectory must be valid, except when bIndex is set to TRUE.

- This function runs in-process or out of process. See Convert Files Out of Process, on page 29.

- When converting out of process, this function must be called after the call to KVHTMLStartOOPSession() and before the call to KVHTMLEndOOPSession(). See KVHTMLStartOOPSession(), on page 191 and KVHTMLEndOOPSession(), on page 187.

- When converting out of process, the values for the KVHTMLTemplateEx, KVHTMLOptionsEx, and KVHTMLTOCOptions structures should be set to NULL. These structures are already passed in the call to KVHTMLStartOOPSession(). See KVHTMLStartOOPSession(), on page 191.

Example

The following sample code is from the cnv2html sample program:

```
if(!(*KVHTMLInt.fpConvertStream)(
            pKVHTML,        /* A pointer returned by fpInit()   */
            NULL,           /* A pointer for callback functions */
            &Input,         /* Input stream                     */
            &Output,        /* Output stream                    */
            &HTMLTemplates, /* Markup and related variables     */
            &HTMLOptions,   /* Options                          */
            NULL,           /* A pointer to callback functions  */
            NULL,           /* TOC options                      */
            FALSE,          /* Index mode                       */
            &error))        /* Error return value               */
{
   printf("Error converting %s to HTML %d\n", argv[i - 1], error);
}
else
{
   printf("Conversion of %s to HTML completed.\n\n", argv[i - 1]);
}
```

fpFileToInputStreamCreate()

This function creates an input stream from an input file.

Syntax

```
BOOL pascal _export 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 converted.

pInput   A pointer to the developer-assigned instance of `KVInputStream`. The `KVInputStream` structure defines the input stream that contains the source for the conversion. See KVInputStream, on page 201.

Returns

- If the call is successful, the return value is `TRUE`.

- If this call is unsuccessful, the return value is `FALSE`. Processing is halted.

Discussion

After the conversion is complete, call `fpFileToInputStreamFree()` to free the memory allocated by this function.

Example

The following sample code is from the `cnv2html` sample program:

```
if(!(*KVHTMLInt.fpFileToInputStreamCreate)(pKVHTML, argv[i++], &Input))
    {
        printf("Error creating input stream\n");
        (*KVHTMLInt.fpShutDown)(pKVHTML);
        mpFreeLibrary(hKVHTML);
        return (5);
    }
```

fpFileToInputStreamFree()

This function frees the memory used to create an input stream.

Syntax

```
BOOL pascal _export fpFileToInputStreamFree(
    void           *pContext,
    KVInputStream  *pInput);
```

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 for the conversion. See
           KVInputStream, on page 201.

Returns

- If the call is successful, the return value is `TRUE`.

- If this call is unsuccessful, the return value is `FALSE`. Processing is halted.

Discussion

After the conversion is complete, call this function to free the memory allocated by
`fpFileToInputStreamCreate()`.

fpFileToOutputStreamCreate()

This function creates an output stream from an output file.

Syntax

```
BOOL pascal _export fpFileToOutputStreamCreate(
    void            *pContext,
    char            *pszFileName,
    KVOutputStream  *pOutput );
```

Arguments

pContext   A pointer returned from `fpInit()`.

pszFileName   A pointer to the name of the output file to create.

pOutput   A pointer to the developer-assigned instance of `KVOutputStream`. The
                `KVOutputStream` structure defines the output stream to which Export writes the
                generated HTML. See KVOutputStream, on page 202.

Returns

- If the call is successful, the return value is `TRUE`.

- If this call is unsuccessful, the return value is `FALSE`. Processing is halted.

Discussion

After the conversion is complete, call `fpFileToOutputStreamFree()` to free the memory allocated by
this function.

Example

The following sample code is from the `cnv2html` sample program:

```
if (!(*KVHTMLInt.fpFileToOutputStreamCreate)(pKVHTML, argv[i], &Output))
   {
      printf("Error creating output stream\n");
      (*KVHTMLInt.fpFileToInputStreamFree)(pKVHTML, &Input);
      (*KVHTMLInt.fpShutDown)(pKVHTML);
      mpFreeLibrary(hKVHTML);
      return 6;
   }
```

fpFileToOutputStreamFree()

This function frees the memory used to create the output stream.

Syntax

```
BOOL pascal _export fpFileToOutputStreamFree(
    void               *pContext,
    KVOutputStream     *pOutput );
```

Arguments

pContext A pointer returned from `fpInit()`.

pOutput   A pointer to the developer-assigned instance of `KVOutputStream`. The `KVOutputStream`
structure defines the output stream to which Export writes the generated HTML. See
KVOutputStream, on page 202.

Returns

- If the call is successful, the return value is `TRUE`.

- If this call is unsuccessful, the return value is `FALSE`. Processing is halted.

Discussion

After the conversion is complete, call this function to free the memory allocated by
`fpFileToOutputStreamCreate()`.

fpGetAnchor()

This function gets the file name automatically generated by Export and used for external graphics
referenced with `<img>` tags and for heading-level table of contents entries.

Syntax

```
BOOL pascal fpGetAnchor(
    void                *pCallingContext,
    KVHTMLAnchorTypeEx   eAnchorTypeEx,
    KVXMLAnchorType      eAnchorType,
    char                *pszAnchor,
    int                  cbAnchorMax,
    BYTE                *pcHTML,
    UINT                 cbHTML);
```

Arguments

pCallingContext A pointer passed back to the callback functions.

eAnchorTypeEx   The graphic or block anchor type for the output stream. It must be one of the
enumerated types defined in `KVHTMLAnchorTypeEx`. See
KVHTMLAnchorTypeEx, on page 242.

pszAnchor   A pointer to the location in which the new anchor is stored.

cbAnchorMax	The maximum number of bytes to place in `pszAnchor`.
pcHTML	A pointer to either the markup defining the contents of the table of contents entry, a pointer to the external graphic name, or `NULL`.
cbHTML	The number of valid bytes in `pcHTML`.

Returns

- If the call is successful, the return value is `TRUE`.

- If this call is unsuccessful, the return value is `FALSE`. Processing is halted.

Discussion

- `pszAnchor` must be assigned. It might be derived from the `cbAnchorMax`, `pcHTML`, and `cbHTML` values that are also provided.

- `pcHTML` can be `NULL` if the graphic is an internal part of the document.

- This function is exposed so that it can be called from the `GetAnchor()` callback function to obtain default behavior for anchor types the callback is not set to handle.

fpGetConvertFileList()

This function gets the list of files automatically converted to HTML during a call to `fpConvertStream()` or `KVHTMLConvertFile()`.

Syntax

```
char ** pascal _export fpGetConvertFileList(
    void    *pContext,
    int     *pnSize );
```

Arguments

pContext	A pointer returned from `fpInit()`.
pnSize	A pointer to the number of files generated by the conversion.

Returns

If no files are converted, the return value is a `NULL` pointer. Otherwise, the return value is a pointer to an array of strings that provides the available path information for each converted file.

Discussion

- The array of file path information includes all externally generated files, including graphic files. Note that the main output file is not included in the array, nor in the count of the number of files converted.

- The memory used by the array of file path information is freed by the API.

- The array is not valid after a call to `fpShutDown()`.

- This function runs in-process or out of process. See Convert Files Out of Process, on page 29.

- When converting out of process, this function must be called after the call to `KVHTMLStartOOPSession()` and before the call to `KVHTMLEndOOPSession()`. See KVHTMLStartOOPSession(), on page 191 and KVHTMLEndOOPSession(), on page 187.

fpGetKvErrorCode

This function gets an extended error code defined in `KVErrorCode`. If a KeyView HTML Export function fails, you can call `fpGetKvErrorCode()` to find extra information on the failure.

Syntax

```
KVErrorCode pascal fpGetKvErrorCode (
    void            *pContext );
```

Arguments

`pContext`   A pointer returned from `fpInit()`. See fpInit(), on page 174.

Returns

The current error code.

Discussion

If there has not been a failure, this function returns `KVERR_Success`.

fpGetKvErrorCodeEx

This function gets an extended error code defined in `KVErrorCodeEx`. It is called to provide additional information when `fpGetKvErrorCode()` returns the error `KVERR_General`.

Syntax

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

Arguments

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

Returns

The current extended error code.

fpGetStreamInfo()

This function extracts file format and character set information from the source document.

Syntax

```
BOOL pascal _export fpGetStreamInfo (
    void            *pContext,
    KVInputStream   *pInput,
    KVStreamInfo    *pStreamInfo );
```

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 for the conversion. See .

pStreamInfo A pointer to the developer-assigned instance of `KVStreamInfo`. The `KVStreamInfo` structure defines the input stream document type and character set. See .

You can examine the fields in the structure to determine the appropriate template to use based on the document type.

Returns

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

fpGetSummaryInfo()

This function extracts all metadata from the input stream. See Extract Metadata, on page 76 for more information.

Syntax

```
BOOL pascal _export fpGetSummaryInfo(
    void                *pContext,
    KVInputStream       *pInput,
    KVSummaryInfoEx     *pSummary,
    BOOL                 bFree );
```

Arguments

pContext   A pointer returned from `fpInit()`.

pInput   A pointer to the developer-assigned instance of `KVInputStream`. The `KVInputStream` structure points to the input stream that contains the source for the conversion. See KVInputStream, on page 201.

pSummary   A pointer to the developer-assigned instance of `KVSummaryInfoEx`.

           In this 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`. See KVSummaryInfoEx, on page 206.

bFree   A flag to free or fill the memory allocated to the document metadata.

Returns

- If the call is successful, the return value is TRUE. When the document does *not* contain metadata, but the document reader can extract metadata from the specified format, this function returns TRUE with `nElem` set to `0`.

- If this call is unsuccessful, the return value is FALSE. This function returns FALSE when the document reader does not support metadata extraction for the specified format, or there is an error in extraction. The section Supported Formats, on page 284 lists the file formats for which metadata can be determined.

Discussion

- For metadata to be extracted by Export, metadata must be defined in the source document, and the document reader must be able to extract metadata for the file format. Supported Formats, on page 284 lists the file formats for which metadata can be determined. Export does not generate metadata automatically from the document contents.

- This function runs in-process or out of process. See Convert Files Out of Process, on page 29.

- You can call this function at any time after the call to `fpInit()`.

- When converting out of process, this function must be called after the call to `KVHTMLStartOOPSession()` and before the call to `KVHTMLEndOOPSession()`. See KVHTMLStartOOPSession(), on page 191 and KVHTMLEndOOPSession(), on page 187.

- Call this function with `bFree` set to **FALSE** to return an array of `KVSummaryInfoEx` structures, each containing an element of available document metadata.

- After processing the information in the structure, call this function with `bFree` set to **TRUE** to free the memory allocated to the document metadata.

fpInit()

This function initializes an Export session. Its return value, `pContext`, is passed as the first parameter to the File Extraction interface and all other Export functions.

Syntax

```
void* pascal _export fpInit(
    KVMemoryStream      *pMemAllocator,
    char                *pszKeyViewDir,
    char                *pszDataFile,
    KVErrorCode         *pError,
    DWORD                dWord);
```

Arguments

pMemAllocator	A pointer to a developer-defined memory allocator. If NULL is passed, the default C run-time memory allocation is used.
pszKeyViewDir	A pointer to the directory where the Export components are located. This is normally the directory $install\OS\bin$, where $install$ is the path name of the Export installation directory and OS is the name of the operating system.
pszDataFile	A pointer to the directory and file name of the Export data file, `formats_e.ini`. This file determines whether a format is supported. If a format does not exist in this file, the conversion fails.
	The `formats_e.ini` file is normally stored in the directory $install\OS\bin$, where $install$ is the path name of the Export installation directory and OS is the name of the operating system. See File Format Detection, on page 393 for more information.
pError	A pointer to an error code defined in `KVErrorCode` or `KVErrorCodeEx` in `kvtypes.h`. See KVErrorCode, on page 236 and KVErrorCodeEx, on page 238.
dWord	Reserved. Must be 0.

Returns

- If the call is successful, the return value is a pointer passed to all other functions.

- If the call is unsuccessful, the return value is a `NULL` pointer.

Discussion

- If `pszKeyViewDir` is `NULL`, the required components cannot be found. Ensure that it is valid.

- If this function returns `NULL`, check `stderr` for the KeyView installation error messages, "`KeyView Export SDK License Key has Expired`" and "`KeyView Export SDK License Key is Invalid`", and pass them to your application. See the *Export SDK Installation Instructions* for more information on the KeyView license feature.

- To ensure multithreaded conversions 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. Creating a context pointer for every thread does not affect performance because the context pointer uses minimal resources.

- When the conversion context is no longer required, it should be terminated by calling `fpShutdown()`. See fpShutDown(), on the next page.

Example

The following sample code is from the `cnv2html` sample program:

```
pKVHTML = (*KVHTMLInt.fpInit)(NULL, ".", NULL, &error, 0);
if(!pKVHTML)
    {
       printf("Error initializing KVHTML: %d\n", error);
       mpFreeLibrary(hKVHTML);
       return 4;
    }
```

fpSetStyleMapping()

This function is used to set the mapping for user-defined styles. Export does not make a distinction between paragraph styles or character styles, but operates under the assumption that each style has a unique name.

Syntax

```
BOOL pascal _export fpSetStyleMapping(
    void       *pContext,
    KVStyle    *pStyles,
```

```
int         iStyles,
BOOL        bCopy);
```

Arguments

pContext   A pointer returned from `fpInit()`.

pStyles   A pointer to the developer-assigned instance of `KVStyle`. See KVStyle , on page 205.
           The `KVStyle` structure defines the elements of a custom style.

iStyles   The number of elements in the `pStyles` array.

bCopy   If Export is to allocate memory to copy the `pStyles` array, set this to **TRUE**. If `pStyles`
           remains valid throughout the conversion process, set this to **FALSE**.

Returns

- If the call is successful, the return value is `TRUE`.

- If this call is unsuccessful, the return value is `FALSE`.

Discussion

- Paragraph styles are presently implemented only for documents in Microsoft Word, RTF, Folio Flat
  files, WordPro, and WordPerfect 6.x.

- This function runs in-process or out of process. See Convert Files Out of Process, on page 29.

- When converting out of process, this function must be called after the call to
  `KVHTMLStartOOPSession()` and before the call to `KVHTMLEndOOPSession()`. See
  KVHTMLStartOOPSession(), on page 191 and KVHTMLEndOOPSession(), on page 187.

- After this API function is called, the styles are valid until `fpShutDown()` is called, or until this
  function is called again with a new style or `NULL`.

fpShutDown()

This function terminates an Export session that was initialized by `fpInit()`, and frees allocated
system resources. It is called when the conversion context is no longer required.

Syntax

```
void pascal _export fpShutDown(KVHTMLContext *pContext);
```

Arguments

pContext   A pointer returned from `fpInit()`.

Returns

None.

Discussion

After this function is called, the `pContext` pointer must not be passed to any HTML Export API.

fpValidateTemplate()

This function is used to ensure that the markup in the structures is valid. It is currently not implemented.

KVHTMLConfig()

This function is called directly and provides a way to configure options prior to document conversion. You can use this function to:

- **Enable PDF conversion to JPEG**

  Enable the graphic-based PDF reader `kppdfrdr` to convert PDF documents to JPEG files.

- **Configure PDF bookmarks**

  Specify whether bookmarks in a PDF file are used to create a table of contents in the HTML output.

- **Configure rotated text**

  Specify whether rotated text is displayed in its original position or at the bottom of the page. Currently, this option applies only to PDF files.

- **Designate temporary directory**

  Specify a directory in which temporary files created during the conversion process are stored.

  > **NOTE:** 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.

- **Configure XML conversion**

  Specify the elements and attributes extracted from an XML document based on a file's document type.

- **Enable PDF logical reading order**

  Convert paragraphs in PDF files in the order in which they appear on the PDF page and with left-to-right or right-to-left paragraph direction. See Convert PDF Files to a Logical Reading Order, on page 96.

- **Configure PDF soft hyphens**

  Specify whether soft hyphens in a PDF file are removed from the HTML output. See Control Hyphenation, on page 102.

- **Enable revision marks**

  Convert text and graphics that were deleted from a document with revision tracking enabled and include revision information in the HTML output. See Include Revision Information, on page 89.

- **Enable empty image tags**

  Prevent graphics from being converted and generate image tags with empty `src` attributes. This makes the conversion faster, but, because placeholders are generated for the graphics, maintains the text flow of the original document. This is similar to the `bNoPictures` parameter; however, `bNoPictures` does not generate an image tag. See bNoPictures, on page 223.

- **Toggle hidden data output from Microsoft Word, Excel, and PowerPoint documents**

  Show or hide information from hidden sources such as comments or slides. See Show Hidden Data, on page 113.

- **Enable a PDF invisible text toggle button**

  Enable a JavaScript button that toggles the display of invisible text and regular content in exported PDF documents. Toggle Invisible Text, on page 100.

- **Specify opacity of invisible text in PDFs**

  Specify the opacity of invisible text in exported PDF documents, from 0 (invisible) to 100 (fully visible). See Specify Opacity of Invisible Text, on page 101.

- **Protected file password**

  Specify the password to use to open a password-protected file for export.

- **Specify output character set for summary information**

  Specify the output character set for the document's metadata, when using `fpGetSummaryInfo()`.

- **Enable tabbed spreadsheet view**

  Enables a tabbed navigation view for spreadsheets.

- **Enable previews for large spreadsheets**

  Limits the number of rows, columns, and sheets that are exported to HTML.

Syntax

```
KVErrorCode pascal KVHTMLConfig(
    void    *pContext,
    int      nType,
    int      nValue,
    void    *p );
```

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 Configuration Flags for KVHTMLConfig(), on the next page.

nValue   The integer value defined for the flags above. This is `TRUE` or `FALSE` for all flags except:

- `KVCFG_LOGICALPDF`—`nValue` is one of the paragraph direction options defined in the `LPDF_DIRECTION` enumerated type in `kvtypes.h`. See LPDF_DIRECTION, on page 251.

- `KVCFG_SETTEMPDIRECTORY`—`nValue` is not set.

- `KVCFG_SETXMLCONFIGINFO`—`nValue` is not set.

- `KVCFG_SETINVISTEXTTOGGLE`—`nValue` is not set.

- `KVCFG_SETINVISTEXTOPACITY`—`nValue` is an integer that specifies text opacity, from 0 (invisible) to 100 (fully visible).

- `KVCFG_SETMETADATACHARSET`—`nValue` is a character set enumerated in `KVCharSet` of `kvtypes.h`. See Convert Character Sets, on page 80.

p   The data for the configuration flag. This is `NULL` for all flags except:

- `KVCFG_SETTEMPDIRECTORY`—This is a pointer to a path to the directory where temporary files are stored.

- `KVCFG_SETXMLCONFIGINFO`—This is a pointer to the `KVXConfigInfo` structure. See KVXConfigInfo, on page 207.

- `KVCFG_INCLREVISIONMARK`—This is a pointer to the `KVRevisionMark` structure. See KVRevisionMark, on page 231.

- `KVCFG_SETINVISTEXTTOGGLE`—This is a null-terminated string that determines the toggle button name.

- `KVCFG_SETPASSWORD`—This is the source file password.

Returns

The return value is one of the error codes defined in `KVErrorCode` in `kvtypes.h`.

Discussion

- You must call this function after the call to `fpInit()` and before the call to `fpConvertStream()` or `KVHTMLConvertFile()`.

- This function runs in-process or out of process. See Convert Files Out of Process, on page 29.

- When converting out of process, you must call this function after the call to

KVHTMLStartOOPSession() and before the call to KVHTMLEndOOPSession().
See KVHTMLStartOOPSession(), on page 191 and KVHTMLEndOOPSession(), on page 187.

- The configuration flags are described in the following table.

Configuration Flags for KVHTMLConfig()

Flag	Description
KVCFG_SETHIFIPDF	This flag enables the graphic-based PDF reader kppdfrdr to convert PDF documents. See Convert PDF Files to Raster Images, on page 95.
	By default, Export uses the basic PDF reader, pdfsr, to convert PDF
KVCFG_ SETMETADATACHARSET	This flag enables you to specify the output character set for metadata when using fpGetSummaryInfo(). nValue is a character set enumerated in KVCharSet of kvtypes.h. See Convert Character Sets, on page 80. You should call this function before fpGetSummaryInfo().
KVCFG_ SUPPRESSTOCPRINTIMAGE	If you set KVCFG_SUPPRESSTOCPRINTIMAGE, bookmarks in a PDF file are not used to generate a table of contents in the HTML output. By default, the table of contents is generated from bookmarks within the PDF file. See Generate a Table of Contents from PDF Bookmarks, on page 99.
KVCFG_SETTEXTROTATE	If you set KVCFG_SETTEXTROTATE, rotated text in a file is displayed at 0 degrees at the bottom of the page on which it appears. The page is enlarged to accommodate the text.
	By default, rotated text in a file is displayed in its original position, at the original font size, and at 0 degrees rotation. Because the text is the original size, but might be displayed in a smaller space, the text might overlap adjacent text in the HTML output. You use the KVCFG_ SETTEXTROTATE option to avoid this problem. See Convert Rotated Text, on page 101.
	HTML markup does not support text rotation.
KVCFG_SETTEMPDIRECTORY	This flag enables you to specify the directory in which temporary files created during conversion processes are stored. By default, the system temporary directory is used.
	To define a directory for temporary files generated during an out-of-process conversion, set the tempfilepath parameter in the formats_ e.ini file. Convert Files Out of Process, on page 29.
	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.
KVCFG_SETXMLCONFIGINFO	This flag enables you to define which elements and attributes are

Configuration Flags for KVHTMLConfig(), continued

Flag	Description
	extracted from XML documents with a specified format ID or root element. You can use this to override the default settings for the supported XML formats (see Convert XML Files, on page 108), or to define settings for custom XML document types. The settings are defined in the `KVXConfigInfo` structure (seeKVXConfigInfo, on page 207). To set custom settings for more than one document type, call the `KVHTMLConfig()` 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 108.
KVCFG_LOGICALPDF	This flag converts paragraphs in a PDF file in the order in which they appear on the page (logical reading order). The `nValue` argument specifies the paragraph direction. See Convert PDF Files to a Logical Reading Order, on page 96.
KVCFG_DELSOFTHYPHEN	If you set this flag, soft hyphens in the source document are removed, and the hyphenated words are joined in the HTML output. By default, soft hyphens are maintained. See Control Hyphenation, on page 102. Micro Focus recommends that you remove soft hyphens if you use Export to generate text output for an indexing engine or are not concerned with maintaining the document's layout. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185 for more information on running Export in index mode.
KVCFG_INCLREVISIONMARK	If you set this flag to `TRUE`, text and graphics that were deleted from a document with revision tracking enabled are converted, and revision information (revision title, reviewer name, and revision date and time) is included in the HTML output. To reset the flag and exclude deleted content and revision information from the HTML output, set the flag to `FALSE`. See Include Revision Information, on page 89. The default is `FALSE`.
KVCFG_BLANKPICTURE	If you set this flag to `TRUE`, graphics in a document are not converted, but an image tag is generated with an empty `src` attribute, creating an empty placeholder for the graphic. For example: `<img src="" height="136" width="101">` This allows you to generate output without graphics, but still maintain the text flow of the original document.

Configuration Flags for KVHTMLConfig(), continued

Flag	Description
	This option applies to word processing formats only. The default is FALSE.
KVCFG_WP_NOCOMMENTS	Set this flag to **TRUE** not to export text from comments in Microsoft Word documents. Comment text is exported by default from Microsoft Word 97 to 2003 files.
	You can also toggle the display of comment output by modifying the formats_e.ini file. See Show Hidden Data, on page 113.
KVCFG_WP_ SHOWHIDDENTEXT	Set this flag to **TRUE** to export hidden text from Microsoft Word documents.
KVCFG_WP_ SHOWDATEFIELDCODE	Set this flag to **TRUE** to export date field codes from Microsoft Word documents.
KVCFG_WP_ SHOWFILENAMEFIELDCODE	Set this flag to **TRUE** to export the file name field code from Microsoft Word documents.
KVCFG_SS_ SHOWHIDDENINFOR	Set this flag to **TRUE** to export hidden information from Microsoft Excel files.
KVCFG_SS_SHOWCOMMENTS	Set this flag to **TRUE** to export comments from Microsoft Excel files.
KVCFG_SS_SHOWFORMULA	Set this flag to **TRUE** to export formulas from Microsoft Excel files.
KVCFG_PG_ HIDEHIDDENSLIDE	Set this flag to **TRUE** not to export hidden slides from Microsoft PowerPoint files.
KVCFG_PG_HIDECOMMENT	Set this flag to **TRUE** not to export comments from Microsoft PowerPoint files. Comments are exported by default from PowerPoint 97 to 2000 files.
KVCFG_PG_ SHOWCOMMENTSSLIDE	Set this flag to **TRUE** to export comments slides from Microsoft PowerPoint 2003 and 2007 files.
KVCFG_PG_SHOWSLIDNOTES	Set this flag to **TRUE** to export slide notes from Microsoft PowerPoint files.
	You can also toggle slide note output by modifying the formats_e.ini file. See Show Hidden Data, on page 113.
KVCFG_ SETPDFINVISTEXTTOGGLE	This flag enables a JavaScript button in exported PDF documents, which you can use to show and hide invisible text.
	Invisible text is hidden by default. See Toggle Invisible Text, on page 100.
KVCFG_ SETPDFINVISTEXTOPACITY	This flag allows you specify the degree of invisible text opacity in exported PDFs, from 0 (invisible) to 100 (invisible). Use this option if

Configuration Flags for KVHTMLConfig(), continued

Flag	Description
	you want to view both the invisible text and the rasterized image in the document. Invisible text opacity is set to 0 by default. See Specify Opacity of Invisible Text, on page 101.
KVCFG_SETPASSWORD	This flag enables you to define a password used to open a password-protected file for export. See Export Password Protected Files, on page 412. nValue is TRUE. p is the source file password, which can have a maximum length of 255 characters (the final byte is null).
KVCFG_TABNAVIGATION	If you set this flag to TRUE, it enables a tabbed navigation view for spreadsheets. A row of tabs is displayed at the bottom of the browser window, and enables the user to switch between multiple sheets in a workbook. **NOTE:** JavaScript must be enabled.
KVCFG_SS_PREVIEW	Specifies whether to export a preview for large spreadsheets rather than exporting all content. Web browsers might take a long time, or fail completely, to render spreadsheets with large numbers of cells. If you set this flag to TRUE, KeyView limits the numbers of rows, columns, and sheets that are exported to HTML.

Examples

- To specify that the graphic-based PDF reader is used to convert PDF files:

  ```
  (*fpHTMLConfig)(pKVHTML, KVCFG_SETHIFIPDF, TRUE, NULL);
  ```

- To specify that bookmarks in a PDF file are not used to generate a table of contents in the HTML output:

  ```
  (*fpHTMLConfig)(pKVHTML, KVCFG_SUPPRESSTOCPRINTIMAGE, TRUE, NULL);
  ```

- To specify that rotated text in a file is displayed at 0 degrees at the bottom of the page on which it appears:

  ```
  (*fpHTMLConfig)(pKVHTML, KVCFG_SETTEXTROTATE, TRUE, NULL);
  ```

- To set a directory for temporary files:

```
char    tmpDir[250];
strcpy (tmpDir, "c:\\temp\\htmlexport");
(*fpHTMLConfig)(pKVHTML, KVCFG_SETTEMPDIRECTORY, 0, tmpDir);
```

- To specify custom extraction settings for conversion of an XML file:

```
KVXConfigInfo    xinfo;
(*fpHTMLConfig)(pKVHTML, KVCFG_SETXMLCONFIGINFO, 0, &xinfo);
```

- To specify that PDF files are converted to a logical reading order, and the paragraph direction for the PDF output is left to right:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_LOGICALPDF, LPDF_LTR, NULL);
```

- To specify that PDF files are converted to a logical reading order, and the paragraph direction for the PDF output is right to left:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_LOGICALPDF, LPDF_RTL, NULL);
```

- To specify that PDF files are converted to a logical reading order, and the paragraph direction for the PDF output is determined automatically for each page:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_LOGICALPDF, LPDF_AUTO, NULL);
```

- To specify that soft hyphens are removed from the HTML output:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_DELSOFTHYPHEN, TRUE, NULL);
```

- To convert text and graphics that are identified by revision marks:

```
KVRevisionMark    RMark;
(*fpHTMLConfig)(pKVHTML, KVCFG_INCLREVISIONMARK, TRUE, &RMark))
```

- To generate a placeholder for all pictures:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_BLANKPICTURE, TRUE, NULL);
```

- To toggle hidden data output from Microsoft Word documents, use one of the KVCFG_WP flags:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_WP_NOCOMMENTS, TRUE, NULL);
```

- To toggle hidden data output from Microsoft Excel documents, use one of the KVCFG_SS flags:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SS_SHOWHIDDENINFOR, TRUE, NULL);
```

- To toggle hidden data output from Microsoft PowerPoint documents, use one of the KVCFG_PG flags:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_PG_HIDEHIDDENSLIDE, TRUE, NULL);
```

- To enable an invisible text toggle button in exported PDF documents:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SETPDFINVISTEXTTOGGLE, 0, szButtonName);
```

where szButtonName is a null-terminated string that determines the button name.

- To specify the opacity of invisible text in exported PDF documents:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SETPDFINVISTEXTOPACITY, iInvisOpacity, NULL);
```

where iInvisOpacity is an integer from 0 (invisible) to 100 (fully visible).

- To specify a password to open a password-protected file for export:

  `(*fpHTMLConfig)(pKVHTML, KVCFG_SETPASSWORD, TRUE, password);`

  where `password` is a null-terminated string of 255 or fewer characters.

- To produce summary information in UTF8:

  `(*fpHTMLConfig)(pKVHTML, KVCFG_SETMETADATACHARSET, KVCS_UTF8, NULL);`

- To export only a preview of spreadsheets to HTML:

  `(*fpHTMLConfig)(pKVHTML, KVCFG_SS_PREVIEW, TRUE, NULL);`

KVHTMLConvertFile()

This function is called directly and converts a source file to an output file.

Syntax

```
BOOL pascal KVHTMLConvertFile  (
    void                    *pContext,
    void                    *pCallingContext,
    char                    *pInFileName,
    char                    *pOutFileName,
    KVHTMLTemplateEx        *pTemplatesEx,
    KVHTMLOptionsEx         *pOptionsEx,
    KVHTMLCallbacksEx       *pCallbacksEx,
    KVHTMLTOCOptions        *pTOCCreateOptions,
    BOOL                     bIndex,
    KVErrorCode             *pError)
```

Arguments

`pContext`	A pointer returned from `fpInit()`.
`pCallingContext`	A pointer passed back to the callback functions.
`pInFileName`	A pointer to the input file.
`pOutFileName`	A pointer to the output file.
`pTemplatesEx`	A pointer to the `KVHTMLTemplateEx` data structure. It defines the overall structure of the output. Individual elements within the structure define the markup written at specific points in the output stream. See KVHTMLTemplateEx, on page 225.
	If this pointer is `NULL`, the default values for the structure are used.
`pOptionsEx`	A pointer to the `KVHTMLOptionsEx` data structure. It defines the options that control the markup written in response to the general style and attributes (font,

color, and so on) of the document. See KVHTMLOptionsEx, on page 216.

If this pointer is NULL, the default values for the structure are used.

pCallbacksEx	A pointer to the KVHTMLCallbacksEx data structure. It is a structure of functions that Export calls for specific, user-defined purposes. See KVHTMLCallbacksEx, on page 208.
	If you do not use callbacks, this can be NULL. Only the Continue() callback can be used with KVHTMLConvertFile().
pTOCCreateOptions	A pointer to the KVHTMLTOCOptions data structure. It specifies whether a heading is included in the table of contents. See KVHTMLTOCOptions, on page 230.
	If this pointer is NULL, the default values for the structure are used.
bIndex	Set this to **TRUE** to generate output with minimal markup and without images. Because the generated output is minimized to textual content, it is suitable for an indexing engine. If you set bIndex to **FALSE**, embedded images in a document are regenerated as separate files and stored in the output directory.
	You can also set this through the bNoPictures, on page 223 member in the template files.
pError	A pointer to an error code if the call to KVHTMLConvertFile() fails.

Returns

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

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

Discussion

- Only pContext, pInFileName, pOutFileName, and bIndex are required. All other pointers should be NULL when they are not set.

- If pCallbacksEx is NULL, pOptionsEx->pszDefaultOutputDirectory must be valid, except when you set bIndex to **TRUE**.

- This function runs in-process or out of process. See Convert Files Out of Process, on page 29.

- When converting out of process, you must call this function after the call to KVHTMLStartOOPSession() and before the call to KVHTMLEndOOPSession(). See KVHTMLStartOOPSession(), on page 191 and KVHTMLEndOOPSession(), on the next page.

- When converting out of process, the values for the KVHTMLTemplateEx, KVHTMLOptionsEx, and KVHTMLTOCOptions structures should be set to NULL. These structures are already passed in the call to KVHTMLStartOOPSession(). SeeKVHTMLStartOOPSession(), on page 191.

Example

```
if(!(*KVHTMLInt.KVHTMLConvertFile)(
            pKVHTML,        /* A pointer returned by fpInit()   */
            NULL,           /* A pointer for callback functions */
            &InputFile,     /* Input file                       */
            &OutputFile,    /* Output file                      */
            &HTMLTemplates, /* Markup and related variables   */
            &HTMLOptions,   /* Options                          */
            NULL,           /* A pointer to callback functions  */
            NULL,           /* TOC options                      */
            FALSE,          /* Index mode                       */
            &error))        /* The error return value           */
{
   printf("Error converting %s to HTML %d\n", argv[i - 1], error);
}
else
{
   printf("Conversion of %s to HTML completed.\n\n", argv[i - 1]);
}
```

KVHTMLEndOOPSession()

This function terminates the current out-of-process conversion session, and releases the source data and resources related to the session.

Syntax

```
BOOL pascal KVHTMLEndOOPSession(
    void           *pContext,
    BOOL            bKeepServantAlive,
    KVErrorCodeEx  *pError
    DWORD           dwOptions,
    void           *pReserved1,
    void           *pReserved2 );
```

Arguments

pContext   A pointer returned from `fpInit()`.

bKeepServantAlive   Set this to **TRUE** to keep a Servant process active after the Export out-of-process session is terminated. If the Servant remains active, subsequent conversion requests are processed more quickly because the Servant is already prepared to receive data.

	Set this to **FALSE** to terminate the Export out-of-process session and the associated Servant process.
pError	A pointer to an error code defined in KVErrorCodeEx in kvtypes.h.
dwOptions	Reserved for future use.
pReserved1	Reserved for future use.
pReserved2	Reserved for future use.

Returns

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

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

Example

The following sample code is from the cnv2htmloop sample program:

```
/* declare endsession function pointer */
BOOL (pascal *fpKVHTMLEndOOPSession)( void  *,
                BOOL                    ,
                KVErrorCode         *,
                DWORD                   ,
                void                *,
                void                *);
/* assign OOP endsession function pointer */
fpKVHTMLEndOOPSession = (BOOL (pascal *)( void  *,
                BOOL                    ,
                KVErrorCode         *,
                DWORD                   ,
                void                *,
                void                * ))mpGetProcAddress(hKVHTML,
"KVHTMLEndOOPSession");
if(!fpKVHTMLEndOOPSession)
{
   printf("Error assigning KVHTMLEndOOPSession() pointer\n");
   (*KVHTMLInt.fpFileToInputStreamFree)(pKVHTML, &Input);
   (*KVHTMLInt.fpFileToOutputStreamFree)(pKVHTML, &Output);
   mpFreeLibrary(hKVHTML);
   return 8;
}
/********END OOP SESSION, DO NOT KEEP SERVANT ALIVE ********/
if(!(*fpKVHTMLEndOOPSession)(pKVHTML,
        FALSE,
        &error,
        0,
        NULL,
```

```
        NULL))
{
    printf("Error calling fpKVHTMLEndOOPSession \n");
    (*KVHTMLInt.fpFileToInputStreamFree)(pKVHTML, &Input);
    (*KVHTMLInt.fpFileToOutputStreamFree)(pKVHTML, &Output);
    (*KVHTMLInt.fpShutDown)(pKVHTML);
    mpFreeLibrary(hKVHTML);
    return 10;
}
```

KVHTMLSetHighlight()

This function is called directly and enables you to specify search terms that are found and highlighted in the HTML output. See Search and Highlight Terms, on page 88. The `htmlini` sample program demonstrates this function. See htmlini, on page 123.

Syntax

```
int pascal_export KVHTMLSetHighlight(
    void            *pContext,
    KVHTMLHighlight *pHLConfig,
    void            *pReserved1,
    void            *pReserved2 );
```

Arguments

pContext   A pointer returned from `fpInit()`.

pHLConfig   A pointer to the `KVHTMLHighlight` data structure. This structure defines the terms to be found and the highlight format applied. See KVHTMLHighlight, on page 211.

pReserved1   Reserved for future use.

pReserved2   Reserved for future use.

Returns

- If the call is successful, the return value is `KVERR_Success`.

- If the call is unsuccessful, the return value is an error code.

Discussion

- This function must be called after the call to `fpInit()` and before the call to `fpConvertStream()` or `KVHTMLConvertFile()`.

- When converting out of process, this function must be called before the call to `KVHTMLStartOOPSession()`. See .

Example

```
KVHTMLHighlight      HTMLHighlight;
int (pascal *fpHTMLSetHighlight)(void *, KVHTMLHighlight *, void *, void *);
//get function pointer and call the function
fpHTMLSetHighlight = (int (pascal *)
   (void *, KVHTMLHighlight *, void *, void*))myGetProcAddress(hKVHTML,
"KVHTMLSetHighlight");
   if(!fpHTMLSetHighlight)
   {
      printf("Error accessing HTMLSetHighlight().\n");
   }
   else
   {
      if(KVERR_Success != (*fpHTMLSetHighlight)(pKVHTML, &amp;HTMLHighlight, NULL,
NULL))
      {
         printf("Error setting HTML highlight.\n");
      }
   }
}
```

KVHTMLSetStyleSheet()

This function is called directly and is used to specify the full path and file name of an external Cascading Style Sheet (CSS).

Syntax

```
BOOL pascal KVHTMLSetStyleSheet(
    void    *pContext,
    char    *pszStyleSheetName,
    char    *pszRef);
```

Arguments

pContext	A pointer returned from `fpInit()`.
pszStyleSheetName	A pointer to the full path and file name of the style sheet.
pszUrlRef	A pointer to the URL or file name of style sheet.

Returns

- If the call is successful, the return value is `TRUE`.

- If this call is unsuccessful, the return value is `FALSE`.

Discussion

- When the value for `eStyleSheetType` in `KVHTMLOptionsEx` is set to CSS_TOFILE, and the token `$STYLESHEET` is specified in the templates, an external CSS file is referenced in the output HTML by a LINK statement of the form:

  ```
  <LINK rel="STYLESHEET" href="pszRef" type="text/css">
  ```

- If the name of the style sheet is not specified by using this function, a style sheet file is created with an automatically generated file name.

- If this function is used to specify the name of the style sheet file, that file name is used in the conversion.

    ○ If the file does not exist in the specified location, it is created.

    ○ If the file exists, but is empty, CSS styles are written to the file.

    ○ If the file exists and is not empty, the conversion attempts to use the predefined styles, and appends any new styles that are required for the conversion.

- If the value for `pszStyleSheetName` includes the output directory, the `href` only consists of the file name because the HTML output resides in the same directory as the CSS file.

- If the value for `pszStyleSheetName` points to a directory other than the output directory, the `href` consists of the full path and file name.

- If the value for `pszStyleSheetName` points to a file that is not a CSS file or to a non-existent directory, the LINK statement is written; but, the style sheet information is added inline (`CSS_INLINE`), and an external CSS file is not generated.

- If there are multiple calls made to `fpConvertStream()` or `KVHTMLConvertFile()`, and the name of the style sheet is set with `KVHTMLSetStyleSheet`, the file name can be disabled by calling `KVHTMLSetStyleSheet` again with the `pszStyleSheetName` and `pszRef` set to `NULL`. The file name can then be set to a different value by calling `KVHTMLSetStyleSheet` with the new file name prior to the next call to `fpConvertStream()` or `KVHTMLConvertFile()`.

- This function runs in-process or out of process. See Convert Files Out of Process, on page 29.

- When converting out of process, this function must be called after the call to `KVHTMLStartOOPSession()` and before the call to `KVHTMLEndOOPSession()`. See KVHTMLStartOOPSession(), below and KVHTMLEndOOPSession(), on page 187.

KVHTMLStartOOPSession()

This function performs the following:

- Initializes the out-of-process session.

- Specifies the input stream or file.

- Passes conversion options from the KVHTMLTemplateEx, KVHTMLOptionsEx, and KVHTMLTOCOptions data structures.

- Creates a Servant process.

- Establishes a communication channel between the application thread and the Servant.

- Sends the data to the Servant.

Syntax

```
BOOL pascal KVHTMLStartOOPSession(
    void                *pContext,
    KVInputStream       *pInputStream,
    char                *pFileName,
    KVHTMLTemplateEx    *pTemplatesEx,
    KVHTMLOptionsEx     *pOptionsEx,
    KVHTMLTOCOptions    *pTOCCreateOptions
    DWORD               *pPID,
    KVErrorCode         *pError
    DWORD                dwOptions,
    void                *pReserved1,
    void                *pReserved2 );
```

Arguments

pContext	A pointer returned from fpInit().
pInputStream	A pointer to the developer-assigned instance of KVInputStream. The KVInputStream structure defines the input stream containing the source for the conversion.
	If pInput is defined, then pFileName must be NULL. The input data can be defined as a data stream or file, but not both.
pFileName	A pointer to the file to be converted. The file must exist on the same file system as the Servant.
	If pFileName is defined, then pInput must be NULL. The input data can be defined as a data stream or file, but not both.
pTemplatesEx	A pointer to the KVHTMLTemplateEx data structure. It defines the overall structure of the output. Individual elements within the structure define the markup written at specific points in the output stream. See KVHTMLTemplateEx, on page 225.
	If this pointer is NULL, the default values for the structure are used.

pOptionsEx	A pointer to the KVHTMLOptionsEx data structure. It defines the options that control the markup written in response to the general style and attributes (font, color, and so on) of the document. See KVHTMLOptionsEx, on page 216.
	If this pointer is NULL, the default values for the structure are used.
pTOCCreateOptions	A pointer to the KVHTMLTOCOptions data structure. It specifies whether a heading is included in the table of contents. SeeKVHTMLTOCOptions, on page 230.
	If this pointer is NULL, the default values for the structure are used.
pPID	Address of a DWORD into which the Servant process ID is returned.
pError	A pointer to an error code defined in KVErrorCode in kvtypes.h.
dwOptions	Reserved for future use.
pReserved1	Reserved for future use.
pReserved2	Reserved for future use.

Returns

If the call is successful, the return value is TRUE.

If the call is unsuccessful, the return value is FALSE.

Discussion

- After the out-of-process session is started successfully, all conversion functions can be called. The data is then processed on the Servant until the session is terminated by a call to KVHTMLEndOOPSession(), on page 187.

- All functions that can run out of process must be called within the out-of-process session, that is, after the call to KVHTMLStartOOPSession(), and before the call to KVHTMLEndOOPSession().

- The KVHTMLConvertFile() function can only be called once in a single out-of-process session.

- Since the KVHTMLTemplateEx, KVHTMLOptionsEx, and KVHTMLTOCOptions data structures are passed by this function, the same pointers in the call to KVHTMLConvertFile() are ignored.

Example

The following sample code is from the cnv2htmloop sample program:

```
/* declare OOP startsession function pointer */
BOOL (pascal *fpKVHTMLStartOOPSession)( void      *,
              KVInputStream      *,
              char               *,
              KVHTMLTemplateEx   *,
              KVHTMLOptionsEx    *,
              KVHTMLTOCOptions   *,
```

```
                DWORD               *,
                KVErrorCode         *,
                DWORD                ,
                void                *,
                void                * );
/* assign OOP startsession function pointer */
fpKVHTMLStartOOPSession = (BOOL (pascal *)( void      *,
                KVInputStream       *,
                char                *,
                KVHTMLTemplateEx    *,
                KVHTMLOptionsEx     *,
                KVHTMLTOCOptions    *,
                DWORD               *,
                KVErrorCode         *,
                DWORD                ,
                void                *,
                void                * ))mpGetProcAddress(hKVHTML,
"KVHTMLStartOOPSession");
if(!fpKVHTMLStartOOPSession)
{
    printf("Error assigning KVHTMLStartOOPSession() pointer\n");
    (*KVHTMLInt.fpFileToInputStreamFree)(pKVHTML, &Input);
    (*KVHTMLInt.fpFileToOutputStreamFree)(pKVHTML, &Output);
    mpFreeLibrary(hKVHTML);
    return 7;
}
/********START OOP SESSION ****************/
if(!(*fpKVHTMLStartOOPSession)(pKVHTML,
        &Input,
        NULL,
        &HTMLTemplates,        /* Markup and related variables */
        &HTMLOptions,           /* Options */
        NULL,                  /* TOC options */
        &oopServantPID,
        &error,
        0,
        NULL,
        NULL))
{
    printf("Error calling fpKVHTMLStartOOPSession \n");
    (*KVHTMLInt.fpFileToInputStreamFree)(pKVHTML, &Input);
    (*KVHTMLInt.fpFileToOutputStreamFree)(pKVHTML, &Output);
    (*KVHTMLInt.fpShutDown)(pKVHTML);
    mpFreeLibrary(hKVHTML);
    return 9;
}
```

Chapter 9: HTML Export API Callback Functions

This section describes the HTML Export API callback functions.

Introduction

The `fpConvertStream()` and `KVHTMLConvertFile()` functions enable you to specify a callback function. A callback function controls the conversion while it is in progress. For example, you can specify a callback function to report progress during the conversion.

To use the API callback functions, declare one or more instances of the `KVHTMLCallbacksEx` structure. Each member of this instance can then be initialized by assigning a function pointer to the application-defined callback functions, cast to the appropriate function prototype. Each instance of `KVHTMLCallbacksEx` can define unique callback functions. Alternatively, the functions can be common to all instances of `KVHTMLCallbacksEx`; these functions take appropriate action, depending on the value of the pointer `pCallingContext`.

The second parameter (`pCallingContext`) of the call to `fpConvertStream()` and `KVHTMLConvertFile()` provides a void pointer used to identify the context of this call. If more than one call to `fpConvertStream()` or `KVHTMLConvertFile()` is made within a single application, any resulting callbacks are identified by the first parameter of the callback function. This enables the callback function to take any appropriate action, depending on which calling context is returned.

The seventh parameter (`pCallbacks`) of the call to `fpConvertStream()` and `KVHTMLConvertFile()` must be set to the address of the `KVHTMLCallbacksEx` structure to be used for this call.

For sample code, see the sample program `callback.c`. It creates a frame-based HTML stream and demonstrates the use of the callback functions.

Continue()

When `fpConvertStream()` or `KVHTMLConvertFile()` is called, control is not returned to the application until the entire document is processed. This callback function provides a means of monitoring progress and terminating the conversion process before the conversion is completed.

Syntax

```
BOOL (pascal *Continue) (
    void      *pCallingContext,
```

```
    int       nPercentComplete);
```

Arguments

pCallingContext   A pointer passed back to the caller-provided callback functions. This pointer, which can be NULL, is specified as the second parameter of the call to fpConvertStream() and KVHTMLConvertFile().

nPercentComplete   The approximate percentage of the current conversion that is completed.

You can monitor the progress of the conversion by checking the value of nPercentDone, which indicates how many blocks out of the total number of blocks have been processed.

Returns

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

- If the call is unsuccessful, the return value is FALSE. Processing is halted.

Discussion

- There is a callback to this function for every entry that appears in the generated table of contents.

- The application is free to execute any required code in the callback function, with the exception of fpShutDown().

GetAnchor()

This function gets the file name automatically generated by Export and used for external graphics referenced with <img> tags, heading-level table of contents entries, and external files (such as CSS files and revision summary files).

Syntax

```
BOOL (pascal *GetAnchor)  (
    void                    *pCallingContext,
    KVHTMLXMLAnchorTypeEx        eAnchorTypeEx,
    char                    *pszAnchor,
    int                      cbAnchorMax,
    BYTE                    *pcHTML,
    UINT                     cbHTML);
```

Arguments

pCallingContext	A pointer that gets passed back to the caller-provided callback functions. This pointer, which can be NULL, is specified as the second parameter of the call to fpConvertStream().
eAnchorTypeEx	The anchor type for the output stream. It must be one of the enumerated types defined in KVHTMLAnchorTypeEx.
pszAnchor	A pointer to the location where the new anchor is stored.
cbAnchorMax	The maximum number of bytes to place in pszAnchor.
pcHTML	This is either NULL or a pointer to one of the following:

- markup defining the contents of a table of contents entry

- the external graphic file name

- the external file name

cbHTML	The number of valid bytes in pcHTML.

Returns

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

- If the call is unsuccessful, the return value is FALSE. Processing is halted.

Discussion

- If this callback is NULL, default anchor names are generated. The generated names are unique across the document.

- This function is called once per block, block chunk, graphic anchor, or extra file. Any required code can be executed here as long as a unique value for pszAnchor is assigned. If this string is not unique, an existing file might be overwritten, producing undesirable results. The callback function should contain the functionality to verify whether files already exist.

- If you want to specify graphic anchor names, but use default anchor names for all other anchors, provide the graphic names when eAnchorTypeEx is VectorPictureAnchorEx or RasterPictureAnchorEx. For all other anchor types, call with the same parameters you were passed.

- pszAnchor must be assigned. It can be derived from the cbAnchorMax, pcHTML, and cbHTML values, which are also provided.

- pcHTML can be null if the graphic is an internal part of the document.

GetAuxOutput()

This callback function enables the calling application to specify an auxiliary output stream for a block or graphic.

Syntax

```
BOOL (pascal *GetAuxOutput) (
    void                    *pCallingContext,
    KVHTMLXMLAnchorTypeEx        eAnchorTypeEx,
    char                    *pszAnchor,
    KVOutputStream          *pNewOutput);
```

Arguments

pCallingContext	A pointer passed back to the caller-provided callback functions. This pointer, which can be NULL, is specified as the second parameter of the call to fpConvertStream().
eAnchorTypeEx	A graphic or block anchor as defined by the enumerated types in KVHTMLAnchorTypeEx.
pszAnchor	A pointer to location where a new anchor is stored. pszAnchor is based on the call to GetAnchor().
pNewOutput	A pointer to a KVOutputStream structure that can be used to write data to the current block.

Returns

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

- If the call is unsuccessful, the return value is FALSE. Processing is halted.

Discussion

- If GetAuxOutput() is NULL, the pszDefaultOutputDirectory member of the instance of KVHTMLOptionsEx is used as the base storage location for auxiliary output files. If pszDefaultOutputDirectory is also NULL, auxiliary files are placed in the current working directory.

- For each pszAnchor provided, create (malloc) an appropriate I/O structure. Assign pNewOutput->pOutputStreamPrivateData to point to that structure. Each remaining member of the KVOutputStream should then be initialized by assigning a function pointer to the additional application-defined functions, cast to the appropriate function prototype for Create(), Write(),

Seek(), Tell(), and Close(). Memory allocated to the I/O structure must be tracked and can be freed up within the call to Close(). See the callback.c sample program.

UserCB()

This callback function is triggered by including the $USERCB token in a member of KVHTMLTemplateEx. For example, placing "$USERCB=my_callback " in pszFirstH1Start results in a callback at the point when pszFirstH1Start is processed. The user callback function is identified by the text assigned to $USERCB, which in this example is my_callback. This identifier is passed to the argument pszUserCBid.

Syntax

```
BOOL (pascal *UserCB) (
    void              *pCallingContext,
    char              *pszUserCBid,
    KVOutputStream    *pNewOutput
    void              *pReserved);
```

Arguments

pCallingContext	A pointer that gets passed back to the caller-provided callback function. This pointer, which can be NULL, is specified as the second parameter of the call to fpConvertStream().
pszUserCBid	A pointer to a string that identifies the source of the callback. The identifier must be delimited by a trailing white space. For example, "my_callback ".
pNewOutput	A pointer to a KVOutputStream structure that can be used to write data to the current block.
pReserved	Reserved for future use.

Returns

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

- If the call is unsuccessful, the return value is FALSE. Processing is halted.

Chapter 10: HTML Export API Structures

This section provides information on the structures used by the HTML Export API. These structures are defined in `kvhtml.h`, `kvtypes.h`, and `adinfo.h`.

ADDOCINFO

This structure provides the format, file class, and version number of the source document. It is defined in `adinfo.h`, and is initialized by calling the `fpGetStreamInfo()` function. See fpGetStreamInfo(), on page 172.

```
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 `ENdocClass` enumerated type in `adinfo.h`.
`eFormat`	The major format of the source document (such as Microsoft Word or Corel Presentation) as defined by the `ENdocFmt` enumerated type in `adinfo.h`.
`lVersion`	The version number of the file format. The number is multiplied by 1000. For example, 1.02 is represented by 1020.
`ulAttributes`	Other attributes of the document as defined by the `ENdocAttributes` enumerated type in `adinfo.h`.

Discussion

When format detection is enhanced in future releases, new format IDs might be added to the `ENdocFmt` enumerated type. When you use 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.

KVInputStream

This structure defines an input stream for the HTML conversion.

```
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 HTML Export. It is initialized by calling `fpInit()`. See fpInit(), on page 174.

```
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.

Discussion

- `fpRealloc()` must handle a `NULL` pointer.

- For systems that do not support `fpRealloc()`, refer to the `callback` sample program, which demonstrates how to use the memory management features.

- If `KVMemoryStream` is not provided, the default C run-time memory allocation is used.

KVOutputStream

This structure defines an output stream for the HTML conversion.

```
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.

KVSTR

This structure is used to identify string types (string text and byte count) for the first three members of `KVStyle`. See KVStyle , on page 205.

```
typedef struct tag_KVSTR
{
    char    *pcString;
    int      cbString;
}
KVSTR;
```

Member Descriptions

pcString   A text string.

cbString   The length of `pcString`, excluding the terminating NULL(s). This allows UNICODE or double bytes to be employed.

KVStreamInfo

This structure defines a document's character set and format. It is initialized by calling `fpGetStreamInfo()`. See fpGetStreamInfo(), on page 172.

```
typedef struct tag_KVStreamInfo
{
    KVCharSet    charset;
    ADDOCINFO    adInfo;
}
KVStreamInfo;
```

Member Descriptions

charset   The character set of the source document, if that information is ascertainable. The available character sets are enumerated in `KVCharSet` in `kvtypes.h`. See Convert Character Sets, on page 80.

adInfo   The file class, major format, and version of the source document. A pointer to the `ADDOCINFO` structure. The structure of `ADDOCINFO` is defined in `adinfo.h`. See ADDOCINFO, on page 200.

- `adInfo.eClass` represents the class of the source document, as defined by the `ENdocClass` enumerated type.

- `adInfo.eFormat` represents the format of the source document, as defined by the `ENdocFmt` enumerated type.

- `adInfo.lVersion` represents the version number of the file format. The number is multiplied by 1000. For example, 1.02 is represented by 1020.

- `adInfo.ulAttributes` represents other attributes of the document as defined by the `ENdocAttributes` enumerated type.

Discussion

When format detection is enhanced in future releases, new format IDs might be added to the `ENdocFmt` enumerated type. When you use 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 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.

KVStructHead

This structure contains the current KeyView version number and is the first member of other structures. It enables Micro Focus to modify the structures in future releases, but to maintain backward compatibility. Before initializing 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

```
KVStructInit(&openArg);
```

KVStyle

This structure defines the style mapping support for KVSTR-defined styles. The first three members of KVStyle are KVSTR structures (see KVSTR, on page 203). Each KVSTR structure contains the text string and byte count for StyleName, MarkUpStart, and MarkUpEnd. The structure is initialized by calling the function fpSetStyleMapping().

See fpSetStyleMapping(), on page 175 and Map Styles, on page 84.

HTML Export supports both paragraph styles and character styles. It works on the assumption that each style has a unique name. Only one paragraph style can be active at one time; therefore, the opening of a new paragraph style automatically closes the previous paragraph style. By contrast, several character styles can be active at once. When HTML Export receives an EndCharStyle token from the format parser, the most recent character style is terminated.

```
typedef struct tag_KVStyles
{
    KVSTR     StyleName;
    KVSTR     MarkUpStart;
    KVSTR     MarkUpEnd;
    DWORD     dwFlags;
}
KVStyle;
```

Member Descriptions

StyleName	The name of the word processing style (for example, "Heading 1") to which style mapping applies. A pointer to the KVSTR structure. See KVSTR, on page 203. Style names are case sensitive.
MarkUpStart	The markup added to the beginning of a paragraph or character style. A pointer to the KVSTR structure. See KVSTR, on page 203.
MarkUpEnd	The markup added to the end of a paragraph or character style. A pointer to the KVSTR structure. See KVSTR, on page 203.
dwFlags	Instructions on how to process the content associated with a paragraph or character style. The flag can be one of the types defined in kvtypes.h. They are described in Flags for Defining Styles, on page 86. The value associated with each flag is a hexadecimal number. You can set an option by either entering the converted decimal value, or by entering the flag's text (for example, KVSTYLE_PRE). The value of Flags in the template files is passed to this member of KVStyle.

Discussion

- This structure applies to word processing documents only.

- By default, HTML Export maps the heading style "Heading 1" to `<h1></h1>`, and so on, for heading levels 1 through 6. If you use style mappings, the default mapping is overridden. Therefore, you must supply markup for *all* heading levels.

- When the user-defined markup in `KVStyle` conflicts with other markup generated by HTML Export, the user-defined markup takes precedence.

KVSumInfoElemEx

This structure defines the individual metadata elements.

```
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.

`type`   The data type of the metadata element. The types are defined in the `KVSumInfoType` structure in `kvtypes.h`. See KVSumInfoType, on page 247.

`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.

`pcType`   A pointer to the name of the metadata field.

KVSummaryInfoEx

This structure provides a count of the number of metadata elements, and a pointer to the first element of the array of individual elements. The structure is initialized by calling the `fpGetSummaryInfo()` function. See fpGetSummaryInfo(), on page 173.

```
typedef struct tag_KVSummaryInfoEx
{
    int                 nElem;
    KVSumInfoElemEx     *pElem;
}
KVSummaryInfoEx;
```

Member Descriptions

nElem The number of metadata elements contained in the array. `nElem` can be zero. This indicates that the document did not contain metadata, such as an ASCII text document.

pElem Points to the first element of the array of document metadata elements defined by the `KVSumInfoElemEx`structure. See KVSumInfoElemEx, on the previous page.

KVXConfigInfo

This structure defines an XML document type and the element extraction settings for that type. The settings can be applied based on the file format ID, or the file's root element. This structure is in `kvtypes.h` and is initialized by calling the `KVHTMLConfig()` function. See Convert XML Files, on page 108 and KVHTMLConfig(), on page 177.

```
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 `ENdocFmt` enumerated type in `adinfo.h`. See File Format Detection, on page 393 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 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 page 112.

pszInMeta   The elements extracted from the file as metadata. All other elements are extracted as text. Multiple entries must be separated by commas.

To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 112.

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 112.

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 112.

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 112.

pszInAttribute   The attribute values extracted from the file. If attributes are not defined, attribute values are not extracted. The namespace (if used), element name, and attribute name must be defined in the following format:

namespace:*elementname@attributename*

For example:

`microfocus:division@name`

KVHTMLCallbacksEx

This structure provides all callbacks that can result from a call to `fpConvertStream()` or `KVHTMLConvertFile()`. See fpConvertStream(), on page 163 and KVHTMLConvertFile(), on page 185. Any and all of the function pointers can be `NULL`.

```
typedef BOOL (pascal *KVHTMLCB_CONTINUE)(
    void                *pcallingContext,
    int                 nPercentDone);
typedef BOOL (pascal *KVHTMLCB_GETANCHOREX)(
    void                *pCallingContext,
```

```
      KVHTMLAnchorTypeEx      eAnchorTypeEx,
      char                    *pszAnchor,
      int                     cbAnchorMax,
      BYTE                    *pcHTML,
      UINT                    cbHTML);
typedef BOOL (pascal *KVHTMLCB_GETAUXOUTPUTEX) (
      void                    *pCallingContext,
      KVHTMLAnchorTypeEx      eAnchorTypeEx,
      char                    *pszAnchor,
      KVOutputStream          *pNewOutput);
typedef BOOL (pascal *KVHTMLCB_USERCBEX) (
      void                    *pCallingContext,
      char                    *psUserCBid,
      KVOutputStream          *pOutput,
      void                    *pReserved);
typedef struct tag_KVHTMLCallbacksEx
{
      KVHTMLCB_CONTINUE           Continue;
      KVHTMLCB_GETANCHOREX        GetAnchor;
      KVHTMLCB_GETAUXOUTPUTEX     GetAuxOutput;
      KVHTMLCB_USERCBEX           UserCB;
}
KVHTMLCallbacksEx;
```

Member Descriptions

- The members of this structure are pointers to the functions described in HTML Export API Callback Functions, on page 195.

- If `GetAuxOutput()` is `NULL`, the `pszDefaultOutputDirectory` member of the instance of `KVHTMLOptionsEx` is used as the base storage location for auxiliary output files. If `pszDefaultOutputDirectory` is also `NULL`, auxiliary files are placed in the current working directory. See KVHTMLOptionsEx, on page 216.

KVHTMLHeadingInfo

This structure defines how HTML Export creates heading information based on the source document's content and attributes. Source text is converted to a heading and included in the table of contents if

- it meets *all* the criteria defined by this structure, and

- the `headingCreateType` member of `KVHTMLTOCOptions` is set to allow automatic heading generation.

HTML Export evaluates the text against each member in the order in which the members appear below.

See KVHTMLTOCOptions, on page 230 for more information on automatic generation of headings.

When you convert PDF files to HTML by using the default reader, `pdfsr`, the table of contents is generated from "bookmarks" within the PDF file. This structure is not used. The table of contents appear either at the beginning of the HTML file or in a separate frame.

```
typedef struct tag_KVHTMLHeadingInfo
{
    int        minParaLen;
    int        maxParaLen;
    int        fontSizeMin;
    int        fontSizeMax;
    BOOL       bMustBeBold;
    BOOL       bMustBeItalic;
    BOOL       bMustBeUnderlined;
    BOOL       bNonZeroIndent;
    BOOL       bNoTabs;
    BOOL       bNoMultiSpaces;
    int        mSpaceBefore;
    int        mSpaceAfter;
}
KVHTMLHeadingInfo;
```

Member Descriptions

minParaLen	The minimum number of characters that text in the source document can contain for the text to meet the criteria for heading conversion.
	This option applies to word processing documents only.
	The default is 3 for heading levels 1 to 3.
maxParaLen	The maximum number of characters that text in the source document can contain for the text to meet the criteria for heading conversion.
	This option applies to word processing documents only.
	The default is 80 for heading levels 1 to 3.
fontSizeMin	The minimum font size of text in the source document for the text to meet the criteria for heading conversion.
	The default is 14 for heading level 1, and 12 for heading levels 2 and 3.
fontSizeMax	The maximum font size of text in the source document for the text to meet the criteria for heading conversion.
	The default is 20 for heading level 1, and 14 for heading levels 2 and 3.
bMustBeBold	If you set `bMustBeBold` to **TRUE**, the text in the source document must be bold to meet the criteria for heading conversion.
	The default is **TRUE** for heading levels 1 and 2, and **FALSE** for heading level 3.
bMustBeItalic	If you set `bMustBeItalic` to **TRUE**, the text in the source document must be italic to meet the criteria for heading conversion.

The default is `FALSE`.

bMustBeUnderlined	If you set `bMustBeUnderlined` to **TRUE**, the text in the source document must be underlined to meet the criteria for heading conversion. The default is `FALSE`.
bNonZeroIndent	If you set `bNonZeroIndent` to `TRUE`, the text in the source document must be indented to meet the criteria for heading conversion. If you set `bNonZeroIndent` to **FALSE**, the text must be aligned left. The default is `FALSE`.
bNoTabs	If you set `bNoTabs` to **TRUE**, the text in the source document must *not* contain tabs to meet the criteria for heading conversion. The default is `FALSE`.
bNoMultiSpaces	If you set `bNoMultiSpaces` to **TRUE**, the text in the source document must *not* contain two or more contiguous white spaces to meet the criteria for heading conversion. The default is `FALSE`.
mSpaceBefore	The amount of space in TWIPS (20th of a point) that must come before a paragraph in the source document for the text to meet the criteria for heading conversion. If –1 is used, the amount of space before the paragraph is not considered in the heading generation. The default is `0`.
mSpaceAfter	The amount of space in TWIPS (20th of a point) that must follow a paragraph in the source document for the text to meet the criteria for heading conversion. If –1 is used, the amount of space after the paragraph is not considered in the heading generation. The default is `0`.

KVHTMLHighlight

This structure defines the search terms that are found and highlighted in the HTML output. It is initialized by calling `KVHTMLSetHighlight()` and is defined in `kvtypes.h`. See .

```
typedef struct tag_KVHTMLHighlight
{
    KVStructHeader;
    char            **ppHLTerms;
    int             nSize;
    KVCharset       eCharset;
    char            *pHLStart;
    char            *pHLEnd;
    BOOL            bMatchCase;
```

```
    int           nReserved;
    void          *pReserved;
}
KVHTMLHighlight;
```

Member Descriptions

`KVStructHeader`	The KeyView version of the structure. See KVStructHead, on page 204.
`ppHLTerms`	An array of terms to be found and highlighted in the HTML output.
`nSize`	The number of terms to be found and highlighted in the HTML output.
`eCharset`	The character set of the term. The available character sets are enumerated in `KVCharSet` in `kvtypes.h`. See Convert Character Sets, on page 80.
	When `eCharset` is specified, the term is converted from this character set to the output character set when the character sets are different. If `eCharset` is not specified, the term's character set is assumed to be the output character set.
`pHLStart`	The start tag that specifies the text attributes used to highlight the text string. For example, you could specify
	`<font style="color:#ff0000;background-color:#00FF00;">`
`pHLEnd`	The end tag used to close the highlighting start tag. The end tag for the example above would be `</font>`
`bMatchCase`	This Boolean applies only to searches on documents with a target character set of 1252. If this Boolean is set, the text search is case sensitive. By default, the text search is case insensitive.
`nReserved`	Reserved for future use.
`pReserved`	Reserved for future use.

KVHTMLInterfaceEx

> **NOTE:**
> This structure has been superseded by KVHTMLInterfaceEx2; `KVHTMLInterfaceEx2` should be used instead of `KVHTMLInterfaceEx`.

The members of this structure are pointers to the API functions described in HTML Export API Functions, on page 161.

```
typedef void* (pascal *KVHTML_INITEX) (
    KVMemoryStream      *pMemAllocator,
    char                *pszKeyViewDir,
    char                *pszDataFile,
    KVErrorCode         *pError,
    DWORD                dWord);
```

```
typedef void (pascal *KVHTML_SHUTDOWN)(void*);
typedef BOOL (pascal *KVHTML_CONVERT_STREAMEX) (
    void                 *pContext,
    void                 *pCallingContext,
    KVInputStream        *pInput,
    KVOutputStream       *pOutput,
    KVHTMLTemplateEx     *pTemplatesEx,
    KVHTMLOptionsEx      *pOptionsEx,
    KVHTMLTOCOptions     *pTOCCreateOptions,
    KVHTMLCallbacksEx    *pCallbacksEx,
    BOOL                  bIndex,
    KVErrorCode          *pError);
typedef char** (pascal *KVHTML_GET_FILE_LIST)(
    void                 *pContext,
    int                  *pnSize );
typedef BOOL (pascal *KVHTML_GET_STREAM_INFO)(
    void                 *pContext,
    KVInputStream        *pInput,
    KVStreamInfo         *pStreamInfo );
typedef BOOL (pascal *KVHTML_GET_ANCHOREX) (
    void                 *pCallingContext,
    KVHTMLAnchorTypeEx    eAnchorTypeEx,
    char                 *pszAnchor,
    int                   cbAnchorMax,
    BYTE                 *pcHTML,
    UINT                  cbHTML);
typedef BOOL (pascal *KVHTML_INPUTSTREAM_CREATE) (
    void                 *pContext,
    char                 *pszFileName,
    KVInputStream        *pInput);
typedef BOOL (pascal *KVHTML_INPUTSTREAM_FREE) (
    void                 *pContext,
    KVInputStream        *pInput);
typedef BOOL (pascal *KVHTML_OUTPUTSTREAM_CREATE) (
    void                 *pContext,
    char                 *pszFileName,
    KVOutputStream       *pOutput );
typedef BOOL (pascal *KVHTML_OUTPUTSTREAM_FREE)(
    void                 *pContext,
    KVOutputStream       *pOutput );
typedef KVLanguageID (pascal *KVHTML_LANGUAGE_ID)(void *pContext);
typedef BOOL (pascal *KVHTML_GET_SUMMARY_INFO)(
    void                 *pContext,
    KVInputStream        *pInput,
    KVSummaryInfoEx      *pSummary,
    BOOL                  bFree );
typedef BOOL (pascal *KVHTML_SET_STYLE_MAPPING) (
    void                 *pContext,
    KVStyle              *pStyles,
```

```
    int                 iStyles,
    BOOL                bCopy);
typedef BOOL (pascal *KVHTML_VALIDATE_TEMPLATE)(
    void                *pContext,
    KVOutputStream      *pOutput,
    KVHTMLTemplate      *pTemplate,
    KVHTMLOptions       *pOptions,
    KVHTMLTOCOptions    *pTOCOptions,
    KVHTMLCallbacks     *pCallBalls,
    KVMemoryStream      *pMemStream)
typedef struct tag_KVHTMLInterfaceEx
{
    KVHTML_INITEX               fpInit;
    KVHTML_SHUTDOWN             fpShutDown;
    KVHTML_CONVERT_STREAMEX     fpConvertStream;
    KVHTML_GET_FILE_LIST        fpGetConvertFileList;
    KVHTML_GET_STREAM_INFO      fpGetStreamInfo;
    KVHTML_GET_ANCHOREX         fpGetAnchor;
    KVHTML_INPUTSTREAM_CREATE   fpFileToInputStreamCreate;
    KVHTML_INPUTSTREAM_FREE     fpFileToInputStreamFree;
    KVHTML_OUTPUTSTREAM_CREATE  fpFileToOutputStreamCreate;
    KVHTML_OUTPUTSTREAM_FREE    fpFileToOutputStreamFree;
    KVHTML_GET_SUMMARY_INFO     fpGetSummaryInfo;
    KVHTML_SET_STYLE_MAPPING    fpSetStyleMapping;
    KVHTML_VALIDATE_TEMPLATE    fpValidateTemplate;
}
            KVHTMLInterfaceEx;
```

Member Descriptions

The members of this structure are function pointers to the functions described in HTML Export API Functions, on page 161.

KVHTML_LANGUAGE_ID and KVHTML_VALIDATE_TEMPLATE are currently not implemented.

KVHTMLInterfaceEx2

The members of this structure are pointers to the API functions described in HTML Export API Functions, on page 161.

This structure supersedes KVHTMLInterfaceEx. KVHTMLInterfaceEx2 should be used instead of KVHTMLInterfaceEx.

Compared to KVHTMLInterfaceEx, KVHTMLInterfaceEx2 adds two functions for checking error codes, and allows for binary compatible extensibility in future releases.

```
typedef void* (pascal *KVHTML_INITEX) (
            KVMemoryStream *pMemAllocator,
            char *pszKeyViewDir,
```

```
        char *pszDataFile,
        KVErrorCode *pError,
        DWORD dWord);
typedef void (pascal *KVHTML_SHUTDOWN)(void*);
typedef BOOL (pascal *KVHTML_CONVERT_STREAMEX) (
        void *pContext,
        void *pCallingContext,
        KVInputStream *pInput,
        KVOutputStream *pOutput,
        KVHTMLTemplateEx *pTemplatesEx,
        KVHTMLOptionsEx *pOptionsEx,
        KVHTMLTOCOptions *pTOCCreateOptions,
        KVHTMLCallbacksEx *pCallbacksEx,
        BOOL bIndex,
        KVErrorCode *pError);
typedef char** (pascal *KVHTML_GET_FILE_LIST)(
        void *pContext,
        int *pnSize );
typedef BOOL (pascal *KVHTML_GET_STREAM_INFO)(
        void *pContext,
        KVInputStream *pInput,
        KVStreamInfo *pStreamInfo );
typedef BOOL (pascal *KVHTML_GET_ANCHOREX) (
        void *pCallingContext,
        KVHTMLAnchorTypeEx eAnchorTypeEx,
        char *pszAnchor,
        int cbAnchorMax,
        BYTE *pcHTML,
        UINT cbHTML);
typedef BOOL (pascal *KVHTML_INPUTSTREAM_CREATE) (
        void *pContext,
        char *pszFileName,
        KVInputStream *pInput);
typedef BOOL (pascal *KVHTML_INPUTSTREAM_FREE) (
        void *pContext,
        KVInputStream *pInput);
typedef BOOL (pascal *KVHTML_OUTPUTSTREAM_CREATE) (
        void *pContext,
        char *pszFileName,
        KVOutputStream *pOutput );
typedef BOOL (pascal *KVHTML_OUTPUTSTREAM_FREE)(
        void *pContext,
        KVOutputStream *pOutput );
typedef KVLanguageID (pascal *KVHTML_LANGUAGE_ID)(void *pContext);
typedef BOOL (pascal *KVHTML_GET_SUMMARY_INFO)(
        void *pContext,
        KVInputStream *pInput,
        KVSummaryInfoEx *pSummary,
        BOOL bFree );
```

```
typedef BOOL (pascal *KVHTML_SET_STYLE_MAPPING) (
        void *pContext,
        KVStyle *pStyles,
        int iStyles,
        BOOL bCopy);
typedef BOOL (pascal *KVHTML_VALIDATE_TEMPLATE)(
        void *pContext,
        KVOutputStream *pOutput,
        KVHTMLTemplate *pTemplate,
        KVHTMLOptions *pOptions,
        KVHTMLTOCOptions *pTOCOptions,
        KVHTMLCallbacks *pCallBalls,
        KVMemoryStream *pMemStream);
typedef KVErrorCode (pascal *KVHTML_GET_KV_ERROR_CODE)(void *);
typedef KVErrorCodeEx (pascal *KVHTML_GET_KV_ERROR_CODE_EX)(void *);

typedef struct tag_KVHTMLInterfaceEx2
{
        KVStructHeader;
        KVHTML_INITEX fpInit;
        KVHTML_SHUTDOWN fpShutDown;
        KVHTML_CONVERT_STREAMEX fpConvertStream;
        KVHTML_GET_FILE_LIST fpGetConvertFileList;
        KVHTML_GET_STREAM_INFO fpGetStreamInfo;
        KVHTML_GET_ANCHOREX fpGetAnchor;
        KVHTML_INPUTSTREAM_CREATE fpFileToInputStreamCreate;
        KVHTML_INPUTSTREAM_FREE fpFileToInputStreamFree;
        KVHTML_OUTPUTSTREAM_CREATE fpFileToOutputStreamCreate;
        KVHTML_OUTPUTSTREAM_FREE fpFileToOutputStreamFree;
        KVHTML_GET_SUMMARY_INFO fpGetSummaryInfo;
        KVHTML_SET_STYLE_MAPPING fpSetStyleMapping;
        KVHTML_VALIDATE_TEMPLATE fpValidateTemplate;
        KVHTML_GET_KV_ERROR_CODE      fpGetKvErrorCode;
        KVHTML_GET_KV_ERROR_CODE_EX  fpGetKvErrorCodeEx;
}
KVHTMLInterfaceEx2;
```

KVHTMLOptionsEx

This structure defines the options that control the HTML markup written in response to the general style
and attributes (font, color, and so on) of the document. The structure is initialized by calling the function
`fpConvertStream()` or `KVHTMLConvertFile()`.

```
typedef struct tag_KVHTMLOptionsEx
{
    KVCharSet               OutputCharSet;
    BOOL                    bUseDocumentColors;
    BOOL                    bUseDocumentFontInfo;
```

```
        BOOL                    bSupportFontFace;
        BOOL                    bSupportUserFontSizeMapping;
        KVFontSizeMap           FontSizeMap;
        BOOL                    bDisplayRelativeFontSize;
        BOOL                    bSupportRFC1942_cols;
        BOOL                    bNbspEmptyCells;
        ENSATableBorder         SATableBorder;
        int                     nTableBorderWidth;
        char                    *pszBaseURL;
        char                    *pszMainURL;
        char                    *pszDefaultOutputDirectory;
        char                    *pszPicPath;
        char                    *pszPicURL;
        char                    *pszJavaURL;
        char                    *pszJavaArchive;
        BOOL                    bRemoveFileNameSpaces;
        BOOL                    bRasterizeFiles
        KVHTMLGraphicType       OutputRasterGraphicType;
        KVHTMLGraphicType       OutputVectorGraphicType;
        int                     cxVectorToRasterXRes;
        int                     cyVectorToRasterYRes;
        BOOL                    bGenerateURLs;
        long                    lcbMaxMemUsage;
        BOOL                    bSupportColumnHeadings;
        BOOL                    bSupportRowHeadings;
        BOOL                    bSupportCellSpan;
        BOOL                    bSupportRowSpan;
        BOOL                    bSupportColumnWidth;
        BOOL                    bRemoveEmptyColumns;
        BOOL                    bRemoveEmptyRows;
        int                     nRowsBeforeSplit;
        KVLanguageID            OutputLanguageID;
        KVHTMLStyleSheetType    eStyleSheetType
        BOOL                    bTabsToTables;
        BOOL                    bForceOutputCharSet;
        BOOL                    bEnableEmptyRows;
        BYTE                    cReplaceChar;
        BYTE                    cRedact;
        KVCharSet               eSrcCharSet;
        BOOL                    bForceSrcCharSet;
        int                     nCompressionQuality;
    }
KVHTMLOptionsEx;
```

Member Descriptions

OutputCharSet   The character set to use for textual output. The available character

	KVCharSet in kvtypes.h. See Convert Character Sets, on page 80
	To make sure that the character set defined here is used, you must set bForceOutputCharSet to **TRUE**.
	The default is KVCS_UNKNOWN.
bUseDocumentColors	Set bUseDocumentColors to **TRUE** to retain the color attributes information contained in the source document. If you set bUseDocumentColors to **FALSE**, no color attributes appear in the <font> tags of the output.
	The default is FALSE.
bUseDocumentColors	Set bUseDocumentColors to **TRUE** to retain the font information contained in the source document. If you set bUseDocumentColors to **FALSE**, no font information appears in the <font> tags in the output.
	The default is FALSE.
bSupportFontFace	Set bSupportFontFace to **TRUE** to retain the font face information contained in the source document. If you set bSupportFontFace to **FALSE**, no FACE attributes appear in the <font> tags of the HTML output.
	This applies only when you set eStyleSheetType to STYLESHEET_ DISABLED. The default is TRUE.
bSupportUser FontSizeMapping	Set bSupportUserFontSizeMapping to **TRUE** to use the font sizes specified in the FontSizeMap. If you set bSupportUserFontSizeMapping to **FALSE**, HTML Export uses default size attributes.
	This applies only when you set eStyleSheetType to STYLESHEET_ DISABLED. The default is FALSE.
FontSizeMap	The font sizes to which the HTML tags <font size=1> through <font size=7> correspond. If bSupportUserFontSizeMapping is FALSE, this member can be left blank.
	See the Discussion section for more information.
bDisplayRelativeFontSize	Set bDisplayRelativeFontSize to **TRUE** to use relative font size tags in the HTML output. For example, the tag <font size=+1> adds one to the base font size, which is normally three.
	The default is FALSE.
bSupportRFC1942_cols	Set bSupportRFC1942_cols to **TRUE** to include cols=integer specifications in the <table> tags of the HTML output.
	The default is TRUE.
bNbspEmptyCells	Set bNbspEmptyCells to **TRUE** to include a non-breaking space (<td> </td>) in the markup for empty table cells in the source

document. If you set `bNbspEmptyCells` to **FALSE**, `<td></td>` is generated for empty table cells.

This option applies to word processing documents and spreadsheets only.

The default is `TRUE`.

`SATableBorder`	Specifies whether table borders are based on the setting in the source document, are always on, or are always off. The options are enumerated in `ENSATableBorder` in `kvtypes.h`. See ENSATableBorder, on page 235.
	This option applies to word processing documents only.
	The default is `SA_BaseOnDocument`.
`nTableBorderWidth`	Sets the width of the table border in pixels.
	This option applies to word processing documents only.
	The default is `1`.
`pszBaseURL`	The base URL that replaces the `$BASE` token in the HTML output. See $BASE, on page 390.
	The default is `NULL`.
`pszMainURL`	The URL that replaces the `$MAINURL` token in the HTML output. See $MAINURL, on page 391.
	The default is `NULL`.
`pszDefaultOutputDirectory`	The default output directory for auxiliary files created during the conversion.
	The default is `NULL`, and the files are placed in the directory in which your application is running.
`pszPicPath`	The output directory for graphics created during the conversion. If specified, this member can also be used by the callback functions `KVHTMLGetAnchorEx` and `KVHTMLGetAuxOutputEx`.
	This option applies to word processing documents only.
	The default is `NULL`, and the files are placed in the directory in which your application is running.
`pszPicURL`	The URL of the picture files created from embedded graphics in the source document. To specify a complete image source, this element must be combined with `pszAnchor` of the `GetAnchor()` callback function. See GetAnchor(), on page 196.
	For example, setting `pszPicURL` to **../cgi-bin/** and setting `pszAnchor` to **pic.jpg** results in the following markup:

```
<IMG SRC="../cgi-bin/pic.jpg">
```

	This element applies to word processing documents only. The default is `NULL`.
`bRemoveFileNameSpaces`	Set `bRemoveFileNameSpaces` to **TRUE** to remove spaces from generated output file names.
	The default is `FALSE`.
`bRasterizeFiles`	Set `bRasterizeFiles` to **TRUE** to rasterize slides from presentations into single images. For this setting to take effect, you must set the bNoPictures, on page 223 member to **FALSE**. The format the images are converted to is determined by the OutputRasterGraphicType, below member.
	Set `bRasterizeFiles` to **FALSE** to convert the text in slides to HTML. When you set this member to **FALSE**, images do not appear in the HTML output.
	The default is `FALSE`.

> **NOTE:**
> When `bRasterizeFiles` is **FALSE**, the export process uses the ordering in the file to produce the output, which does not necessarily match the logical reading order for the presentation. To use a logical reading order instead, you can set the `LogicalOrder` parameter in the `[Options]` section of `formats_e.ini`. See Convert Presentation Files, on page 107.

`OutputRasterGraphicType`	The output format of rasterized embedded graphics. There are six options enumerated in `KVHTMLGraphicType` in `kvhtml.h`. See KVHTMLGraphicType, on page 243.
	The default is `KVGFX_JPEG`.
`OutputVectorGraphicType`	The output format of vector graphics. The options are enumerated in `KVHTMLGraphicType` in `kvhtml.h`. See KVHTMLGraphicType, on page 243. For more information on displaying vector graphics on UNIX or Linux, see Display Vector Graphics on UNIX and Linux, on page 87.
	The default is `KVGFX_JPEG`.
`cxVectorToRasterXRes`	Specifies the horizontal resolution when converting presentation files and vector graphics. This is set in conjunction with `cyVectorToRasterYRes`. For more information, see Set the Resolution of Presentations and Vector Graphics, on page 224.
	The default value is `0`, which means the original resolution is retained.
`cyVectorToRasterYRes`	Specifies the vertical resolution when converting presentation files and vector graphics. This is set in conjunction with `cxVectorToRasterXRes`. For more information, see Set the

The default value is `0`, which means the original resolution is retained.

bGenerateURLs	Set `bGenerateURLs` to **TRUE** to add anchor tags (`<a ...></a></a>`) to text starting with "`www`", "`http:`" or "`file:`". This option applies to word processing documents only. The default is `FALSE`.
lcbMaxMemUsage	The maximum memory allocated dynamically for token buffers during file processing. If this maximum is reached, Export performs a swap-to-disk operation internally, and then reuses the memory blocks. Export maintains an internal minimum memory size. This option applies to word processing or text documents only. The default is `LONG_MAX`. The unit is in bytes.
bSupportColumnHeadings	Set `bSupportColumnHeadings` to **TRUE** to include column headings from the source spreadsheet in the output. This option applies to spreadsheets only. The default is `FALSE`.
bSupportRowHeadings	Set `bSupportRowHeadings` to **TRUE** to include row headings from the source spreadsheet in the output. This option applies to spreadsheets only. The default is `FALSE`.
bSupportCellSpan	Set `bSupportCellSpan` to **TRUE** to include `colspan="n"` markup in the output. If text in the source document spans across empty columns, and `bSupportCellSpan` is enabled, the text is output across columns in the HTML. If this option is disabled, text that spans across columns is output in a single cell. This option applies to spreadsheets only. The default value is `FALSE`.
bSupportRowSpan	Set `bSupportRowSpan` to **TRUE** to include row span data from the source spreadsheet in the output. This option applies to spreadsheets only. The default value is `FALSE`. Currently not supported.
bSupportColumnWidth	Set `bSupportColumnWidth` to **TRUE** to include column width data from the source document in the output. This option applies to spreadsheets only. The default value is `FALSE`.
bRemoveEmptyColumns	Set `bRemoveEmptyColumns` to **TRUE** to remove columns that do not contain text or color. To remove empty columns, you must set `bSupportCellSpan` to **FALSE**. This option applies to spreadsheets only. The default is `FALSE`.

`bRemoveEmptyRows`	Set `bRemoveEmptyRows` to **TRUE** to remove empty rows that do not contain text or color. This option applies to spreadsheets only.
	The default is `FALSE`.
`nRowsBeforeSplit`	The approximate number of spreadsheet rows to be processed before splitting a table. This helps to prevent large spreadsheet tables from occurring in a single document, which can cause speed and processing problems for the browser.
	This option applies to spreadsheets only. The default is `0`.
`OutputLanguageID`	The language for the textual output of language-specific data such as time and date. `OutputLanguageID` must be in the system locale. If `OutputLanguageID` is invalid or not supplied, the system default is used. Language IDs are defined in `KVLanguageID` in `kvtypes.h`.
	The default is `Language_UNKNOWN`.
`eStyleSheetType`	One of the enumerated options for processing style sheet information. The options are defined in `KVHTMLStyleSheetType` in `kvhtml.h`. See KVHTMLStyleSheetType, on page 241.
	`STYLESHEET_DISABLED`—Disables Cascading Style Sheet (CSS) formatting.
	`CSS_INLINE`—Enables CSS formatting, and adds style sheet information inline to the HTML output file.
	`CSS_TOFILE`—Enables CSS formatting, and generates an external file or uses an existing external file, which is referenced in a `<link...>` element.
	The default is `STYLESHEET_DISABLED`.
`bTabsToTables`	Set `bTabsToTables` to **TRUE** to convert tabbed columns to tables. This option applies to word processing documents only.
	When you use `bTabsToTables`, the original font in the cells is not maintained. The `bTabsToTables` option is intended for use on plain text files. Its purpose is to make the text more readable when viewed in a browser. Font information is not written into the table cells that are generated. Instead, the default font of the browser is used.
	The default is `FALSE`.
`bForceOutputCharSet`	Set `bForceOutputCharSet` to **TRUE** to use the output character set specified in `OutputCharSet`, regardless of the internal document information or the source character set specified by `eSrcCharSet`. Forcing a character set to `KVCS_UNKNOWN` is always ignored. SeeConvert Character Sets, on page 80.
	The default is `FALSE`.

bEnableEmptyRows	Set bEnableEmptyRows to **TRUE** to display empty rows in spreadsheets. If you set bEnableEmptyRows to **FALSE**, empty rows are not displayed. This option applies only to 20 or more consecutive empty rows.
	This option applies to spreadsheets only. The default is FALSE.
cReplaceChar	The character used when a character in the source document's character set cannot be mapped to the output character set.
	The default replacement character is a question mark (?).
cRedact	The character used to replace text that is designated through style mapping to be omitted from the output. This functionality is useful when you need to hide confidential or sensitive information.
	The specified character is used for all text that is mapped to a style processed with the KVSTYLE_REDACT flag (defined in kvtypes.h). See Map Styles, on page 84.
	This option applies to word processing documents only. The default replacement character is "X".
eSrcCharSet	Specifies the source character set of the document. The available character sets are enumerated in KVCharSet in kvtypes.h. See Convert Character Sets, on page 80.
	To make sure that the character set defined here is used, you must set bForceSrcCharSet to **TRUE**.
	The default is KVCS_UNKNOWN.
bForceSrcCharSet	Set bForceSrcCharSet to **TRUE** to use the source character set specified in eSrcCharSet, regardless of the internal document information. See Convert Character Sets, on page 80.
	Forcing a character set to KVCS_UNKNOWN is always ignored.
	The default is FALSE.
nCompressionQuality	Controls the output quality of graphics that support compression quality (for example, JPEG). A value of 0 means default quality (85 compression); 1 is the lowest quality (highest compression and therefore the smallest file size); 100 is the highest quality (no compression and therefore the largest file size).
	This option applies to word processing documents only. The default is 0.
bNoPictures	This member is only available in the structure in the template files. It is not part of the structure in the API.
	If you set bNoPictures to **TRUE**, embedded graphics in a document are not generated as separate files, image tags are not included in the output, and verbose markup is generated. Because graphics are not generated, the conversion is faster.

If you set `bNoPictures` to **FALSE**, embedded graphics in a document are regenerated as separate files, stored in the output directory, and image tags are included in the output.

This member is passed to the bIndex argument of the `fpConvertStream()` or `KVHTMLConvertFile()` functions.

The default is `FALSE`.

Discussion

- A pointer to this structure is passed as an argument to `fpConvertStream()` or `KVHTMLConvertFile()`. If the pointer to the structure is not `NULL`, the values of the members specified in the structure are used. If the pointer to the structure is `NULL`, the default values are used.

- `bNoPictures` is similar to the `KVCFG_BLANKPICTURE` flag in the call to `KVHTMLConfig()`. Unlike `bNoPictures`, the `KVCFG_BLANKPICTURE` flag generates image tags with an empty `src` attribute. KVHTMLConfig(), on page 177.

- To output graphics for presentations, you must set `bNoPictures` to **FALSE**, and set bRasterizeFiles to **TRUE**.

- The values in FontSizeMap, on page 218 indicate the range for the HTML tag `<font size=#>`. For example, if you specify `8`, `10`, `12`, `14`, `18`, and `24`:

  - font size <= 8 in the source document is mapped to `<font size=1>` in the output HTML

  - else, font size <=10 in the source document is mapped to `<font size=2>` in the output HTML

  - else, font size <=12 in the source document is mapped to `<font size=3>` in the output HTML

  - else, font size <=14 in the source document is mapped to `<font size=4>` in the output HTML

  - else, font size <=18 in the source document is mapped to `<font size=5>` in the output HTML

  - else, font size <=24 in the source document is mapped to `<font size=6>` in the output HTML

  - font size >24 in source the document is mapped to `<font size=7>` in the output HTML

When the HTML output is viewed, the browser maps `<font size=#>` to a specific font size.

Set the Resolution of Presentations and Vector Graphics

The cxVectorToRasterXRes and cyVectorToRasterYRes members are set in conjunction to specify the resolution (width and height) at which presentation files and vector graphics are converted.

You can specify the resolution as an absolute size in pixels, or as a proportion of the original size.

KeyView always maintains the aspect ratio of the original graphic and does not increase the resolution. If you set values that would enlarge a graphic, KeyView only changes the size of the HTML element.

To set the resolution in pixels

To specify the resolution in pixels, specify the width (`cxVectorToRasterXRes`) and/or height (`cyVectorToRasterYRes`).

To export the largest image that fits within a bounding box, without changing the original aspect ratio, set both the width and height. For example, to export the largest image that fits in an 800x500 bounding box:

```
cxVectorToRasterXRes=800
cyVextorToRasterYRes=500
```

Alternatively you can fix one of the dimensions. Set one value and set the other to zero. For example, to export images with a height of 1500 pixels and whatever width is necessary to maintain the original aspect ratio:

```
cxVectorToRasterXRes=0
cyVextorToRasterYRes=1500
```

The maximum size permitted for either dimension is 4000 pixels.

To set the resolution proportionally

To set the resolution proportionally, set `cxVectorToRasterXRes` to a negative value. A negative value represents a percentage of the original resolution. Set `cyVectorToRasterYRes` to `0` (zero). Negative (percentage) values for `cyVectorToRasterYRes` are ignored.

The following example exports a graphic at 50 percent of its original resolution:

```
cxVectorToRasterXRes=-50
cyVectorToRasterYRes=0
```

KVHTMLTemplateEx

This structure defines the overall framework of the HTML output. Members in this structure define the HTML markup written at specific points in the output stream. The pointers contain HTML markup that might include embedded KeyView tokens. See Export Tokens, on page 390 for more information on tokens. The structure is initialized by calling the `fpConvertStream()` or `KVHTMLConvertFile()` function. See fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

```
typedef struct tag_KVHTMLTemplateEx
{
    char        *pszMainTop;
    char        *pszMainBottom;
    char        *pszFirstH1Start;
    char        *pszFirstH1End;
    char        *pszMiddleH1Start;
    char        *pszMiddleH1End;
    char        *pszLastH1Start;
    char        *pszLastH1End;
    char        *pszH[2..6]HTML;
    char        *pszTOCH[1..6]Start;
    char        *pszTOC_H[1..6];
    char        *pszTOCH[1..6]End;
    char        *pszXFile;
    char        *pszStartBlock;
    char        *pszEndBlock;
```

```
    BOOL        bPutBlocksInSeparateFiles;
    BOOL        bHardPageMakesNewBlock
    long        lcbBlockSize;
    char        *pszChunkTemplate;
    char        *pszTableHTML;
    BOOL        bTableHTMLForSpreadsheetOnly;
    char        *pszUserSummary;
    char        *pszXStartBlock;
    char        *pszXEndBlock;
    char        *pszTOCH[1..6]LeafNode;
}
KVHTMLTemplateEx;
```

Member Descriptions

pszMainTop	The markup and tokens inserted at the beginning of the main HTML file. Most of the template files feature `<meta>` tags with tokens that store the input document's metadata. This member should at least contain the `<html>` and `<body>` tags. For frame-style output, this member must include the `<frame>` tag.
	The default is `NULL`.
pszMainBottom	The markup and tokens inserted at the end of the main HTML file. This member should at least contain the `</html>` and `</body>` tags, and should close the elements that are opened by `pszMainTop`.
	The default is `NULL`.
pszFirstH1Start	The markup and tokens inserted at the beginning of the first created H1 HTML block (that is, the block associated with the first H1 table of contents entry).
	The default is `NULL`.
pszFirstH1End	The markup and tokens inserted at the end of the first created H1 HTML block (that is, the block associated with the first H1 table of contents entry).
	The default is `NULL`.
pszMiddleH1Start	The markup and tokens inserted at the beginning of those H1 HTML blocks that are neither the first nor the last H1 blocks created (that is, blocks associated with all but the first and last H1 table of contents entries).
	The default is `NULL`.
pszMiddleH1End	The markup and tokens inserted at the end of those H1 HTML blocks that are neither the first nor the last H1 blocks created

	(that is, blocks associated with all but the first and last H1 table of contents entries).
	The default is `NULL`.
`pszLastH1Start`	The markup and tokens inserted at the beginning of the last created H1 HTML block (that is, the block associated with the last H1 table of contents entry).
	The default is `NULL`.
`pszLastH1End`	The markup and tokens inserted at the end of the last created H1 HTML block (that is, the block associated with the last H1 table of contents entry).
	The default is `NULL`.
`pszH[2..6]HTML`	The markup and tokens inserted in an HTML block for heading levels 2 through 6.
	The default is `NULL`.
`pszTOCH[1..6]Start`	The markup and tokens inserted at the beginning of a table of contents block for heading levels 1 through 6 entries. For example:
	`<ol type="I">`
	`<ol type="1">`
	`<ol type="i">`
	The default is `NULL`.
`pszTOC_H[1..6]`	The markup and tokens required to process the table of contents entries for heading levels 1 through 6. For example:
	`<a href="$ANCHOR" target="right">$TOCTE</a>`
	The default is `NULL`.
`pszTOCH[1..6]End`	The markup and tokens inserted at the end of a table of contents block for heading levels 1 through 6 entries. For example:
	`</ol>`
	The default is `NULL`.
`pszXFile`	The markup and tokens generated and placed in an extra HTML file. This file holds content from the source document. For example, it could contain the table of contents (using the `$TOC` token), which could then be displayed in an HTML frame. To process this file, you would use the `$XANCHOR` token.
	For example, if the extra file contains the table of contents in frame-based HTML, set `pszXFile=$TOC` and place the following HTML markup in the `pszMainTop` element:

```
<frame src="$XANCHOR" name="left" scrolling="auto"
target="right">
```

See Export Tokens, on page 390 for more information on Export tokens.

The default is NULL.

pszStartBlock	The markup and tokens inserted at the beginning of each block created as a result of lcbBlockSize or bHardPageMakesNewBlock. The default is NULL.
pszEndBlock	The markup and tokens inserted at the end of each block created as a result of the of lcbBlockSize or bHardPageMakesNewBlock. The default is NULL.
bPutBlocksInSeparateFiles	Set bPutBlocksInSeparateFiles to **TRUE** to create a separate HTML file for each heading level 1 block. Each new block uses the markup defined in pszStartBlock and pszEndBlock. If you set bPutBlocksInSeparateFiles to **FALSE**, each heading level 1 block is placed sequentially in the same file after the initial markup is written. The default is FALSE.
bHardPageMakesNewBlock	Set bHardPageMakesNewBlock to TRUE to have hard page breaks in the source document generate new HTML files during the conversion process. The bPutBlocksInSeparateFiles, above member must also be set to **TRUE**, and the pszChunkTemplate, below member must specify a table of contents entry for the new file. This option applies to word processing documents and spreadsheets only. Page boundaries in PDF documents are considered page breaks. The default is FALSE.
lcbBlockSize	The maximum size (in bytes) of heading level 1 HTML output files. This number is used as a guideline and can be exceeded to break content at a logical location. This setting is not used when exporting spreadsheets. The default is 0. This means that the size is undefined and unlimited.
pszChunkTemplate	If a heading level 1 HTML block is subdivided into separate files because the block exceeds the size limitations specified in lcbBlockSize, or bHardPageMakesNewBlock is set, this member defines the table of contents entry for the new file.

The page number can be included in the table of contents entry by inserting the $SPLITBLOCKNUMBER token. For example:

`Page $SPLITBLOCKNUMBER`

The default is NULL.

pszTableHTML	Specifies the markup (no tokens) inserted at the beginning of each table created during the conversion process. If you set pszTableHTML, table cell color and border information from the document is ignored. This is used in conjunction with bTableHTMLForSpreadsheetOnly to control the look of generated spreadsheets.

For example, to center the table, set the background color to teal, and set the border width to 13, use:

`pszTableHTML=<table bgcolor="teal" border="13" align="center">`

The default is NULL.

bTableHTMLForSpreadsheetOnly	If set to TRUE, bTableHTMLForSpreadsheetOnly controls how spreadsheets are displayed in the output. If set to FALSE, cell color and border information from the source document is used. Use this member in conjunction with pszTableHTML.

The default is FALSE.

pszUserSummary	The markup and tokens generated when the tokens $USERSUMMARY or $SUMMARY are used. For example:

`<meta name="$NAME" content="$CONTENT" />`

The default is NULL.

pszXStartBlock	The markup and tokens inserted at the beginning of each HTML block generated by the $XANCHOR token. If either this member or pszXEndBlock is defined, both pszStartBlock and pszEndBlock are ignored. See Export Tokens, on page 390 for more information on $XANCHOR.

The default is NULL.

pszXEndBlock	The markup and tokens inserted at the end of each HTML block generated by the $XANCHOR token. If either this member or pszXStartBlock is defined, both pszStartBlock and pszEndBlock are ignored.See Export Tokens, on page 390 for more information on $XANCHOR.

The default is NULL.

pszTOCH[1..6]LeafNode	The markup that replaces pszTOC_H[1..6] entries for leaf nodes in the table of contents. A leaf node is a node that has no children.

The default is NULL.

Discussion

A pointer to this structure is passed as an argument to `fpConvertStream()` and `KVHTMLConvertFile`
`()`. If the pointer to the structure is not `NULL`, the values of the members specified in the structure are
used. If the pointer to the structure is `NULL`, the default values are used.

KVHTMLTOCOptions

This structure defines whether a heading is included in the table of contents. Source text is converted
to a heading in the HTML output if

- it meets *all* the criteria defined by the members of `KVHTMLHeadingInfo` (see KVHTMLHeadingInfo, on page 209), and

- the `headingCreateType` member of `KVHTMLTOCOptions` is set to allow automatic heading generation.

The structure is initialized by calling the `fpConvertStream()` or `KVHTMLConvertFile()` function. See
fpConvertStream(), on page 163 or KVHTMLConvertFile(), on page 185.

When PDF files are converted to HTML by using the default reader, `pdfsr`, the table of contents is
generated from "bookmarks" within the PDF file. This structure is not used.

```
Typedef struct tag_KVHTMLTOCOptions
{
    BOOL                    bAllowHeadingsInTables;
    KVHeadingCreateOptions  headingCreateType;
    KVHTMLHeadingInfo       *pH1;
    KVHTMLHeadingInfo       *pH2;
    KVHTMLHeadingInfo       *pH3;
    KVHTMLHeadingInfo       *pH4;
    KVHTMLHeadingInfo       *pH5;
    KVHTMLHeadingInfo       *pH6;
}
KVHTMLTOCOptions;
```

Member Descriptions

`bAllowHeadingsInTables`   This option determines if the text in tables is considered for automatic
heading generation. If you set `bAllowHeadingsInTables` to **TRUE**, the
text in tables is included in the determination of headings and table of
contents entries.

This option applies to word processing documents and spreadsheets
only.

The default is `FALSE`.

`headingCreateType`   This option determines how HTML Export subdivides the source

in `kvhtml.h`. See KVHeadingCreateOptions, on page 244.

The determination of table of contents entries is based on whether the source document contains *heading styles* or whether *text attributes* conform to the criteria defined in the `KVHTMLHeadingInfo` structure. See KVHTMLHeadingInfo, on page 209.

Heading styles are predefined style tags, such as "Heading 1" and "Heading 2" tags in a Microsoft Word document. Text attributes are bold, underlined, italic, and so on.

This option applies to word processing documents only.

The default is `KVCS_DocHeadingsOnly`.

KVHTMLHeadingInfo

A pointer to the `KVHTMLHeadingInfo` structure. SeeKVHTMLHeadingInfo, on page 209.

When the table of contents entries are not based on the source documents heading styles, the table of contents entries are determined by whether text attributes (such as bold, underlined, and italic text) in the source document meet all the criteria defined in `KVHTMLHeadingInfo`.

Discussion

A pointer to this structure is passed as an argument to `fpConvertStream()` and `KVHTMLConvertFile()`. If the pointer to the structure is not `NULL`, the values of the members specified in the structure are used. If the pointer to the structure is `NULL`, the default values are used.

KVRevisionMark

This structure defines the information generated when the revision feature is enabled and how the information is displayed. (see Include Revision Information, on page 89). It defines the following:

- the contents of the `title` attribute for an `<ins>` or `<del>` tag. See Configure the Revision Title, on page 90.

- the style used to display revised text by different reviewers. See Configure the Revision Style, on page 91.

- the revision summary file. See Generate a Revision Summary, on page 92.

```
typedef struct tag_KVRevisionMark
{
    KVStructHeader;
    KV_RM_Title    InsTitle;
    KV_RM_Title    DelTitle;
    char           **ppAuthorStyles;
    int            nAuthorStyles;
    BOOL           bCreateSummary;
    char           *pszRevSumStartBlock;
```

```
    char          *pszRevSumEndBlock;
    int           nReserved;
    void          *pReserved;
}
KVRevisionMark;
```

Member Description

KVStructHeader	The KeyView version of the structure. See KVStructHead, on page 204.
InsTitle	The prefix and revision information for the `<ins>` tag, as defined by the KV_ RM_TITLE structure in `kvtypes.h`. See KV_RM_Title, below.
DelTitle	The prefix and revision information for the `<del>` tag, as defined by the KV_ RM_TITLE structure in `kvtypes.h`. See KV_RM_Title, below.
ppAuthorStyles	The HTML style you want applied to the revised content for a particular reviewer.
nAuthorStyles	The number of HTML styles to be defined.
bCreateSummary	When you set this flag, a revision summary file is created in the directory where the HTML output is generated. The default file name is `output_ filename.revsum.htm`. You can change this file name by using the `fpGetAnchor` callback function. See GetAnchor(), on page 196.
pszRevSumStartBlock	The markup and tokens inserted at the beginning of the revision summary file.
pszRevSumEndBlock	The markup and tokens inserted at the end of the revision summary file.
nReserved	Reserved for internal use.
pReserved	Reserved for internal use.

KV_RM_Title

This structure defines the contents (prefix, reviewer name, date, and time) of the `title` attribute for the InsTitle and DelTitle members of KVRevisionMark (see KVRevisionMark, on the previous page).

```
typedef struct tag_KV_RM_Title
{
    RM_Title_Flag    eFlag;
    BYTE             *pPrefix;
    int              nSize;
    KVCharSet        eCharSet;
}
KV_RM_Title;
```

Member Description

eFlag	Specifies whether the reviewer name, date, and time appear in the `title` attribute. The options are enumerated in `RM_Title_Flag` in `kvtypes.h`. See RM_Title_Flag, on page 252.
pPrefix	A pointer to the text string that is prefixed to the `eFlag` value in the `title` attribute of the `<ins>` or `<del>` tags. By default, the string "`inserted:`" or "`deleted:`" is the first entry in the attribute.
nSize	The size of the prefix.
eCharSet	The character set of the prefix characters. The available character sets are enumerated in `KVCharSet` in `kvtypes.h`. See Convert Character Sets, on page 80.
	If you set `eCharSet` to **KVCS_UNKNOWN**, the character set of the prefix is not converted and the prefix is written directly to the HTML output.

Chapter 11: Enumerated Types

This section provides information on some of the enumerated types used by the HTML Export API.

Introduction

The enumerated types are in `adinfo.h`, `kvtypes.h`, `kvhtml.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

Enumerated Type	Setting
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

ENSATableBorder

This enumerated type defines the type of border to display around tables. This enumerated type is defined in kvtypes.h.

Definition

```
typedef enum tag_ENSATableBorder
{
    SA_BaseOnDocument,
    SA_NoBorder,
    SA_Border
}
ENSATableBorder;
```

Enumerators

SA_BaseOnDocument	Border type is based on the document.
SA_NoBorder	Table borders are always off.
SA_Border	Table borders are always on.

KVCredKeyType

This enumerated type defines the type of credential used to open a protected file. See KVCredentialComponent, on page 146. 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_Password	The credential in `KVCredentialComponent` is a password.

KVErrorCode

This enumerated type defines the type of error generated if Export 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.
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`.

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. This enumerator is used by KeyView Filter.
`KVError_ InterfaceFunctionNotFound`	An interface function was not found during out-of-process filtering. This is an extended error for the `KVERR_General` code. This enumerator is used by KeyView Filter.
`KVError_InputFileNotFound`	Could not find the input file during out-of-process filtering. This is an extended error for the `KVERR_General` code. This enumerator is used by KeyView Filter.
`KVError_ OpenOutputFileFailed`	Could not open the output file during out-of-process filtering. This is an extended error for the `KVERR_General` code. This enumerator is used by KeyView Filter.
`KVError_MemoryLeak`	A memory leak occurred during out-of-process filtering. This is an extended error for the `KVERR_General` code. This enumerator is used by KeyView Filter.
`KVError_MemoryOverwrite`	A memory overwrite occurred during out-of-process filtering. This is an extended error for the `KVERR_General` code. This enumerator is used by KeyView Filter.
`KVError_GPF`	An exception occurred during out-of-process filtering. This is an extended error for the `KVERR_General` code. This enumerator is used by KeyView Filter.
`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. This enumerator is used by KeyView Filter.
`KVError_KVoopLogFailed`	The creation of the out-of-process error log failed. This is an extended error for the `KVERR_General` code. This enumerator is used by KeyView Filter.
`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 convert 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 Micro Focus sales representative to get an updated license key.
`KVError_OopBadConfig`	Information in the `kvxconfig.ini` file is incomplete and cannot be used to the XML file. This is used by KeyView Filter.

`KVError_OopBrokenPipe`	Data was not transferred between the parent and child processes during out-of-process filtering because either the parent or child failed. This is used by KeyView Filter.
`KVError_OopPipeOEF`	Data was not transferred between the parent and child processes during out-of-process filtering because the parent process was shut down. This is used by KeyView Filter.
`KVError_IPCTimeOut`	Either the parent or child process is waiting for a reply or request during out-of-process filtering. This is used by KeyView Filter.
`KVError_ InvalidOopDriverSignature`	A client sent a request to an out-of-process server, but the context driver does not exist on the server. This is used by KeyView Filter.
`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. This is used by KeyView Filter.
`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 235.

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

KVHTMLStyleSheetType

This enumerated type defines the options for processing style sheet information. This enumerated type is defined in `kvhtml.h`.

Definition

```
typedef enum tag_KVHTMLStyleSheetType{    STYLESHEET_DISABLED = 0,

    CSS_INLINE,

    CSS_TOFILE

}
KVHTMLStyleSheetType;
```

Enumerators

STYLESHEET_ DISABLED	Disables Cascading Style Sheet (CSS) formatting.
CSS_INLINE	Enables CSS formatting and adds style sheet information inline to the HTML output file.
CSS_TOFILE	Enables CSS formatting, and generates an external file or uses an existing external file, which is referenced in a `<link...>` element. If `CSS_TOFILE` is set, you must use the `$STYLESHEET` token to specify the URL of the style sheet in the HTML output.

CSS_TOFILE (continued):

The `-c` option can be used in the `htmlini` sample program to specify the full path and file name of an external CSS file.

Applies to word processing documents only.

KVHTMLAnchorTypeEx

This enumerated type defines the anchor types for the output stream. This enumerated type is defined in `kvhtml.h`.

Definition

```
typedef enum tag_KVHTMLAnchorTypeEx

{

    VectorPictureAnchorEx = 0,

    RasterPictureAnchorEx,

    H1AnchorEx,

    H2AnchorEx,

    H3AnchorEx,

    H4AnchorEx,
```

```
        H5AnchorEx,

        H6AnchorEx,

        XAnchorEx,

        AnimatedGIFAnchorEx,

        CSSAnchorEx,

        GeneralAnchorEx,

        DBAnchorEx,

        JPEGAnchorEx

}

KVHTMLAnchorTypeEx;
```

Enumerators

`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.

KVHTMLGraphicType

This enumerated type defines graphic formats to which embedded graphics and presentations are converted. This enumerated type is defined in `kvhtml.h`.

Definition

```
typedef enum tag_KVHTMLGraphicType

{

    KVGFX_GIF,

    KVGFX_JPEG,

    KVGFX_PNG,

    KVGFX_CGM,

    KVGFX_WMF,

    KVGFX_HTML

    KVGFX_JAVA

}

KVHTMLGraphicType;
```

Enumerators

`KVGFX_GIF`	Specifies GIF (Graphics Interchange Format) as the graphic type.
`KVGFX_JPEG`	Specifies JPEG (Joint Photographic Experts Group) as the graphic type.
`KVGFX_PNG`	Specifies PNG (Portable Network Graphics) as the graphic type.
`KVGFX_CGM`	Deprecated.
`KVGFX_WMF`	Specifies WMF (Windows Metafile) as the graphic type.
`KVGFX_JAVA`	Deprecated.
`KVGFX_HTML`	Specifies that text in presentations are converted to HTML.

KVHeadingCreateOptions

This enumerated type defines whether Export generates blocks and block chunks based only on the heading styles defined in a source document (if they are available), or based on both the source document's heading styles and headings that are created automatically by Export. Headings that are created automatically by Export are based on the text attributes of the source document as defined by `KVHTMLHeadingInfo`). This enumerated type is defined in `kvhtml.h`.

Definition

```
typedef enum tag_KVHeadingCreateOptions
{
```

```
    KVHC_DocHeadingsOnly,
    KVHC_CreateHeadingsAlways
}
KVHeadingCreateOptions;
```

Enumerators

`KVHC_DocHeadingsOnly`	This instructs Export to rely exclusively on heading styles defined in the source document. However, if the source document does not contain heading styles, Export generates blocks on its own using the criteria defined by the structure `KVHeadingInfo`.
`KVHC_CreateHeadingsAlways`	This instructs Export to use the heading styles in the source document when available, and to also automatically create table of contents entries based on the criteria defined by the structure `KVHeadingInfo`.

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 235.

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 153. 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.

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.

RM_Title_Flag

This enumerated type defines the information that appears in the `title` attribute in the `<ins>` and `<del>` tags for the `eFlag` member of the `KV_RM_Title` structure. This applies when revision information is enabled and included in the HTML output.

Definition

```
typedef enum
{
    RMT_Off = 0,
    RMT_Author,
    RMT_Datetime,
    RMT_AuthorDatetime
}

RM_Title_Flag;
```

Enumerators

RMT_Off	A `title` attribute is not included in the `<ins>` and `<del>` tags. This would generate markup similar to the following: `<ins cite="mailto:JohnD" datetime="2006-04-24T14:47:00">This text was added</ins>`

RMT_Author	The reviewer's name is generated in the title attribute of the `<ins>` and `<del>` tags. This would generate markup similar to the following:

```
<ins title="Inserted: JohnD"
cite="mailto:JohnD" datetime="2006-
04-24T14:47:00">This text was
added</ins>
```

RMT_Datetime	The date and time is included in the title attribute of the `<ins>` and `<del>` tags. The date is in ISO-8601 format: YYYY-MM-DDThh:mm:ss. This would generate markup similar to the following:

```
<ins title="Inserted: 2006-04-
24T14:47:00" cite="mailto:JohnD"
datetime="2006-04-24T14:47:00">This
text was added</ins>
```

RMT_AuthorDatetime	The reviewer's name, date, and time are included in the title attribute of the `<ins>` and `<del>` tags. This would generate markup similar to the following:

```
<ins title="Inserted: JohnD, 2006-04-
24T14:47:00" cite="mailto:JohnD"
datetime="2006-04-24T14:47:00">This
text was added</ins>
```

This is the default.

Part IV: COM API Reference

This section provides detailed reference information for the COM implementation of the HTML Export API.

- COM Interface Methods and Events
- COM Interface Properties

Chapter 12: COM Interface Methods and Events

This section describes the methods that HTML Export uses to manage the input and output streams, and to perform the actual document conversion. It also describes the events that allow the calling application to monitor and affect the HTML conversion while it is in progress.

The methods' syntax are described as they would be used from within Visual Basic, where `HTM` is the COM Automation Server object. The events are described with the Interface Description Language (IDL). See the Template Wizard or `comsamp` sample programs for examples of how to use the events from Visual Basic.

Methods

Events

AddStyleMapping

This method maps a character or paragraph style to arbitrary markup. See Map Styles, on page 84 for more information on style mapping.

```
HTM.AddStyleMapping(StyleName, MarkupStart, MarkupEnd, nFlags)
```

where:

- `StyleName` is the name of the word processing style (for example, "Heading 1") to which style mapping is applied. Style names are case sensitive.

- `MarkupStart` is the markup to be added at the beginning of the content (for example, `<h1>`).

- `MarkupEnd` is the markup to be added at the end of the content (for example, `<\h1>`).

- `nFlags` is the flag set for this markup. A flag specifies how the content associated with the style should be processed. Possible values are enumerated in `kvtypes.h`. See Flags for Defining Styles, on page 86 for a description of the flags.

> **NOTE:** By default, HTML Export maps the heading style "Heading 1" to `<h1></h1>`, and so on, for heading levels 1 through 6. If you use style mappings, the default mapping is overridden. Therefore, you must supply markup for *all* heading levels. HTML Export uses heading levels to define the overall structure of the HTML output.

ConvertFileToFile

This method converts a specified file to HTML.

`nRet = HTM.ConvertFileToFile(szTargetFile)`

where:

- `szTargetFile` is the complete path and file name of the HTML output.

- `nRet` is the return error code as defined in the `KVErrorCode` enumerated type defined in `kvtypes.h`.

GetFileInfo

This method retrieves information about the source file (as defined in the `pszInputFile` property).

```
HTM.GetFileInfo(lcharSet, ldocClass, ldocFmt, ldocVersion,
ldocAttributes)
```

where:

- `lcharSet` is the same as the `InputCharSet` property

- `docClass` is the same as the `adInfo_docClass` property

- `ldocFmt` is the same as the `adInfo_docFmt` property

- `ldocVersion` is the same as the `adInfo_docVersion` property

- `ldocAttributes` is the same as the `bAllowHeadingsInTables` property

See COM Interface Properties, on page 260 for a description of these properties.

> **NOTE:** This method fills in all of these parameters (long) based on the source document.

GetStyleMapping

This method gets the formatting specifically assigned to various styles.

```
HTM.GetStyleMapping(nItem, StyleName, MarkupStart, MarkupEnd,
nFlags)
```

See for a description of `StyleName`, `MarkupStart`, `MarkupEnd`, and `nFlags`.

By specifying `nItem` (which can be any value from 0 to `NumStyles` −1 where `pszBaseURL` is a property), the `StyleName`, `MarkupStart`, `MarkupEnd`, and `nFlags` parameters are filled in.

GetSummaryInfo

This method retrieves metadata from the input file.

```
HTM.GetSummaryInfo(nItem, lTotaltems, lValid, lType, lVal, szVal, szValUser)
```

where:

- `nItem` is an integer which is the metadata item number. See `KVSumtype` in `kvtypes.h` for a list of values.

  For example, `KV_Author` is value 3, so setting `nItem` to **3** would retrieve the Author of the source document if this information was available. This is identical to using the `pszAuthor` property (some common metadata items are exposed as properties as well as being accessible through this method).

- `lTotalItems` is a long which is returned to give the total number of metadata items possible. This is 34 for most documents (as defined in `KVSumType` in `kvtypes.h`), although all 34 might not be valid (see `lValid` parameter). If there is user-defined metadata available, `lTotalItems` might be greater than 34.

- `lValid` is a long which is 0 if the item is invalid (not available) and 1 if the information associated with this item is available.

- `lType` is a long which corresponds to the `KVSumInfoType` enumerated type defined in `kvtypes.h`. A value of 1 indicates that `szVal` contains the string associated with this item (if `lValid` is also 1). A value of 2 indicates that `lVal` contains a long associated with this item (if `lValid` is also 1). See .

- `szValUser` is a string description of the metadata item.

Sample Code

The following code from the `comsamp` sample program demonstrates how to use the `GetSummaryInfo` method:

```
Dim nTotal As Long
Dim nValid As Long
Dim nType As Long
Dim nVal As Long
Dim szVal As String
Dim szUserVal As String
On Error GoTo Handler
' Get the Author if available (item 3 is the Author. See KVSumType
' in kvtypes.h for a list of items and their values)
Call MyRef.GetSummaryInfo(3, nTotal, nValid, nType, nVal, szVal, szUserVal)
MsgBox szUserVal & " = " & szVal
```

RemoveStyleMapping

This method disables an instance of style mapping.

`HTM.RemoveStyleMapping(nItem)`

where:

- `nItem` is the style mapping item to remove (See GetStyleMapping, on page 256).

Unload

This method unloads the `htmserv` object.

`HTM.Unload()`

This method applies only to the out-of-process COM object and normally is not necessary. Note that using this method forces an unload, even if the object's reference count is not zero.

UpdateFromIniFile

This method updates parameters from the template file.

`HTM.UpdateFromIniFile(szIniFile)`

where:

- `szIniFile` is the complete path and file name of a supplied template file (see the directory *install*`\htmlexport\programs\ini` for examples of template files).

HTMLConfig

This method provides a way to configure options prior to document conversion. Currently, the function is used to specify a PDF reader. For more information, see Convert PDF Files to Raster Images, on page 95.

`HTM.HTMLConfig(`*nType, nValue, p*`)`

where:

- *nType* is a symbolic constant defined in `kvhtml.h` and used to configure options. Currently, you can set this to **KVCFG_SETHIFIPDF**. This option specifies that the graphic-based PDF reader (`kppdfrdr`) is used to convert PDF documents. See Convert PDF Files to Raster Images, on page 95.

- *nValue* is an integer value defined for the above type.

- *p* is reserved, and must be `NULL`.

The `comsamp` sample program demonstrates how to use this method. See comsamp, on page 132.

Continue

This event has two purposes: to enable the calling process to monitor the progress of the HTML conversion, and to provide a way of terminating a conversion before it is completed.

```
Continue([in] int PercentDone,
         [out, retval] int *bQuit);
```

The sample program `compsamp` demonstrates how to use this event.

UserCallback

This event allows the calling process to insert optional data into selected points of the HTML output. This event is triggered by the inclusion of the `$USERCB=X` token in one of the properties, where `X` identifies the callback. The text in `szUserString` is inserted in the HTML. See $USERCB, on page 392.

```
UserCallback([in] BSTR szUserString,
             [out, retval] BSTR *pszVal);
```

The `compsamp` sample program demonstrates how to use this event.

Chapter 13: COM Interface Properties

This section contains an alphabetized list of all the properties in the COM Interface.

Some of the descriptions refer to enumerated types in `adinfo.h`, `kvhtml.h`, or `kvtypes.h`. These header files are located in the `include` directory. See Enumerated Types, on page 234 for more information on enumerated types.

Some of the classes use HTML Export tokens. See Export Tokens, on page 390 for a description of these tokens.

adInfo_docAttributes

This is a read-only property.

The attributes of the source document. The document attributes are enumerated in `ENdocAttributes` of `adinfo.h`:

- `kEncrypted`
- `kMacBinaryEncoded`
- `kAppleSingleENcoded`
- `kAppleDoubleEncoded`
- `kWangGDIencoded`

adInfo_docClass

This is a read-only property.

The class of the source document. The format classes are enumerated in `ENdocClass` of `adinfo.h`.

A value of –1 indicates that HTML Export encountered an error while attempting to detect the class of the source document. Zero (0) indicates that HTML Export is unable to determine the format of the source document.

adInfo_docFmt

This is a read-only property.

The format of the source document. This information determines which document reader is used to generate stream information during the conversion. The formats are enumerated in `ENdocFmt` of `adinfo.h`.

adInfo_docVersion

This is a read-only property.

The version of the source document's format.

This property is a long integer corresponding to the version number of the format `version#` * 1000. For example, version number 1.02 would be 1020.

bAllowHeadingsInTables

This is a read and write property.

This property determines whether or not the contents of tables are considered for automatic heading generation. If you set `bAllowHeadingsInTables` to **TRUE**, HTML Export considers converting the contents of tables to headings in the HTML output.

See headingCreateType, on page 268 for more information on automatic generation of headings.

This property applies to word processing documents and spreadsheets only.

bDisplayRelativeFontSize

This is a read and write property.

Set `bDisplayRelativeFontSize` to **TRUE** to use relative font size tags in the HTML output. For example, the tag `<font size=+1>` adds one to the base font size, which is normally three.

bEnableEmptyRows

This is a read and write property.

Set `bEnableEmptyRows` to **TRUE** to display empty rows in a spreadsheet format. If you set `bEnableEmptyRows` to **FALSE**, empty rows are not displayed. This property applies only to 20 or more consecutive empty rows. The default is `FALSE`.

This property applies to spreadsheets only.

bForceOutputCharSet

This is a read and write property.

Set `bForceOutputCharSet` to **TRUE** to use the output character set specified in `OutputCharSet`, regardless of the internal document information or the source character set specified by `SrcCharSet`.

See Convert Character Sets, on page 80 for more information on character set mapping.

> **NOTE:** Forcing a character set to `KVCS_UNKNOWN` is always ignored.

bForceSrcCharSet

This is a read and write property.

Set `bForceSrcCharSet` to **TRUE** to use the source character set specified in `SrcCharSet`, regardless of the internal document information.

See Convert Character Sets, on page 80 for more information on character set mapping.

> **NOTE:** Forcing a character set to `KVCS_UNKNOWN` is always ignored.

bGenerateURLs

This is a read and write property.

Set `bGenerateURLs` to **TRUE** to add anchor tags (`<a ...></a>`) to text starting with "`www`", "`http:`", or "`file:`".

This property applies to word processing documents only.

bHardPageMakesNewBlock

This is a read and write property.

Set `bHardPageMakesNewBlock` to **TRUE** to have hard page breaks in the source document generate new HTML files during the conversion process. `pszchunktemplate` provides the appropriate table of contents entry for the new block. See pszChunkTemplate, on page 275.

This property applies to word processing documents and spreadsheets only.

bNbspEmptyCells

This is a read and write property.

Set `bNbspEmptyCells` to **TRUE** to include a non-breaking space (`<td> </td>`) in the markup for empty table cells in the source document. If you set `bNbspEmptyCells` to **FALSE**, `<td></td>` is generated for empty table cells.

This property applies to word processing documents and spreadsheets only.

bNoPictures

This is a read and write property.

Set `bNoPictures` to **TRUE** to generate verbose markup only. Embedded graphics are not generated as separate files, and image tags are not included in the output.

If you set `bNoPictures` to **FALSE**, embedded graphics in a document are regenerated as separate files, stored in the output directory, and image tags are included in the output.

To output graphics for presentations, you must set `bNoPictures` to **FALSE**, and set bRasterizeFiles to **TRUE**.

bPutBlocksInSeparateFiles

This is a read and write property.

Set `bPutBlocksInSeparateFiles` to **TRUE** to create a separate HTML file for each heading level 1 block. Each new block uses the markup defined in `pszStartBlock` and `pszEndBlock`. See pszStartBlock, on page 279 and pszEndBlock, on page 276. If you set `bPutBlocksInSeparateFiles` to **FALSE**, each heading level 1 block is placed sequentially in the same file, after the initial markup is written.

bRasterizeFiles

This is a read and write property.

Set `bRasterizeFiles` to **TRUE** to rasterize slides from presentations into single images. For this setting to take effect, you must also set the bNoPictures property to **FALSE**. The format the images are converted to is determined by the OutputRasterGraphicType property.

Set `bRasterizeFiles` to **FALSE** to convert the text in slides to HTML. When you set this property to **FALSE**, images do not appear in the HTML output.

bRemoveEmptyColumns

This is a read and write property.

Set `bRemoveEmptyColumns` to **TRUE** to remove spreadsheet columns that do not contain data and to disable cell merge. The default is `FALSE`.

This property applies to spreadsheets only.

bRemoveFileNameSpaces

This is a read and write property.

Set `bRemoveFileNameSpaces` to **TRUE** to remove spaces from generated output file names.

bSupportCellSpan

This is a read and write property.

Set `bSupportCellSpan` to **TRUE** to include `colspan="n"` markup in the output. If text in the source document spans across empty columns, and `bSupportCellSpan` is enabled, the text is output across columns in the HTML. If this option is disabled, text that spans across columns is output in a single cell. This property applies to spreadsheets only. The default value is `FALSE`.

bSupportColumnHeadings

This is a read and write property.

Set `bSupportColumnHeadings` to **TRUE** to include column headings from the source spreadsheet in the HTML output. This property applies to spreadsheets only. The default is **FALSE**.

bSupportColumnWidth

This is a read and write property.

Set `bSupportColumnWidth` to **TRUE** to include column width data from the source spreadsheet in the HTML output. This property applies to spreadsheets only. The default value is **FALSE**.

bSupportFontFace

This is a read and write property.

Set `bSupportFontFace` to **TRUE** to retain the font face information contained in the source document. If you set `bSupportFontFace` to **FALSE**, no `FACE` attributes appear in the `<font>` tags of the HTML output.

bSupportRFC1942_cols

This is a read and write property.

Set `bSupportRFC1942_cols` to **TRUE** to include `cols=x` specifications in the `<table>` tags of the HTML output markup.

bSupportRowHeadings

This is a read and write property.

Set `bSupportRowHeadings` to **TRUE** to include row headings from the source spreadsheet in the HTML output. This property applies to spreadsheets only. The default is **FALSE**.

bSupportRowSpan

This is a read and write property.

Set `bSupportRowSpan` to **TRUE** to include row span data from the source spreadsheet in the HTML output. This property applies to spreadsheets only. The default value is **FALSE**.

bSupportUserFontSizeMapping

This is a read and write property.

Set `bSupportUserFontSizeMapping` to **TRUE** to use the font sizes specified in the `FontSizeMap` (see FontSizeMap_nSize[1...7], on page 268). If you set `bSupportUserFontSizeMapping` to **FALSE**, HTML Export uses default `SIZE` attributes.

bTableHTMLForSpreadsheetOnly

This is a read and write property.

Set `bTableHTMLForSpreadsheetOnly` to **TRUE** to control how spreadsheets are displayed in the HTML output. If you set `bTableHTMLForSpreadsheetOnly` to **FALSE**, cell color and border information from the source document is used. Use this parameter in conjunction with `pszTableHTML`.

bTabsToTables

This is a read and write property.

Set `bTabsToTables` to **TRUE** to convert tabbed columns to tables. This property applies to word processing documents only.

bUseDocumentColors

This is a read and write property.

Set `bUseDocumentColors` to **TRUE** to retain the color attributes information contained in the source document. If you set `bUseDocumentColors` to **FALSE**, no color attributes appear in the `<font>` tags of the HTML output.

bUseDocumentFontInfo

This is a read and write property.

Set `bUseDocumentFontInfo` to **TRUE** to retain the font information contained in the source document. If you set `bUseDocumentFontInfo` to **FALSE**, no font information appears in the `<font>` tags in the HTML output.

CodePage

This is a read-only property.

The character encoding of the document if available.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

cRedact

This is a read and write property.

The character used to replace tagged text designated through style mapping to be omitted from the HTML output. This functionality is useful when you need to hide confidential or sensitive information. The default replacement character is "X".

The specified character is used for all text that is mapped to a style which is processed by using the `KVSTYLE_REDACT` flag (defined in `kvtypes.h`). See Flags for Defining Styles, on page 86 for more information on style mapping and the `REDACT` flag.

This property applies to word processing documents only.

cReplaceChar

This is a read and write property.

The character used when a character in the source document's character set cannot be mapped to the output character set. The default replacement character is a question mark (?).

cxVectorToRasterXRes

This is a read and write property.

This property controls the horizontal resolution at which presentations and vector graphics are converted. The default value is `0`, which means that HTML Export retains the original resolution.

This property is set in conjunction with cyVectorToRasterYRes, on the next page.

You can specify the resolution as an absolute size in pixels, or as a proportion of the original size.

KeyView always maintains the aspect ratio of the original graphic and does not increase the resolution. If you set values that would enlarge a graphic, KeyView only changes the size of the HTML element.

To set the resolution in pixels

To specify the resolution in pixels, specify the width (`cxVectorToRasterXRes`) and/or height (`cyVectorToRasterYRes`).

To export the largest image that fits within a bounding box, without changing the original aspect ratio, set both the width and height. For example, to export the largest image that fits in an 800x500 bounding box:

```
cxVectorToRasterXRes=800
cyVextorToRasterYRes=500
```

Alternatively you can fix one of the dimensions. Set one value and set the other to zero. For example, to export images with a height of 1500 pixels and whatever width is necessary to maintain the original aspect ratio:

```
cxVectorToRasterXRes=0
cyVextorToRasterYRes=1500
```

The maximum size permitted for either dimension is 4000 pixels.

To set the resolution proportionally

To set the resolution proportionally, set `cxVectorToRasterXRes` to a negative value. A negative value represents a percentage of the original resolution. Set `cyVectorToRasterYRes` to `0` (zero). Negative (percentage) values for `cyVectorToRasterYRes` are ignored.

The following example exports a graphic at 50 percent of its original resolution:

```
cxVectorToRasterXRes=-50
cyVectorToRasterYRes=0
```

cyVectorToRasterYRes

This is a read and write property.

This property controls the vertical resolution at which presentations and vector graphics are converted. The default value is `0`, which means HTML Export retains the original resolution.

This property is set in conjunction with cxVectorToRasterXRes, on the previous page.

You can specify the resolution as an absolute size in pixels, or as a proportion of the original size.

KeyView always maintains the aspect ratio of the original graphic and does not increase the resolution. If you set values that would enlarge a graphic, KeyView only changes the size of the HTML element.

To set the resolution in pixels

To specify the resolution in pixels, specify the width (`cxVectorToRasterXRes`) and/or height (`cyVectorToRasterYRes`).

To export the largest image that fits within a bounding box, without changing the original aspect ratio, set both the width and height. For example, to export the largest image that fits in an 800x500 bounding box:

```
cxVectorToRasterXRes=800
cyVextorToRasterYRes=500
```

Alternatively you can fix one of the dimensions. Set one value and set the other to zero. For example, to export images with a height of 1500 pixels and whatever width is necessary to maintain the original aspect ratio:

```
cxVectorToRasterXRes=0
cyVextorToRasterYRes=1500
```

The maximum size permitted for either dimension is 4000 pixels.

To set the resolution proportionally

To set the resolution proportionally, set `cxVectorToRasterXRes` to a negative value. A negative value represents a percentage of the original resolution. Set `cyVectorToRasterYRes` to `0` (zero). Negative (percentage) values for `cyVectorToRasterYRes` are ignored.

The following example exports a graphic at 50 percent of its original resolution:

```
cxVectorToRasterXRes=-50
cyVectorToRasterYRes=0
```

dwFlags

This is a read and write property.

Instructions on how to process the content associated with a paragraph or character style. See Flags for Defining Styles, on page 86 for descriptions of each flag.

> **NOTE: Note**: The value of `Flags` in the template files is used in `dwFlags`.

This property applies to word processing documents only.

FontSizeMap_nSize[1...7]

This is a read and write property.

The font sizes to which the HTML tags `<font size=1>` through `<font size=7>` correspond. If you set `bSupportUserFontSizeMapping` to **FALSE**, this parameter can be left blank.

The values in `FontSizeMap` indicate the range for the HTML tag `<font size=#>`. For example if you specify 6, 9, 12, 18, 21, 24, and 28:

- font size 6 in the source document is mapped to `<font size=1>` in the output HTML

- font size 9 in the source document is mapped to `<font size=2>` in the output HTML

- font size 12 in the source document is mapped to `<font size=3>` in the output HTML

- and so on, up to <font size=7>

When the HTML output is viewed, the browser maps `<font size=#>` to a specific font size.

The default font sizes are 8, 10, 12, 14, 18, 24, and 36.

headingCreateType

This is a read and write property.

This property determines how HTML Export subdivides the source document into table of contents entries. This should be set to one of the two options that are enumerated in `KVHeadingCreateOptions` in `kvhtml.h`. The determination of table of contents entries is based on whether the source document contains *heading styles* or whether *text attributes* (bold, underlined, italic, and so on) conform to the criteria defined in `KVHTMLHeadingInfo`. Heading styles are predefined style tags, such as "Heading 1" and "Heading 2" tags in a Microsoft Word document.

This property applies to word processing documents only. The enumerated types are described below.

KVHC_DocHeadingsOnly

This instructs HTML Export to rely exclusively on heading styles defined in the source document. However, if the source document does not contain heading styles, HTML Export generates blocks on its own by using the criteria defined by the parameters in `KVHTMLHeadingInfo`.

KVHC_CreateHeadingsAlways

This instructs HTML Export to use the heading styles in the source document when available, and to also automatically create table of contents entries based on the criteria defined by the parameters in `KVHTMLHeadingInfo`.

> **NOTE: Note**: When the determination of table of contents entries is based on text attributes, source text must meet *all* the criteria defined by the parameters of `KVHTMLHeadingInfo` before it is converted to a heading and included as a table of contents entry.

InputCharSet

This is a read-only property.

The character set of the source document if that information is ascertainable. The character sets that are available are enumerated in `KVCharSet` of `kvtypes.h`.

IcbBlockSize

This is a read and write property.

The maximum size (in bytes) of heading level 1 HTML output files. This number is used as a guideline and can be exceeded to break content at a logical location. This setting is not used when exporting spreadsheets.

IcbMaxMemUsage

This is a read and write property.

The maximum memory allocated dynamically for token buffers during file processing. If this maximum is reached, Export performs a swap-to-disk operation internally, and then reuses the memory blocks. Export maintains an internal minimum memory size.

This property applies to word processing or text documents only.

The default is `LONG_MAX`. The unit is in bytes.

MarkUpEnd

This is a read and write property.

The markup to be added to the end of a paragraph or character style. See Map Styles, on page 84 for more information on mapping styles.

This property applies to word processing documents only.

MarkUpStart

This is a read and write property.

The markup to be added to the beginning of a paragraph or character style. See Map Styles, on page 84 for more information on mapping styles.

This property applies to word processing documents only.

nCompressionQuality

This is a read and write property.

This property controls the output quality of graphics that support compression quality (for example, JPEG). A value of 0 means default quality (85 compression), 1 is the lowest quality (highest compression and therefore smallest file size), and 100 is the highest quality (no compression and therefore the largest file size).

This property applies to word processing documents only.

nRowsBeforeSplit

This is a read and write property.

The approximate number of spreadsheet rows to be processed before splitting a table. This helps to prevent large spreadsheet tables from occurring in a single document, which can cause speed and processing problems for the browser.

This property applies to spreadsheets only.

nTableBorderWidth

This is a read and write property.

This property sets the width of the table border in pixels.

This property applies to word processing documents only.

NumStyles

The number of developer-defined styles to be used by HTML Export in formatting Cascading Style Sheet (CSS) information passed by the source document.

See AddStyleMapping, on page 255, GetStyleMapping, on page 256, and RemoveStyleMapping, on page 258 for more information.

This property applies to word processing documents only.

OutputCharSet

This is a read and write property.

The character set to use for textual output if the document character set cannot be determined from the document, and the input character set is not specified by `SrcCharSet`. To ensure that the source character set defined here is used, you must set `bForceOutputCharSet` to **TRUE**. The character sets that are available are enumerated in `KVCharSet` in `kvtypes.h`. See Convert Character Sets, on page 80 for more information on mapping character sets.

Supported Formats, on page 284 lists the file formats for which character set information can be determined.

OutputLanguageID

This is a read and write property.

The language for the textual output of language-specific data like time and date. Note that `OutputLanguageID` must be in the system locale. If `OutputLanguageID` is invalid or not supplied, the system default is used. Language IDs are enumerated in `KVLanguageID` in `kvtypes.h`.

OutputRasterGraphicType

This is a read and write property.

The output format of rasterized embedded graphics. There are six options enumerated in `KVHTMLGraphicType` in `kvhtml.h`. The default is JPEG.

OutputVectorGraphicType

This is a read and write property.

The output format of vector graphics. The options are enumerated in `KVHTMLGraphicType` in `kvhtml.h`. The default is JPEG.

For more information on displaying vector graphics, see Display Vector Graphics on UNIX and Linux, on page 87.

pH*n*_bNoMultiSpaces

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. If you set `pHn_bNoMultiSpaces` to **TRUE**, the text in the source document must *not* contain two or more contiguous white spaces in order for HTML Export to consider converting it to a heading. The default is `FALSE`.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_bNonZeroIndent

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. If you set `pHn_bNonZeroIndent` to **TRUE**, the text in the source document must be indented in order for HTML Export to consider converting it to a heading. If you set `pHn_bNonZeroIndent` to `FALSE`, the text must be aligned left. The default is `FALSE`.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_bNoTabs

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. If you set `pHn_bNoTabs` to **TRUE**, the text in the source document must *not* contain tabs in order for HTML Export to consider converting it to a heading. The default is `FALSE`.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_bMustBeBold

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. If you set `pHn_bMustBeBold` to **TRUE**, the text in the source document must be bold in order for HTML Export to consider converting it to a heading. The default is `TRUE`.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_bMustBeItalic

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. If you set pH*n*_bMustBeItalic to **TRUE**, the text in the source document must be italic in order for HTML Export to consider converting it to a heading. The default is TRUE.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of KVHTMLHeadingInfo.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_bMustBeUnderlined

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. If you set pH*n*_bMustBeUnderlined to **TRUE**, the text in the source document must be underlined in order for HTML Export to consider converting it to a heading. The default is TRUE.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of KVHTMLHeadingInfo.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_fontSizeMax

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. It specifies the maximum font size that text in the source document can be in order for HTML Export to consider converting it to a heading. The default is 20 for heading level 1 and 14 for heading levels 2 to 6.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of KVHTMLHeadingInfo.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_fontSizeMin

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. It specifies the minimum font size that text in the source document can be in order for HTML Export to consider converting it to a heading. The default is 14 for heading level 1 and 14 for heading levels 2 to 6.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

pH*n*_maxParaLen

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. It specifies the maximum number of characters that text in the source document can contain in order for HTML Export to consider converting it to a heading. The default is 80.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

This property applies to word processing documents only.

pH*n*_minParaLen

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. It specifies the minimum number of characters that text in the source document can contain in order for HTML Export to consider converting it to a heading. The default is 3.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

This property applies to word processing documents only.

pH*n*_mSpaceAfter

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. It specifies the amount of space in TWIPS (20th of a point) that must follow a paragraph in the source document in order for HTML Export to consider converting the paragraph to a heading. If –1 is used, the amount of space after the paragraph is not considered in the heading generation.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

See headingCreateType, on page 268 for more information on automatic generation of headings.

For this property, *n* can be heading level 1 through 6.

This property applies to word processing documents only.

pH*n*_mSpaceBefore

This is a read and write property.

This is one of the criteria used to determine whether source text should be converted to a heading in the HTML output. It specifies the amount of space in TWIPS (20th of a point) that must come before a paragraph in the source document in order for HTML Export to consider converting the paragraph to a heading. If –1 is used, the amount of space before the paragraph is not considered in the heading generation.

To determine whether source text should be converted to a heading, HTML Export also considers whether the text meets the criteria defined by the other parameters of `KVHTMLHeadingInfo`.

For this property, *n* can be heading level 1 through 6.

This property applies to word processing documents only.

pszAuthor

The contents of the Author property field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszBaseURL

This is a read and write property.

The base URL that replaces the `$BASE` token in the HTML output.

pszComments

The contents of the Comments property field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszChunkTemplate

This is a read and write property.

If an H1 HTML block is subdivided into separate files as a result of the size limitations specified in `lcbBlockSize`, `pszChunkTemplate` provides a template for creating a table of contents entry for the new file. The chunk number can be made a part of this template by inserting the token `$SPLITBLOCKNUMBER` (for example, `"Page $SPLITBLOCKNUMBER"`).

pszDefaultOutputDirectory

This is a read and write property.

The default output directory for auxiliary files that are created. The default is the directory in which your application is running.

pszEndBlock

This is a read and write property.

The markup and HTML Export tokens to be output at the end of each block created as a result of the of `lcbBlockSize` or `bHardPageMakesNewBlock`.

pszFirstH1End

This is a read and write property.

The markup and HTML Export tokens to include in the output at the end of the first H1 HTML block created, that is, the block associated with the first H1 table of contents entry.

pszFirstH1Start

This is a read and write property.

The markup and HTML Export tokens to include in the output at the beginning of the first H1 HTML block created, that is, the block associated with the first H1 table of contents entry.

pszH[2..6]HTML

This is a read and write property.

The markup and HTML Export tokens to include in the output in an HTML block for heading levels 2 through 6.

pszInputFile

This is a read and write property.

The full path and file name of the file that is converted to HTML, or has its metadata or format information extracted. You must set this property for the conversion to proceed.

pszKeyViewDir

This is a read and write property.

The location of the directory where HTML Export is installed. If this property is NULL, required components might not be found.

pszKeywords

This is a read-only property.

The contents of the Keywords property field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszLastH1End

This is a read and write property.

The markup and HTML Export tokens to include in the output at the end of the last H1 HTML block created, that is, the block associated with the last H1 table of contents entry.

pszLastH1Start

This is a read and write property.

The markup and HTML Export tokens to include in the output at the beginning of the last H1 HTML block created, that is, the block associated with the last H1 table of contents entry.

pszLastSavedby

This is a read-only property.

The contents of the "Last Saved by" field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszMainBottom

This is a read and write property.

The markup and tokens to include in the output at the end of the main HTML file created during document conversion. This parameter should at least contain the `</html>` tag.

pszMainTop

This is a read and write property.

The markup and tokens to include in the output at the start of the main HTML file created during document conversion. Most of the template files feature `<meta>` tags with tokens that store the input document's metadata. This parameter should at least contain the `<html>` tag. For frames-style HTML output, this parameter must include the `<frame>` tag.

pszMainURL

This is a read and write property.

The URL that replaces the `$MAINURL` token in the HTML output.

pszMiddleH1End

This is a read and write property.

The markup and HTML Export tokens to include in the output at the end of those H1 HTML blocks that are neither the first nor the last H1 blocks created, that is, blocks that are associated with all but the first and last H1 table of contents entry.

pszMiddleH1Start

This is a read and write property.

The markup and HTML Export tokens to include in the output at the beginning of those H1 HTML blocks that are neither the first nor the last H1 blocks created, that is, blocks that are associated with all but the first and last H1 table of contents entry.

pszPicPath

This is a read and write property.

The output directory for picture files created during the conversion. The default is the directory in which your application is running.

If specified, this parameter can also be used by the callback functions `KVHTMLGetAnchorEx` and `KVHTMLGetAuxOutputEx`.

This property applies to word processing documents only.

pszPicURL

This is a read and write property.

The URL of the picture files created from embedded graphics in the source document.

This property applies to word processing documents only.

pszRevNumber

This is a read-only property.

The contents of the Revision number property field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszStartBlock

This is a read and write property.

The markup and HTML Export tokens to include in the output at the beginning of each block created as a result of `lcbBlockSize` or `bHardPageMakesNewBlock`.

pszSubject

This is a read-only property.

The contents of the Subject property field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszTableHTML

This is a read and write property.

Specifies the markup (no tokens) output at the beginning of each table created during the HTML conversion process. If you set this property, table cell color and border information from the document is ignored. This property is used in conjunction with `bTableHTMLForSpreadsheetOnly` to control the look of generated spreadsheets.

For example, to set the tables to be centered, colored, and with big borders, use:

```
pszTableHTML=<table bgcolor="teal" border="13" align="center">
```

pszTemplate

This is a read-only property.

The contents of the Template property field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszTitle

This is a read-only property.

The contents of the Title property field taken from the source document.

See GetSummaryInfo, on page 257 for more information on all metadata that you can obtain from a document.

pszTOC_H[1..6]

This is a read and write property.

The markup and HTML Export tokens required to process the table of contents entries for heading levels 1 through 6. For example:

```
<a href="$ANCHOR" target="right">$TOCTE</a>
```

pszTOCH[1..6]End

This is a read and write property.

The markup and HTML Export tokens to include in the output at the end of a table of contents block for heading levels 1 through 6 TOC entries. An example of `pszTOCH[1..6]End` is `</ol>`.

pszTOCH[1..6]LeafNode

This is a read and write property.

The markup that replaces `pszTOC_H[1..6]` entries for leaf nodes in the table of contents. A leaf node is a node that has no children.

pszTOCH[1..6]Start

This is a read and write property.

The markup and HTML Export tokens to include in the output at the beginning of a table of contents block for heading levels 1 through 6 TOC entries. An example of `pszTOC_H[1..6]Start` would be `<ol type=I>`.

pszUserSummary

This is a read and write property.

The markup and tokens that are generated when the tokens `$USERSUMMARY`, `$SUMMARY`, or `$SUMMARYNN` are used. For example:

"`<meta title="$SUMMARY01" author="$SUMMARY03"/>`".

pszXFile

This is a read and write property.

The markup and HTML Export tokens that are generated and placed in an extra HTML file. This file holds content from the source document (for example, a separate file containing `$TOC` that can be displayed within an HTML frame). To process this file, use the `$XANCHOR` token. See Export Tokens, on page 390 for more information on `$XANCHOR`.

For example, if the extra file is to contain the table of contents in frame-based HTML, set `pszXFile` to **$TOC** and place the following HTML markup in the `pszMainTop` element:

`<frame src="$XANCHOR" name="left" scrolling="auto" target="right">`.

pszXStartBlock

This is a read and write property.

The markup and tokens to include in the output at the beginning of each HTML block generated by the `$XANCHOR` token. If either this parameter or `pszXEndBlock` is defined, both `pszStartBlock` and `pszEndBlock` are ignored. See Export Tokens, on page 390 for more information on `$XANCHOR`.

pszXEndBlock

This is a read and write property.

The markup and tokens to include in the output at the end of each HTML block generated by the `$XANCHOR` token. If either this parameter or `pszXStartBlock` is defined, both `pszStartBlock` and `pszEndBlock` are ignored. See Export Tokens, on page 390 for more information on `$XANCHOR`.

SATableBorder

This is a read and write property.

This property specifies whether table borders in the HTML output are based on the setting in the source document, are always on, or are always off. The options are enumerated in `ENSATableBorder` of `kvtypes.h`.

This property applies to word processing documents only.

SrcCharSet

This is a read and write property.

This property specifies the source character set if the reader for the document type cannot determine the character set. To ensure that the source character set defined here is used, you might have to set

`bForceSrcCharSet` to **TRUE**. The character sets that are available are enumerated in `KVCharSet` of `kvtypes.h`. See Convert Character Sets, on page 80 for more information on character set mapping.

The section Supported Formats, on page 284 lists the file formats for which character set information can be determined.

StyleSheetType

This is a read and write property.

One of the three enumerated options for processing style sheet information.

- To disable style sheet formatting, set `StyleSheetType` to **STYLESHEET_DISABLED**.

- To enable Cascading Style Sheet (CSS) formatting and output the generated formatting data within the HTML output stream, set `StyleSheetType` to **CSS_INLINE**.

- To enable CSS formatting and output the generated formatting data in an external CSS file referenced in the HTML output as a tag, set `StyleSheetType` to **CSS_TOFILE**.

Refer to KVHTMLStyleSheetType, on page 241 for more information on these enumerated options.

StyleName

This is a read and write property.

The name of the word processing style (for example, "Heading 1") to which style mapping is applied. Style names are case sensitive. See Map Styles, on page 84 for more information on mapping styles.

This property applies to word processing documents only.

Timeout

This is a read and write property.

The number of seconds that HTML Export should allow for converting a source document before killing the process. This property guards against hung processes. Setting this property to **-1** deprives HTML Export of a timeout guideline, that is, no processes are terminated. This property applies only to the out-of-process COM object.

Part V: Appendixes

This section lists supported formats, supported character sets and redistributed files, and provides information on format detection.

- Supported Formats

- Detected Formats

- Character Sets

- Extract and Format Lotus Notes Subfiles

- Export Tokens

- File Format Detection

- Files Required for Redistribution

- Password Protected Files

Appendix A: Supported Formats

This section lists the file formats that KeyView can process (either filter, convert, or display).

Supported Formats

The tables in this section provide the following information:

- The file formats supported by the Filter API, Export API, Viewing API, and File Extraction API. The supported versions and the format's extension are also listed. All of the formats listed in this section can be detected by the KeyView format detection module (`kwad`). For a complete list of formats that can be detected, see Detected Formats, on page 314.

- The file formats for which KeyView can detect and extract the character set and metadata information (properties such as title, author, and subject).

  Even though a file format might be able to provide character set information, some documents might not contain character set information. Therefore, the document reader would not be able to determine the character set of the document. In this case, either the operating system code page or the character set specified in the API is used.

- The document reader used to filter each format.

Key to Support Tables

Symbol	Description
Y	The format is supported. You can extract metadata for this format. You can determine the character set for this format.
N	The format is not supported. You cannot extract metadata for this format. You cannot determine the character set for this format.
P	Partial metadata is extracted from this format. Some non-standard fields are not extracted.
T	Only text is extracted from this format. Formatting information is not extracted.
M	Only metadata (title, subject, author, and so on) is extracted from this format. Text and formatting information are not extracted.

Archive Formats

Supported Archive Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
7-Zip	4.57	z7zsr, multiarcsr[1]	7Z	N	N	Y	Y	N	n/a	N
AD1	n/a	ad1sr	AD1	N	N	Y	Y	N	n/a	N
ARJ	n/a	multiarcsr	ARJ	N	N	N	Y	N	n/a	N
B1	n/a	b1sr	B1	N	N	Y	Y	N	n/a	N
BinHex	n/a	kvhqxsr	HQX	N	N	Y	Y	N	n/a	N
Bzip2	n/a	bzip2sr	BZ2	N	N	Y	Y	N	n/a	N
CPIO (copy-in-and-out archiver)	n/a	multiarcsr		N	N	N	Y	N	n/a	N
Debian binary package	n/a	multiarcsr	DEB	N	N	N	Y	N	n/a	N
DOS/Windows Object Library	n/a	multiarcsr	LIB, A	N	N	N	Y	N	n/a	N
Expert Witness Compression Format (EnCase)	6	encasesr	E01, L01	N	N	Y	Y	N	n/a	N
	7	encase2sr	Lx01	N	N	Y	Y	N	n/a	N

[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
GZIP	2	kvgzsr	GZ	N	N	N	Y	N	n/a	N
		kvgz	GZ	N	N	Y	N	N	n/a	N
ISO	n/a	isosr	ISO	N	N	Y	Y	N	n/a	N
Java Archive	n/a	unzip	JAR	N	N	Y	Y	N	n/a	N
Legato EMailXtender Archive	n/a	emxsr	EMX	N	N	Y	Y	N	n/a	N
LZMA compressed data	n/a	multiarcsr	LZMA	N	N	N	Y	N	n/a	N
MacBinary	n/a	macbinsr	BIN	N	N	Y	Y	N	n/a	N
Mac Disk Copy Disk Image	n/a	dmgsr	DMG	N	N	Y	Y	N	n/a	N
Mac OS-X (Mach-O) executable	n/a	multiarcsr		N	N	N	Y	N	n/a	N
Microsoft Backup File	n/a	bkfsr	BKF	N	N	Y	Y	N	n/a	N
Microsoft Cabinet format	1.3	cabsr	CAB	N	N	Y	Y	N	n/a	N
Microsoft Compiled HTML Help	3	chmsr	CHM	N	N	Y	Y	N	n/a	N
Microsoft Compressed Folder	n/a	lzhsr	LZH LHA	N	N	N	Y	N	n/a	N
Microsoft Power BI Desktop format	n/a	unzip	PBIX	N	N	N	Y	N	n/a	N
MSI (Microsoft Installer)	n/a	multiarcsr	MSI	N	N	N	Y	N	n/a	N

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
PKZIP	through 9.0	unzip	ZIP	N	N	Y	Y	N	n/a	N
RAR archive	2.0 through 3.5	rarsr	RAR	N	N	N	Y	N	n/a	N
RAR5 archive	5	multiarcsr	RAR5	N	N	N	Y	N	n/a	N
RPM (package manager file)	n/a	multiarcsr	RPM	N	N	N	Y	N	n/a	N
SUN PEX Binary Archive	n/a	multiarcsr		N	N	Y	Y	N	n/a	N
Tableau Packaged Data Source format	n/a	unzip	TDSX	N	N	N	Y	N	n/a	N
Tableau Packaged Workbook format	n/a	unzip	TWBX	N	N	N	Y	N	n/a	N
Tape Archive	n/a	tarsr	TAR	N	N	Y	Y	N	n/a	N
UNIX Compress	n/a	kvzeesr	Z	N	N	N	Y	N	n/a	N
		kvzee	Z	N	N	Y	N	N	n/a	N
UUEncoding	all versions	uudsr	UUE	N	N	Y	Y	N	n/a	N
XZ	n/a	multiarcsr	XZ	N	N	N	Y	N	n/a	N
Windows Imaging Format	n/a	multiarcsr	WIM	N	N	N	Y	N	n/a	N

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Windows Scrap File	n/a	olesr	SHS	N	N	N	Y	N	n/a	N
WinZip	through 10	unzip	ZIP	N	N	Y	Y	N	n/a	N
XAR (Extensible Archive)	n/a	multiarcsr		N	N	N	Y	N	n/a	N
Zipped Keyhole Markup Language	n/a	unzip	ZIP	N	N	N	Y	N	n/a	N

Binary Format

Supported Binary Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Executable	n/a	exesr	EXE	N	N	Y	N	N	n/a	N
Link Library	n/a	exesr	DLL	N	N	Y	N	N	n/a	N

Computer-Aided Design Formats

Supported CAD Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
AutoCAD Drawing	R13, R14, R15/2000, 2004, 2007, 2010, 2013, 2018	kpODArdr kpDWGrdr[1]	DWG	Y	Y	Y	N	Y	Y	N
AutoCAD Drawing Exchange	R13, R14, R15/2000, 2004, 2007, 2010, 2013	kpODArdr kpDXFrdr[2]	DXF	Y	Y	Y	N	Y	Y	N
CATIA formats	5	kpCATrdr	CAT[3]	Y	N	N	N	Y	N	N
Microsoft Visio	4, 5, 2000, 2002, 2003, 2007, 2010[4]	vsdsr	VSD	Y	Y	Y	Y[5]	Y	Y	N
		kpVSD2rdr	VSD, VSS VST	Y	Y	Y	N	Y	Y	N

[1]The kpODArdr reader can filter, export, and view all versions but is supported only on Windows, Linux, and OSX. The kpDWGrdr reader is used on AIX, FreeBSD, Solaris, and SPARC platforms, but does not support graphics for versions after 2004 or text for versions after 2013.
[2]The kpODArdr reader can filter, export, and view all versions but is supported only on Windows, Linux, and OSX. The kpDXFrdr reader is used on AIX, FreeBSD, Solaris, and SPARC platforms, but does not support graphics for versions after 2004.
[3]All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.
[4]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.
[5]Extraction of embedded OLE objects is supported for Filter on Windows platforms only.

Supported CAD Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
	2013	ActiveX components	VSDM VSSM VSTM VSDX VSSX VSTX	N	N	Y[1]	N	Y	N	N
		kpVSDXrdr	VSDM VSSM VSTM VSDX VSSX VSTX	Y	Y	Y	Y	Y	Y	N
Unigraphics (UG) NX		kpUGrdr	PRT	Y	N	N	N	N	N	N

Database Formats

Supported Database Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
dBase Database	III+, IV	dbfsr	DBF	Y	Y	Y	N	N	N	N

[1]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 Database Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Access	95, 97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	mdbsr	MDB, ACCDB	Y	T	T	N	N	Y[1]	N
Microsoft Project	2000, 2002, 2003, 2007, 2010, 2013, 2016	mppsr	MPP	Y	Y	Y	Y	Y	Y	N

Desktop Publishing

Supported Desktop Publishing Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Publisher	98 to 2016	mspubsr	PUB	Y	T	T	Y	Y	Y	N

Display Formats

Supported Display Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe PDF	1.1 to 1.7	pdfsr	PDF	Y	Y	N	Y[2]	Y	Y	N
		pdf2sr	PDF	N	Y	N	N	N	N	N

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

[2]Includes support for extraction of subfiles from PDF Portfolio documents.

Supported Display Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
		kppdfrdr	PDF	N	Y	Y	N	N	N	N
		kppdf2rdr[1]	PDF	N	N	Y	N	N	N	N

Graphic Formats

Supported Graphic Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Computer Graphics Metafile	n/a	kpcgmrdr[2]	CGM	Y	Y	Y	N	N	N	N
CorelDRAW[3]	through 9.0 10, 11, 12, X3	kpcdrrdr	CDR	N	Y	Y	N	N	N	N
DCX Fax System	n/a	kpdcxrdr	DCX	N	Y	Y	N	N	N	N
Digital Imaging & Communications in	n/a	dcmsr	DCM	M	N	N	N	Y	N	N

[1]kppdf2rdr is an alternate graphic-based reader that produces high-fidelity output but does not support other features such as highlighting or text searching.
[2]Files with non-partitioned data are supported.

[3]CDR/CDR with TIFF header.

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Medicine (DICOM)										
Encapsulated PostScript (raster)	TIFF header	kpepsrdr	EPS	N	Y	Y	N	N	N	N
Enhanced Metafile	n/a	kpemfrdr	EMF	Y	Y	Y	N	Y	N	N
GIF	87, 89	kpgifrdr	GIF	N	Y	Y	N	N	N	N
		gifsr		M	M	N	N	Y	N	N
ISO-BMFF JPEG 2000 compound image	n/a	kpjp2000rdr	JPM	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
ISO-BMFF JPEG 2000 image	n/a	kpjp2000rdr	JP2	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
ISO-BMFF JPEG 2000 with extensions	n/a	kpjp2000rdr	JPX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
JBIG2	n/a	kpJBIG2rdr	JBIG2	N	Y	Y	N	N	N	N
JPEG	n/a	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

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
JPEG 2000 PGX Verification Model image	n/a	kpjp2000rdr	PGX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
Lotus AMIDraw Graphics	n/a	kpsdwrdr	SDW	N	Y	Y	N	N	N	N
Lotus Pic	n/a	kppicrdr	PIC	Y	Y	Y	N	N	N	N
Macintosh Raster	2	kppctrdr	PIC PCT	N	Y	Y	N	N	N	N
MacPaint	n/a	kpmacrdr	PNTG	N	Y	Y	N	N	N	N
Microsoft Office Drawing	n/a	kpmsordr	MSO	N	Y	Y	N	N	N	N
Omni 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
Scalable Vector Graphics	n/a	xmlsr	SVG	Y	T	T	N	Y	Y	N
SGI RGB Image	n/a	kpsgirdr	RGB	N	Y	Y	N	N	N	N
Sun Raster Image	n/a	kpsunrdr	RS	N	Y	Y	N	N	N	N

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Tagged Image File	through 6.0[1]	tifsr	TIFF	M	M	N	N	Y	N	N
		kptifrdr	TIFF	N	Y	Y	N	N	N	N
Truevision Targa	2	kpTGArdr	TGA	N	Y	Y	N	N	N	N
Windows Animated Cursor	n/a	kpanirdr	ANI	N	Y	Y	N	N	N	N
Windows Bitmap	n/a	kpbmprdr	BMP	N	Y	Y	N	N	N	N
		bmpsr	BMP	M	M	N	N	Y	N	N
Windows Icon Cursor	n/a	kpicordr	ICO	N	Y	Y	N	N	N	N
Windows Metafile	3	kpwmfrdr	WMF	Y[2]	Y	Y	N	N	N	N
WordPerfect Graphics 1	1	kpwpgrdr	WPG	N	Y	Y	N	N	N	N
WordPerfect Graphics 2	2, 7	kpwg2rdr	WPG	N	Y	Y	N	N	N	N

[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.

[2]Windows Metafiles can contain both raster images (KeyView file class 4) and vector graphics (KeyView file class 5). Filtering is supported only for vector graphics (class 5).

Mail Formats

Supported Mail Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Documentum EMCMF	n/a	msgsr	EMCMF	N	N	Y	Y	Y	Y	N
Domino XML Language[1]	n/a	dxlsr	DXL	N	N	Y	Y	Y	N	N
GroupWise FileSurf	n/a	gwfssr	GWFS	N	N	Y	Y	Y	N	N
Legato Extender	n/a	onmsr	ONM	N	N	Y	Y	Y	N	N
Lotus Notes database	4, 5, 6.0, 6.5, 7.0, 8.0	nsfsr	NSF	N	N	Y	Y	Y	N	N
Mailbox[2]	Thunderbird 1.0, Eudora 6.2	mbxsr[3]	MBX	N	N	T	Y	Y	Y	N
Microsoft	2004	entsr	various	N	N	Y	Y	Y	Y	N

[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
Entourage Database										
Microsoft Outlook	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016, 2019	msgsr[1]	MSG, OFT	Y	T	T	Y	Y	Y [2]	N
Microsoft Outlook DBX	5.0, 6.0	dbxsr	DBX	N	N	Y	Y	Y	Y	N
Microsoft Outlook Express	Windows 6 MacIntosh 5	emlsr[3]	EML	Y	T	T	Y	Y	Y	N
		mbxsr[4]	EML	N	N	T	Y	Y	Y	N
Microsoft Outlook iCalendar	1.0, 2.0	icssr	ICS, VCS	N	N	Y	Y	Y	Y	N
Microsoft Outlook for Macintosh	2011	olmsr	OLM	N	N	Y	Y	N	Y	N
Microsoft Outlook Offline Storage File	97, 2000, 2002, 2003, 2007, 2010, 2013	pffsr[5]	OST	N	N	Y	Y	Y	Y	N

[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
Microsoft Outlook Personal Folder[1]	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016, 2019	pstsr[2]	PST	N	N	Y	Y	Y	N	N
	97, 2000, 2002, 2003, 2007, 2010, 2013	pstnsr	PST	N	N	Y	Y	Y	Y	N
	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016, 2019	pstxsr	PST	N	N	Y	Y	Y	Y	N
Microsoft Outlook vCard Contact	2.1, 3.0, 4.0	vcfsr	VCF	Y	Y	T	N	Y	N	N
Text Mail (MIME)	n/a	emlsr[3]	various	Y	T	T	Y	Y	Y	N
		mbxsr[4]	various	Y	T	T	Y	Y	Y	N
Transport Neutral Encapsulation Format	n/a	tnefsr	various	N	N	Y	Y	Y	Y	N

[1]KeyView provides several readers capable of processing PST files. The `pstsr` reader uses the Microsoft Messaging Application Programming Interface (MAPI), works only on Windows, and requires that you have Microsoft Outlook installed. The `pstxsr` reader is available for Windows (32-bit and 64-bit) and Linux (64-bit only) and does not require Microsoft Outlook. The `pstnsr` reader is an alternative reader that does not require Microsoft Outlook, for all platforms not supported by `pstxsr`. For more information about these readers, see "Extract Subfiles from Outlook Personal Folders Files" in Chapter 3.

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

[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
3GPP video file	n/a	mpeg4sr	3GP	M	N	N	N	Y	N	N
3GPP2 video file	n/a	mpeg4sr	3G2	M	N	N	N	Y	N	N
Adobe Flash Player audio	n/a	mpeg4sr	F4A	M	N	N	N	Y	N	N
Adobe Flash Player audio book	n/a	mpeg4sr	F4B	M	N	N	N	Y	N	N
Adobe Flash Player protected video	n/a	mpeg4sr	F4P	M	N	N	N	Y	N	N
Adobe Flash Player video	n/a	mpeg4sr	F4V	M	N	N	N	Y	N	N
Apple ISO-BMFF QuickTime video	n/a	MCI	QT MOV	N	N	Y	N	N	N	N
Apple MPEG-4 Part 14 audio	n/a	mpeg4sr	M4A	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14 audio book	n/a	mpeg4sr	M4B	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14 protected audio	n/a	mpeg4sr	M4P	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14	n/a	mpeg4sr	M4V	M	N	N	N	Y	N	N

Supported Multimedia Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
video										
Audible Enhanced Audiobook	n/a	mpeg4sr	AAX	M	N	N	N	Y	N	N
KDDI video file	n/a	MCI		N	N	Y	N	N	N	N
Advanced Systems Format	1.2	asfsr	ASF WMA WMV	N	N	N	N	Y	N	N
Audio Interchange File Format	n/a	MCI	AIFF	N	N	Y	N	N	N	N
		aiffsr	AIFF	M	N	N	N	Y	N	N
ISO-BMFF MPEG-4 with AVC extension	n/a	mpeg4sr		M	N	N	N	Y	N	N
Microsoft Wave Sound	n/a	MCI	WAV	N	N	Y	N	N	N	N
		riffsr	WAV	M	N	N	N	Y	N	N
MIDI	n/a	MCI	MID	N	N	Y	N	N	N	N
Mobile QuickTime video	n/a	mpeg4sr	MQV	M	N	N	N	Y	N	N
Motion JPEG 2000	n/a	kpjp2000rdr	MJ2 MJP2	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
MPEG-1 Audio layer 3	ID3 v1 and v2	MCI	MP3	N	N	Y	N	N	N	N
		mp3sr	MP3	M	M	Y	N	Y	N	N

Supported Multimedia Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
MPEG-1 Video	2, 3	MCI	MPG	N	N	Y	N	N	N	N
MPEG-2 Audio	n/a	MCI	MPEGA	N	N	Y	N	N	N	N
MPEG-21	n/a	mpeg4sr		M	N	N	N	Y	N	N
MPEG-4 Audio	n/a	mpeg4sr	MP4 3GP	M	N	N	N	Y	N	N
Nero AAC audio	n/a	mpeg4sr		M	N	N	N	Y	N	N
Nero MPEG-4 profile	n/a	mpeg4sr		M	N	N	N	Y	N	N
Nero MPEG-4 profile with AVC extension	n/a	mpeg4sr		M	N	N	N	Y	N	N
NeXT/Sun Audio	n/a	MCI	AU	N	N	Y	N	N	N	N
NTT MPEG-4	n/a	mpeg4sr		M	N	N	N	Y	N	N
QuickTime Movie	2, 3, 4	MCI	QT MOV	N	N	Y	N	N	N	N
Sony PSP MPEG-4	n/a	mpeg4sr	MP4	M	N	N	N	Y	N	N
Sony XAVC video	n/a	mpeg4sr		M	N	N	N	Y	N	N
Windows Video	2.1	MCI	AVI	N	N	Y	N	N	N	N

NOTE:
Depending on the default multimedia player installed on your computer, the View API might not be able to play some supported multimedia formats. To play multimedia files, the View API uses the Windows Media Control Interface (MCI) to communicate with the multimedia player installed on your computer. If the player does not play a multimedia file that is supported by the Viewing SDK, the View API cannot

> play the file.
>
> If you cannot play a supported multimedia file by using the View API, install a different multimedia player or compressor/decompressor (codec) component.

Presentation Formats

Supported Presentation Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Keynote	2, 3, '08, '09	kpIWPGrdr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16, '18 iCloud 2018	kpIWPG13rdr [1]	KEY	Y	T	N	N	N	N	N
Applix Presents	4.0, 4.2, 4.3, 4.4	kpagrdr	AG	Y	Y	Y	N	N	N	N
Corel Presentations	6, 7, 8, 9, 10, 11, 12, X3	kpshwrdr	SHW	Y	Y	Y	N	N	N	N
Extensible Forms Description Language	n/a	kpXFDLrdr	XFD XFDL	Y	Y	Y	N	Y	Y	N
Lotus Freelance Graphics	96, 97, 98, R9, 9.8	kpprzrdr	PRZ	Y	Y	Y	N	N	N	N
Lotus Freelance Graphics 2	2	kpprerdr	PRE	Y	Y	Y	N	N	N	N

[1]This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Macromedia Flash	through 8.0	swfsr	SWF	Y	Y	Y	N	N	Y[1]	N
Microsoft 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, 2019	kpppxrdr	PPTX PPTM POTX POTM PPSX	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
			PPSM PPAM							
OASIS Open Document Format	1, 2[1]	kpodfrdr	SXD SXI ODG ODP	Y	Y	Y	Y[2]	Y	Y	N
OpenOffice Impress, LibreOffice Impress	1 to 5	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N
StarOffice Impress	3, 4, 5	kpsddrdr	SDA SDD	Y	T	N	N	N	N	N
	6, 7, 8, 9	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N

[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.

Spreadsheet Formats

Supported Spreadsheet Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Numbers	'08, '09	iwsssr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16, '18, iCloud 2018	iwss13sr 1	NUMBERS	Y	T	T	N	N	Y	N
Applix Spreadsheets	4.2, 4.3, 4.4	assr	AS	Y	Y	Y	N	N	Y	N
Comma Separated Values	n/a	csvsr	CSV	Y	Y	Y	N	N	N	N
Corel Quattro Pro	5, 6, 7, 8	qpssr	WB2 WB3	Y	Y	Y	N	P	Y	N
	X4	qpwsr	QPW	Y	N	Y	N	P	Y	N
Data Interchange Format	n/a	difsr		Y	Y	Y	N	N	N	N
Lotus 1-2-3	96, 97, R9, 9.8	l123sr	123	Y	Y	Y	N	P	Y	N
Lotus 1-2-3	2, 3, 4, 5	wkssr	WK4	Y	Y	Y	N	N	Y	N
Lotus 1-2-3 Charts	2, 3, 4, 5	kpchtrdr	123	N	Y	Y	N	N	N	N
Microsoft Excel Charts	2, 3, 4, 5, 6, 7	kpchtrdr	XLS	N	Y	Y	N	N	N	N

[1]This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Excel Macintosh	98, 2001, v.X, 2004	xlssr	XLS	Y	Y	Y	Y[1]	Y	Y	N
Microsoft Excel Windows	2.2 through 2003	xlssr	XLS XLW XLT XLA	Y	Y	Y	Y[2]	Y	Y	Y
Microsoft Excel Windows XML	2007, 2010, 2013, 2016, 2019	xlsxsr	XLSX XLTX XLSM XLTM XLAM	Y	Y	Y	Y	Y	Y	Y
Microsoft Excel Binary Format	2007, 2010, 2013, 2016	xlsbsr	XLSB	Y	Y	Y	N	N	N	N
Microsoft Works Spreadsheet	2, 3, 4	mwssr	S30 S40	Y	Y	Y	N	N	Y	N
OASIS Open Document Format	1, 2[3]	odfsssr	ODS SXC STC	Y	Y	Y	Y[4]	Y	Y	N
OpenOffice Calc, LibreOffice Calc	1 to 5	sosr	SXC ODS	Y	T	T	N	Y	Y	N

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

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

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

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

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
			OTS							
StarOffice Calc	3, 4, 5	starcsr	SDC	Y	T	T	N	N	N	N
	6, 7, 8, 9	sosr	SXC ODS	Y	T	T	N	Y	Y	N

Text and Markup Formats

Supported Text and Markup Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
ANSI	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
ASCII	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
HTML	3, 4	htmsr	HTM	Y	Y	Y	N	P	Y	N
Microsoft Excel Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N
Microsoft Word Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N
Microsoft Visio XML	2003	xmlsr	VDX VTX	Y	T	T	N	Y	Y	N
MIME HTML	n/a	mhtsr	MHT	Y	Y	Y	N	Y	Y	N
Rich Text Format	1 through 1.7	rtfsr	RTF	Y	Y	Y	N	P	Y	Y

Supported Text and Markup Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Tableau Data Source format	n/a	xmlsr	TDS	Y	T	T	N	Y	Y	N
Tableau Map Source format	n/a	xmlsr	TMS	Y	T	T	N	Y	Y	N
Tableau Preferences format	n/a	xmlsr	TPS	Y	T	T	N	Y	Y	N
Tableau Workbook format	n/a	xmlsr	TWB	Y	T	T	N	Y	Y	N
Unicode HTML	n/a	unihtmsr	HTM	Y	Y	Y	N	Y	Y	N
Unicode Text	3, 4	unisr	TXT	Y	Y	Y	N	N	Y	N
Vector Open Diagnostic Data Exchange Format	n/a	xmlsr	ODX	Y	T	T	N	Y	Y	N
XHTML	1.0	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, '18 iCloud 2018	iwwp13sr [1]	PAGES	Y	T	T	N	N	N	N
Applix Words	3.11, 4, 4.1, 4.2, 4.3, 4.4	awsr	AW	Y	Y	Y	N	N	Y	Y
Corel WordPerfect Linux	6.0, 8.1	wp6sr	WPS	Y	Y	Y	N	P	Y	N
Corel WordPerfect Macintosh	1.02, 2, 2.1, 2.2, 3, 3.1	wpmsr	WPM	Y	Y	Y	N	N	Y	N
Corel WordPerfect Windows	5, 5.1	wosr	WO	Y	Y	Y	N	P	Y	Y
Corel WordPerfect Windows	6, 7, 8, 9, 10, 11, 12, X3	wp6sr	WPD	Y	Y	Y	N	P	Y	Y
DisplayWrite	4	dw4sr	IP	Y	Y	Y	N	N	Y	N
Folio Flat File	3.1	foliosr	FFF	Y	Y	Y	N	Y	Y	Y
Founder Chinese E-paper Basic	3.2.1	cebsr[2]	CEB	Y	N	N	N	N	N	N

[1]This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.
[2]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
Fujitsu Oasys	7	oa2sr	OA2	Y	Y	Y	N	P	N	N
Haansoft Hangul	97	hwpsr	HWP	Y	Y	Y	N	Y	Y	N
	2002, 2005, 2007, 2010	hwposr	HWP	Y	Y	Y	Y	Y	Y	N
Health level7	2.0	hl7sr	HL7	Y	Y	Y	N	Y	Y	N
IBM DCA/RFT (Revisable Form Text)	SC23-0758-1	dcasr	DC	Y	Y	Y	N	N	Y	N
JustSystems Ichitaro	8 to 2013, 2018	jtdsr	JTD	Y	Y	Y	N	P	N	Y
Lotus AMI Pro	2, 3	lasr	SAM	Y	Y	Y	N	P	Y	Y
Lotus AMI Professional Write Plus	2.1	lasr	AMI	Y	Y	Y	N	N	N	Y
Lotus Word Pro	96, 97, R9	lwpsr	LWP	Y	Y	Y	N	P	N	Y
Lotus SmartMaster	96, 97	lwpsr	MWP	Y	Y	Y	N	N	N	N
Microsoft 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

[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 PC	4, 5, 5.5, 6	mwsr	DOC	Y	Y	Y	N	N	N	Y
Microsoft Word Windows	1.0, 2.0	misr	DOC	Y	Y	Y	N	N	N	Y
Microsoft Word Windows	6, 7, 8, 95	mw6sr	DOC	Y	Y	Y	N	Y	Y	Y
Microsoft Word Windows	97, 2000, 2002, 2003	mw8sr	DOC DOT	Y	Y	Y	Y[1]	Y	Y	Y
Microsoft Word Windows XML	2007, 2010, 2013, 2016, 2019	mwxsr	DOCM DOCX DOTX DOTM	Y	Y	Y	Y	Y	Y	Y
Microsoft Word Windows Flat XML	2007, 2010, 2013, 2016	mwxsr	XML	Y	Y	Y	Y	Y	Y	Y
Microsoft Works	1, 2, 3, 4	mswsr	WPS	Y	Y	Y	N	N	N	Y
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

[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
OASIS Open Document Format	1, 2[1]	odfwpsr	ODT SXW STW	Y	Y	Y	Y[2]	Y	Y	Y
Omni Outliner	v3, OPML, OOutline	oo3sr	OO3 OPML OOUTLINE	Y	Y	Y	N	N	Y	N
OpenOffice Writer, LibreOffice Writer	1 to 5	sosr	SXW ODT	Y	T	T	N	Y	Y	N
Open Publication Structure eBook	2.0, 3.0	epubsr	EPUB	Y	Y	Y	N	Y	Y	N
StarOffice Writer	3, 4, 5	starwsr	SDW	Y	T	T	N	N	N	N
	6, 7, 8, 9	sosr	SXW ODT	Y	T	T	N	Y	Y	N
Skype Log	3	skypesr	DBB	Y	Y	Y	N	N	N	N
WordPad	through 2003	rtfsr	RTF	Y	Y	Y	N	P	Y	N
XML Paper Specification	n/a	xpssr	XPS	Y	T	T	N	N	N	N
XyWrite	4.12	xywsr	XY4	Y	Y	Y	N	N	N	N

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

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

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
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.

Appendix B: Detected Formats

This section lists the file formats that KeyView can detect.

Key to Detected Formats Table

The detected formats table includes the following information:

Column	Description
Format Name	The format name that is returned by KeyView format detection. • In the C API, these values are defined in the `ENdocFmt` enumeration in `adDocFmt.h`. • In the .NET API these values are defined in the `Autonomy.API.Export.DocFormat` enumeration. • In the Java API these values are defined in the `com.verity.api.DocFormat` enumeration.
Number	The format number that is returned by KeyView format detection. This is the value associated with the Format Name in the relevant enumeration.
Category	This value is used in the KeyView configuration file `formats.ini` to specify the reader to use to filter, export, or view the format. Several formats might have the same category value.
Description	A short description of the file format.
MIME Type	The MIME type (if any).
Extension	A list of common file extensions for the file format. **NOTE:** This is not a complete list of file extensions. KeyView does not distinguish between file types based on their extension. Instead, it detects the file format based on the file content. This is more reliable because content cannot always be predicted from the file extension, and because some file extensions are associated with multiple formats.
File Class	The KeyView file class. • In the C API, these values are defined in the `ENdocClass` enumeration in `adinfo.h`. • In the .NET API these values are defined in the

	`Autonomy.API.Export.DocClass` enumeration. • In the Java API these values are defined in the `com.verity.api.DocClass` enumeration.

Detected Formats

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Reserved__Fmt	-1	-1				AutoDetNoFormat
Unknown_Fmt	0	0				AutoDetNoFormat
AES_Multiplus_Comm_Fmt	1	1	Multiplus (AES)		PTF	adWORDPROCESSOR
ASCII_Text_Fmt	2	2	Plain Text file	text/plain	TXT	adWORDPROCESSOR
MSDOS_Batch_File_Fmt	3	2	MS-DOS Batch File	application/x-bat	BAT	adEXECUTABLE
Applix_Alis_Fmt	4	3	APPLIX ASTERIX		AX	adWORDPROCESSOR
BMP_Fmt	5	4	Windows Bitmap Image (BMP)	image/bmp	BMP	adRASTERIMAGE
CT_DEF_Fmt	6	5	Convergent Technologies DEF Comm. Format			adWORDPROCESSOR
Corel_Draw_Fmt	7	6	Corel Draw (up to version 13/X3)	application/coreldraw	CDR	adVECTORGRAPHIC
CGM_ClearText_Fmt	8	8	Computer Graphics Metafile (CGM)		CGM	adVECTORGRAPHIC
CGM_Binary_Fmt	9	8	Computer Graphics Metafile (CGM)	image/cgm	CGM	adVECTORGRAPHIC
CGM_Character_Fmt	10	8	Computer Graphics Metafile (CGM)		CGM	adVECTORGRAPHIC
Word_Connection_Fmt	11	9	Word Connection		CN	adWORDPROCESSOR
COMET_TOP_Word_Fmt	12	10	Nixdorf COMET TOP Financial Accounting software			adWORDPROCESSOR
CEOwrite_Fmt	13	11	CEOwrite		CW	adWORDPROCESSOR
DSA101_Fmt	14	12	DSA101 (Honeywell Bull)			adWORDPROCESSOR
DCA_RFT_Fmt	15	13	DCA-RFT (IBM Revisable Form)	application/dca-rft	RFT, DC	adWORDPROCESSOR
CDA_DDIF_Fmt	16	14	CDA / DDIF		DDIF	adWORDPROCESSOR
DG_CDS_Fmt	17	16	DG Common Data Stream (CDS)		CDS	adWORDPROCESSOR
Micrografx_Draw_Fmt	18	18	Windows Draw (Micrografx)		DRW	adVECTORGRAPHIC
Data_Point_VistaWord_Fmt	19	19	Vistaword		DV	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
DECdx_Fmt	20	20	DECdx		DX	adWORDPROCESSOR
Enable_WP_Fmt	21	21	Enable Word Processing		WPF	adWORDPROCESSOR
EPSF_Fmt	22	22	Encapsulated PostScript	application/postscript	EPS	AutoDetNoFormat
Preview_EPSF_Fmt	23	22	Encapsulated PostScript	application/postscript		AutoDetNoFormat
MS_Executable_Fmt	24	23	MSDOS/Windows Program	application/x-msdownload	EXE	adEXECUTABLE
G31D_Fmt	25	24	CCITT G3 1D			adRASTERIMAGE
GIF_87a_Fmt	26	25	Graphics Interchange Format (GIF87a)	image/gif	GIF	adRASTERIMAGE
GIF_89a_Fmt	27	25	Graphics Interchange Format (GIF89a)	image/gif	GIF	adRASTERIMAGE
HP_Word_PC_Fmt	28	26	HP Word PC		HW	adWORDPROCESSOR
IBM_1403_LinePrinter_Fmt	29	27	IBM 1403 Line Printer		I4	adWORDPROCESSOR
IBM_DCF_Script_Fmt	30	28	DCF Script		IC	adWORDPROCESSOR
IBM_DCA_FFT_Fmt	31	29	DCA-FFT (IBM Final Form)		IF, FFT	adWORDPROCESSOR
Interleaf_Fmt	32	30	Interleaf			adWORDPROCESSOR
GEM_Image_Fmt	33	31	GEM Bit Image		IMG	adRASTERIMAGE
IBM_Display_Write_Fmt	34	32	Display Write		IP	adWORDPROCESSOR
Sun_Raster_Fmt	35	33	Sun Raster	image/x-cmu-raster	RAS	adRASTERIMAGE
Ami_Pro_Fmt	36	35	Lotus Ami Pro	application/x-lotus-amipro	SAM	adWORDPROCESSOR
Ami_Pro_StyleSheet_Fmt	37	35	Lotus Ami Pro Style Sheet			adWORDPROCESSOR
MORE_Fmt	38	36	MORE Database MAC			adOUTLINE
Lyrix_Fmt	39	37	Lyrix Word Processing			adWORDPROCESSOR
MASS_11_Fmt	40	38	MASS-11		M1	adWORDPROCESSOR
MacPaint_Fmt	41	39	MacPaint		PNTG	adRASTERIMAGE
MS_Word_Mac_Fmt	42	40	Microsoft Word for Macintosh (up to version 3)	application/msword	DOC	adWORDPROCESSOR
SmartWare_II_Comm_Fmt	43	41	SmartWare II			adCOMMUNICATION
MS_Word_Win_Fmt	44	42	Microsoft Word for Windows (up to version 6)	application/msword	DOC, WPS	adWORDPROCESSOR
Multimate_Fmt	45	43	MultiMate		MM	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Multimate_Fnote_Fmt	46	43	MultiMate Footnote File			adWORDPROCESSOR
Multimate_Adv_Fmt	47	43	MultiMate Advantage			adWORDPROCESSOR
Multimate_Adv_Fnote_Fmt	48	43	MultiMate Advantage Footnote File			adWORDPROCESSOR
Multimate_Adv_II_Fmt	49	43	MultiMate Advantage II			adWORDPROCESSOR
Multimate_Adv_II_Fnote_Fmt	50	43	MultiMate Advantage II Footnote File		FBX, FNX	adWORDPROCESSOR
Multiplan_PC_Fmt	51	44	Multiplan (PC)			adSPREADSHEET
Multiplan_Mac_Fmt	52	44	Multiplan (Mac)			adSPREADSHEET
MS_RTF_Fmt	53	45	Rich Text Format (RTF)	application/rtf	RTF	adWORDPROCESSOR
MS_Word_PC_Fmt	54	46	Microsoft Word for PC (up to version 6)	application/x-ms-wordpc	MW	adWORDPROCESSOR
MS_Word_PC_StyleSheet_Fmt	55	46	Microsoft Word for PC (up to version 6) Style Sheet			adWORDPROCESSOR
MS_Word_PC_Glossary_Fmt	56	46	Microsoft Word for PC (up to version 6) Glossary			adWORDPROCESSOR
MS_Word_PC_Driver_Fmt	57	46	Microsoft Word for PC (up to version 6) Driver			adWORDPROCESSOR
MS_Word_PC_Misc_Fmt	58	46	Microsoft Word for PC (up to version 6) Miscellaneous File			adWORDPROCESSOR
NBI_Async_Archive_Fmt	59	47	NBI Async Archive Format			adWORDPROCESSOR
Navy_DIF_Fmt	60	48	Navy DIF (document interchange format)		ND	adWORDPROCESSOR
NBI_Net_Archive_Fmt	61	49	NBI Net Archive Format		NN	adWORDPROCESSOR
NIOS_TOP_Fmt	62	50	NIOS TOP			adWORDPROCESSOR
FileMaker_Mac_Fmt	63	51	Filemaker MAC		FP5, FP7	adDATABASE
ODA_Q1_11_Fmt	64	52	ODA / ODIF Q1 11		OD	adWORDPROCESSOR
ODA_Q1_12_Fmt	65	52	ODA / ODIF Q1 12		OD	adWORDPROCESSOR
OLIDIF_Fmt	66	53	OLIDIF (Olivetti)			adWORDPROCESSOR
Office_Writer_Fmt	67	55	Office Writer		OW	adWORDPROCESSOR
PC_Paintbrush_Fmt	68	56	PC Paintbrush Graphics (PCX)	image/vnd.zbrush.pcx	PCX	adRASTERIMAGE
CPT_Comm_Fmt	69	57	CPT Corporation word processor		PF	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Lotus_PIC_Fmt	70	58	Lotus PIC	image/x-pict	PIC	adVECTORGRAPHIC
Mac_PICT_Fmt	71	59	QuickDraw Picture	image/x-pict	PCT	AutoDetNoFormat
Philips_Script_Word_Fmt	72	60	Philips Script			adWORDPROCESSOR
PostScript_Fmt	73	61	PostScript	application/postscript	PS	adVECTORGRAPHIC
PRIMEWORD_Fmt	74	62	PRIMEWORD			adWORDPROCESSOR
Quadratron_Q_One_v1_Fmt	75	63	Q-One V1.93J		Q1, QX	adWORDPROCESSOR
Quadratron_Q_One_v2_Fmt	76	64	Q-One V2.0		Q1, QX	adWORDPROCESSOR
SAMNA_Word_IV_Fmt	77	65	SAMNA Word		SAM	adWORDPROCESSOR
Ami_Pro_Draw_Fmt	78	66	Lotus Ami Pro Draw		SDW	adVECTORGRAPHIC
SYLK_Spreadsheet_Fmt	79	67	SYmbolic LinK (SYLK) format		SLK	adSPREADSHEET
SmartWare_II_WP_Fmt	80	68	Informix SmartWare II word processor		DOC	adWORDPROCESSOR
Symphony_Fmt	81	69	Lotus Symphony spreadsheet		WR1	adSPREADSHEET
Targa_Fmt	82	70	Targa image	image/x-tga	TGA	adRASTERIMAGE
TIFF_Fmt	83	71	Tag Image File Format (TIFF)	image/tiff	TIF, TIFF	AutoDetNoFormat
Targon_Word_Fmt	84	72	Targon Word		TW	adWORDPROCESSOR
Uniplex_Ucalc_Fmt	85	73	Uniplex Ucalc		SS	adSPREADSHEET
Uniplex_WP_Fmt	86	74	Uniplex word processor		UP	adWORDPROCESSOR
MS_Word_UNIX_Fmt	87	75	Microsoft Word UNIX	application/msword		adWORDPROCESSOR
WANG_PC_Fmt	88	76	WANG PC			adWORDPROCESSOR
WordERA_Fmt	89	77	WordERA			adWORDPROCESSOR
WANG_WPS_Comm_Fmt	90	78	WANG WPS		WF	adWORDPROCESSOR
WordPerfect_Mac_Fmt	91	79	WordPerfect MAC	application/x-corel-wordperfect		adWORDPROCESSOR
WordPerfect_Fmt	92	86	WordPerfect version 4	application/x-corel-wordperfect	WP, WP4	adWORDPROCESSOR
WordPerfect_VAX_Fmt	93	139	WordPerfect VAX	application/x-corel-wordperfect		adWORDPROCESSOR
WordPerfect_Macro_Fmt	94	139	WordPerfect Macro	application/vnd.wordperfect	MRS	adWORDPROCESSOR
WordPerfect_Dictionary_Fmt	95	139	WordPerfect Spelling Dictionary	application/vnd.wordperfect	SPW	adWORDPROCESSOR
WordPerfect_Thesaurus_Fmt	96	139	WordPerfect Thesaurus	application/vnd.wordperfect		adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
WordPerfect_Resource_Fmt	97	139	WordPerfect Resource File	application/vnd.wordperfect	WWK, PRS	adWORDPROCESSOR
WordPerfect_Driver_Fmt	98	139	WordPerfect Driver	application/vnd.wordperfect	IRS, VRS	adWORDPROCESSOR
WordPerfect_Cfg_Fmt	99	139	WordPerfect Configuration File	application/vnd.wordperfect	PFX	adWORDPROCESSOR
WordPerfect_Hyphenation_Fmt	100	139	WordPerfect Hyphenation Dictionary	application/vnd.wordperfect	HYC	adWORDPROCESSOR
WordPerfect_Misc_Fmt	101	139	WordPerfect Miscellaneous File	application/vnd.wordperfect		adWORDPROCESSOR
WordMARC_Fmt	102	82	WordMARC Composer	video/x-ms-wm	WM, PW	adWORDPROCESSOR
Windows_Metafile_Fmt	103	83	Windows Metafile	image/wmf	WMF	adVECTORGRAPHIC
Windows_Metafile_NoHdr_Fmt	104	83	Windows Metafile (no header)	image/wmf	WMF	adVECTORGRAPHIC
SmartWare_II_DB_Fmt	105	84	Informix SmartWare II database			adDATABASE
WordPerfect_Graphics_Fmt	106	195	WordPerfect Graphics (version 2 and higher)	application/vnd.wordperfect	WPG, QPG	AutoDetNoFormat
WordStar_Fmt	107	87	WordStar		WS, WSD	adWORDPROCESSOR
WANG_WITA_Fmt	108	88	WANG WITA		WT	adWORDPROCESSOR
Xerox_860_Comm_Fmt	109	89	Xerox 860			adWORDPROCESSOR
Xerox_Writer_Fmt	110	91	Xerox Writer			adWORDPROCESSOR
DIF_SpreadSheet_Fmt	111	92	Data Interchange Format (DIF)	application/dif+xml	DIF	adSPREADSHEET
Enable_Spreadsheet_Fmt	112	93	Enable Spreadsheet	application/vnd.epson.ssf	SSF	adSPREADSHEET
SuperCalc_Fmt	113	94	Sorcim SuperCalc spreadsheet		CAL	adSPREADSHEET
UltraCalc_Fmt	114	95	UltraCalc spreadsheet			adSPREADSHEET
SmartWare_II_SS_Fmt	115	96	Informix SmartWare II spreadsheet			adSPREADSHEET
SOF_Encapsulation_Fmt	116	97	Serialized Object Format (SOF)	application/java-serialized-object	SOF	adENCAPSULATION
PowerPoint_Win_Fmt	117	98	Microsoft PowerPoint PC (up to version 4)	application/x-ms-powerpoint	PPT	adPRESENTATION
PowerPoint_Mac_Fmt	118	99	Microsoft PowerPoint MAC (up to version 4)	application/x-ms-powerpoint	PPT	adPRESENTATION
PowerPoint_95_Fmt	119	212	Microsoft PowerPoint 95	application/x-ms-powerpoint	PPT	adPRESENTATION
PowerPoint_97_Fmt	120	272	Microsoft PowerPoint 97	application/x-ms-powerpoint	PPT	adPRESENTATION
PageMaker_Mac_Fmt	121	100	PageMaker for Macintosh			adDESKTOPPUBLSH
PageMaker_Win_Fmt	122	101	PageMaker for Windows			adDESKTOPPUBLSH

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Works_Mac_WP_Fmt	123	103	Microsoft Works Word Processor for MAC	application/x-msworks	MWK	adWORDPROCESSOR
MS_Works_Mac_DB_Fmt	124	104	Microsoft Works Database for MAC	application/x-msworks		adDATABASE
MS_Works_Mac_SS_Fmt	125	105	Microsoft Works Spreadsheet for MAC	application/x-msworks		adSPREADSHEET
MS_Works_Mac_Comm_Fmt	126	106	Microsoft Works Communication for MAC	application/x-msworks		adCOMMUNICATION
MS_Works_DOS_WP_Fmt	127	107	Microsoft Works Word Processor for DOS	application/x-msworks	WPS	adWORDPROCESSOR
MS_Works_DOS_DB_Fmt	128	108	Microsoft Works Database for DOS	application/x-msworks	WDB	adDATABASE
MS_Works_DOS_SS_Fmt	129	109	Microsoft Works Spreadsheet for DOS	application/x-msworks		adSPREADSHEET
MS_Works_Win_WP_Fmt	130	227	Microsoft Works Word Processor for Windows	application/x-msworks	WPS, W40	adWORDPROCESSOR
MS_Works_Win_DB_Fmt	131	231	Microsoft Works Database for Windows	application/x-msworks		adDATABASE
MS_Works_Win_SS_Fmt	132	228	Microsoft Works Spreadsheet for Windows	application/x-msworks	S30, S40	adSPREADSHEET
PC_Library_Fmt	133	111	DOS/Windows Object Library	application/x-archive	LIB, A	adLIBRARY
MacWrite_Fmt	134	112	MacWrite	application/macwriteii		adWORDPROCESSOR
MacWrite_II_Fmt	135	113	MacWrite II	application/macwriteii		adWORDPROCESSOR
Freehand_Fmt	136	114	Freehand MAC	image/x-freehand		adVECTORGRAPHIC
Disk_Doubler_Fmt	137	115	Disk Doubler			adENCAPSULATION
HP_GL_Fmt	138	116	HP Graphics Language	vector/x-hpgl	HPGL	adVECTORGRAPHIC
FrameMaker_Fmt	139	136	FrameMaker	application/vnd.framemaker	FM, FRM	adDESKTOPPUBLSH
FrameMaker_Book_Fmt	140	136	FrameMaker Book	application/vnd.framemaker	BOOK	adDESKTOPPUBLSH
Maker_Markup_Language_Fmt	141	174	Maker Markup Language	application/vnd.mif		adDESKTOPPUBLSH
Maker_Interchange_Fmt	142	117	Maker Interchange Format (MIF)	application/x-mif	MIF	adWORDPROCESSOR
JPEG_File_Interchange_Fmt	143	118	JPEG Interchange Format	image/jpeg	JPG, JPEG	adRASTERIMAGE
Reflex_Fmt	144	119	Borland Reflex database			adDATABASE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Framework_Fmt	145	276	Framework office suite			adMIXED
Framework_II_Fmt	146	120	Framework II office suite		FW3	adMIXED
Paradox_Fmt	147	121	Borland Paradox database		DB	adDATABASE
MS_Windows_Write_Fmt	148	123	Microsoft Windows Write	application/x-ms-write	WRI	adWORDPROCESSOR
Quattro_Pro_DOS_Fmt	149	124	Quattro Pro for DOS	application/x-quattropro	WQ1	adSPREADSHEET
Quattro_Pro_Win_Fmt	150	184	Quattro Pro for Windows	application/x-quattro-win	WB1, WB2, WB3	adSPREADSHEET
Persuasion_Fmt	151	126	Adobe Persuasion			adPRESENTATION
Windows_Icon_Fmt	152	128	Windows Icon Format	image/ico	ICO	adRASTERIMAGE
Windows_Cursor_Fmt	153	133	Windows Cursor	image/x-win-bitmap	CUR	adRASTERIMAGE
MS_Project_Activity_Fmt	154	129	Microsoft Project (up to version 3) activity file			adSCHEDULE
MS_Project_Resource_Fmt	155	129	Microsoft Project (up to version 3) resource file			adSCHEDULE
MS_Project_Calc_Fmt	156	129	Microsoft Project (up to version 3) calc file			adSCHEDULE
PKZIP_Fmt	157	132	ZIP Archive	application/zip	ZIP, ZIPX	AutoDetNoFormat
Quark_Xpress_Fmt	158	134	Quark Xpress MAC			adDESKTOPPUBLSH
ARC_PAK_Archive_Fmt	159	135	PAK/ARC Archive		ARC, PAK	adENCAPSULATION
MS_Publisher_Fmt	160	137	Microsoft Publisher (up to version 3)	application/x-mspublisher	PUB	adDESKTOPPUBLSH
PlanPerfect_Fmt	161	138	PlanPerfect			adSCHEDULE
WordPerfect_Auxiliary_Fmt	162	139	WordPerfect auxiliary file		WPW	adMISC
MS_WAVE_Audio_Fmt	163	141	Microsoft Wave	audio/wav	WAV	adSOUND
MIDI_Audio_Fmt	164	142	MIDI audio	audio/mid	MID, MIDI	adSOUND
AutoCAD_DXF_Binary_Fmt	165	143	AutoCAD DXF	image/x-dxf	DXF	adVECTORGRAPHIC
AutoCAD_DXF_Text_Fmt	166	143	AutoCAD DXF	image/x-dxf	DXF	adVECTORGRAPHIC
dBase_Fmt	167	144	dBase	application/x-dbf	DBF, VCX	adDATABASE
OS_2_PM_Metafile_Fmt	168	145	OS/2 PM Metafile		MET	adVECTORGRAPHIC
Lasergraphics_Language_Fmt	169	146	Lasergraphics Language			adVECTORGRAPHIC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
AutoShade_Rendering_Fmt	170	147	AutoShade Rendering			adVECTORGRAPHIC
GEM_VDI_Fmt	171	148	GEM VDI Metafile image		GEM, GDI	adVECTORGRAPHIC
Windows_Help_Fmt	172	149	Windows Help File	application/winhlp	HLP	adMISC
Volkswriter_Fmt	173	150	Volkswriter word processor		VW4	adWORDPROCESSOR
Ability_WP_Fmt	174	151	Ability Word Processor			adWORDPROCESSOR
Ability_DB_Fmt	175	151	Ability Database			adDATABASE
Ability_SS_Fmt	176	151	Ability Spreadsheet			adSPREADSHEET
Ability_Comm_Fmt	177	151	Ability Presentation			adCOMMUNICATION
Ability_Image_Fmt	178	151	Ability Image			adRASTERIMAGE
XyWrite_Fmt	179	152	XYWrite / Nota Bene		XY4	adWORDPROCESSOR
CSV_Fmt	180	153	CSV (Comma Separated Values)	text/csv	CSV	adSPREADSHEET
IBM_Writing_Assistant_Fmt	181	154	IBM Writing Assistant		IWA	adWORDPROCESSOR
WordStar_2000_Fmt	182	155	WordStar 2000		WS2	adWORDPROCESSOR
HP_PCL_Fmt	183	157	HP Printer Control Language	application/pcl	PCL	adVECTORGRAPHIC
UNIX_Exe_PreSysV_VAX_Fmt	184	158	Unix Executable (PDP-11/pre-System V VAX)	application/octet-stream		adEXECUTABLE
UNIX_Exe_Basic_16_Fmt	185	158	Unix Executable (Basic-16)	application/octet-stream		adEXECUTABLE
UNIX_Exe_x86_Fmt	186	158	Unix Executable (x86)	application/octet-stream		adEXECUTABLE
UNIX_Exe_iAPX_286_Fmt	187	158	Unix Executable (iAPX 286)	application/octet-stream		adEXECUTABLE
UNIX_Exe_MC68k_Fmt	188	158	Unix Executable (MC680x0)	application/octet-stream		adEXECUTABLE
UNIX_Exe_3B20_Fmt	189	158	Unix Executable (3B20)	application/octet-stream		adEXECUTABLE
UNIX_Exe_WE32000_Fmt	190	158	Unix Executable (WE32000)	application/octet-stream		adEXECUTABLE
UNIX_Exe_VAX_Fmt	191	158	Unix Executable (VAX)	application/octet-stream		adEXECUTABLE
UNIX_Exe_Bell_5_Fmt	192	158	Unix Executable (Bell 5.0)	application/octet-stream		adEXECUTABLE
UNIX_Obj_VAX_Demand_Fmt	193	159	Unix Object Module (VAX Demand)			adOBJECTMODULE
UNIX_Obj_MS8086_Fmt	194	159	Unix Object Module (old MS 8086)			adOBJECTMODULE
UNIX_Obj_Z8000_Fmt	195	159	Unix Object Module (Z8000)			adOBJECTMODULE
AU_Audio_Fmt	196	161	NeXT/Sun Audio Data	audio/basic	AU	adSOUND

Format Name	Number	Category	Description	MIME Type	Extension	File Class
NeWS_Font_Fmt	197	162	NeWS bitmap font			adFONT
cpio_Archive_CRChdr_Fmt	198	163	cpio archive (CRC Header)	application/x-cpio		adENCAPSULATION
cpio_Archive_CHRhdr_Fmt	199	163	cpio archive (CHR Header)	application/x-cpio		adENCAPSULATION
PEX_Binary_Archive_Fmt	200	164	SUN PEX Binary Archive			adENCAPSULATION
Sun_vfont_Fmt	201	165	SUN vfont Definition			adFONT
Curses_Screen_Fmt	202	166	Curses Screen Image			adRASTERIMAGE
UUEncoded_Fmt	203	167	UU encoded	text/x-uuencode	UUE	adENCAPSULATION
WriteNow_Fmt	204	168	WriteNow MAC			adWORDPROCESSOR
PC_Obj_Fmt	205	169	DOS/Windows Object Module	application/octet-stream	OBJ	adOBJECTMODULE
Windows_Group_Fmt	206	170	Windows Group			adMISC
TrueType_Font_Fmt	207	171	TrueType Font	application/x-font-ttf	TTF	adFONT
Windows_PIF_Fmt	208	172	Program Information File (PIF)	application/octet-stream	PIF	adMISC
MS_COM_Executable_Fmt	209	173	PC (.COM)	application/octet-stream	COM	adEXECUTABLE
StuffIt_Fmt	210	175	StuffIt (MAC)	application/x-stuffit	HQX	adENCAPSULATION
PeachCalc_Fmt	211	176	PeachCalc		CAL	adSPREADSHEET
Wang_GDL_Fmt	212	177	WANG Office GDL Header			adENCAPSULATION
Q_A_DOS_Fmt	213	179	Q & A for DOS			adWORDPROCESSOR
Q_A_Win_Fmt	214	180	Q & A for Windows		JW	adWORDPROCESSOR
WPS_PLUS_Fmt	215	181	WPS-PLUS	application/vnd.ms-wpl	WPL	adWORDPROCESSOR
DCX_Fmt	216	182	DCX FAX Format(PCX images)	image/dcx	DCX	adFAXFORMAT
OLE_Fmt	217	183	OLE Compound Document		OLE	adENCAPSULATION
EBCDIC_Fmt	218	186	EBCDIC Text			adWORDPROCESSOR
DCS_Fmt	219	187	DCS			adWORDPROCESSOR
UNIX_SHAR_Fmt	220	190	SHAR shell archive format	application/x-shar	SHAR	adENCAPSULATION
Lotus_Notes_BitMap_Fmt	221	191	Lotus Notes Bitmap			adRASTERIMAGE
Lotus_Notes_CDF_Fmt	222	193	Lotus Notes CDF	application/cdf	CDF	adWORDPROCESSOR
Compress_Fmt	223	192	Unix Compress	application/x-compress	Z	adENCAPSULATION
GZ_Compress_Fmt	224	198	GZ Compress	application/gzip	GZ	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
TAR_Fmt	225	194	TAR archive	application/tar	TAR	adENCAPSULATION
ODIF_FOD26_Fmt	226	196	Open Document Architecture (ODA / ODIF) FOD26	application/oda	F26	adWORDPROCESSOR
ODIF_FOD36_Fmt	227	196	Open Document Architecture (ODA / ODIF) FOD36	application/oda	F36	adWORDPROCESSOR
ALIS_Fmt	228	197	ALIS			adWORDPROCESSOR
Envoy_Fmt	229	199	WordPerfect Envoy	application/envoy	EVY	adWORDPROCESSOR
PDF_Fmt	230	200	Portable Document Format	application/pdf	PDF	adWORDPROCESSOR
BinHex_Fmt	231	206	BinHex	application/mac-binhex40	HQX	adENCAPSULATION
SMTP_Fmt	232	207	SMTP	message/rfc822	SMTP	adENCAPSULATION
MIME_Fmt	233	208	MIME (EML, MBX email)[1]	message/rfc822	EML, MBX	adENCAPSULATION
USENET_Fmt	234	264	USENET	message/news		adWORDPROCESSOR
SGML_Fmt	235	209	SGML	text/sgml	SGML	adWORDPROCESSOR
HTML_Fmt	236	210	HTML	text/html	HTM, HTML	adWORDPROCESSOR
ACT_Fmt	237	211	ACT! CRM software		ACT	adWORDPROCESSOR
PNG_Fmt	238	213	Portable Network Graphics (PNG)	image/png	PNG	adRASTERIMAGE
MS_Video_Fmt	239	214	Video for Windows (AVI)	video/avi	AVI	adMOVIE
Windows_Animated_Cursor_Fmt	240	215	Windows Animated Cursor		ANI	adRASTERIMAGE
Windows_CPP_Obj_Storage_Fmt	241	216	Windows C++ Object Storage			adMIXED
Windows_Palette_Fmt	242	217	Windows Palette		PAL	adRASTERIMAGE
RIFF_DIB_Fmt	243	218	RIFF Device Independent Bitmap			adRASTERIMAGE
RIFF_MIDI_Fmt	244	219	RIFF MIDI	audio/midi	RMI	adSOUND
RIFF_Multimedia_Movie_Fmt	245	220	RIFF Multimedia Movie			adMOVIE
MPEG_Fmt	246	221	MPEG Movie	video/mpeg		adMOVIE
QuickTime_Fmt	247	222	QuickTime Movie, MPEG-4 audio	video/quicktime	MOV, QT, MP4	adMOVIE
AIFF_Fmt	248	223	Audio Interchange File Format (AIFF)	audio/aiff	AIF, AIFF	adSOUND
Amiga_MOD_Fmt	249	224	Amiga MOD		MOD	adSOUND
Amiga_IFF_8SVX_Fmt	250	225	Amiga IFF (8SVX) Sound	audio/x-8svx	IFF	adSOUND

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Creative_Voice_Audio_Fmt	251	226	Creative Voice (VOC)		VOC	adSOUND
AutoDesk_Animator_FLI_Fmt	252	229	AutoDesk Animator FLIC	video/x-fli	FLI	adANIMATION
AutoDesk_AnimatorPro_FLC_Fmt	253	230	AutoDesk Animator Pro FLIC	video/x-flc	FLC	adANIMATION
Compactor_Archive_Fmt	254	233	Compactor / Compact Pro	application/mac-compactpro		adENCAPSULATION
VRML_Fmt	255	234	VRML	model/vrml	WRL	adVECTORGRAPHIC
QuickDraw_3D_Metafile_Fmt	256	235	QuickDraw 3D Metafile			adVECTORGRAPHIC
PGP_Secret_Keyring_Fmt	257	236	PGP Secret Keyring	application/pgp		adENCAPSULATION
PGP_Public_Keyring_Fmt	258	237	PGP Public Keyring	application/pgp		adENCAPSULATION
PGP_Encrypted_Data_Fmt	259	238	PGP Encrypted Data	application/pgp		adENCAPSULATION
PGP_Signed_Data_Fmt	260	239	PGP Signed Data	application/pgp		adENCAPSULATION
PGP_SignedEncrypted_Data_Fmt	261	240	PGP Signed and Encrypted Data	application/pgp		adENCAPSULATION
PGP_Sign_Certificate_Fmt	262	241	PGP Signature Certificate	application/pgp-signature	SIG	adENCAPSULATION
PGP_Compressed_Data_Fmt	263	246	PGP Compressed Data	application/pgp		adENCAPSULATION
PGP_ASCII_Public_Keyring_Fmt	264	242	ASCII-armored PGP Public Keyring	application/pgp	PGP	adENCAPSULATION
PGP_ASCII_Encoded_Fmt	265	243	ASCII-armored PGP encoded	application/pgp		adENCAPSULATION
PGP_ASCII_Signed_Fmt	266	244	ASCII-armored PGP signed	application/pgp		adENCAPSULATION
OLE_DIB_Fmt	267	245	OLE DIB object			adRASTERIMAGE
SGI_Image_Fmt	268	247	SGI Image	image/sgi	RGB	adRASTERIMAGE
Lotus_ScreenCam_Fmt	269	248	Lotus ScreenCam	application/vnd.lotus-screencam	SCM	adANIMATION
MPEG_Audio_Fmt	270	249	MPEG Audio	audio/mpeg	MPEGA, MPG, MP3	adSOUND
FTP_Software_Session_Fmt	271	250	FTP Session Data		STE	adCOMMUNICATION
Netscape_Bookmark_File_Fmt	272	210	Netscape Bookmark File	text/html		adWORDPROCESSOR
Corel_Draw_CMX_Fmt	273	252	Corel CMX	application/cmx	CMX	adVECTORGRAPHIC
AutoDesk_DWG_Fmt	274	253	AutoDesk Drawing (DWG)	image/x-dwg	DWG	adVECTORGRAPHIC
AutoDesk_WHIP_Fmt	275	254	AutoDesk WHIP		WHP	adVECTORGRAPHIC
Macromedia_Director_Fmt	276	255	Macromedia Director	application/x-director	DCR	adANIMATION
Real_Audio_Fmt	277	256	Real Audio	audio/x-pn-realaudio	RM, RA	adSOUND

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MSDOS_Device_Driver_Fmt	278	257	MSDOS Device Driver	application/octet-stream	SYS	adEXECUTABLE
Micrografx_Designer_Fmt	279	258	Micrografx Designer		DSF	adVECTORGRAPHIC
SVF_Fmt	280	259	Simple Vector Format (SVF)	image/x-svf	SVF	adVECTORGRAPHIC
Applix_Words_Fmt	281	261	Applix Words	application/x-applix-word	AW	adWORDPROCESSOR
Applix_Graphics_Fmt	282	262	Applix Graphics		AG	adPRESENTATION
MS_Access_Fmt	283	263	Microsoft Access (versions 1 and 2)	application/x-msaccess	MDB	adDATABASE
MS_Access_95_Fmt	284	263	Microsoft Access 95	application/msaccess	MDB	adDATABASE
MS_Access_97_Fmt	285	263	Microsoft Access 97	application/msaccess	MDB	adDATABASE
MacBinary_Fmt	286	265	MacBinary	application/x-macbinary	BIN	adENCAPSULATION
Apple_Single_Fmt	287	266	Apple Single			adENCAPSULATION
Apple_Double_Fmt	288	267	Apple Double	multipart/appledouble	AD	adENCAPSULATION
Enhanced_Metafile_Fmt	289	270	Enhanced Metafile	image/x-emf	EMF	adVECTORGRAPHIC
MS_Office_Drawing_Fmt	290	271	Microsoft Office Drawing			adVECTORGRAPHIC
XML_Fmt	291	285	XML	text/xml	XML	adWORDPROCESSOR
DeVice_Independent_Fmt	292	274	DeVice Independent file (DVI)	application/x-dvi	DVI	adVECTORGRAPHIC
Unicode_Fmt	293	275	Unicode text file	text/plain	UNI	adWORDPROCESSOR
Lotus_123_Worksheet_Fmt	294	81	Lotus 1-2-3	application/x-lotus-123	WKS, WK1, WK3, WK4	adSPREADSHEET
Lotus_123_Format_Fmt	295	81	Lotus 1-2-3 Formatting	application/x-123	FM3	adSPREADSHEET
Lotus_123_97_Fmt	296	81	Lotus 1-2-3 97	application/x-lotus-123	123	adSPREADSHEET
Lotus_Word_Pro_96_Fmt	297	268	Lotus Word Pro 96	application/vnd.lotus-wordpro	LWP, MWP	adWORDPROCESSOR
Lotus_Word_Pro_97_Fmt	298	268	Lotus Word Pro 97	application/vnd.lotus-wordpro	LWP, MWP	adWORDPROCESSOR
Freelance_DOS_Fmt	299	140	Lotus Freelance for DOS	application/x-freelance	PRZ	adPRESENTATION
Freelance_Win_Fmt	300	140	Lotus Freelance for Windows	application/x-freelance	PRE	adPRESENTATION
Freelance_OS2_Fmt	301	140	Lotus Freelance for OS/2	application/x-freelance	PRS	adPRESENTATION
Freelance_96_Fmt	302	140	Lotus Freelance 96	application/x-freelance	PRZ	adPRESENTATION
Freelance_97_Fmt	303	140	Lotus Freelance 97	application/x-freelance	PRZ	adPRESENTATION
MS_Word_95_Fmt	304	189	Microsoft Word 95	application/msword	DOC	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Word_97_Fmt	305	269	Microsoft Word 97	application/msword	DOC, WPS, WBK	adWORDPROCESSOR
Excel_Fmt	306	90	Microsoft Excel (up to version 5)	application/x-ms-excel	XLS	adSPREADSHEET
Excel_Chart_Fmt	307	90	Microsoft Excel (up to version 5) chart	application/x-ms-excel	XLC	adSPREADSHEET
Excel_Macro_Fmt	308	90	Microsoft Excel (up to version 5) macro	application/vnd.ms-excel	XLM	adSPREADSHEET
Excel_95_Fmt	309	188	Microsoft Excel 95	application/x-ms-excel	XLS	adSPREADSHEET
Excel_97_Fmt	310	188	Microsoft Excel 97	application/x-ms-excel	XLS	adSPREADSHEET
Corel_Presentations_Fmt	311	127	Corel Presentations	application/x-corelpresentations	XFD, XFDL	adPRESENTATION
Harvard_Graphics_Fmt	312	131	Harvard Graphics		PR4	adPRESENTATION
Harvard_Graphics_Chart_Fmt	313	131	Harvard Graphics Chart		CH3, CHT	adVECTORGRAPHIC
Harvard_Graphics_Symbol_Fmt	314	131	Harvard Graphics Symbol File		SY3	adVECTORGRAPHIC
Harvard_Graphics_Cfg_Fmt	315	131	Harvard Graphics Configuration File			adVECTORGRAPHIC
Harvard_Graphics_Palette_Fmt	316	131	Harvard Graphics Palette			adVECTORGRAPHIC
Lotus_123_R9_Fmt	317	81	Lotus 1-2-3 Release 9	application/x-lotus-123	123	adSPREADSHEET
Applix_Spreadsheets_Fmt	318	278	Applix Spreadsheets	application/x-applix-spreadsheet	AS	adSPREADSHEET
MS_Pocket_Word_Fmt	319	45	Microsoft Pocket Word		PWD	adWORDPROCESSOR
MS_DIB_Fmt	320	279	Microsoft Device Independent Bitmap	image/bmp	DIB	adRASTERIMAGE
MS_Word_2000_Fmt	321	269	Microsoft Word 2000	application/msword	DOC	adWORDPROCESSOR
Excel_2000_Fmt	322	188	Microsoft Excel 2000	application/x-ms-excel	XLS	adSPREADSHEET
PowerPoint_2000_Fmt	323	272	Microsoft PowerPoint 2000	application/x-ms-powerpoint	PPT	adPRESENTATION
MS_Access_2000_Fmt	324	263	Microsoft Access 2000	application/x-msaccess	MDB	adDATABASE
MS_Project_4_Fmt	325	281	Microsoft Project 4		MPP	adSCHEDULE
MS_Project_41_Fmt	326	281	Microsoft Project 4.1		MPP	adSCHEDULE
MS_Project_98_Fmt	327	281	Microsoft Project 98	application/vnd.ms-project	MPP	adSCHEDULE
Folio_Flat_Fmt	328	282	Folio Flat File		FFF	adWORDPROCESSOR
HWP_Fmt	329	283	HWP (Arae-Ah Hangul)	application/x-hwp	HWP	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ICHITARO_Fmt	330	284	ICHITARO (v4-10)		JTD	adWORDPROCESSOR
IS_XML_Fmt	331	273	Extended or Custom XML	text/xml	XML	adWORDPROCESSOR
Oasys_Fmt	332	286	Oasys	application/vnd.fujitsu.oasys	OAS, OA2, OA3	adWORDPROCESSOR
PBM_ASC_Fmt	333	287	Portable Bitmap Utilities ASCII format (PBM)	image/pbm	PBM	adRASTERIMAGE
PBM_BIN_Fmt	334	287	Portable Bitmap Utilities BINARY format (PBM)	image/pbm	PBM	adRASTERIMAGE
PGM_ASC_Fmt	335	288	Portable Greymap Utilities ASCII format (PGM)	image/x-pgm	PGM	adRASTERIMAGE
PGM_BIN_Fmt	336	288	Portable Greymap Utilities BINARY format (PGM)	image/x-pgm	PGM	adRASTERIMAGE
PPM_ASC_Fmt	337	289	Portable Pixmap Utilities ASCII format (PPM)	image/x-portable-pixmap	PPM	adRASTERIMAGE
PPM_BIN_Fmt	338	289	Portable Pixmap Utilities BINARY format (PPM)	image/x-portable-pixmap	PPM	adRASTERIMAGE
XBM_Fmt	339	290	X Bitmap format (XBM)	image/x-xbitmap	XBM	adRASTERIMAGE
XPM_Fmt	340	291	X Pixmap format (XPM)	image/xpm	XPM	adRASTERIMAGE
FPX_Fmt	341	292	Kodak FlashPix FPX Image format	image/fpx	FPX	adRASTERIMAGE
PCD_Fmt	342	293	PCD Image format	image/pcd	PCD	adRASTERIMAGE
MS_Visio_Fmt	343	294	Microsoft Visio (up to version 11)	image/x-vsd	VSD	adPRESENTATION
MS_Project_2000_Fmt	344	281	Microsoft Project 2000	application/vnd.ms-project	MPP	adSCHEDULE
MS_Outlook_Fmt	345	295	Microsoft Outlook message	application/vnd.ms-outlook	MSG, OFT	adENCAPSULATION
ELF_Relocatable_Fmt	346	159	ELF Relocatable	application/octet-stream	O	adOBJECTMODULE
ELF_Executable_Fmt	347	158	ELF Executable	application/octet-stream		adEXECUTABLE
ELF_Dynamic_Lib_Fmt	348	160	ELF Dynamic Library	application/octet-stream	SO	adLIBRARY
MS_Word_XML_Fmt	349	285	Microsoft Word 2003 XML	text/xml	XML	adWORDPROCESSOR
MS_Excel_XML_Fmt	350	285	Microsoft Excel 2003 XML	text/xml	XML	adWORDPROCESSOR
MS_Visio_XML_Fmt	351	285	Microsoft Visio 2003 XML	text/xml	VDX	adWORDPROCESSOR
SO_Text_XML_Fmt	352	314	OpenDocument format (OpenOffice 1/StarOffice 6,7) Text XML	application/vnd.sun.xml.writer	SXW	adWORDPROCESSOR
SO_Spreadsheet_XML_Fmt	353	315	OpenDocument format	application/vnd.sun.xml.calc	SXC, STC	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			(OpenOffice 1/StarOffice 6,7) Spreadsheet XML			
SO_Presentation_XML_Fmt	354	316	OpenDocument format (OpenOffice 1/StarOffice 6,7) Presentation XML	application/vnd.sun.xml.impress	SXD, SXI	adPRESENTATION
XHTML_Fmt	355	296	XHTML	text/xhtml	XML, ASP	adWORDPROCESSOR
MS_OutlookPST_Fmt	356	297	Microsoft Outlook Personal Folders File (.pst)	application/vnd.ms-outlook-pst	PST	adENCAPSULATION
RAR_Fmt	357	298	RAR archive format	application/x-rar-compressed	RAR	adENCAPSULATION
Lotus_Notes_NSF_Fmt	358	299	IBM Lotus Notes Database NSF/NTF	application/x-lotus-notes	NSF	adENCAPSULATION
Macromedia_Flash_Fmt	359	300	Macromedia Flash (.swf)	application/x-shockwave-flash	SWF	adWORDPROCESSOR
MS_Word_2007_Fmt	360	301	Microsoft Word 2007 XML - Docx	application/x-ms-word07	DOCX, DOTX	adWORDPROCESSOR
MS_Excel_2007_Fmt	361	302	Microsoft Excel 2007 XML	application/x-ms-excel07	XLSX, XLTX	adSPREADSHEET
MS_PPT_2007_Fmt	362	303	Microsoft PowerPoint 2007 XML	application/x-ms-powerpoint07	PPTX, POTX, PPSX	adPRESENTATION
OpenPGP_Fmt	363	304	OpenPGP Message Format (with new packet format)	application/pgp-encrypted	PGP	adENCAPSULATION
Intergraph_V7_DGN_Fmt	364	305	Intergraph Standard File Format (ISFF) V7 DGN (non-OLE)		DGN	adVECTORGRAPHIC
MicroStation_V8_DGN_Fmt	365	306	MicroStation V8 DGN (OLE)		DGN	adVECTORGRAPHIC
MS_Word_Macro_2007_Fmt	366	307	Microsoft Word Macro 2007 XML	application/x-ms-word07m	DOCM, DOTM	adWORDPROCESSOR
MS_Excel_Macro_2007_Fmt	367	308	Microsoft Excel Macro 2007 XML	application/x-ms-excel07m	XLSM, XLTM, XLAM	adSPREADSHEET
MS_PPT_Macro_2007_Fmt	368	309	Microsoft PPT Macro 2007 XML	application/x-ms-powerpoint07m	PPTM, POTM, PPSM, PPAM	adPRESENTATION
LZH_Fmt	369	310	LZH Archive	application/x-lzh-compressed	LZH, LHA	adENCAPSULATION
Office_2007_Fmt	370	311	Office 2007 document		XLSB	adMISC
MS_XPS_Fmt	371	312	Microsoft XML Paper Specification (XPS)	application/vnd.ms-xpsdocument	XPS	adWORDPROCESSOR
Lotus_Domino_DXL_Fmt	372	313	IBM Domino Data in XML format (.dxl)	text/xml	DXL	adENCAPSULATION
ODF_Text_Fmt	373	314	ODF Text	application/vnd.oasis.opendocument.text	ODT	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ODF_Spreadsheet_Fmt	374	315	ODF Spreadsheet	application/vnd.oasis.opendocument.spreadsheet	ODS	adSPREADSHEET
ODF_Presentation_Fmt	375	316	ODF Presentation	application/vnd.oasis.opendocument.presentation	ODP	adPRESENTATION
Legato_Extender_ONM_Fmt	376	317	Legato Extender Native Message ONM	application/x-lotus-notes	ONM	adENCAPSULATION
bin_Unknown_Fmt	377	318	Bin unknown format (.xxx)			adWORDPROCESSOR
TNEF_Fmt	378	319	Transport Neutral Encapsulation Format (TNEF)	application/vnd.ms-tnef		adENCAPSULATION
CADAM_Drawing_Fmt	379	320	CADAM Drawing		CDD	adVECTORGRAPHIC
CADAM_Drawing_Overlay_Fmt	380	321	CADAM Drawing Overlay		CDO	adVECTORGRAPHIC
NURSTOR_Drawing_Fmt	381	322	NURSTOR Drawing		NUR	adVECTORGRAPHIC
HP_GLP_Fmt	382	323	HP Graphics Language (Plotter)	vector/x-hpgl2	HPG	adVECTORGRAPHIC
ASF_Fmt	383	324	Advanced Systems Format (ASF)	application/x-ms-asf	ASF	adMISC
WMA_Fmt	384	325	Windows Media Audio Format (WMA)	audio/x-ms-wma	WMA	adSOUND
WMV_Fmt	385	326	Windows Media Video Format (WMV)	video/x-ms-wmv	WMV	adMOVIE
EMX_Fmt	386	327	Legato EMailXtender Archives Format (EMX)		EMX	adENCAPSULATION
Z7Z_Fmt	387	328	7 Zip Format (7z)	application/7z	7Z	adENCAPSULATION
MS_Excel_Binary_2007_Fmt	388	329	Microsoft Excel Binary 2007	application/vnd.ms-excel.sheet.binary.macroenabled.12	XLSB	adSPREADSHEET
CAB_Fmt	389	330	Microsoft Cabinet File (CAB)	application/vnd.ms-cab-compressed	CAB	adENCAPSULATION
CATIA_Fmt	390	331	CATIA Formats (CAT*)		CATPART, CATPRODUCT 2	adVECTORGRAPHIC
YIM_Fmt	391	332	Yahoo Instant Messenger History		DAT	adWORDPROCESSOR
ODF_Drawing_Fmt	392	316	ODF Drawing/Graphics	application/vnd.oasis.opendocument.graphics	ODG	adVECTORGRAPHIC
Founder_CEB_Fmt	393	333	Founder Chinese E-paper Basic (ceb)	application/ceb	CEB	adWORDPROCESSOR
QPW_Fmt	394	334	Corel Quattro Pro 9+ for Windows	application/quattro-pro	QPW	adSPREADSHEET
MHT_Fmt	395	335	MHTML format (MHT)[1]	multipart/related	MHT, MHTML	adWORDPROCESSOR
MDI_Fmt	396	336	Microsoft Document Imaging Format	image/vnd.ms-modi	MDI	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
GRV_Fmt	397	337	Microsoft Office Groove Format	application/vnd.groove-injector	GRV	adWORDPROCESSOR
IWWP_Fmt	398	338	Apple iWork Pages format	application/vnd.apple.pages	PAGES	adWORDPROCESSOR
IWSS_Fmt	399	339	Apple iWork Numbers format	application/vnd.apple.numbers	NUMBERS	adSPREADSHEET
IWPG_Fmt	400	340	Apple iWork Keynote format	application/vnd.apple.keynote	KEY	adPRESENTATION
BKF_Fmt	401	341	Windows Backup File		BKF	adENCAPSULATION
MS_Access_2007_Fmt	402	342	Microsoft Access 2007	application/msaccess	ACCDB	adDATABASE
ENT_Fmt	403	343	Microsoft Entourage Database Format			adENCAPSULATION
DMG_Fmt	404	344	Mac Disk Copy Disk Image File	application/x-apple-diskimage	DMG	adENCAPSULATION
CWK_Fmt	405	345	AppleWorks File	application/appleworks	CWK	adWORDPROCESSOR
OO3_Fmt	406	346	Omni Outliner V3 File		OO3	adWORDPROCESSOR
OPML_Fmt	407	347	Omni Outliner OPML File		OPML	adWORDPROCESSOR
Omni_Graffle_XML_Fmt	408	348	Omni Graffle XML File		GRAFFLE	adVECTORGRAPHIC
PSD_Fmt	409	349	Photoshop Document	image/vnd.adobe.photoshop	PSD, PSB	adRASTERIMAGE
Apple_Binary_PList_Fmt	410	350	Apple Binary Property List format		PLIST	adMISC
Apple_iChat_Fmt	411	351	Apple iChat format		ICHAT	adWORDPROCESSOR
OOUTLINE_Fmt	412	352	OOutliner File		OOUTLINE	adWORDPROCESSOR
BZIP2_Fmt	413	353	Bzip 2 Compressed File	application/x-bzip2	BZ2	adENCAPSULATION
ISO_Fmt	414	354	ISO-9660 CD Disc Image Format	application/x-iso9660-image	ISO	adENCAPSULATION
DocuWorks_Fmt	415	355	DocuWorks Format	application/vnd.fujixerox.docuworks	XDW	adWORDPROCESSOR
RealMedia_Fmt	416	356	RealMedia Streaming Media	application/vnd.rn-realmedia	RM, RA	adMOVIE
AC3Audio_Fmt	417	357	AC3 Audio File Format	audio/ac3	AC3	adSOUND
NEF_Fmt	418	358	Nero Encrypted File		NEF	adENCAPSULATION
SolidWorks_Fmt	419	359	SolidWorks Format Files		SLDASM, SLDPRT, SLDDRW, SLDDRT	adVECTORGRAPHIC
XFDL_Fmt	420	366	Extensible Forms Description Language	application/x-xfdl	XFDL, XFD	adPRESENTATION
Apple_XML_PList_Fmt	421	367	Apple XML Property List format		PLIST	adMISC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
OneNote_Fmt	422	368	OneNote Note Format	application/onenote	ONE	adPRESENTATION
IFilter_Fmt	423	369	iFilter			adWORDPROCESSOR
Dicom_Fmt	424	370	Digital Imaging and Communications in Medicine (Dicom)	application/dicom	DCM	adRASTERIMAGE
EnCase_Fmt	425	371	Expert Witness Compression Format (EnCase)		E01, L01, Lx01	adENCAPSULATION
Scrap_Fmt	426	372	Shell Scrap Object File		SHS	adENCAPSULATION
MS_Project_2007_Fmt	427	373	Microsoft Project 2007	application/vnd.ms-project	MPP	adSCHEDULE
MS_Publisher_98_Fmt	428	374	Microsoft Publisher from version 98	application/x-mspublisher	PUB	adDESKTOPPUBLSH
Skype_Fmt	429	375	Skype Log File		DBB	adWORDPROCESSOR
Hl7_Fmt	430	377	Health level7 message		HL7	adWORDPROCESSOR
MS_OutlookOST_Fmt	431	378	Microsoft Outlook Offline Folders File (OST)	application/vnd.ms-outlook-pst	OST	adENCAPSULATION
Epub_Fmt	432	379	Electronic Publication	application/epub+zip	EPUB	adWORDPROCESSOR
MS_OEDBX_Fmt	433	380	Microsoft Outlook Express DBX Message Database		DBX	adENCAPSULATION
BB_Activ_Fmt	434	381	BlackBerry Activation File		DAT	adWORDPROCESSOR
DiskImage_Fmt	435	382	Disk Image		DMG	adENCAPSULATION
Milestone_Fmt	436	383	Milestone Document		MLS, ML3, ML4, ML5, ML6, ML7, ML8, ML9, MLA	adRASTERIMAGE
E_Transcript_Fmt	437	384	RealLegal E-Transcript File		PTX	adWORDPROCESSOR
PostScript_Font_Fmt	438	385	PostScript Type 1 Font	application/x-font	PFB	adFONT
Ghost_DiskImage_Fmt	439	386	Ghost Disk Image File		GHO, GHS	adENCAPSULATION
JPEG_2000_JP2_File_Fmt	440	387	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	image/jp2	JP2, JPF, J2K, JPWL, JPX, PGX	adRASTERIMAGE
Unicode_HTML_Fmt	441	388	Unicode HTML	text/html	HTM, HTML	adWORDPROCESSOR
CHM_Fmt	442	389	Microsoft Compiled HTML Help	application/x-chm	CHM	adENCAPSULATION
EMCMF_Fmt	443	390	Documentum EMCMF format		EMCMF	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Access_2007_Tmpl_Fmt	444	391	Microsoft Access 2007 Template		ACCDT	adDATABASE
Jungum_Fmt	445	392	Samsung Electronics Jungum Global document		GUL	adWORDPROCESSOR
JBIG2_Fmt	446	393	JBIG2 File Format	image/jbig2	JB2, JBIG2	adRASTERIMAGE
EFax_Fmt	447	394	eFax file		EFX	adRASTERIMAGE
AD1_Fmt	448	395	AD1 Evidence file		AD1	adENCAPSULATION
SketchUp_Fmt	449	396	Google SketchUp		SKP	adVECTORGRAPHIC
GWFS_Email_Fmt	450	397	Group Wise File Surf email		GWFS	adENCAPSULATION
JNT_Fmt	451	398	Windows Journal format		JNT	adWORDPROCESSOR
Yahoo_yChat_Fmt	452	399	Yahoo! Messenger chat log		YCHAT	adWORDPROCESSOR
PaperPort_MAX_File_Fmt	453	400	PaperPort MAX image file	image/max	MAX	adRASTERIMAGE
ARJ_Fmt	454	402	ARJ (Archive by Robert Jung) file format	application/arj	ARJ	adENCAPSULATION
RPMSG_Fmt	455	403	Microsoft Outlook Restricted Permission Message	application/x-microsoft-rpmsg-message	RPMSG	adENCAPSULATION
MAT_Fmt	456	404	MATLAB file format	application/x-matlab-data	MAT, FIG	adWORDPROCESSOR
SGY_Fmt	457	405	SEG-Y Seismic Data format		SGY, SEGY	adWORDPROCESSOR
CDXA_MPEG_PS_Fmt	458	406	MPEG-PS container with CDXA stream	video/mpeg	MPG	adMOVIE
EVT_Fmt	459	407	Microsoft Windows NT Event Log		EVT	adMISC
EVTX_Fmt	460	408	Microsoft Windows Vista Event Log		EVTX	adMISC
MS_OutlookOLM_Fmt	461	409	Microsoft Outlook for Macintosh format		OLM	adENCAPSULATION
WARC_Fmt	462	410	Web ARChive	application/warc	WARC	adENCAPSULATION
JAVACLASS_Fmt	463	411	Java Class format	application/x-java-class	CLASS	adWORDPROCESSOR
VCF_Fmt	464	412	Microsoft Outlook vCard file format	text/vcard	VCF	adWORDPROCESSOR
EDB_Fmt	465	413	Microsoft Exchange Server Database file format		EDB	adENCAPSULATION
ICS_Fmt	466	414	Microsoft Outlook iCalendar file format	text/calendar	ICS, VCS	adENCAPSULATION
MS_Visio_2013_Fmt	467	415	Microsoft Visio 2013	application/vnd.visio	VSDX, VSTX,	adPRESENTATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
					VSSX	
MS_Visio_2013_Macro_Fmt	468	415	Microsoft Visio 2013 macro	application/vnd.visio	VSDM, VSTM, VSSM	adPRESENTATION
ICHITARO_Compr_Fmt	469	417	ICHITARO Compressed format	application/x-js-taro	JTDC	adWORDPROCESSOR
IWWP13_Fmt	470	418	Apple iWork 2013 Pages format		IWA, PAGES	adWORDPROCESSOR
IWSS13_Fmt	471	419	Apple iWork 2013 Numbers format		IWA, NUMBERS	adSPREADSHEET
IWPG13_Fmt	472	420	Apple iWork 2013 Keynote format		IWA, KEY	adPRESENTATION
XZ_Fmt	473	421	XZ archive format	application/x-xz	XZ	adENCAPSULATION
Sony_WAVE64_Fmt	474	422	Sony Wave64 format	audio/wav64	W64	adSOUND
Conifer_WAVPACK_Fmt	475	423	Conifer Wavpack format	audio/x-wavpack	WV	adSOUND
Xiph_OGG_VORBIS_Fmt	476	424	Xiph Ogg Vorbis format	audio/ogg	OGG	adSOUND
MS_Visio_2013_Stencil_Fmt	477	415	MS Visio 2013 stencil format	application/vnd.visio	VSSX	adPRESENTATION
MS_Visio_2013_Stencil_Macro_Fmt	478	415	MS Visio 2013 stencil Macro format	application/vnd.visio	VSSM	adPRESENTATION
MS_Visio_2013_Template_Fmt	479	415	MS Visio 2013 template format	application/vnd.visio	VSTX	adPRESENTATION
MS_Visio_2013_Template_Macro_Fmt	480	415	MS Visio 2013 template Macro format	application/vnd.visio	VSTM	adPRESENTATION
Borland_Reflex_2_Fmt	481	425	Borland Reflex 2 format		R2D	adDATABASE
PKCS_12_Fmt	482	426	PKCS #12 (p12) format	application/x-pkcs12	P12, PFX	adWORDPROCESSOR
B1_Fmt	483	427	B1 format	application/x-b1	B1	adENCAPSULATION
ISO_IEC_MPEG_4_Fmt	484	428	ISO/IEC MPEG-4 (ISO 14496) format	video/mp4	MP4	adMOVIE
RAR5_Fmt	485	429	RAR5 Format	application/x-rar-compressed	RAR	adENCAPSULATION
Unigraphics_NX_Fmt	486	362	Unigraphics (UG) NX CAD Format		PRT	adVECTORGRAPHIC
PTC_Creo_Fmt	487	430	PTC Creo CAD Format		ASM, PRT	adVECTORGRAPHIC
KML_Fmt	488	431	Keyhole Markup Language	application/vnd.google-earth.kml+xml	KML	adWORDPROCESSOR
KMZ_Fmt	489	432	Zipped Keyhole Markup Language	application/vnd.google-earth.kmz	KMZ	adWORDPROCESSOR
WML_Fmt	490	433	Wireless Markup Language	text/vnd.wap.wml	WML	adWORDPROCESSOR
ODF_Formula_Fmt	491	434	ODF Formula	application/vnd.oasis.opendocument.formula	ODF	adWORDPROCESSOR
SO_Text_Fmt	492	435	Star Office 4,5 Writer Text	application/vnd.stardivision.writer	SDW, SGL,	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
					VOR	
SO_Spreadsheet_Fmt	493	436	Star Office 4,5 Calc Spreadsheet	application/vnd.stardivision.calc	SDC	adSPREADSHEET
SO_Presentation_Fmt	494	437	Star Office 4,5 Impress Presentation	application/vnd.stardivision.draw	SDD, SDA	adPRESENTATION
SO_Math_Fmt	495	438	Star Office 4,5 Math	application/vnd.stardivision.math	SMF	adMISC
STEP_Fmt	496	439	ISO 10303-21 STEP format			adMISC
STL_Fmt	497	364	3D Systems STL ASCII format			adMISC
AppleScript_Fmt	498	440	AppleScript Source Code[3]	text/x-applescript	APPLESCRIPT	adSOURCECODE
Assembly_Fmt	499	441	Assembly Code[3]	text/x-assembly		adSOURCECODE
C_Fmt	500	442	C Source Code[3]	text/x-c	C, H	adSOURCECODE
Csharp_Fmt	501	443	C# Source Code[3]	text/x-csharp	CS	adSOURCECODE
CPlusPlus_Fmt	502	444	C++ Source Code[3]	text/x-c++	CPP, HPP	adSOURCECODE
Css_Fmt	503	445	Cascading Style Sheet[3]	text/css	CSS	adSOURCECODE
Clojure_Fmt	504	446	Clojure Source Code[3]	text/x-clojure	CLJ, CL2	adSOURCECODE
CoffeeScript_Fmt	505	447	CoffeeScript Source Code[3]	text/x-coffeescript	COFFEE, CAKE	adSOURCECODE
Lisp_Fmt	506	448	Common Lisp Source Code[3]	text/x-common-lisp	EL	adSOURCECODE
Dockerfile_Fmt	507	449	Dockerfile[3]	text/x-dockerfile		adSOURCECODE
Eiffel_Fmt	508	450	Eiffel Source Code[3]	text/x-eiffel	E	adSOURCECODE
Erlang_Fmt	509	451	Erlang Source Code[3]	text/x-erlang	ERL, ES	adSOURCECODE
Fsharp_Fmt	510	452	F# Source Code[3]	text/x-fsharp	FS	adSOURCECODE
Fortran_Fmt	511	453	Fortran Source Code[3]	text/x-fortran	F	adSOURCECODE
Go_Fmt	512	454	Go Source Code[3]	text/x-go	GO	adSOURCECODE
Groovy_Fmt	513	455	Groovy Source Code[3]	text/x-groovy	GRT, GVY	adSOURCECODE
Haskell_Fmt	514	456	Haskell Source Code[3]	text/x-haskell	HS	adSOURCECODE
Ini_Fmt	515	457	Initialization (INI) file[3]	text/x-ini		adSOURCECODE
Java_Fmt	516	458	Java Source Code[3]	text/x-java-source	JAVA	adSOURCECODE
Javascript_Fmt	517	459	Javascript Source Code[3]	text/javascript	JS	adSOURCECODE
Lua_Fmt	518	460	Lua Source Code[3]	text/x-lua	LUA	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Makefile_Fmt	519	461	Makefile[3]	text/x-makefile	MAKE	adSOURCECODE
Mathematica_Fmt	520	462	Wolfram Mathematica Source Code[3]	text/x-mathematica	M	adSOURCECODE
ObjC_Fmt	521	464	Objective-C Source Code[3]	text/x-objc		adSOURCECODE
ObjCpp_Fmt	522	465	Objective-C++ Source Code[3]	text/x-objectivec++		adSOURCECODE
ObjJ_Fmt	523	466	Objective-J Source Code[3]	text/x-objectivej	J	adSOURCECODE
PHP_Fmt	524	467	PHP Source Code[3]	text/x-php	PHP	adSOURCECODE
PLSQL_Fmt	525	468	PLSQL Source Code[3]	text/x-plsql		adSOURCECODE
Pascal_Fmt	526	469	Pascal Source Code[3]	text/x-pascal	PASCAL	adSOURCECODE
Perl_Fmt	527	470	Perl Source Code[3]	text/x-perl	PL	adSOURCECODE
Powershell_Fmt	528	471	PowerShell Source Code[3]	text/x-powershell	PS1	adSOURCECODE
Prolog_Fmt	529	472	Prolog Source Code[3]	text/x-prolog	PRO, PROLOG	adSOURCECODE
Puppet_Fmt	530	473	Puppet Source Code[3]	text/x-puppet	PP	adSOURCECODE
Python_Fmt	531	474	Python Source Code[3]	text/x-python	PY	adSOURCECODE
R_Fmt	532	475	R Source Code[3]	text/x-rsrc	R	adSOURCECODE
Ruby_Fmt	533	476	Ruby Source Code[3]	text/x-ruby	RB	adSOURCECODE
Rust_Fmt	534	477	Rust Source Code[3]	text/x-rust	RS	adSOURCECODE
Scala_Fmt	535	478	Scala Source Code[3]	text/x-scala	SC	adSOURCECODE
Shell_Fmt	536	479	Shell Script[3]	application/x-sh	SH	adSOURCECODE
Smalltalk_Fmt	537	480	Smalltalk Source Code[3]	text/x-stsrc	ST	adSOURCECODE
ML_Fmt	538	481	Standard ML Source Code[3]	text/x-ml	ML	adSOURCECODE
Swift_Fmt	539	482	Swift Source Code[3]	text/x-swift	SWIFT	adSOURCECODE
Tcl_Fmt	540	483	Tool Command Language (Tcl) Source Code[3]	text/x-tcl	TM	adSOURCECODE
Tex_Fmt	541	484	TeX Typesetting File[3]	application/x-tex		adSOURCECODE
TypeScript_Fmt	542	485	TypeScript Source Code[3]	text/x-typescript	TS	adSOURCECODE
Verilog_Fmt	543	486	Verilog Source Code[3]	text/x-verilog	V	adSOURCECODE
YAML_Fmt	544	487	YAML File[3]	text/x-yaml	YML	adSOURCECODE
Wiki_Fmt	545	488	MediaWiki File	text/x-mediawiki		adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
MS_Word_2007_Flat_XML_Fmt	546	301	Microsoft Word 2007 XML - Flat xml	text/xml	XML	adWORDPROCESSOR
Matroska_Fmt	547	489	Matroska video File	video/x-matroska	MKV	adMOVIE
SVG_Fmt	548	490	Scalable Vector Graphics image	image/svg+xml	SVG	adVECTORGRAPHIC
Shapefile_Fmt	549	491	Shapefile	application/x-shapefile	SHP, SHX	adMISC
Flash_Video_Fmt	550	492	Flash video File	video/x-flv	FLV	adMOVIE
Embedded_OpenType_Fmt	551	493	Embedded OpenType font	application/vnd.ms-fontobject	EOT	adFONT
Web_Open_Font_Fmt	552	494	Web Open Font Format	font/woff	WOFF, WOFF2	adFONT
OpenType_Fmt	553	495	OpenType Font	font/otf	OTF	adFONT
MNG_Fmt	554	496	Multiple-image Network Graphics	video/x-mng	MNG	adANIMATION
JNG_Fmt	555	497	JPEG Network Graphics	image/x-jng	JNG	adRASTERIMAGE
AppleScript_Binary_Fmt	556	498	AppleScript Binary Source Code		SCPT	adSOURCECODE
Maya_Binary_Fmt	557	499	Autodesk Maya binary file		MB	adCAD
Jupiter_Tesselation_Fmt	558	363	UGS Jupiter Tesselation file		JT	adCAD
OGV_Fmt	559	500	Ogg Theora Video format	video/ogg	OGV	adMOVIE
OGG_Container_Fmt	560	501	General Ogg Container format	application/ogg	OGG	adMISC
GNU_Message_Catalog_Fmt	561	502	GNU Message Catalog format		MO	adMISC
Windows_Shortcut_Fmt	562	503	Windows shortcut file		LNK	adMISC
Apple_Typedstream_Fmt	563	504	Apple/NeXT typedstream data format			adMISC
XCF_Fmt	564	505	GIMP XCF image	image/x-xcf	XCF	adRASTERIMAGE
PaintShop_Pro_Fmt	565	506	PaintShop Pro image		PSP, PSPIMAGE	adRASTERIMAGE
SQLite_Database_Fmt	566	507	SQLite database format	application/x-sqlite3	QHC	adDATABASE
MySQL_Table_Fmt	567	508	MySQL table definition file		FRM	adDATABASE
Microsoft_Program_DB_Fmt	568	509	Microsoft Program Database format		PDB	adDATABASE
OpenEXR_Fmt	569	510	OpenEXR image format		EXR	adRASTERIMAGE
XMV_Fmt	570	511	4X Movie File			adMOVIE
AMV_Fmt	571	512	AMV video file		AMV	adMOVIE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
NIFF_Fmt	572	513	Notation Interchange File Format		NIF	adSOUND
CuBase_Fmt	573	514	Steinberg CuBase file			adSOUND
SoundFont_Fmt	574	515	SoundFont file			adSOUND
WebP_Fmt	575	516	WebP image	image/webp	WEBP	adRASTERIMAGE
ICC_Fmt	576	517	International Color Consortium files	application/vnd.iccprofile	ICC, ICM	adMISC
PCF_Fmt	577	518	X11 Portable Compiled Font file	application/x-font-pcf	PCF	adFONT
WebM_Fmt	578	519	WebM video file	video/webm	WEBM	adMOVIE
AMFF_Fmt	579	520	Amiga Metafile		AMF	adVECTORGRAPHIC
ANBM_Fmt	580	521	IFF Animated Bitmap			adRASTERIMAGE
ANIM_Fmt	581	522	IFF Amiga animated raster graphics format			adRASTERIMAGE
DEEP_Fmt	582	523	IFF-DEEP TVPaint image		DEEP	adRASTERIMAGE
FAXX_Fmt	583	524	IFF-FAXX Facsimile image			adRASTERIMAGE
ICON_Fmt	584	525	IFF Glow Icon image			adRASTERIMAGE
ILBM_Fmt	585	526	Interleaved BitMap image		IFF	adRASTERIMAGE
LWOB_Fmt	586	527	LightWave Object format		LWOB	adMISC
MAUD_Fmt	587	528	IFF-MAUD MacroSystem audio format			adSOUND
PBM_Fmt	588	529	IFF Planar BitMap			adRASTERIMAGE
TDDD_Fmt	589	530	IFF TDDD and Imagine Object animation format		TDD	adRASTERIMAGE
DjVu_Fmt	590	531	AT&T DjVu format	image/vnd.djvu	DJVU	adWORDPROCESSOR
InDesign_Fmt	591	532	Adobe InDesign document	application/x-indesign		adDESKTOPPUBLSH
Calamus_Fmt	592	533	Calamus Desktop Publishing			adDESKTOPPUBLSH
Adaptive_MultiRate_Fmt	593	534	Adaptive Multi-Rate audio format	audio/amr	AMR	adSOUND
FLAC_Fmt	594	535	Free Lossless Audio Codec format	audio/flac	FLAC	adSOUND
Ogg_FLAC_Fmt	595	536	Ogg Container FLAC audio format		OGG	adSOUND
SAS7BDAT_Fmt	596	537	SAS7BDAT database storage format		SAS7BDAT	adDATABASE
Design_Web_Format_Fmt	597	538	Autodesk Design Web Format	model/vnd.dwf	DWF	adCAD

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Adobe_Flash_Audio_Book_Fmt	598	539	Adobe Flash Player audio book	audio/mp4	F4B	adSOUND
Adobe_Flash_Audio_Fmt	599	540	Adobe Flash Player audio	audio/mp4	F4A	adSOUND
Adobe_Flash_Protected_Video_Fmt	600	541	Adobe Flash Player protected video	video/mp4	F4P	adMOVIE
Adobe_Flash_Video_Fmt	601	542	Adobe Flash Player video	video/x-f4v	F4V	adMOVIE
Audible_Audiobook_Fmt	602	543	Audible Enhanced Audiobook		AAX	adSOUND
Canon_Camera_Fmt	603	544	Canon Digital Camera image			adRASTERIMAGE
Canon_Raw_Fmt	604	545	Canon Raw image		CR3	adRASTERIMAGE
Casio_Camera_Fmt	605	546	Casio Digital Camera image			adRASTERIMAGE
Convergent_Design_Fmt	606	547	Convergent Design file			adRASTERIMAGE
DMB_MAF_Audio_Fmt	607	548	DMB MAF audio			adSOUND
DMB_MAF_Video_Fmt	608	549	DMB MAF video			adMOVIE
DMP_Content_Fmt	609	550	Digital Media Project Content Format			adMISC
DVB_Fmt	610	551	Digital Video Broadcast format	video/vnd.dvb.file	DVB	adMOVIE
Dirac_Wavelet_Compression_Fmt	611	552	ISO-BMFF Dirac Wavelet compression			adMISC
HEICS_Image_Sequence_Fmt	612	553	High Efficiency Image Format HEVC image sequence	image/heic-sequence	HEICS	adRASTERIMAGE
HEIC_Image_Fmt	613	554	High Efficiency Image Format HEVC image	image/heic	HEIC	adRASTERIMAGE
HEIFS_Image_Sequence_Fmt	614	555	High Efficiency Image Format image sequence	image/heif-sequence	HEIFS	adRASTERIMAGE
HEIF_Image_Fmt	615	556	High Efficiency Image Format image	image/heif	HEIF	adRASTERIMAGE
ISMACryp_Fmt	616	557	ISMACryp 2.0 Encrypted format			adENCAPSULATION
ISO_3GPP2_Fmt	617	558	3GPP2 video file	video/3gpp2	3G2	adMOVIE
ISO_3GPP_Fmt	618	559	3GPP video file	video/3gpp	3GP	adMOVIE
ISO_JPEG2000_JP2_Fmt	619	560	ISO-BMFF JPEG 2000 image	image/jp2	JP2	adRASTERIMAGE
ISO_JPEG2000_JPM_Fmt	620	561	ISO-BMFF JPEG 2000 compound image	image/jpm	JPM	adRASTERIMAGE
ISO_JPEG2000_JPX_Fmt	621	562	ISO-BMFF JPEG 2000 with extensions	image/jpx	JPX	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ISO_QuickTime_Fmt	622	563	Apple ISO-BMFF QuickTime video	video/quicktime	QT, MOV	adMOVIE
KDDI_Video_Fmt	623	564	KDDI Video file	video/3gpp2		adMOVIE
MAF_Photo_Player_Fmt	624	565	MAF Photo Player			adMISC
MPEG4_AVC_Fmt	625	566	ISO-BMFF MPEG-4 with AVC extension	video/mp4		adMOVIE
MPEG4_M4A_Fmt	626	567	Apple MPEG-4 Part 14 audio	audio/x-m4a	M4A	adSOUND
MPEG4_M4B_Fmt	627	568	Apple MPEG-4 Part 14 audio book	audio/mp4	M4B	adSOUND
MPEG4_M4P_Fmt	628	569	Apple MPEG-4 Part 14 protected audio	audio/mp4	M4P	adSOUND
MPEG4_M4V_Fmt	629	570	Apple MPEG-4 Part 14 video	video/x-m4v	M4V	adMOVIE
MPEG4_Sony_PSP_Fmt	630	571	Sony PSP MPEG-4	audio/mp4	MP4	adSOUND
MPEG_21_Fmt	631	572	MPEG-21	audio/mp4		adMISC
Mobile_QuickTime_Fmt	632	573	Mobile QuickTime video	video/quicktime	MQV	adMOVIE
Motion_JPEG_2000_Fmt	633	574	Motion JPEG 2000	video/mj2	MJ2, MJP2	adMOVIE
NTT_MPEG4_Fmt	634	575	NTT MPEG-4	video/mp4		adMOVIE
Nero_MPEG4_AVC_Profile	635	576	Nero MPEG-4 profile with AVC extension	video/mp4		adMOVIE
Nero_MPEG4_Audio_Fmt	636	577	Nero AAC audio	audio/mp4		adSOUND
Nero_MPEG4_Profile	637	578	Nero MPEG-4 profile	video/mp4		adMOVIE
OMA_DRM_Fmt	638	579	OMA DRM Format			adMISC
Panasonic_Camera_Fmt	639	580	Panasonic Digital Camera image			adRASTERIMAGE
Ross_Video_Fmt	640	581	Ross video			adMOVIE
SDA_Video_Fmt	641	582	SDA SD Memory Card video			adMOVIE
Samsung_Stereoscopic_Fmt	642	583	Samsung stereoscopic stream			adMISC
Sony_XAVC_Fmt	643	584	Sony XAVC video			adMOVIE
JPEG_2000_PGX_Fmt	644	585	JPEG 2000 PGX Verification Model image		PGX	adRASTERIMAGE
Apple_Desktop_Services_Store_Fmt	645	586	Apple Desktop Services Store file		DS_Store	adMISC
Core_Audio_Fmt	646	587	Apple Core Audio Format	audio/x-caf	CAF	adSOUND
VICAR_Fmt	647	588	VICAR image format		IMG	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
FITS_Fmt	648	589	Flexible Image Transport System FITS image	image/fits	FIT	adRASTERIMAGE
DIF_Fmt	649	590	Digital Interface Format (DIF) DV video		DV	adMOVIE
MPEG_Transport_Stream_Fmt	650	591	MPEG Transport Stream data	video/MP2T	TS	adMISC
MPEG_Sequence_Fmt	651	592	MPEG Sequence format	video/mpeg		adMISC
Ogg_OGM_Fmt	652	593	Ogg OGM video format	video/ogg	OGM	adMOVIE
Ogg_Speex_Fmt	653	594	Ogg Speex audio format	audio/ogg	SPX	adSOUND
Ogg_Opus_Fmt	654	595	Ogg Opus audio format	audio/ogg	OGG	adSOUND
Musepack_Audio_Fmt	655	596	Musepack audio format	audio/x-musepack	MPC	adSOUND
ART_Image_Fmt	656	597	ART image format		ART	adRASTERIMAGE
Vivo_Fmt	657	598	Vivo audio-video format	video/vnd.vivo	VIV	adMOVIE
QCP_Fmt	658	599	Qualcomm QCP audio	audio/qcelp	QCP	adSOUND
CSP_Codec_Fmt	659	600	Creative Signal Processor codec		CSP	adMISC
TwinVQ_Fmt	660	601	NTT TwinVQ audio format		VQF	adSOUND
Interplay_MVE_Fmt	661	602	Interplay MVE video format		MVE	adMOVIE
IRIX_Moviemaker_Fmt	662	603	IRIX Silicon Graphics moviemaker video file	video/x-sgi-movie	MV, MOVIE	adMOVIE
Sega_FILM_Fmt	663	604	Sega FILM video format		CPK, CAK	adMOVIE
SMAF_Fmt	664	605	Synthetic music Mobile Application Format	application/vnd.smaf	MMF	adSOUND
NIST_SPHERE_Fmt	665	606	NIST SPeech HEader REsources format		NIST	adSOUND
Chinese_AVS_Fmt	666	607	Chinese AVS video format			adMOVIE
VQA_Fmt	667	608	Westwood Studios Vector Quantized Animation video file		VQA	adANIMATION
YAFA_Fmt	668	609	Wildfire YAFA animation		YAFA	adANIMATION
Origin_MVE_Fmt	669	610	Origin Wing Commander III MVE movie format		MVE	adMOVIE
BBC_Dirac_Fmt	670	611	BBC Dirac video format	video/x-dirac	DRC	adMOVIE
Maya_ASCII_Fmt	671	612	Autodesk Maya ASCII file format		MA	adCAD

Format Name	Number	Category	Description	MIME Type	Extension	File Class
RenderMan_Fmt	672	613	Pixar RenderMan Interface Bytestream file		RIB	adVECTORGRAPHIC
NOFF_Binary_Fmt	673	614	NOFF 3D Object File Format		NOFF	adVECTORGRAPHIC
VTK_ASCII_Fmt	674	615	Visualization Toolkit VTK ASCII format		VTK	adVECTORGRAPHIC
VTK_Binary_Fmt	675	616	Visualization Toolkit VTK Binary format		VTK	adVECTORGRAPHIC
Wolfram_CDF_Fmt	676	617	Wolfram Mathematica Computable Document Format	application/cdf	CDF	adMISC
Wolfram_Notebook_Fmt	677	618	Wolfram Mathematica Notebook Format		NB	adMISC
HDF4_Fmt	678	619	Hierarchical Data Format HDF4	application/x-hdf	HDF, H4	adMISC
HDF5_Fmt	679	620	Hierarchical Data Format HDF5	application/x-hdf	HDF, H5	adMISC
ARMovie_Fmt	680	621	Acorn RISC ARMovie video format		RPL	adMOVIE
Windows_TV_DVR_Fmt	681	622	Windows Television DVR format		WTV	adMOVIE
InstallShield_Z_Fmt	682	623	InstallShield Z archive format	application/x-compress	Z	adENCAPSULATION
MS_DirectDraw_Surface_Fmt	683	624	Microsoft DirectDraw Surface container format		DDS	adENCAPSULATION
Bink_Fmt	684	625	Bink audio-video container format		BIK, BK2	adMOVIE
LZMA_Fmt	685	626	LZMA compressed data format	application/x-lzma	LZMA	adENCAPSULATION
True_Audio_Fmt	686	627	True Audio format	audio/x-tta	TTA	adSOUND
Keepass_Fmt	687	628	Keepass Password file		KDB, KDBX	adMISC
RPM_Fmt	688	629	RPM Package Manager file	application/x-rpm	RPM	adENCAPSULATION
Printer_Font_Metrics_Fmt	689	630	Adobe Printer Font Metrics format	application/x-font-printer-metric	PFM	adFONT
Adobe_Font_Metrics_Fmt	690	631	Adobe Font Metrics ASCII format	application/x-font-adobe-metric	AFM	adFONT
Printer_Font_ASCII_Fmt	691	632	Adobe Printer Font ASCII format	application/x-font-type1	PFA	adFONT
Netware_Loadable_Module_Fmt	692	633	Netware Loadable Module format		NLM	adMISC
TCPdump_pcap_Fmt	693	634	TCPdump packet stream capture savefile format	application/vnd.tcpdump.pcap	PCAP	adMISC
Multiple_Master_Font_Fmt	694	635	Adobe Multiple master font format		MMM	adFONT
TrueType_Font_Collection_Fmt	695	636	TrueType font collection format	application/x-font-ttf	TTC	adFONT

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Shapefile_Spatial_Index_Fmt	696	637	Shapefile binary spatial index format	application/x-shapefile	SBX, SBN	adMISC
Java_Key_Store_Fmt	697	638	Java Key Store format	application/x-java-keystore	KS	adMISC
Java_JCE_Key_Store_Fmt	698	639	Java JCE Key Store format	application/x-java-jce-keystore		adMISC
Quark_Xpress_Intel_Fmt	699	640	QuarkXPress Intel format	application/vnd.quark.quarkxpress	QXB	adDESKTOPPUBLSH
Windows_Imaging_Fmt	700	641	Microsoft Windows Imaging Format WIM		WIM	adMISC
VMware_Virtual_Disk_Fmt	701	642	VMware Virtual Disk Format 5.0	application/x-vmdk	VMDK	adMISC
XPConnect_Typelib_Fmt	702	643	XPConnect Typelib Format		XPT	adMISC
MS_DOS_Compression_Fmt	703	644	Microsoft MS-DOS installation 'Quantum' compression		EX_	adENCAPSULATION
DLS_Fmt	704	645	DLS Downloadable Sounds format		DLS	adSOUND
MS_Windows_Registry_Fmt	705	646	Microsoft Windows Registry format			adMISC
Microsoft_Help_2_Fmt	706	647	Microsoft Help 2.0 format		HXD, HXW, HXH	adENCAPSULATION
Qt_Translation_Fmt	707	648	Qt binary translation file format		QM	adMISC
PEM_SSL_Certificate_Fmt	708	649	PEM-encoded SSL certificate	application/pkix-cert	CRT, PEM, CER, KEY	adENCAPSULATION
PostScript_Printer_Description_Fmt	709	650	Adobe PostScript Printer Description file	application/vnd.cups-ppd	PPD	adMISC
Speedo_Font_Fmt	710	651	Speedo Font format		SPD	adFONT
InstallShield_Cabinet_Fmt	711	652	InstallShield Cabinet Archive format		CAB, HDR	adENCAPSULATION
InstallShield_Uninstall_Fmt	712	653	InstallShield Uninstall format		ISU	adENCAPSULATION
MS_OEDBX_Folder_Fmt	713	654	Outlook Express DBX folder database format		DBX	adENCAPSULATION
LabVIEW_Fmt	714	655	National Instruments LabVIEW file format		VI	adMISC
SAP_Archive_SAR_Fmt	715	656	SAP compression archive SAR format		SAR	adENCAPSULATION
Netscape_Address_Book_Fmt	716	657	Netscape Address Book format		NAB	adMISC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Universal_3D_Fmt	717	658	Universal 3D file format		U3D	adVECTORGRAPHIC
Open_Inventor_ASCII_Fmt	718	659	Open Inventor ASCII format		IV	adVECTORGRAPHIC
Open_Inventor_Binary_Fmt	719	660	Open Inventor Binary format		IV	adVECTORGRAPHIC
X_Window_Dump_Fmt	720	661	X Window Dump image	image/x-xwindowdump	XWD	adRASTERIMAGE
Git_Packfile_Fmt	721	662	Git Packfile format		PACK	adENCAPSULATION
Xara_Xar_Fmt	722	663	Xara X Xar image format	application/vnd.xara	XAR	adVECTORGRAPHIC
Internet_Archive_ARC_Fmt	723	664	Internet Archive ARC format	application/x-ia-arc	ARC	adENCAPSULATION
Applix_Builder_Fmt	724	665	Applix Builder format		AB	adMISC
Applix_Bitmap_Fmt	725	666	Applix Bitmap image format		IM	adRASTERIMAGE
PEM_RSA_Private_Key_Fmt	726	667	PEM-encoded RSA private key		PEM	adENCAPSULATION
MIFF_Fmt	727	668	Magick Image File Format		MIFF	adRASTERIMAGE
Subversion_Dump_Fmt	728	669	Subversion Dump format			adENCAPSULATION
Virtual_Hard_Disk_Fmt	729	670	Microsoft Virtual Hard Disk format	application/x-vhd	VHD	adENCAPSULATION
Direct_Access_Archive_Fmt	730	671	PowerISO Direct Access Archive format		DAA	adENCAPSULATION
Debian_Binary_Fmt	731	672	Debian binary package format	application/x-debian-package	DEB	adENCAPSULATION
XUL_Fastload_Fmt	732	673	Mozilla XUL Fastload format		MFL	adMISC
Nastran_OP2_Fmt	733	674	Nastran OP2 format		OP2	adCAD
Binary_Logging_Fmt	734	675	CAD Binary Logging Format		BLF	adCAD
Measurement_Data_Fmt	735	676	CAD Measurement Data Format		MDF	adCAD
Abaqus_ODB_Fmt	736	677	Abaqus ODB Format		ODB	adCAD
Open_Diagnostic_Data_Exchange_Fmt	737	678	Vector Open Diagnostic Data Exchange format		ODX	adCAD
Vector_ASCII_Fmt	738	679	Vector CAD ASCII ASC format		ASC	adCAD
LSDYNA_State_Database_Fmt	739	680	LS-DYNA State Database format			adCAD
LSDYNA_Binary_Output_Fmt	740	681	LS-DYNA binary output (binout) format			adCAD
MS_Power_BI_Fmt	741	682	Microsoft Power BI Desktop format		PBIX	adANALYTICS
Tableau_Workbook_Fmt	742	683	Tableau Workbook format		TWB	adANALYTICS
Tableau_Packaged_Workbook_Fmt	743	684	Tableau Packaged Workbook		TWBX	adANALYTICS

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			format			
Tableau_Extract_Fmt	744	685	Tableau Extract format		TDE	adANALYTICS
Tableau_Data_Source_Fmt	745	686	Tableau Data Source format		TDS	adANALYTICS
Tableau_Packaged_Data_Source_Fmt	746	687	Tableau Packaged Data Source format		TDSX	adANALYTICS
Tableau_Preferences_Fmt	747	688	Tableau Preferences format		TPS	adANALYTICS
Tableau_Map_Source_Fmt	748	689	Tableau Map Source format		TMS	adANALYTICS
ABAP_Fmt	749	690	ABAP Source Code[4]	text/x-abap	ABAP	adSOURCECODE
AMPL_Fmt	750	691	AMPL Source Code[4]		AMPL	adSOURCECODE
APL_Fmt	751	692	APL Source Code[4]		APL	adSOURCECODE
ASN1_Fmt	752	693	ASN.1 Source Code[4]		ASN	adSOURCECODE
ATS_Fmt	753	694	ATS Source Code[4]			adSOURCECODE
Agda_Fmt	754	695	Agda Source Code[4]	text/x-agda	AGDA	adSOURCECODE
Alloy_Fmt	755	696	Alloy Source Code[4]	text/x-alloy	ALS	adSOURCECODE
Apex_Fmt	756	697	Apex Source Code[4]		CLS	adSOURCECODE
Arduino_Fmt	757	698	Arduino Source Code[4]	text/x-arduino	INO	adSOURCECODE
AsciiDoc_Fmt	758	699	AsciiDoc Source Code[4]	text/x-asciidoc	ASC	adSOURCECODE
AspectJ_Fmt	759	700	AspectJ Source Code[4]	text/x-aspectj	AJ	adSOURCECODE
Awk_Fmt	760	701	Awk Source Code[4]	text/x-awk	AWK	adSOURCECODE
BlitzMax_Fmt	761	702	BlitzMax Source Code[4]	text/x-bmx	BMX	adSOURCECODE
Bluespec_Fmt	762	703	Bluespec Source Code[4]		BSV	adSOURCECODE
Brainfuck_Fmt	763	704	Brainfuck Source Code[4]	text/x-brainfuck	B, BF	adSOURCECODE
Brightscript_Fmt	764	705	Brightscript Source Code[4]		BRS	adSOURCECODE
CLIPS_Fmt	765	706	CLIPS Source Code[4]		CLP	adSOURCECODE
CMake_Fmt	766	707	CMake Source Code[4]	text/x-cmake	CMAKE	adSOURCECODE
COBOL_Fmt	767	708	COBOL Source Code[4]	text/x-cobol	CBL, CCP, COB, CPY	adSOURCECODE
CWeb_Fmt	768	709	CWeb Source Code[4]		W	adSOURCECODE
CartoCSS_Fmt	769	710	CartoCSS Source Code[4]		MSS	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Ceylon_Fmt	770	711	Ceylon Source Code[4]	text/x-ceylon	CEYLON	adSOURCECODE
Chapel_Fmt	771	712	Chapel Source Code[4]		CHPL	adSOURCECODE
Clarion_Fmt	772	713	Clarion Source Code[4]		CLW	adSOURCECODE
Clean_Fmt	773	714	Clean Source Code[4]		DCL, ICL	adSOURCECODE
Component_Pascal_Fmt	774	715	Component Pascal Source Code[4]	text/x-component-pascal	CP	adSOURCECODE
Cool_Fmt	775	716	Cool Source Code[4]		CL	adSOURCECODE
Coq_Fmt	776	717	Coq Source Code[4]	text/x-coq	V	adSOURCECODE
Creole_Fmt	777	718	Creole Source Code[4]		CREOLE	adSOURCECODE
Crystal_Fmt	778	719	Crystal Source Code[4]		CR	adSOURCECODE
Csound_Fmt	779	720	Csound Source Code[4]		ORC	adSOURCECODE
Csound_Document_Fmt	780	721	Csound Document Source Code[4]		CSD	adSOURCECODE
Cuda_Fmt	781	722	Cuda Source Code[4]	text/x-cuda	CU	adSOURCECODE
D_Fmt	782	723	D Source Code[4]	text/x-d	DCL, ICL	adSOURCECODE
DIGITAL_Command_Language_Fmt	783	724	DIGITAL Command Language Source Code[4]		COM	adSOURCECODE
DTrace_Fmt	784	725	DTrace Source Code[4]		D	adSOURCECODE
Dart_Fmt	785	726	Dart Source Code[4]	text/x-dart	DART	adSOURCECODE
E_Fmt	786	727	E Source Code[4]		E	adSOURCECODE
ECL_Fmt	787	728	ECL Source Code[4]	application/x-ecl	ECL	adSOURCECODE
Elm_Fmt	788	729	Elm Source Code[4]	text/x-elm	ELM	adSOURCECODE
Emacs_Lisp_Fmt	789	730	Emacs Lisp Source Code[4]	text/x-emacs-lisp	EL	adSOURCECODE
EmberScript_Fmt	790	731	EmberScript Source Code[4]		EM	adSOURCECODE
Fantom_Fmt	791	732	Fantom Source Code[4]	application/x-fantom	FAN	adSOURCECODE
Forth_Fmt	792	733	Forth Source Code[4]	text/x-forth	FOR, FORTH	adSOURCECODE
FreeMarker_Fmt	793	734	FreeMarker Source Code[4]		FTL	adSOURCECODE
Frege_Fmt	794	735	Frege Source Code[4]		FR	adSOURCECODE
G_code_Fmt	795	736	G-code Source Code[4]		G	adSOURCECODE
GAMS_Fmt	796	737	GAMS Source Code[4]		GMS	adSOURCECODE
GAP_Fmt	797	738	GAP Source Code[4]			adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
GDScript_Fmt	798	739	GDScript Source Code[4]		GD	adSOURCECODE
GLSL_Fmt	799	740	GLSL Source Code[4]	text-x-glslsrc	GLSL	adSOURCECODE
Game_Maker_Language_Fmt	800	741	Game Maker Language Source Code[4]		GML	adSOURCECODE
Gnuplot_Fmt	801	742	Gnuplot Source Code[4]	text/x-gnuplot	GNU, GP	adSOURCECODE
Golo_Fmt	802	743	Golo Source Code[4]		GOLO	adSOURCECODE
Gosu_Fmt	803	744	Gosu Source Code[4]	text/x-gosu	GS	adSOURCECODE
Gradle_Fmt	804	745	Gradle Source Code[4]		GRADLE	adSOURCECODE
GraphQL_Fmt	805	746	GraphQL Source Code[4]		GRAPHQL	adSOURCECODE
Graphviz_DOT_Fmt	806	747	Graphviz (DOT) Source Code[4]		DOT	adSOURCECODE
HLSL_Fmt	807	748	HLSL Source Code[4]		HLSL	adSOURCECODE
Hack_Fmt	808	749	Hack Source Code[4]			adSOURCECODE
Haml_Fmt	809	750	Haml Source Code[4]	text/x-haml	HAML	adSOURCECODE
Handlebars_Fmt	810	751	Handlebars Source Code[4]		HBS	adSOURCECODE
Hy_Fmt	811	752	Hy Source Code[4]	text/x-hy	HY	adSOURCECODE
IDL_Fmt	812	753	IDL Source Code[4]	text/x-idl	PRO	adSOURCECODE
IGOR_Pro_Fmt	813	754	IGOR Pro Source Code[4]	text/ipf	IPF	adSOURCECODE
Idris_Fmt	814	755	Idris Source Code[4]	text/x-idris	IDR	adSOURCECODE
Inform_7_Fmt	815	756	Inform 7 Source Code[4]		I7X	adSOURCECODE
Ioke_Fmt	816	757	Ioke Source Code[4]	text/x-iokesrc	IK	adSOURCECODE
Isabelle_Fmt	817	758	Isabelle Source Code[4]	text/x-isabelle		adSOURCECODE
J_Fmt	818	759	J Source Code[4]	text/x-j	IJS	adSOURCECODE
JSONiq_Fmt	819	760	JSONiq Source Code[4]		JQ	adSOURCECODE
JSX_Fmt	820	761	JSX Source Code[4]		JSX	adSOURCECODE
Jasmin_Fmt	821	762	Jasmin Source Code[4]		J	adSOURCECODE
Jolie_Fmt	822	763	Jolie Source Code[4]			adSOURCECODE
Julia_Fmt	823	764	Julia Source Code[4]	text/x-julia	JL	adSOURCECODE
KiCad_Layout_Fmt	824	765	KiCad Layout Source Code[4]			adSOURCECODE
KiCad_Schematic_Fmt	825	766	KiCad Schematic Source Code[4]		SCH	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Kotlin_Fmt	826	767	Kotlin Source Code[4]		KT	adSOURCECODE
LFE_Fmt	827	768	LFE Source Code[4]	text/x-kotlin	LFE	adSOURCECODE
LOLCODE_Fmt	828	769	LOLCODE Source Code[4]		LOL	adSOURCECODE
Lasso_Fmt	829	770	Lasso Source Code[4]	text/x-lasso	LAS, LASSO	adSOURCECODE
Limbo_Fmt	830	771	Limbo Source Code[4]	text/limbo		adSOURCECODE
LiveScript_Fmt	831	772	LiveScript Source Code[4]	text/x-livescript	LS	adSOURCECODE
M_Fmt	832	773	M Source Code[4]		M	adSOURCECODE
MAXScript_Fmt	833	774	MAXScript Source Code[4]		MS	adSOURCECODE
Markdown_Fmt	834	775	Markdown Source Code[4]		MD	adSOURCECODE
Matlab_Fmt	835	463	Matlab Source Code[4]	text/x-matlab	M	adSOURCECODE
Max_Code_Fmt	836	776	Max Source Code[4]		MXT	adSOURCECODE
Mercury_Fmt	837	777	Mercury Source Code[4]			adSOURCECODE
Modelica_Fmt	838	778	Modelica Source Code[4]	text/x-modelica	MO	adSOURCECODE
Modula_2_Fmt	839	779	Modula-2 Source Code[4]	text/x-modula2	MOD	adSOURCECODE
Monkey_Fmt	840	780	Monkey Source Code[4]	text/x-monkey	MONKEY	adSOURCECODE
Moocode_Fmt	841	781	Moocode Source Code[4]	text/x-moocode	MOO	adSOURCECODE
NL_Fmt	842	782	NL Source Code[4]		NL	adSOURCECODE
NSIS_Fmt	843	783	NSIS Source Code[4]	text/x-nsis	NSI	adSOURCECODE
NetLogo_Fmt	844	784	NetLogo Source Code[4]		NLOGO	adSOURCECODE
NewLisp_Fmt	845	785	NewLisp Source Code[4]	text/x-newlisp	NL	adSOURCECODE
Nginx_Fmt	846	786	Nginx Source Code[4]	text/x-nginx-conf	VHOST	adSOURCECODE
Nix_Fmt	847	787	Nix Source Code[4]	text/x-nix	NIX	adSOURCECODE
Nu_Fmt	848	788	Nu Source Code[4]		NU	adSOURCECODE
OCaml_Fmt	849	789	OCaml Source Code[4]	text/x-ocaml		adSOURCECODE
OpenCL_Fmt	850	790	OpenCL Source Code[4]		CL	adSOURCECODE
OpenEdge_ABL_Fmt	851	791	OpenEdge ABL Source Code[4]	text/x-openedge		adSOURCECODE
OpenSCAD_Fmt	852	792	OpenSCAD Source Code[4]		SCAD	adSOURCECODE
Ox_Fmt	853	793	Ox Source Code[4]		OX	adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Oxygene_Fmt	854	794	Oxygene Source Code[4]		OXYGENE	adSOURCECODE
Oz_Fmt	855	795	Oz Source Code[4]		OZ	adSOURCECODE
PAWN_Fmt	856	796	PAWN Source Code[4]	text/x-pawn	PWN	adSOURCECODE
PLpgSQL_Fmt	857	797	PLpgSQL Source Code[4]	text/x-plpgsql	PLSQL	adSOURCECODE
Pan_Fmt	858	798	Pan Source Code[4]		PAN	adSOURCECODE
Parrot_Assembly_Fmt	859	799	Parrot Assembly Source Code[4]		PASM	adSOURCECODE
PicoLisp_Fmt	860	800	PicoLisp Source Code[4]			adSOURCECODE
Pike_Fmt	861	801	Pike Source Code[4]	text/x-pike	PIKE	adSOURCECODE
Pony_Fmt	862	802	Pony Source Code[4]		PONY	adSOURCECODE
Processing_Fmt	863	803	Processing Source Code[4]		PDE	adSOURCECODE
PureBasic_Fmt	864	804	PureBasic Source Code[4]		PB	adSOURCECODE
QMake_Fmt	865	805	QMake File[4]			adSOURCECODE
RAML_Fmt	866	806	RAML Source Code[4]		RAML	adSOURCECODE
RDoc_Fmt	867	807	RDoc Source Code[4]		RDOC	adSOURCECODE
REXX_Fmt	868	808	REXX Source Code[4]	text/x-rexx	REXX	adSOURCECODE
Racket_Fmt	869	809	Racket Source Code[4]	text/x-racket		adSOURCECODE
Ragel_Fmt	870	810	Ragel Source Code[4]			adSOURCECODE
Rascal_Fmt	871	811	Rascal Source Code[4]		RSC	adSOURCECODE
Rebol_Fmt	872	812	Rebol Source Code[4]	text/x-rebol	REB, REBOL	adSOURCECODE
Red_Fmt	873	813	Red Source Code[4]	text/x-red	RED	adSOURCECODE
RenPy_Fmt	874	814	Ren'Py Source Code[4]		RPY	adSOURCECODE
RenderScript_Fmt	875	815	RenderScript Source Code[4]		RS	adSOURCECODE
Ring_Fmt	876	816	Ring Source Code[4]		RING	adSOURCECODE
RobotFramework_Fmt	877	817	RobotFramework Source Code[4]	text/x-robotframework	ROBOT	adSOURCECODE
SAS_Fmt	878	818	SAS Source Code[4]		SAS	adSOURCECODE
SPARQL_Fmt	879	819	SPARQL format[4]	application/sparql-query		adSOURCECODE
SQL_Fmt	880	820	SQL format[4]	text/x-sql		adSOURCECODE
SQLPL_Fmt	881	821	SQLPL Source Code[4]			adSOURCECODE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
SaltStack_Fmt	882	822	SaltStack Source Code[4]		SLS	adSOURCECODE
Scheme_Fmt	883	823	Scheme Source Code[4]	text/x-scheme		adSOURCECODE
Scilab_Fmt	884	824	Scilab Source Code[4]	text/scilab	SCI	adSOURCECODE
Squirrel_Fmt	885	825	Squirrel Source Code[4]		NUT	adSOURCECODE
Stan_Fmt	886	826	Stan Source Code[4]		STAN	adSOURCECODE
Stata_Fmt	887	827	Stata Source Code[4]			adSOURCECODE
Stylus_Fmt	888	828	Stylus Source Code[4]		STYL	adSOURCECODE
SuperCollider_Fmt	889	829	SuperCollider Source Code[4]	text/supercollider	SC	adSOURCECODE
SystemVerilog_Fmt	890	830	SystemVerilog Source Code[4]	text/x-systemverilog	SV	adSOURCECODE
TXL_Fmt	891	831	TXL Source Code[4]		TXL	adSOURCECODE
Turing_Fmt	892	832	Turing Source Code[4]		T	adSOURCECODE
Turtle_Fmt	893	833	Turtle Source Code[4]	text/turtle	TTL	adSOURCECODE
UrWeb_Fmt	894	834	UrWeb Source Code[4]		UR, URS	adSOURCECODE
Vim_script_Fmt	895	835	Vim script File[4]	text/x-vim	VIM	adSOURCECODE
Visual_Basic_Fmt	896	836	Visual Basic Source Code[4]	text/x-vbasic	VB	adSOURCECODE
WebAssembly_Fmt	897	837	WebAssembly Source Code[4]		WAT	adSOURCECODE
WebIDL_Fmt	898	838	WebIDL Source Code[4]		WEBIDL	adSOURCECODE
X10_Fmt	899	839	X10 Source Code[4]	text/x-x10	X10	adSOURCECODE
XQuery_Fmt	900	840	XQuery Source Code[4]	text/xquery	XQM	adSOURCECODE
Xojo_Fmt	901	841	Xojo Source Code[4]			adSOURCECODE
Xtend_Fmt	902	842	Xtend Source Code[4]	text/x-xtend	XTEND	adSOURCECODE
YANG_Fmt	903	843	YANG Source Code[4]		YANG	adSOURCECODE
Zephir_Fmt	904	844	Zephir Source Code[4]		ZEP	adSOURCECODE
eC_Fmt	905	845	eC Source Code[4]	text/x-ecsrc	EC	adSOURCECODE
reStructuredText_Fmt	906	846	reStructuredText Source Code[4]	text/x-rst		adSOURCECODE
xBase_Fmt	907	847	xBase Source Code[4]			adSOURCECODE
Windows_Installer_Fmt	908	848	MSI Windows Installer format	application/x-ole-storage	MSI	adENCAPSULATION
Autodesk_3ds_Max_Fmt	909	849	Autodesk 3ds Max format		MAX	adCAD

Format Name	Number	Category	Description	MIME Type	Extension	File Class
PhotoDraw_Mix_Fmt	910	850	PhotoDraw MIX image	image/vnd.mix	MIX	adRASTERIMAGE
Softimage_SCN_Fmt	911	851	Softimage Scene SCN format		SCN	adCAD
Parasolid_XT_Fmt	912	852	Parasolid ascii XT format		X_T	adCAD
Parasolid_XB_Fmt	913	853	Parasolid binary XB format		X_B	adCAD
IGES_Fmt	914	854	Initial Graphics Exchange Specification format	model/iges	IGS	adCAD
ACE_Archive_Fmt	915	855	ACE archive format	application/x-ace-compressed	ACE	adENCAPSULATION
Grasshopper_GHX_Fmt	916	856	Grasshopper GHX format		GHX	adCAD
MS_FrontPage_Macro_Fmt	917	857	Microsoft FrontPage macro file format		FPM	adWORDPROCESSOR
MS_AtWork_Fax_Fmt	918	858	Microsoft AtWork Fax format		AWD	adFAXFORMAT
MS_Image_Composer_Fmt	919	859	Microsoft Image Composer format		MIC	adRASTERIMAGE
MS_Visual_InterDev_Fmt	920	860	Microsoft Visual InterDev web project items file		WDM	adMISC
Macromedia_Flash_FLA_OLE_Fmt	921	861	Macromedia Flash FLA Project File OLE format		FLA	adWORDPROCESSOR
Corel_Draw_X4_Fmt	922	862	CorelDRAW version X4 onwards	application/x-vnd.corel.zcf.draw.document+zip	CDRX	adVECTORGRAPHIC
Ogg_Daala_Fmt	923	863	Ogg Daala video format	video/daala	OGV	adMOVIE
Ogg_BBC_Dirac_Fmt	924	864	Ogg BBC Dirac video format	video/x-dirac	OGV	adMOVIE
PKCS_7_Fmt	925	865	PKCS #7 cryptographic format	application/pkcs7-signature	P7S	adWORDPROCESSOR
Time_Stamped_Data_Fmt	926	866	Time-stamped data format	application/timestamped-data	TSD	adENCAPSULATION
Sereal_Fmt	927	867	Sereal data serialization format	application/sereal	SRL	adMISC
Associated_Signature_Simple_Fmt	928	868	Associated Signature Container Simple format	application/vnd.etsi.asic-s+zip	ASICS	adENCAPSULATION
Associated_Signature_Extended_Fmt	929	869	Associated Signature Container Extended format	application/vnd.etsi.asic-e+zip	ASICE	adENCAPSULATION
iBooks_Fmt	930	870	Apple iBooks format	application/x-ibooks+zip	IBOOKS	adWORDPROCESSOR
PDF_Forms_Data_Fmt	931	871	PDF Forms Data Format	application/vnd.fdf	FDF	adWORDPROCESSOR
PDF_XML_Forms_Data_Fmt	932	872	PDF XML Forms Data Format	application/vnd.adobe.xfdf	XFDF	adWORDPROCESSOR
AxCrypt_Fmt	933	873	AxCrypt encrypted document	application/x-axcrypt	AXX	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Unix_Archive_Fmt	934	874	Unix Archive ar format	application/x-archive	AR	adENCAPSULATION
Berkeley_Btree_Database_Fmt	935	875	Berkeley DB btree database format	application/x-berkeley-db	DB	adDATABASE
Berkeley_Hash_Database_Fmt	936	876	Berkeley DB hash database format	application/x-berkeley-db	DB	adDATABASE
Berkeley_Log_Database_Fmt	937	877	Berkeley DB log database format	application/x-berkeley-db		adDATABASE
Berkeley_Queue_Database_Fmt	938	878	Berkeley DB queue database format	application/x-berkeley-db		adDATABASE
BitTorrent_Fmt	939	879	BitTorrent file format	application/x-bittorrent	TORRENT	adMISC
Chrome_Extension_Fmt	940	880	Google Chrome Extension format	application/x-chrome-package	CRX	adENCAPSULATION
Dalvik_Executable_Fmt	941	881	Dalvik Executable dex format	application/x-dex	DEX	adEXECUTABLE
Foxmail_Fmt	942	882	Foxmail email format	application/x-foxmail	BOX	adWORDPROCESSOR
GRIB_Fmt	943	883	General Regularly-distributed Information in Binary form GRIB format	application/x-grib	GRB, GRIB2	adMISC
Zstandard_Fmt	944	884	Zstandard compression format	application/zstd	ZSTD	adENCAPSULATION
LZ4_Fmt	945	885	LZ4 compressed file	application/x-lz4	LZ4	adENCAPSULATION
MS_Money_Fmt	946	886	Microsoft Money format	application/x-msmoney	MNY	adSPREADSHEET
NetCDF_Fmt	947	887	Network Common Data Form NetCDF format	application/x-netcdf	NC	adMISC
SAS6_Data_Fmt	948	888	SAS 6 Data storage format	application/x-sas-data-v6	SD2	adDATABASE
SAS_Transport_Fmt	949	889	SAS Transport File XPORT format	application/x-sas-xport	XPT, XPORT	adDATABASE
Snappy_Framed_Fmt	950	890	Snappy Framed compression format	application/x-snappy-framed	SZ	adENCAPSULATION
Stata_Data_Fmt	951	891	Stata Data Format	application/x-stata-dta	DTA	adDATABASE
SPSS_SAV_Fmt	952	892	SPSS Statistics Data File Format		SAV	adDATABASE
Zoo_Archive_Fmt	953	893	Zoo Compressed Archive Format	application/x-zoo	ZOO	adENCAPSULATION
CDX_Fmt	954	894	ChemDraw CDX format	chemical/x-cdx	CDX	adMISC
CDXML_Fmt	955	895	ChemDraw CDXML format	application/vnd.chemdraw+xml	CDXML	adMISC
BPG_Fmt	956	896	Better Portable Graphics BPG format	image/x-bpg	BPG	adRASTERIMAGE
Apple_Icon_Fmt	957	897	Apple Icon image format	image/icns	ICNS	adRASTERIMAGE
NITF_Fmt	958	898	National Imagery Transmission	image/nitf	NTF, NITF	adRASTERIMAGE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			Format NITF image			
ERDAS_Imagine_Fmt	959	899	ERDAS Imagine image format	application/x-erdas-hfa	HFA	adRASTERIMAGE
MS_Office_Temporary_Owner_Fmt	960	900	Microsoft Office temporary owner file	application/x-ms-owner		adMISC
EAC3_Audio_Fmt	961	901	Enhanced-AC3 (EAC3) Audio File format	audio/eac3	AC3	adSOUND
COFF_Relocatable_Fmt	962	902	Common Object File Format (COFF) relocatable object	application/x-object-file	O	adOBJECTMODULE
COFF_Executable_Fmt	963	903	Common Object File Format (COFF) executable	application/x-executable-file		adEXECUTABLE
COFF_Dynamic_Lib_Fmt	964	904	Common Object File Format (COFF) dynamic library	application/x-library-file		adLIBRARY
ELF_Core_Fmt	965	905	ELF Core file			adMISC
Purify_Fmt	966	906	Rational Purify data file		PFY	adMISC
Kryptel_Fmt	967	907	Kryptel encrypted file		EDC	adENCAPSULATION
Windows_Core_Dump_Fmt	968	908	Windows heap or mini core dump file	application/x-dmp	DMP	adMISC
Qt_Prerendered_Font_Fmt	969	909	Qt Prerendered Font format		QPF2	adFONT
AIX_Relocatable_Fmt	970	910	AIX/RISC COFF relocatable object	application/x-object-file		adOBJECTMODULE
AIX_Executable_Fmt	971	911	AIX/RISC COFF executable	application/x-executable-file		adEXECUTABLE
AIX_Dynamic_Lib_Fmt	972	912	AIX/RISC COFF dynamic library	application/x-library-file	A	adLIBRARY
HPUX_Relocatable_Fmt	973	913	HPUX/PA-RISC COFF relocatable object	application/x-object-file		adOBJECTMODULE
HPUX_Executable_Fmt	974	914	HPUX/PA-RISC COFF executable	application/x-executable-file		adEXECUTABLE
HPUX_Dynamic_Lib_Fmt	975	915	HPUX/PA-RISC COFF dynamic library	application/x-library-file	SL	adLIBRARY
XML_EBCDIC_Fmt	976	916	EBCDIC-encoded XML file	application/xml	XML	adWORDPROCESSOR
MPEG_JVT_H264_Fmt	977	917	MPEG JVT-NAL sequence H264 video	video/h264	264	adMOVIE
Material_Exchange_Fmt	978	918	Material Exchange Format audio-video container format	application/mxf	MXF	adMOVIE
MS_Agent_Character_Fmt	979	919	Microsoft Agent Character file		ACS	adMOVIE

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Quicken_Fmt	980	920	Quicken data file		QDF	adMISC
MS_Outlook_Address_Fmt	981	921	Microsoft Outlook address file		WAB	adMISC
MS_Answer_Wizard_Fmt	982	922	Microsoft Answer Wizard file			adMISC
ADX_Fmt	983	923	ADX audio file		ADX	adSOUND
System_Deployment_Image_Fmt	984	924	Microsoft System Deployment Image SDI format		SDI	adMISC
Free_Lossless_Image_Fmt	985	925	Free Lossless Image Format (FLIF)	image/flif	FLIF	adRASTERIMAGE
DPX_Fmt	986	926	Digital Picture Exchange (DPX) image format	image/dpx	DPX	adRASTERIMAGE
Avro_Fmt	987	927	Apache Avro binary format		AVRO	adMISC
InstallShield_Archive_Fmt	988	928	InstallShield archive (early versions) format		EX_	adENCAPSULATION
Mac_Executable_Fmt	989	929	Mac OS-X (Mach-O) executable format			adEXECUTABLE
GDSII_Fmt	990	930	GDSII data format		GDS	adMISC
ActiveMime_Fmt	991	931	Microsoft ActiveMime (mso) documents	application/x-mso	MSO	adMISC
SmartCharts_Fmt	992	932	BizInt SmartCharts data format		CHP, CHRR	adMISC
Webex_ARF_Fmt	993	933	Webex advanced network ARF recordings		ARF	adMOVIE
Webex_WRF_Fmt	994	934	Webex local WRF recordings		WRF	adMOVIE
PGP_NetShare_Fmt	995	935	Symantec PGP NetShare encrypted file			adENCAPSULATION
Ability_WP_OLE_Fmt	996	936	Ability Write later versions format		AWW	adWORDPROCESSOR
Ability_SS_OLE_Fmt	997	937	Ability Spreadsheet later versions format		AWS	adSPREADSHEET
InDesign_IDML_Fmt	998	938	Adobe InDesign IDML format	application/vnd.adobe.indesign-idml-package	IDML	adDESKTOPPUBLSH
Executable_JAR_Fmt	999	939	Executable Java Archive (jar) file	application/java-archive	JAR	adENCAPSULATION
IDOL_IDX_Fmt	1000	940	IDOL Server IDX file		IDX	adENCAPSULATION
Android_Package_Kit_Fmt	1001	941	Android Package Kit (APK) format	application/vnd.android.package-archive	APK	adEXECUTABLE
Android_Binary_XML_Fmt	1002	942	Android Binary XML (compressed	application/xml	XML	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
			by aapt) format			
Java_WAR_Fmt	1003	943	Java WAR file format		WAR	adENCAPSULATION
Java_EAR_Fmt	1004	944	Java EAR file format		EAR	adENCAPSULATION
Atom_Syndication_Fmt	1005	945	Atom Syndication Format	application/atom+xml	ATOM	adWORDPROCESSOR
RSS_Fmt	1006	946	RSS syndication XML format	application/rss+xml	RSS	adWORDPROCESSOR
SMIL_Fmt	1007	947	Synchonized Multimedia Integration Language (SMIL) XML format	application/smil+xml	SMIL	adWORDPROCESSOR
XSLT_Fmt	1008	948	Extensible Stylesheet Language Transformations (XSLT) format	application/xslt+xml	XSL, XSLT	adWORDPROCESSOR
XML_Shareable_Playlist_Fmt	1009	949	XML Shareable Playlist Format (XSPF)	application/xspf+xml	XSPF	adWORDPROCESSOR
FictionBook_Fmt	1010	950	FictionBook e-book XML format	application/x-fictionbook+xml	FB2	adWORDPROCESSOR
Adobe_Premiere_Project_Fmt	1011	951	Adobe Premiere project format	image/vnd.adobe.premiere	PPJ	adMISC
RDF_XML_Fmt	1012	952	RDF/XML format	application/rdf+xml	RDF	adWORDPROCESSOR
Really_Simple_Discovery_Fmt	1013	953	Really Simple Discovery (RSD) XML format	application/rsd+xml	RSD	adWORDPROCESSOR
SBML_Fmt	1014	954	Systems Biology Markul Language (SBML) XML format	application/sbml+xml	SBML	adWORDPROCESSOR
SRU_Fmt	1015	955	Search/Retrieve via URL (SRU) XML format	application/sru+xml	SRU	adWORDPROCESSOR
SSML_Fmt	1016	956	Speech Synthesis Markup Language (SSML) XML format	application/ssml+xml	SSML	adWORDPROCESSOR
PLS_Fmt	1017	957	Pronunciation Lexicon Specification (PLS) XML format	application/pls+xml	PLS	adWORDPROCESSOR
TEI_Fmt	1018	958	Text Encoding Initiative (TEI) XML format	application/tei+xml	TEI	adWORDPROCESSOR
METS_Fmt	1019	959	Metadata Encoding and Transmission Standard (METS) XML format	application/mets+xml	METS	adWORDPROCESSOR
MODS_Fmt	1020	960	Metadata Object Description Schema (MODS) XML format	application/mods+xml	MODS	adWORDPROCESSOR
Metalink_Fmt	1021	961	Metalink XML format	application/metalink4+xml	METALINK	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
Open_eBook_Fmt	1022	962	Open eBook (OEBPS) XML format	application/oebps-package+xml	OPF	adWORDPROCESSOR
SRGS_Fmt	1023	963	Speech Recognition Grammar Specification (SRGS) XML format	application/srgs+xml	SRGS	adWORDPROCESSOR
SPARQL_Results_Fmt	1024	964	SPARQL Query Results XML format	application/sparql-results+xml	SRX	adWORDPROCESSOR
Adobe_XML_Data_Package_Fmt	1025	965	Adobe XML Data Package format	application/vnd.adobe.xdp+xml	XDP	adWORDPROCESSOR
ESzigno_Fmt	1026	966	e-Szigno signed xml document	application/vnd.eszigno3+xml	ES3	adWORDPROCESSOR
Mozilla_XUL_Fmt	1027	967	Mozilla XML User Interface Language (XUL) XML format	application/vnd.mozilla.xul+xml	XUL	adWORDPROCESSOR
SyncML_Fmt	1028	968	Synchronization Markup Language (SyncML) XML format	application/vnd.syncml+xml	XML	adWORDPROCESSOR
VoiceXML_Fmt	1029	969	VoiceXML (VXML) XML format	application/voicexml+xml	VXML	adWORDPROCESSOR
TI_Target_Configuration_Fmt	1030	970	Texas Instruments CCXML target configuration XML format		CCXML	adWORDPROCESSOR
LZFSE_Fmt	1031	971	Lempel-Ziv Finite State Entropy (LZFSE) compression format		LZFSE	adENCAPSULATION
Kindle_eBook_Fmt	1032	972	Amazon Kindle or Mobipocket eBook format	application/vnd.amazon.ebook	AZW, PRC	adWORDPROCESSOR
Oasis_Stream_Fmt	1033	973	Open Artwork System Interchange Standard (OASIS) format		OAS	adMISC
Amazon_KFX_Fmt	1034	974	Amazon KFX eBook format		KFX	adWORDPROCESSOR
KTX_Fmt	1035	975	KTX image format	image/ktx	KTX	adRASTERIMAGE
GMSH_Mesh_Fmt	1036	976	GMSH Mesh polygon format	model/mesh	MSH	adCAD
Collada_DAE_Fmt	1037	977	Collada Digital Asset Exchange (DAE) format	model/vnd.collada+xml	DAE	adCAD
YIN_Fmt	1038	978	YIN XML format	application/yin+xml	YIN	adWORDPROCESSOR
MPEG_Playlist_Fmt	1039	979	MPEG audio playlist format	audio/mpegurl	M3U	adSOUND
Windows_Audio_Playlist_Fmt	1040	980	Windows Audio playlist format	audio/x-ms-wax	WAX	adSOUND
DTS_Audio_Fmt	1041	981	DTS Coherent Acoustics audio format	audio/vnd.dts	DTS	adSOUND
Chemical_Markup_Language_Fmt	1042	982	Chemical Markup Language (CML) XML format	chemical/x-cml	CML	adWORDPROCESSOR

Format Name	Number	Category	Description	MIME Type	Extension	File Class
CrystalMaker_Fmt	1043	983	CrystalMaker chemical format	chemical/x-cmdf	CMDF	adMISC
VTK_XML_Fmt	1044	984	Visualization Toolkit VTK XML format	model/vnd.vtu	VTU	adVECTORGRAPHIC
IPFIX_Fmt	1045	985	IP Flow Information Export (IPFIX) format	application/ipfix	IPFIX	adMISC
Portable_Font_Resource_Fmt	1046	986	Portable Font Resource font format	application/font-tdpfr	PFR	adFONT
MARC_Fmt	1047	987	Machine-Readable Cataloging (MARC21) format	application/marc	MARC	adDATABASE
MARC_XML_Fmt	1048	988	Machine-Readable Cataloging (MARC) XML format	application/marcxml+xml	XML	adWORDPROCESSOR
XAR_Fmt	1049	989	Extensible Archive (XAR) format			adENCAPSULATION
Symbian_Installer_Fmt	1050	990	Symbian installer format	application/vnd.symbian.install	SIS	adENCAPSULATION
SO_Drawing_XML_Fmt	1051	316	OpenDocument format (OpenOffice 1/StarOffice 6.7) Drawing XML	application/vnd.sun.xml.draw	SXD	adVECTORGRAPHIC
SO_Text_Global_XML_Fmt	1052	991	OpenDocument format (OpenOffice 1/StarOffice 6.7) Writer Master document XML	application/vnd.sun.xml.writer.global	SXG	adWORDPROCESSOR
ODF_Chart_Fmt	1053	992	ODF Chart	application/vnd.oasis.opendocument.chart	ODC	adVECTORGRAPHIC
ODF_Database_Fmt	1054	993	ODF Database	application/vnd.sun.xml.base	ODB	adDATABASE
ODF_Image_Fmt	1055	994	ODF Image	application/vnd.oasis.opendocument.image	ODI	adRASTERIMAGE
ODF_Text_Master_Fmt	1056	995	ODF Text Master	application/vnd.oasis.opendocument.text-master	ODM	adWORDPROCESSOR
ODF_Text_Web_Fmt	1057	996	ODF Text Web	application/vnd.oasis.opendocument.text-web	OTH	adWORDPROCESSOR
ODF_Chart_Template_Fmt	1058	997	ODF Chart Template	application/vnd.oasis.opendocument.chart-template	OTC	adVECTORGRAPHIC
ODF_Formula_Template_Fmt	1059	998	ODF Formula Template	application/vnd.oasis.opendocument.formula-template	OTF	adWORDPROCESSOR
ODF_Drawing_Template_Fmt	1060	316	ODF Drawing/Graphics Template	application/vnd.oasis.opendocument.graphics-template	OTG	adVECTORGRAPHIC
ODF_Image_Template_Fmt	1061	999	ODF Image Template	application/vnd.oasis.opendocument.image-template	OTI	adRASTERIMAGE
ODF_Presentation_Template_Fmt	1062	316	ODF Presentation Template	application/vnd.oasis.opendocument.presentation-template	OTP	adPRESENTATION
ODF_Spreadsheet_Template_Fmt	1063	315	ODF Spreadsheet Template	application/vnd.oasis.opendocument.spreadsheet-template	OTS	adSPREADSHEET

Format Name	Number	Category	Description	MIME Type	Extension	File Class
ODF_Text_Template_Fmt	1064	314	ODF Text Template	application/vnd.oasis.opendocument.text-template	OTT	adWORDPROCESSOR
ODF_Chart_XML_Fmt	1065	1000	ODF Chart flat XML format	application/vnd.oasis.opendocument.chart.xml	FODC	adVECTORGRAPHIC
ODF_Drawing_XML_Fmt	1066	1001	ODF Drawing/Graphics flat XML format	application/vnd.oasis.opendocument.formula.xml	FODG	adWORDPROCESSOR
ODF_Formula_XML_Fmt	1067	1002	ODF Formula flat XML format	application/vnd.oasis.opendocument.graphics.xml	FODF	adVECTORGRAPHIC
ODF_Image_XML_Fmt	1068	1003	ODF Image flat XML format	application/vnd.oasis.opendocument.image.xml	FODI	adRASTERIMAGE
ODF_Presentation_XML_Fmt	1069	1004	ODF Presentation flat XML format	application/vnd.oasis.opendocument.presentation.xml	FODP	adPRESENTATION
ODF_Spreadsheet_XML_Fmt	1070	1005	ODF Spreadsheet flat XML format	application/vnd.oasis.opendocument.spreadsheet.xml	FODS	adSPREADSHEET
ODF_Text_XML_Fmt	1071	1006	ODF Text flat XML format	application/vnd.oasis.opendocument.text.xml	FODT	adWORDPROCESSOR
ODF_Extension_Fmt	1072	1007	ODF Extension format	application/vnd.openofficeorg.extension	OXT	adMISC
StarView_Metafile_Fmt	1073	1008	OpenOffice StarView MetaFile format	image/x-svm	SVM	adRASTERIMAGE
BBeB_LRF_eBook_Fmt	1074	1009	Broad Band eBook (BBeB) in LRF format		LRF	adWORDPROCESSOR
GPG_Trust_DB_Fmt	1075	1010	GPG trust database format		GPG	adMISC
VICE_Emulator_Fmt	1076	1011	VICE (Versatile Commodore Emulator) format		VSF	adMISC
Portable_Game_Notation_Fmt	1077	1012	Portable Game Notation chess format	application/vnd.chess-pgn	PGN	adWORDPROCESSOR
Doom_WAD_Fmt	1078	1013	Doom IWAD/PWAD format	application/x-doom	WAD	adMISC
Device_Tree_Blob_Fmt	1079	1014	Linux Device Tree Blob format		DTB	adMISC
BDF_Font_Fmt	1080	1015	Glyph Bitmap Distribution Format	application/x-font-bdf	BDF	adFONT
PC_Screen_Font_Fmt	1081	1016	PC Screen Font format	application/x-font-psf	PSF	adFONT
JNLP_Fmt	1082	1017	Java Network Launching Protocol	application/x-java-jnlp-file	JNLP	adWORDPROCESSOR
XAML_Browser_Application_Fmt	1083	1018	XAML Browser Application (XBAP) format	application/x-ms-xbap	XBAP	adWORDPROCESSOR
MS_Binder_Fmt	1084	1019	Microsoft Office Binder format	application/x-msbinder	OBP	adENCAPSULATION
XAP_Fmt	1085	1020	Microsoft Silverlight application (XAP) format	application/x-silverlight-app	XAP	adENCAPSULATION
StuffIt_X_Fmt	1086	1021	StuffIt X (SITX) archive format	application/x-stuffitx	SITX	adENCAPSULATION
FIG_Fmt	1087	1022	Facility for Interactive Generation of figures (FIG) image format	application/x-xfig	FIG	adVECTORGRAPHIC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
XPInstall_Fmt	1088	1023	XPInstall Cross-Platform Installer Module (XPI) format	application/x-xpinstall	XPI	adENCAPSULATION
XDF_Fmt	1089	1024	Extensible Data Format (XDF) XML format		XDF	adWORDPROCESSOR
MXML_Fmt	1090	1025	MXML UI markup language XML format		MXML	adWORDPROCESSOR
MusicXML_Fmt	1091	1026	MusicXML format	application/vnd.recordare.musicxml	MXL	adENCAPSULATION
Finale_Fmt	1092	1027	Finale audio format		MUS	adSOUND
Spotfire_DXP_Fmt	1093	1028	TIBCO Spotfire DXP data format	application/vnd.spotfire.dxp	DXP	adANALYTICS
MS_Office_Theme_2007_Fmt	1094	1029	Microsoft Office theme format	application/vnd.ms-officetheme	THMX	adMISC
Adobe_AIR_Installer_Fmt	1095	1030	Adobe AIR application installer package	application/vnd.adobe.air-application-installer-package+zip	AIR	adENCAPSULATION
Flex_Project_Fmt	1096	1031	Adobe Flash Flex project file format	application/vnd.adobe.fxp	FXP	adENCAPSULATION
FoxPro_Fmt	1097	1032	FoxPro compiled source format		FXP	adLIBRARY
VST_Preset_Fmt	1098	1033	Virtual Studio Technology (VST) preset format		FXP	adSOUND
Mischief_Image_Fmt	1099	1034	Mischief vector graphics image format		ART	adVECTORGRAPHIC
FreeArc_Fmt	1100	1035	FreeArc archive format	application/x-freearc	ARC	adENCAPSULATION
Autodesk_3ds_Fmt	1101	1036	Autodesk 3ds format	application/x-3ds	3DS	adCAD
Monkeys_Audio_Fmt	1102	1037	Monkey's Audio format		APE	adSOUND
CALS_Fmt	1103	1038	CALS raster image format		CAL	adRASTERIMAGE
Dr_Halo_PAL_Fmt	1104	1039	Dr Halo raster image PAL file format		PAL	adRASTERIMAGE
DPG_Fmt	1105	1040	Nintendo DS DPG video format		DPG	adMOVIE
JPEG_XR_Fmt	1106	1041	JPEG XR (extended range) image format	image/vnd.ms-photo	JXR, HDP	adRASTERIMAGE
TCR_eBook_Fmt	1107	1042	TCR (Text Compression for Reader) eBook format		TCR	adWORDPROCESSOR
IHEX_Fmt	1108	1043	Intel Hex format		IHEX	adENCAPSULATION
QCOW_Fmt	1109	1044	QEMU Copy On Write		QCOW	adENCAPSULATION
VDI_Fmt	1110	1045	VirtualBox Disk Image		VDI	adENCAPSULATION

Format Name	Number	Category	Description	MIME Type	Extension	File Class
OneNote_Alternate_Fmt	1111	1046	OneNote Alternative Packaging Format			adPRESENTATION
RMS_Protected_Fmt	1112	1047	Rights Management Services (RMS)-protected format		PFILE, PPDF, PJPG, PTXT	adENCAPSULATION
Portfolio_PDF_Fmt	1113	1048	Portfolio PDF File	application/pdf	PDF	adWORDPROCESSOR
Crystal_Reports_Fmt	1114	1049	SAP Crystal Reports format	application/x-rpt	RPT	adANALYTICS
Thumbs_db_Fmt	1115	1050	Microsoft Windows thumbs.db format		DB	adENCAPSULATION
PagePlus_Fmt	1116	1051	Serif PagePlus format		PPP	adDESKTOPPUBLSH
MS_Project_Exchange_Fmt	1117	1052	Microsoft Project Exchange format		MPX	adSCHEDULE
MS_Management_Pack_MPX_Fmt	1118	1053	Microsoft Systems Center Operation Manager (SCOM) management pack MPX format		MPX	adMISC
AutoCAD_VBA_Project_Fmt	1119	1054	AutoCAD VBA project format		DVB	adMISC
PLY_ASCII_Fmt	1120	1055	Polygon File Format (PLY) ASCII format		PLY	adCAD
PLY_Binary_Fmt	1121	1056	Polygon File Format (PLY) binary format		PLY	adCAD
JavaView_JVX_Fmt	1122	1057	JavaView XML (JVX) format		JVX	adCAD
X3D_Fmt	1123	1058	Extensible 3d Graphics (X3D) XML format	model/x3d+xml	X3D	adCAD
ZBrush_Project_Fmt	1124	1059	ZBrush ZProject (ZPR) format		ZPR	adCAD
ZBrush_Tool_Fmt	1125	1060	ZBrush ZTtool (ZTL) format		ZTL	adCAD
Windows_Installer_Patch_Fmt	1126	1061	Microsoft Windows Installer Patch Package (MSP) format		MSP	adENCAPSULATION
Windows_Installer_Transform_Fmt	1127	1062	Microsoft Windows Installer Transform (MST) format		MST	adENCAPSULATION
Lotus_Approach_Fmt	1128	1063	Lotus Approach format	application/vnd.lotus-approach	APR, MPR	adDATABASE
Outlook_SendRcv_Settings_Fmt	1129	1064	Microsoft Outlook 2002 Send-Receive Settings		SRS	adMISC
MS_Publisher_Scheme_Fmt	1130	1065	Microsoft Publisher colour scheme		SCM	adMISC
SO_Chart_Fmt	1131	1066	Star Office 4,5 Chart	application/vnd.stardivision.chart	SDS	adVECTORGRAPHIC

Format Name	Number	Category	Description	MIME Type	Extension	File Class
SO_Database_Fmt	1132	1067	Star Office 4,5 Database	application/vnd.stardivision.base	SDB	adDATABASE
SO_Library_Fmt	1133	1068	Star Office 4,5 Library		SBL	adLIBRARY
PageMaker_Document_Fmt	1134	1069	Adobe PageMaker document	application/pagemaker	PMD	adDESKTOPPUBLSH
MS_DTS_Fmt	1135	1070	Microsoft Data Transformation Services (DTS) package file		DTS	adMISC
Cognos_PowerPlay_PPR_Fmt	1136	1071	Cognos PowerPlay up to version 7 (PPR) format		PPR	adANALYTICS
Visual_Studio_SUO_Fmt	1137	1072	Microsoft Visual Studio solution user options (suo) file		SUO	adMISC
MS_GraphEdit_Fmt	1138	1073	Microsoft GraphEdit File format		GRF	adMISC
ArcGIS_Graph_Fmt	1139	1074	ArcGIS Graph format		GRF	adMISC
SID_Audio_Fmt	1140	1075	SID Audio format	audio/prs.sid	SID	adSOUND
MrSID_Fmt	1141	1076	LizardTech MrSID image format	image/x-mrsid	SID	adRASTERIMAGE
Cardfile_Fmt	1142	1077	Microsoft Windows Cardfile address book format	application/x-mscardfile	CRD	adWORDPROCESSOR
MS_Word_Mac_4_Fmt	1143	205	Microsoft Word for Macintosh (version 4,5)	application/msword	DOC	adWORDPROCESSOR
WordPerfect_5_Fmt	1144	80	WordPerfect (version 5)	application/x-corel-wordperfect	WOP, DOC	adWORDPROCESSOR
WordPerfect_6_Fmt	1145	178	WordPerfect (version 6 and higher)	application/x-corel-wordperfect	WPD	adWORDPROCESSOR
WordPerfect_Graphics_1_Fmt	1146	85	WordPerfect Graphics (version 1)	application/vnd.wordperfect	WPG, QPG	AutoDetNoFormat

[1]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.

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

[3]This format is returned only if you enable source code identification. See Source Code Identification, on page 115.

[4]This format is returned only if you enable extended source code identification. See Source Code Identification, on page 115.

Appendix C: Character Sets

This section provides information on the handling of character sets in the KeyView suite of products, which includes KeyView Filter SDK, KeyView Export SDK, and KeyView Viewing SDK.

Multibyte and Bidirectional Support

The KeyView SDKs can process files that contain multibyte characters. A multibyte character encoding represents a single character with consecutive bytes. KeyView can also process text from files that contain bidirectional text. Bidirectional text contains both Latin-based text which is read from left to right, and text that is read from right to left (Hebrew and Arabic).

The following table indicates which character encodings are supported by KeyView for each format.

Multibyte and bidirectional support

Format	Single-byte	Multibyte	Bidirectional
Archive			
7-Zip (7Z)	n/a	n/a	n/a
AD1 Evidence file	n/a	n/a	n/a
ADJ	n/a	n/a	n/a
B1	n/a	n/a	n/a
BinHex (HQX)	n/a	n/a	n/a
Bzip2 (BZ2)	n/a	n/a	n/a
EnCase – Expert Witness Compression Format (E01)	n/a	n/a	n/a
GZIP (GZ)	n/a	n/a	n/a
ISO (ISO)	n/a	n/a	n/a
Java Archive (JAR)	n/a	n/a	n/a
Legato EMailXtender Archive (EMX)	n/a	n/a	n/a
MacBinary (BIN)	n/a	n/a	n/a
Mac Disk Copy Disk Image (DMG)	n/a	n/a	n/a
Microsoft Backup File (BKF)	n/a	n/a	n/a

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Microsoft Cabinet format (CAB)	n/a	n/a	n/a
Microsoft Compiled HTML Help (CHM)	n/a	n/a	n/a
Microsoft Compressed Folder (LZH)	n/a	n/a	n/a
PKZip (ZIP)	n/a	n/a	n/a
Microsoft Outlook DBX (DBX)	Y	Y	Y
Microsoft Outlook Offline Storage File (OST)	Y	Y	Y
RAR Archive (RAR)	n/a	n/a	n/a
Tape Archive (TAR)	n/a	n/a	n/a
UNIX Compress (Z)	n/a	n/a	n/a
UUEncoding (UUE)	n/a	n/a	n/a
Windows Scrap File (SHS)	n/a	n/a	n/a
WinZip (ZIP)	n/a	n/a	n/a
Binary			
Executable (EXE)	n/a	n/a	n/a
Link Library (DLL)	n/a	n/a	n/a
Computer-aided Design			
AutoCAD Drawing (DWG)	Y	Y	Y
AutoCAD Drawing Exchange (DXF)	Y	Y	Y
CATIA formats (CAT)	Y	N	N
Microsoft Visio (VSD)	Y	Y	Y
Database			
dBase Database	Y	N	N
Microsoft Access (MDB)	Y	Y	N
Microsoft Project (MPP)	Y	Y	N
Desktop Publishing			
Microsoft Publisher	N	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Display			
Adobe Portable Document Format (PDF) (basic reader)	Y	Y[1]	Y
Adobe Portable Document Format (PDF) (graphic-based reader)	Y	Y1	Y
Graphics			
Computer Graphics Metafile (CGM)	Y	N	N
Corel DRAW (CDR)	n/a	n/a	n/a
DCX Fax System (DCX)	Y	N	N
DICOM – Digital Imaging and Communications in Medicine (DCM)	n/a	n/a	n/a
Encapsulated PostScript (EPS)	Y	N	N
Enhanced Metafile (EMF)	Y	Y	N
Graphic Interchange Format (GIF)	n/a	n/a	n/a
JBIG2	n/a	n/a	n/a
JPEG	n/a	n/a	n/a
JPEG 2000	n/a	n/a	n/a
Lotus AMIDraw Graphics (SDW)	n/a	n/a	n/a
Lotus Pic (PIC)	n/a	n/a	n/a
Macintosh Raster (PICT/PCT)	n/a	n/a	n/a
MacPaint (PNTG)	n/a	n/a	n/a
Microsoft Office Drawing (MSO)	n/a	n/a	n/a

[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.

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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[1]
Lotus 1-2-3 (123)	Y	Y	Y
Lotus 1-2-3 (WK4)	Y	Y	N
Lotus 123 Charts (123)	Y	Y	N
Microsoft Excel Charts (XLS)	Y	Y	N
Microsoft Excel Macintosh (XLS)	Y	N	N
Microsoft Excel Windows (XLS)	Y	Y	Y [2]
Microsoft Excel Windows XML 2007 (XLSX)	Y	Y	N
Microsoft Office Excel Binary Format (XLSB)	Y	Y	N
Microsoft Works Spreadsheet	Y	N	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
(S30/S40)			
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, 2]
Microsoft Excel Windows XML 2003	Y	Y	Y
Microsoft Word for Windows XML 2003	Y	Y	Y
Microsoft Visio XML 2003	Y	Y	Y
Rich Text Format (RTF)	Y	Y	Y[3]
Unicode HTML	Y	Y	Y[2,3]
Unicode Text (TXT)	Y	Y	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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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]The text direction in the output file might not be correct.

[2]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 Export class.

Code Character Sets

Coded Character Set	Description	Can be set as target charset?
KVCS_UNKNOWN	Unknown character set	N
KVCS_SJIS	Japanese (uses multibyte encoding), cp932	Y
KVCS_GB	Simplified Chinese (China, Singapore, Malaysia) cp936	Y
KVCS_BIG5	Traditional Chinese (Taiwan, Hong Kong, Macaw) cp950	Y
KVCS_KSC	Korean, cp949	Y
KVCS_1250	Windows Latin 2 (Central Europe)	Y
KVCS_1251	Windows Cyrillic (Slavic)	Y

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_1252	Windows Latin 1 (ANSI)	Y
KVCS_1253	Windows Greek	Y
KVCS_1254	Windows Latin 5 (Turkish)	Y
KVCS_1255	Windows Hebrew	Y
KVCS_1256	Windows Arabic	Y
KVCS_1257	Windows Baltic Rim	Y
KVCS_1258	Windows Vietnamese	Y
KVCS_8859_1	ISO 8859-1 Latin 1 (Western Europe, Latin America)	Y
KVCS_8859_2	ISO 8859-2 Latin 2 (Central Eastern Europe)	Y
KVCS_8859_3	ISO 8859-3 Latin 3 (S.E. Europe)	Y
KVCS_8859_4	ISO 8859-4 Latin 4 (Scandinavia/Baltic)	Y
KVCS_8859_5	ISO 8859-5 Latin/Cyrillic	Y
KVCS_8859_6	ISO 8859-6 Latin/Arabic	Y
KVCS_8859_7	ISO 8859-7 Latin/Greek	Y
KVCS_8859_8	ISO 8859-8 Latin/Hebrew	Y
KVCS_8859_9	ISO 8859-9 Latin/Turkish	Y
KVCS_8859_14	ISO 8859-14	Y
KVCS_8859_15	ISO 8859-15	Y
KVCS_437	DOS Latin US	Y
KVCS_737	DOS Greek	Y
KVCS_775	DOS Baltic Rim	Y
KVCS_850	DOS Latin 1	Y
KVCS_851	DOS Greek	Y
KVCS_852	DOS Latin 2	Y
KVCS_855	DOS Cyrillic	Y

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_857	DOS Turkish	Y
KVCS_860	DOS Portuguese	Y
KVCS_861	DOS Icelandic	Y
KVCS_862	DOS Hebrew	Y
KVCS_863	DOS Canadian French	Y
KVCS_864	DOS Arabic	Y
KVCS_865	DOS Nordic	Y
KVCS_866	DOS Cyrillic Russian	Y
KVCS_869	DOS Greek 2	Y
KVCS_874	Thai	Y
KVCS_PDFMACDOC	PDF MAC DOC	N
KVCS_PDFWINDOC	PDF WIN DOC	N
KVCS_STDENC	Adobe Standard Encoding	N
KVCS_PDFDOC	Adobe standard PDF character set	N
KVCS_037	EBCDIC code page 037	Y
KVCS_1026	EBCDIC code page 1026	Y
KVCS_500	EBCDIC code page 500	Y
KVCS_875	EBCDIC code page 875	Y
KVCS_LMBCS	Lotus multibyte character set Group 1 and Group 2	N
KVCS_UNICODE	Unicode, UCS-2	N
KVCS_UTF16	16-bit Unicode transformation format	N
KVCS_UTF8	8-bit Unicode transformation format	Y
KVCS_UTF7	7-bit Unicode transformation format	Y
KVCS_2022_JP	ISO 2022-JP, Japanese mail and news safe encoding (JIS-7)	N

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_2022_CN	ISO 2022-CN, Chinese mail and news safe encoding	N
KVCS_2022_KR	ISO 2022-KR, Korean mail and news safe encoding	N
KVCS_WP6X	Word Perfect 6.x and higher character mapping	N
KVCS_10000	Western European (Macintosh)	Y
KVCS_KSC5601	Unified Hangul	Y
KVCS_GB2312	Simplified Chinese (China, Singapore, Hong Kong)	Y
KVCS_GB12345	Traditional Chinese (China) - analogue of GB2312	Y
KVCS_CNS11643	Traditional Chinese - Taiwan. Supplement to Big5	Y
KVCS_JIS0201	Japanese - contains ASCII character set (JIS-Roman)	N
KVCS_JIS0212	Japanese. Supplement to JIS0208.	Y
KVCS_EUC_JP	Japanese Extended UNIX Code	Y
KVCS_EUC_GB	Simplified Chinese Extended UNIX Code	Y
KVCS_EUC_BIG5	Traditional Chinese Extended UNIX Code	N
KVCS_EUC_KSC	Korean Extended UNIX Code	N
KVCS_424	EBCDIC Hebrew	N
KVCS_856	PC Hebrew (old)	N
KVCS_1006	IBM AIX Pakistan (Urdu)	N
KVCS_KOI8R	Cyrillic (Russian)	Y
KVCS_PDF_JAPAN1	Adobe-Japan1-2 character collection	N
KVCS_PDF_KOREA1	Adobe-Korea1-0 character collection	N
KVCS_PDF_GB1	Adobe-GB1-3 character collection	N
KVCS_PDF_	Adobe-CNS1-2 character collection	N

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
CNS1		
KVCS_2022_JP_8	ISO 2022-JP, Japanese mail and news safe encoding (JIS8)	N
KVCS_720	Arabic DOS-720	Y
KVCS_VISCII	Vietnamese VISCII	Y
KVCS_8859_10	ISO 8859-10 (Latin 6 Nordic)	Y[1]
KVCS_8859_13	ISO 8859-13 (Latin 7 Baltic)	Y [1]
KVCS_57002	ISCII Devanagari (x-iscii-de)	Y [1]
KVCS_57003	ISCII Bengali (x-iscii-be)	Y [1]
KVCS_57004	ISCII Tamil (x-iscii-ta)	Y[1]
KVCS_57005	ISCII Telugu (x-iscii-te)	Y[1]
KVCS_57006	ISCII Assamese (x-iscii-as)	Y[1]
KVCS_57007	ISCII Oriya (x-iscii-or)	Y[1]
KVCS_57008	ISCII Kannada (x-iscii-ka)	Y[1]
KVCS_57009	ISCII Malayalam (x-iscii-ma)	Y[1]
KVCS_57010	ISCII Gujarathi (x-iscii-gu)	Y[1]
KVCS_57011	ISCII Panjabi (x-iscii-pa)	Y [1]
KVCS_GB18030b2	Reserved for internal use	n/a
KVCS_GB18030	GB18030 (Chinese 4-byte character set)	Y
KVCS_8859_11	ISO 8859-11 (Thai)	Y
KVCS_8859_16	ISO 8859-16 (Latin-10 South-Eastern Europe)	Y
KVCS_ARABICMAC	Arabic Mac (x-mac-arabic)	Y
KVCS_KOI8U	Cyrillic (KOI8U Ukrainian)	Y
KVCS_HZGB2312	The 7-bit representation of GB 2312 / RFC 1842	n/a

[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 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 379.

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"`

Conditional elements, continued

Element	Description
`<iftzeq>`, `<iftzne>`	Respectively, if the time zone equals or does not equal the comparison `text`, for example `CDT`, `EST`, and so on.
`<ifini>`	If the value of the INI option specified in `name` equals the `text` value.
`<else>`	If the condition from the last `<if>` or `<switch>` was false, process XML.
`<switch>`	If a `name` value exists, process XML. **Attributes** • `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**

Control Elements, continued

Element	Description
	• `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. • `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>, above.
`<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.

Data elements, continued

Element	Description
	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>, on the previous page, images are output in the next part or parts of the MHTML.
`<form>`	Output the message form (usually `$Body` field) in rich text (MHTML). **Attributes** • `name`. (Required if there is no parent) The name of the item to output.
`<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 page 384 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.

Data elements, continued

Element	Description
	• `format`. (Optional) See <format>, on the previous page. 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>, on the previous page. You can set the following values: ◦ TZ ◦ 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 381 and <body>, on the previous page.
`<image_ uri>`	Output an image URI, in quotation marks. The actual image is output to a different part of the MHTML subfile. **Attributes**

Data elements, continued

Element	Description
	• `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

Lotus Notes date and time formats, continued

Format	Integer Value	Description
TZFMT_NEVER	0	(Notes default) All time zones are converted to the current time zone
TZFMT_SOMETIMES	1	Show only when outside the current time zone
TZFMT_ALWAYS	2	Show for all time zones
TSFMT_DATE	0	Date
TSFMT_TIME	1	Time
TSFMT_DATETIME	2	(The Notes default) Date and time
TSFMT_CDATETIME	4	Date and time, or time today or time yesterday

KeyView Date and Time Formats

This section lists KeyView date and time formats. The KeyView formats use the following syntax:

Month   `Month` = full month name

             `Mon` = abbreviated month name

             `m` = month (number)

             `mm` = two-digit month (leading 0)

Weekday   `Weekday` = full weekday name

             `Wday` = abbreviated weekday name

Year   `yy` = two-digit year

             `yyyy` = four-digit year

>Day   `d` = day (number)

             `dd` = two-digit day (leading 0)

Time   `h` = 12-hour

             `H` = 24-hour

             `m` = minutes

             `s` = seconds

             `P` = AM/PM

             `p` = am/pm

Separators	_ = space
	c = comma
	s = slash
	a = dash
	o = dot

KeyView date and time formats

Format	Output	Integer Value
12-Hour and 24-Hour Time Formats		
KVDTF_P	P	1
KVDTF_P_hmm	P h:mm	2
KVDTF_hmm_P	h:mm P	3
KVDTF_P_hhmm	P hh:mm	4
KVDTF_hhmm_P	hh:mm P	5
KVDTF_P_hmmss	P h:mm:ss	6
KVDTF_hmmss_P	h:mm:ss P	7
KVDTF_P_hhmmss	P hh:mm:ss	8
KVDTF_hhmmss_P	hh:mm:ss P	9
KVDTF_Hmm	H:mm	10
KVDTF_HHmm	HH:mm	11
KVDTF_mmss	mm:ss	12
KVDTF_Hmmss	H:mm:ss	13
KVDTF_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

KeyView date and time formats, continued

Format	Output	Integer Value
`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
`KVDTF_mmsddsyy_hmm_P`	`mm/dd/yy h:mm P`	38
`KVDTF_mmsdd_P_hhmm`	`mm/dd P hh:mm`	39
`KVDTF_mmsdd_hhmm_P`	`mm/dd hh:mm P`	40
`KVDTF_mmsddsyy_P_hhmmss`	`mm/dd/yy P hh:mm:ss`	41
`KVDTF_mmsddsyy_hhmmss_P`	`mm/dd/yy hh:mm:ss P`	42
`KVDTF_msd`	`m/d`	43
`KVDTF_yysm`	`yy/m`	44
`KVDTF_yysmm`	`yy/mm`	45

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_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
KVDTF_ddaMonayyyy	dd-Mon-yyyy	62
KVDTF_Mon	Mon	63
KVDTF_Monayy	Mon-yy	64
KVDTF_Monayyyy	Mon-yyyy	65
KVDTF_Monaddayy	Mon-dd-yy	66
KVDTF_yyammadd_P_hhmmss	yy-mm-dd P hh:mm:ss	67
KVDTF_mmadd_P_hhmm	mm-dd P hh:mm	68

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_Mon_yy	Mon yy	69
KVDTF_Monc_yy	Mon, yy	70
KVDTF_Month	Month	71
KVDTF_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
KVDTF_Weekdayc_d_Monthc_yyyy	Weekday, d Month, yyyy	88

Appendix E: Export Tokens

This section contains an alphabetized list of the Export tokens.

Tokens are special strings inserted into the `KVHTMLTemplateEx` structure, `HtmlTemplateInfo` class, and template files. They are placeholders for markup that appears in the HTML output. For example, the `$CHARSET` token marks the place in the HTML output where the name of the source document's character set is inserted. It would be used in the tag `<charset=$CHARSET>`.

Word documents are split into blocks by heading level. By default, each section of text between Heading Level 1 headings will be a single block.

See the template files for examples of how to use tokens.

Export Tokens

Token	Description
`$ANCHOR`	Inserts an anchor for a heading level (`h2-h6`) for the current block.
`$BASE`	Inserts the base URL for the HTML file. Use in the `<base href=xx>` tag.
`$CHARSET`	Inserts the character set of the source document, if that information is ascertainable. Supported Formats, on page 284 lists the file formats for which character set information can be determined.
`$CONTENT`	Inserts the content of the metadata field specified by the `$NAME` token. This token is used in conjunction with the `$SUMMARY`, `$USERSUMMARY`, and `$NAME` tokens to insert source document metadata into the HTML output. An example of this token's use is: `pszUserSummary=<meta name="$NAME" content="$CONTENT">` Supported Formats, on page 284 lists file formats that support metadata.
`$ENDNOTE`	Inserts endnotes from the current block at this point in the output stream. Currently implemented for Microsoft Word documents only.
`$ENDNOTEALL`	Inserts all endnotes at this point in the output stream. Currently implemented for Microsoft Word documents only.
`$FOOTER`	Inserts the footer from the current block at this point in the output stream.
`$FOOTNOTE`	Inserts footnotes from the current block at this point in the output stream. Currently implemented for Microsoft Word documents only.
`$FOOTNOTEALL`	Inserts all footnotes at this point in the output stream. Currently implemented for Microsoft Word documents only.
`$HEADER`	Inserts the header from the current block at this point in the output stream.

Export Tokens, continued

Token	Description
$MAINURL	Inserts the URL to the file containing the start of the generated HTML, that is, the main output stream.
$NAME	Inserts the name of a metadata field. This token is used in conjunction with the $SUMMARY, below, $USERSUMMARY, on the next page, and $CONTENT, on the previous page tokens to insert source document metadata into the HTML output. An example of this token's use is: `pszUserSummary=<meta name="$NAME" content="$CONTENT">` The section Supported Formats, on page 284 lists file formats that support metadata.
$NEXT	Inserts the anchor to the next block. If this is the last block, a link to the first block is inserted.
$PREV	Inserts the anchor to the previous block. If the current block is the first block, a link to the last block is inserted.
$STYLESHEET	Inserts the path to the style sheet. Only available in `KVHTMLOptionsEx`.
$SUMMARY	Inserts the data from standard metadata fields using the markup provided in the `pszUserSummary` member of the structure `KVHTMLTemplateEx`. Standard fields are enumerated from 0 to 33 in `KVSumType` in `kvtypes.h`. See the tokens $USERSUMMARY, on the next page, $NAME, above, and $CONTENT, on the previous page. The section Supported Formats, on page 284 lists file formats that support metadata.
$SUMMARY*NN*	Inserts the data from a *specified* metadata field. *NN* is a number from 0 through 33 enumerated in the `KVSumType` structure in `kvtypes.h`. An example of this token's use is: `pszMainTop=<head> <title> $SUMMARY01 </title> </head> <body>` The section Supported Formats, on page 284 lists file formats that support metadata.
$SPLITBLOCKNUMBER	Inserts the page number for each block generated as a result of `bHardPageMakesNewBlock` or `lcbBlockSize`.
$TOC	Inserts the table of contents at this point in the current output stream. This token is typically embedded in `pszMainTop`.
$TOCB	Inserts the table of contents at this point for the current block.
$TOCBE	Inserts the beginning entry for the table of contents at this point in the current output stream.

Export Tokens, continued

Token	Description
`$TOCE`	Inserts a table of contents entry at this point in the current output stream.
`$TOCTE`	Inserts a text entry without HTML markup at this point in the current output stream.
`$TOCPE`	Inserts a partial table of contents entry at this point in the current output stream. HTML tags are removed; however, character entities are retained. This enables angle brackets to appear in the table of contents entries (for example, `<text>`). Without this token, `<text>` would be interpreted as a non-valid HTML tag and would be ignored by the browser.
`$TOPANCHOR`	Inserts the anchor for the top heading level (h1) for the current block.
`$USERCB`	Triggers the callback function `UserCB()` and identifies the callback used in the function.
`$USERSUMMARY`	Inserts the data from *every* valid non-standard metadata field using the markup provided in the `pszUserSummary` member of the `KVHTMLTemplateEx` structure. Non-standard metadata are any fields not listed from 0 to 33 in `KVSumType`, such as user-defined fields (for example, custom property fields in Word documents), or fields that are unique to a particular file type (for example, "Artist" or "Genre" fields in MP3 files). See the tokens $SUMMARY, on the previous page, $NAME, on the previous page, and $CONTENT, on page 390. The section Supported Formats, on page 284 lists file formats that support metadata.
`$XANCHOR`	Inserts the anchor to an extra file into the HTML output. An example of this token's use is: `<frame src="$XANCHOR" name="Left" scrolling="auto" target="right">` The contents of the extra file is defined by `pszXFile`, and the block generated by this token is defined by `pszXStartBlock` and `pszXEndBlock`.

Appendix F: File Format Detection

This section describes how file formats are detected in the KeyView Export 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 284.

Extract Format Information

You can extract format information from a document by using the `fpGetStreamInfo()` function. If required, this format information can then be reported to the developer's application. The `fpGetStreamInfo()` function extracts format information, such as file class, format, and version, and populates the `ADDOCINFO` structure. This structure is defined in the `adinfo.h` header file.

For information on how to translate the extracted format information, see Translate Format Information, on page 395.

Determine Format Support

After the file format is extracted, the detection module uses the `formats_e.ini` file to determine whether the format is supported by KeyView, and the appropriate structured access layer and reader to load.

The `formats_e.ini` file is in the directory *install**OS*\bin, where *install* is the path name of the Export 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 395.

- The reader associated with each format. See Determine a Document Reader, on page 397.

- Configuration parameters for out-of-process conversions.

- Locale settings for internal use.

Below are some entries from the `formats_e.ini` file:

```
123=mw
152=xyw
178=wp6
189=mw6
2=af
200=pdf
205=mb
210=htm
251=htm
```

> **NOTE:** The `formats_e.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 analyzes 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 allow 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_e.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 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_e.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`.

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).

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 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).

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)

The example above applies to `formats_e.ini` file. When extracting format information by using the `fpGetStreamInfo()` function method, the same format information is represented as `294.2.0.9`.

> **NOTE:** The format values returned by `fpGetStreamInfo()` differ from those in `formats_e.ini` because the former defines a unique ID for each major format, whereas the latter uses a major version, minor version, and minor format to distinguish between formats.

Distinguish Between Formats

The `ADDOCINFO` structure method provides a unique ID for each major format. For example, a call to `fpGetStreamInfo()` returns `351.1.0` for a Microsoft Word 2003 XML format. The major format `351` is unique to this format.

Unlike `ADDOCINFO`, the `formats_e.ini` file distinguishes between formats by using the major version number. For example, in `formats_e.ini`, 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 in `formats_e.ini` to specify the following formats:

- The Microsoft Office 2003 XML 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_e.ini` file to determine whether a format is supported and which reader should be used to parse a format. The entries in the `formats_e.ini` file lists each format's coded value, and an abbreviation for the format's reader. For example:

`81.2.0.9.0=l123`

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. The example entry above specifies that a Lotus 1-2-3 Spreadsheet file version 9.0 is parsed by the Lotus 1-2-3 reader, `l123sr`.

Files Required for Redistribution, on page 401 lists the document readers provided with KeyView.

Category Values in formats_e.ini

The Detected Formats section lists all of the file formats that can be detected by KeyView, with associated category values for use in the `formats_e.ini` file. The following tables provide the list of possible file classes and minor formats.

- File Classes

- Minor Formats

File Classes

Attribute Number	Description	File class
0	No file class	AutoDetNoFormat
01	Word processor	adWORDPROCESSOR
02	Spreadsheet	adSPREADSHEET
03	Database	adDATABASE
04	Raster image	adRASTERIMAGE
05	Vector graphic	adVECTORGRAPHIC
06	Presentation	adPRESENTATION

File Classes, continued

Attribute Number	Description	File class
07	Executable	adEXECUTABLE
08	Encapsulation	adENCAPSULATION
09	Sound	adSOUND
10	Desktop publishing	adDESKTOPPUBLSH
11	Outline/planning	adOUTLINE
12	Miscellaneous	adMISC
13	Mixed format	adMIXED
14	Font	adFONT
15	Time scheduling	adSCHEDULE
16	Communications	adCOMMUNICATION
17	Object module	adOBJECTMODULE
18	Library module	adLIBRARY
19	Fax	adFAXFORMAT
20	Movie	adMOVIE
21	Animation	adANIMATION
22	Source Code	adSOURCECODE
23	Computer-Aided Design	adCAD
24	BI and analysis tools	adANALYTICS

Minor Formats

Attribute Number	Minor Format
00	Minor format not defined
01	Standard
02	Book
03	Chart
04	Macro

Minor Formats, continued

Attribute Number	Minor Format
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
30	Miscellaneous

Minor Formats, continued

Attribute Number	Minor Format
31	UNIX
32	VAX
33	Driver
34	Archive

Appendix G: Files Required for Redistribution

This section lists the Export files that can be redistributed in your applications under the licensing agreement. Unless noted, these files are in the directory *install**OS*\bin, where *install* is the path of the Export installation directory and *OS* is the operating system platform.

- Core Files
- Support Files
- Document Readers and Writers

> **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_e.ini	Initialization file. For more information on this file, see Determine Format Support, on page 393.
htmlexport.	Required by the Java API.
htmserv.dll	The in-process version of the HTML Export COM interface.
htmserv.exe	The out-of-process version of the HTML Export COM interface.
htmcnv.*	HTML converter for the document token stream.
KeyView.jar	Interface for Java support. > **NOTE:** > This file can be found at the path *install*/javaapi/KeyView.jar where *install* is the Export SDK installation directory.
kpifcnvt.*	Graphic conversion routines.
kpifutil.*	Graphic utility routines.
kvdecrypt.*	Decryption utility functions
kvxtract.*	File Extraction interface.
kvhtml.*	HTML Export C API.
kvexport.*	Export C API. Interface to the HTML and XML Export C APIs.

File	Description
kvexportdotnet.*	Interface for .NET support.
kvolefio.*	Embedded OLE object writer.
kvutil.*	Internal KeyView utility functions.
kvxpgsa.*	Interface between presentations or graphic readers and the Export API.
kvxsssa.*	Interface between spreadsheet readers and the Export API.
kvxwpsa.*	Interface between word processing readers and the Export API.
kvzip.*	Zip writer
kwad.*	File auto-recognition module.
regsvr32.exe	A Microsoft Windows program used to register in-process COM objects.
txtcnv.*	Converter for document token stream.
xmlexport.	Required by the Java API.
\vcredist	(Windows platforms only) Microsoft Visual C++ Redistributable Packages. **NOTE:** This folder can be found in the Export SDK installation directory.

Support Files

The following support files can be redistributed with your application.

File	Description
datafiles\	(Folder) Required by kvlangdetect.
NSFtemplates\	(Folder) Templates used by nsfsr to format Lotus mail notes.
7z.*	Required by z7zsr and multiarcsr.
bentofio.*	Required by l123sr.* and kpprzrdr.*.
cbmap.map	Character mappings for Adobe Portable Document Format (PDF).
CEBDLL.*	Required by cebsr.
chartbls.ux	Character mapping tables.
chmdll.*	Required by chmsr.
codeidentifierplugin	Required for source code identification.

File	Description
DFECore.*	Required by `cebsr`.
Filter.*	Required by `cebsr`.
kp3dwrld.*	Required for 3D charts.
kpchtrdr.*	Required for all spreadsheets (chart support).
kpjavwrt.*	Java utility routines.
kpjpeg.*	JPEG file interchange format shared routines.
kppng.*	Portable Network Graphics (PNG) utilities.
kvlangdetect.*	Utility functions for language and character set detection.
kvxconfig.ini	Contains element extraction settings for source XML files.
kvgraph.*	Required for all spreadsheets (chart support).
kvpie.*	Required for all spreadsheets (chart support).
kvradar.*	Required for all spreadsheets (chart support).
kv.lic	Contains license information for KeyView products. This file is opened and validated when a KeyView API is used.
kvraster.class	Java program used to convert vector graphics on UNIX and Linux.
kvVector.class	Java applet used to convert vector graphics on UNIX and Linux.
kvvector.jar	Java applet used to convert vector graphics on UNIX and Linux. This must reside in the output directory.
langdetectext.*	Required by `kvlangdetect.*`
libeay32.dll	(Windows platforms only) SSL utility functions used by KeyView mail format readers.
libpff.*	Required by `pffsr`.
libstlport.so.1	(Solaris platforms only) Solaris Studio Redistributable.
oleaut32.*	Microsoft OLE Automation Controls.
olepro32.*	Microsoft OLE property support library.
servant.exe	Executable required for out-of-process conversions.
unzipjpg.*	Required for JPEG decompression.
wpmap.*	Extended character mapping for WordPerfect and Corel Presentation.
xmlsh.*	Contains a library of content handlers for each XML file type. Required by the Expat XML parser.

Document Readers and Writers

The following readers and writers can be redistributed with your application.

File	Description
ad1sr.*	AD1 Evidence file reader
afsr.*	ASCII reader
assr.*	Applix spreadsheet reader
awsr.*	Applix Words 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
difsr.*	Data Interchange Format reader
dmgsr.*	Mac Disk Copy Disk Image File reader
dw4sr.*	DisplayWrite 4 reader
dxlsr.*	Domino XML Language reader
emlsr.*	Microsoft Outlook Express (EML) reader. This is used to convert EML files when the MBX reader is not licensed.
emxsr.*	Legato EMailXtender archive (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

File	Description
`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
`hwposr.*`	Hangul 2002, 2005, 2007 reader
`hwpsr.*`	Hangul 97 reader
`ichatsr.*`	Apple iChat Log reader
`icssr.*`	Microsoft Outlook iCalendar reader
`isosr.*`	ISO-9660 CD Disc Image Format reader
`iwss13sr.*`	iWork 13 Numbers reader
`iwsssr.*`	Apple iWork Numbers reader
`iwwp13sr.*`	iWork 13 Pages reader
`iwwpsr.*`	Apple iWork Pages reader
`jp2000sr.*`	JPEG 2000 metadata reader
`jpgsr.*`	JPEG metadata reader
`jtdsr.*`	JustSystems Ichitaro reader
`kpagrdr.*`	Applix Presents reader
`kpanirdr.*`	Animated cursor reader
`kpbmprdr.*`	Windows Bitmap reader
`kpbmpwrt.*`	Windows Bitmap writer
`kpcdrrdr.*`	Corel Draw
`kpcgmrdr.*`	Computer Graphics Metafile reader
`kpcgmwrt.*`	Computer Graphics Metafile writer
`kpdcxrdr.*`	DCX (fax) reader
`kpDWGrdr.*`	AutoCAD Drawing format reader
`kpDXFrdr.*`	AutoCAD Drawing Exchange format reader

File	Description
kpemfrdr.*	Enhanced Metafile reader
kpemfwrt.*	Enhanced Metafile writer
kpepsrdr.*	Encapsulated PostScript (EPS) reader
kpgflrdr.*	OmniGraffle Picture reader
kpgifrdr.*	Graphic Interchange Format (GIF) reader
kpgifwrt.*	Graphic Interchange Format (GIF) writer
kpicordr.*	Windows Icon reader
kpiwpgrdr.*	Apple iWork Keynote reader
kpjbig2rdr.*	JBIG2 reader
kpjp2000rdr.*	JPEG 2000 reader
kpjpgrdr.*	JPEG file interchange format reader
kpjpgwrt.*	JPEG file interchange format writer
kpnbmprdr.*	IBM Notes Bitmap reader (for embedded images in DXL files)
kpmacrdr.*	MacPaint reader
kpmsordr.*	Microsoft Office Drawing Objects (office 97, 2000, and XP) reader
kpodfrdr.*	Oasis Open Document Format presentation (ODP) reader
kpODArdr.*	AutoCAD reader (Windows only)
kpONErdr.*	Microsoft OneNote reader
kpoxdrdr.*	Open Office XML Diagram Graphics reader
kppdfrdr.*	Adobe Portable Document File (PDF) graphic-based reader
kppdf2rdr.*	High-fidelity Adobe Portable Document File (PDF) graphic-based 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
kppcxrdr.*	PC Paintbrush (PCX) reader
kppdfrdr.*	Graphic-based Adobe Portable Document File (PDF) reader

File	Description
kppicrdr.*	Pictor PC Paint format (PIC) reader
kppngrdr.*	Portable Network Graphics (PNG) reader
kppngwrt.*	Portable Network Graphics (PNG) writer
kpppxrdr.*	Microsoft PowerPoint XML reader 2007
kpprerdr.*	Lotus Freelance Graphics for Windows V2.0 reader
kpprzrdr.*	Lotus Freelance Graphics 96/97/98 reader
kprawrdr.*	ODA Internal Raster (RAW) Picture reader
kpsddrdr.*	StarOffice Draw / Impress reader
kpsdwrdr.*	Lotus Ami Pro Graphics reader
kpsgirdr.*	SGI RGB reader
kpshwrdr.*	Corel Presentations reader
kpsprdr.*	Shape Stream reader
kpsunrdr.*	Sun Raster reader
kptgardr.*	Truevision Targa reader
kptifrdr.*	Tagged Image File Format (TIFF) reader
kpvsd2rdr.*	Microsoft Visio reader
kpvsdxrdr.*	Microsoft Visio 2013 reader
kpwg2rdr.*	WordPerfect Graphics 2 reader
kpwmfrdr.*	Windows Metafile reader
kpwmfwrt.*	Windows Metafile writer
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

File	Description
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)[1] and Microsoft Outlook Express (EML) reader
mdbsr.*	Microsoft Access reader.
mhtsr.*	MIME HTML reader
mifsr.*	Adobe Maker Interchange Format reader
misr.*	Microsoft Word 2 reader
mp3sr.*	MP3 reader for metadata extraction
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

[1]This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

File	Description
nsfsr.*	IBM Notes Database reader[1]
oa2sr.*	Fujitsu Oasys reader
odfsssr.*	Oasis Open Document Format spreadsheets (ODS) reader
odfwpsr.*	Oasis Open Document Format word processing (ODT) reader
olesr.*	Embedded OLE object reader.
olmsr.*	Microsoft Outlook for Macintosh reader
onmsr.*	Legato EMailXtender Native Message reader
oo3sr.*	Omni Outliner reader
pdf2sr.*	Alternative Adobe Portable Document Format file (PDF) reader
pdfsr.*	Adobe Portable Document 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)[2]
pstnsr.*	Microsoft Outlook Personal Folders file native reader[3]
pstxsr.*	Microsoft Outlook Personal Folders file native reader[4]
qpssr.*	Quattro Pro spreadsheet reader
rarsr.*	RAR Archive reader
rtfsr.*	Microsoft Rich Text Format reader
skypesr.*	Skype log file reader
sosr.*	StarOffice/OpenOffice reader
starcsr.*	StarOffice Calc reader
starwsr.*	StarOffice Writer reader

[1]This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

[2]This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

[3]This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

[4]This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

File	Description
swfsr.*	Macromedia Flash reader
tarsr.*	Tape archive reader
tifsr.*	TIFF reader (metadata only)
tnefsr.*	Transfer Neutral Encapsulation Format reader
unihtmsr.*	Unicode HTML reader
unisr.*	Unicode reader
unzip.*	Zip file reader
uudsr.*	UUEncoding reader
vsdsr.*	Microsoft Visio reader
vcfsr.*	Microsoft Outlook vCard Contact reader
wkssr.*	Lotus 1-2-3 v2.0 through 5.0 reader
wosr.*	WordPerfect 5.x reader
wp6sr.*	WordPerfect 6.0 through 10.0 reader
wpmsr.*	WordPerfect for Macintosh reader
xlsbsr.*	Microsoft Office 2007 Excel Binary Format reader
xlssr.*	Microsoft Excel reader
xlsxsr.*	Microsoft Excel 2007 XML reader
xmlsr.*	Generic XML reader
xpssr.*	XML Paper Specification reader
xywsr.*	XYWrite reader
yimsr.*	Yahoo! Instant Messenger reader
z7zsr.*	7-Zip reader

Appendix 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

[1]The native PST readers, `pstxsr` and `pstnsr`, do not require credentials to open password-protected PST files that use compressible encryption.

Supported password-protected file types, continued

File Type	Version	Filter	Export	Extract	View	Credentials
Lotus Notes NSF	n/a	N	N	Y	N	C
Adobe PDF	n/a	Y	Y	Y	V	P
Microsoft Office	97-2003 2007 2010	Y	Y	Y	V	P

Open Password Protected Container Files

This section describes how to extract password-protected container files using the C API. The following guidelines apply to specific file types.

- **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 IBM Notes API. If the Notes client is not running with an active user, KeyView does not require credentials to access the client.

- **PST files**. To open password-protected PST files that use high encryption (Microsoft Outlook 2003 only), you must use the MAPI-based PST reader (`pstsr`). The native PST readers (`pstxsr` and `pstnsr`) do not support files that use high encryption and return the error message `KVERR_PasswordProtected` if a PST file is encrypted with high encryption.

To open container files

1. Define the credential information in the `KVOpenFileArg` data structure.

2. Pass `KVOpenFileArg` to the fpOpenFile() function.

3. Call `fpCloseFile()`.

Export Password Protected Files

This section describes how to export password-protected non-container files with the C API.

To export password-protected files

1. Call the `fpInit()` function.

2. Call the `KVHTMLConfig()` function with the following arguments :

Argument	Parameter
nType	KVCFG_SETPASSWORD
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:

```
(*fpHTMLConfig)(pKVHTML, KVCFG_SETPASSWORD, TRUE, password);
```

where `password` is a null-terminated string of 255 or fewer characters.

3. Call the `fpConvertStream()` or `KVHTMLConvertFile()` 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 HTML Export SDK C and COM Programming Guide (Micro Focus KeyView 12.2)

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to swpdl.idoldocsfeedback@microfocus.com.

We appreciate your feedback!