

KeyView

Software Version 12.1

XML Export SDK C Programming Guide



Document Release Date: October 2018
Software Release Date: October 2018

Legal notices

Copyright notice

© Copyright 2006-2018 Micro Focus or one of its affiliates.

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

Documentation updates

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

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

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 XML Export	11
Chapter 1: Introducing XML Export	12
Overview	12
Features	13
Platforms, Compilers, and Dependencies	13
Supported Platforms	14
Supported Compilers	14
C++ Filter SDK	15
Software Dependencies	15
Windows Installation	16
UNIX Installation	17
Package Contents	18
License Information	18
Enable Advanced Document Readers	19
Update License Information	19
Directory Structure	20
Definition of Terms	22
Chapter 2: Getting Started	23
Architectural Overview	23
Memory Abstraction	25
Enhance Performance	25
File Caching	25
Convert Files Out of Process	25
Configure Out-of-Process Conversions	26
Run Export Out of Process—Overview	28
Recommendations	29
Run Export Out of Process in the C API	29
Example—KVXMLStartOOPSession	30
Example—KVXMLEndOOPSession	31
Convert Files	32
Subfile Extraction	33
Convert Outlook Email without Using the Extraction API	33
Set Conversion Options	33
Set Conversion Options by Using the API	34
Set Conversion Options by Using the Template Files	34
Templates	34
Use the Export Demo Program	36
Change Input/Output Directories	36
Set Configuration Options	37
Suppress Images	37
Use PDF Position Information	38

Convert Files	38
Use the C-Language Implementation of the API	39
Input/Output Operations	39
Convert Files	40
Multithreaded Conversions	41
Use the Verity Document Type Definition (DTD)	42
Use XML Style Language Transformation (XSLT)	43
Add Elements and Attributes to the DTD	43
Move the DTD	43
 Part II: Use the Export API	 44
Chapter 3: Use the File Extraction API	45
Introduction	45
Extract Subfiles	46
Sanitize Absolute Paths	47
Extract Images	48
Recreate a File's Hierarchy	48
Create a Root Node	48
Recreate a File's Hierarchy—Example	49
Extract Mail Metadata	49
Default Metadata Set	50
Extract the Default Metadata Set	50
Microsoft Outlook (MSG) Metadata	51
Extract MSG-Specific Metadata	52
Microsoft Outlook Express (EML) and Mailbox (MBX) Metadata	53
Extract EML- or MBX-Specific Metadata	53
Lotus Notes Database (NSF) Metadata	53
Extract NSF-Specific Metadata	54
Microsoft Personal Folders File (PST) Metadata	54
MAPI Properties	54
Extract PST-Specific Metadata	55
Exclude Metadata from the Extracted Text File	56
Extract Subfiles from Outlook Files	56
Extract Subfiles from Outlook Express Files	56
Extract Subfiles from Mailbox Files	57
Extract Subfiles from Outlook Personal Folders Files	57
Use the Native or MAPI-based Reader	57
Use the Native PST Reader (pstnsr)	58
Use the MAPI Reader (pstsar)	58
System Requirements	59
MAPI Attachment Methods	59
Open Secured PST Files	60
Detect PST Files While the Outlook Client is Running	60
Extract Subfiles from Lotus Domino XML Language Files	61
Extract .DXL Files to HTML	61

Extract Subfiles from Lotus Notes Database Files	61
System Requirements	62
Installation and Configuration	62
Windows	62
Solaris	63
AIX 5.x	63
Linux	64
Open Secured NSF Files	64
Format Note Subfiles	64
Extract Subfiles from PDF Files	64
Improve Performance for PDFs with Many Small Images	65
Extract Embedded OLE Objects	65
Extract Subfiles from ZIP Files	65
Default File Names for Extracted Subfiles	66
Default File Name for Mail Formats	66
Default File Name for Embedded OLE Objects	67
Chapter 4: Use the XML Export API	68
Extract Metadata	68
Extract Metadata by Using the API	68
Use the C API	69
Extract Metadata by Using a Template File	69
Examples	70
\$SUMMARYNN	70
\$SUMMARY	70
\$USERSUMMARY	71
Extract File Format Information	71
Use the C API	71
Convert Character Sets	71
Determine the Character Set of the Output Text	72
Guidelines for Character Set Conversion	72
Examples of Character Set Conversion	73
Document Character Set Can be Determined	73
Document Character Set Cannot be Determined	74
Set the Character Set During Conversion	74
Set the Character Set During File Extraction from a Container	75
Map Styles	75
Use the C API	76
Use a Template file	76
Use Style Sheets	78
Use Extensible Style Sheet Language (XSL)	78
Use Cascading Style Sheets (CSS)	79
Display Vector Graphics on UNIX and Linux	79
Convert Revision Tracking Information	80
Convert PDF Files	81
Use the pdf2sr Reader	81
Convert PDF Files to a Logical Reading Order	82

Logical Reading Order and Paragraph Direction	83
Enable Logical Reading Order	84
Use the C API	84
Use the formats_e.ini File	84
Control Hyphenation	85
Extract Custom Metadata from PDF Files	86
Configure the Size of Exported Images	87
Convert Spreadsheet Files	88
Convert Hidden Text in Microsoft Excel Files	88
Convert Headers and Footers in Microsoft Excel 2003 Files	88
Specify Date and Time Format on UNIX Systems	88
Convert Very Large Numbers in Spreadsheet Cells to Precision Numbers	89
Extract Microsoft Excel Formulas	89
Set Minimum Image Size	91
Convert Presentation Files	91
Convert Presentation Files to Raster Images	91
Convert Presentation Files to a Logical Reading Order	91
Convert XML Files	92
Configure Element Extraction for XML Documents	92
Modify Element Extraction Settings	93
Use the C API	93
Use an Initialization File	94
Modify Element Extraction Settings in the kvxconfig.ini File	94
Specify an Element's Namespace and Attribute	96
Add Configuration Settings for Custom XML Document Types	96
Show Hidden Data	97
Hidden Data in Microsoft Documents	97
Toggle Word Comment Settings in the formats_e.ini File	98
Toggle PowerPoint Slide Note Settings in the formats_e.ini File	98
Exclude Japanese Guide Text	99
Obtain Image Info	99
Example	100
Source Code Identification	100
Chapter 5: Sample Programs	102
Introduction	102
C Sample Programs	102
Compile the Visual Basic Sample Program	103
tstxtract	103
cnv2xml	104
cnv2xmloop	105
metadata	106
xmlindex	106
xmlini	107
Use Style Sheets with xmlini	108
xmlcallback	108
xmlonefile	108

xmlmulti	109
Export Demo	109
Part III: C API Reference	110
Chapter 6: File Extraction API Functions	111
KVGetExtractInterface()	111
fpCloseFile()	112
fpExtractSubFile()	112
fpFreeStruct()	114
fpGetMainFileInfo()	115
fpGetSubFileInfo()	116
fpGetSubFileMetaData()	117
fpOpenFile()	119
Chapter 7: File Extraction API Structures	121
KVCredential	121
KVCredentialComponent	122
KVExtractInterface	122
KVExtractSubFileArg	123
KVGetSubFileMetaArg	126
KVMainFileInfo	127
KVMetadataElem	128
KVMetaName	129
KVOpenFileArg	129
KVOutputStream	131
KVSubFileExtractInfo	131
KVSubFileInfo	132
KVSubFileMetaData	135
Chapter 8: XML Export API Functions	137
KVXMLGetInterface()	137
KVXMLGetInterfaceEx()	138
fpConvertStream()	139
fpFileToInputStreamCreate()	142
fpFileToInputStreamFree()	143
fpFileToOutputStreamCreate()	143
fpFileToOutputStreamFree()	144
fpFreeImageInfos()	145
fpGetAnchor()	146
fpGetConvertFileList()	147
fpGetKvErrorCode	148
fpGetKvErrorCodeEx	148
fpGetOutputImageCount()	149
fpGetOutputImageInfo()	150
fpGetOutputImageInfos()	150
fpGetStreamInfo()	151
fpGetSummaryInfo()	152

fpInit()	153
fpSetStyleMapping()	155
fpShutDown()	156
fpValidateTemplate()	156
KVXMLConfig()	156
Configuration Flags	158
Examples	162
KVXMLConvertFile()	164
KVXMLEndOOPSession()	167
KVXMLSetStyleSheet()	168
KVXMLStartOOPSession()	170
Discussion	172
Example	172
Chapter 9: XML Export API Callback Functions	174
Introduction	174
Continue()	174
GetAnchor()	175
GetAuxOutput()	177
UserCB()	178
Chapter 10: XML Export API Structures	179
ADDOCINFO	179
KVInputStream	180
KVMemoryStream	181
KVOutputStream	181
KVSTR	182
KVStreamInfo	182
KVStructHead	183
KVStyle	184
KVSumInfoElemEx	185
KVSummaryInfoEx	185
KVXConfigInfo	186
KVXMLCallbacks	187
KVXMLHeadingInfo	188
KVXMLImageInfo	190
KVXMLInterface	191
KVXMLInterfaceEx	193
KVXMLOptions	195
Set the Resolution of Presentations and Vector Graphics	203
KVXMLTemplate	203
KVXMLTOCOptions	207
Chapter 11: Enumerated Types	209
Introduction	209
Programming Guidelines	210
ENSATableBorder	210
KVCredKeyType	211

KVErrorCode	211
KVErrorCodeEx	213
KVXMLStyleSheetType	216
KVXMLAnchorType	217
KVXMLGraphicType	218
KVHeadingCreateOptions	219
KVXMLEmptyParaType	220
Definition	220
Enumerators	220
KVXMLHardPageBreakType	221
Definition	221
Enumerators	221
KVMetadataType	221
KVMetaNameType	223
KVSumInfoType	223
KVSumType	224
LPDF_DIRECTION	228
 Part IV: Appendixes	 229
Appendix A: Supported Formats	230
Supported Formats	230
Archive Formats	231
Binary Format	233
Computer-Aided Design Formats	234
Database Formats	235
Desktop Publishing	236
Display Formats	236
Graphic Formats	237
Mail Formats	241
Multimedia Formats	244
Presentation Formats	247
Spreadsheet Formats	250
Text and Markup Formats	252
Word Processing Formats	253
Appendix B: Detected Formats	259
Detected Formats	260
Appendix C: Character Sets	320
Multibyte and Bidirectional Support	320
Coded Character Sets	328
Appendix D: Extract and Format Lotus Notes Subfiles	334
Overview	334
Customize XML Templates	334
Use Demo Templates	335
Use Old Templates	335

Disable XML Templates	335
Template Elements and Attributes	336
Conditional Elements	336
Control Elements	337
Data Elements	338
Date and Time Formats	341
Lotus Notes Date and Time Formats	341
KeyView Date and Time Formats	342
Appendix E: Export Tokens	347
Appendix F: File Format Detection	350
Introduction	350
Extract Format Information	350
Determine Format Support	350
Refine Detection of Text Files	351
Change the Amount of File Data to Read	351
Change the Percentage of Allowed Non-ASCII Characters	352
Use the File Extension for Detection	352
Allow Consecutive NULL Bytes in a Text File	352
Translate Format Information	352
Distinguish Between Formats	353
Determine a Document Reader	354
Category Values in formats_e.ini	354
Appendix G: Files Required for Redistribution	398
Core Files	398
Support Files	399
Document Readers and Writers	401
Document Type Definition Files	407
Appendix H: Password Protected Files	409
Supported Password Protected File Types	409
Open Password Protected Container Files	410
Export Password Protected Files	410
 Send documentation feedback	 412

Part I: Overview of XML 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.

- [Introducing XML Export, on page 12](#)
- [Getting Started, on page 23](#)

Chapter 1: Introducing XML Export

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

This section describes the KeyView Export SDK package.

• Overview	12
• Features	13
• Platforms, Compilers, and Dependencies	13
• Windows Installation	16
• UNIX Installation	17
• Package Contents	18
• License Information	18
• Directory Structure	20
• Definition of Terms	22

Overview

XML Export is part of the KeyView Export SDK. It enables you to convert virtually any document, spreadsheet, presentation, or graphic into well-formed, valid XML which is validated against a predefined Document Type Definition (DTD). With XML Export, you control the content, structure, and format of the XML output using either easily customized templates, or the flexible and robust APIs.

The main purpose of XML Export is to apply an XML vocabulary to the data structures in a document so that content and metadata can be indexed and subsequently searched in context.

Data structures in a source document can be:

- metadata (title, author, subject, and so on)
- document components (headers, footers, footnotes, endnotes, captions, bookmarks, and so on)
- tagged text (chapters, sections, bulleted lists, and so on)
- table components (sheet names, rows, columns, cell ranges, and so on)
- presentation components (notes, slide titles, slide descriptions, and so on)

Although viewing is not the main purpose of XML Export, Extensible Stylesheet Language (XSL) style sheets or Cascading Style Sheets (CSS) can be used to display the XML data.

Export SDK supports a number of programming environments, such as Visual Basic, Java, 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 well-formed, valid, and 1.0-compliant XML. The XML output is validated against a predefined DTD named the "Verity.dtd."
- 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 XML markup at specific points in the output stream.
- Apply XSL or 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.

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)

Platform	Architecture	Compiler Name	Compiler Version
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 client or later versions is required when processing Microsoft Outlook Personal Folders (PST) files using the MAPI-based reader (*pstsr*). The native PST reader (*pstnsr*) does not require an Outlook client.

NOTE:

If you are using 32-bit KeyView, you must install 32-bit Outlook. If you are using 64-bit KeyView, you must install 64-bit Outlook.

If the bit editions do not match, an error message from Microsoft Office Outlook is displayed:

Either there is a no default mail client or the current mail client cannot fulfill the messaging request. Please run Microsoft Outlook and set it as the default mail client.

Additionally, KeyView displays the following return code:

Error 32: KVErrors_PSTAccessFailed.

- Lotus Notes or Lotus Domino is required for Lotus Notes database (NSF) file processing. The minimum requirement is 6.5.1, but version 8.5 is recommended.
- 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_12.1_Windows_X86_64.exe`


The installation wizard opens.

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

The License Agreement page opens.

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

The Installation Directory page opens.

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

The License Key page opens.

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

NOTE:

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

The Pre-Installation Summary dialog box opens.

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

The SDK is installed.

7. Click **Finish**.

UNIX Installation

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

To install the SDK from the graphical interface

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

To install the SDK from the console

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

```
./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, below](#).

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, well-formed XML (see [Files Required for Redistribution, on page 398](#)).
- The include files that define the functions and structures used by the application to establish an interface with Export:
`adinfo.h`
`kvxml.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 XML to stripped-down, text-only output suitable for use with an indexing engine.
- The predefined DTD, `Verity.dtd`, used to validate all XML output.
- Sample style sheets: `wp.xml` (for word processing documents), `ss.xml` (for spreadsheets), and `pg.xml` (for presentation graphics).

License Information

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

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

Company Name
XXXXXXX-XXXXXXX-XXXXXXX-XXXXXXX

The license key controls whether the following are enabled:

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

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

Enable Advanced Document Readers

To enable advanced readers in one of the KeyView SDKs, you must obtain an appropriate license key from 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-XXXXXX-XXXXXX-XXXXXX* 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 XML 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 XML Export directory is named `/xmlexport`.

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`.

XML Export Installed Directory Structure

Directory	Contents
<i>install</i> \OS\bin	Contains the libraries, executables for the sample programs Export Demo and <code>cnv2xml</code> , the Java program (<code>kvraster.class</code>), the Java applet (<code>kvvector.jar</code>), the format detection file, <code>formats_e.ini</code> , the license key file (<code>kv.lic</code>), and a number of other supporting files.
<i>install</i> \javaapi\ini	Contains the template files used with the Java API.
<i>install</i> \javaapi\javadoc	Contains the Javadoc for the Java API.
<i>install</i> \javaapi\sample	Contains the source files and sample programs for the Java API.
<i>install</i> \testdocs	Contains sample word processing, spreadsheet, and presentation graphics files that can be used to test XML Export's options. You might also find this directory useful when testing your own applications.

XML Export Installed Directory Structure, continued

Directory	Contents
<i>instal</i> \XML Export\guide	Contains the <i>XML Export C Programming Guide</i> and <i>XML Export Java Programming Guide</i> in HTML and PDF format.
<i>instal</i> \XML Export\include	Contains the header files (<code>adinfo.h</code> , <code>kvxml.h</code> , and <code>kvtypes.h</code>) for the C API.
<i>instal</i> \XML Export\programs\bin	Contains the executable files for the sample Visual Basic program called Export Demo.
<i>instal</i> \XML Export\programs\cnv2xml	Contains the C source code files for a sample program that creates a single XML file. The executable for this sample program is in the <code>bin</code> directory.
<i>instal</i> \XML Export\programs\cnv2xmloop	Contains the C source code for a sample program that creates a single XML file out of process.
<i>instal</i> \XML Export\programs\ExportDemo	Contains the source code for a sample Visual Basic program. The executable for this sample program is in the <code>bin</code> directory. Export Demo is available through the Start menu.
<i>instal</i> \XML Export\programs\ini	Contains the template files used to set the conversion options in the C API.
<i>instal</i> \XML Export\programs\metadata	Contains the C source code and supporting files for a sample program that creates a valid XML file containing only the document's metadata.
<i>instal</i> \XML Export\programs\pdfini	Contains the template file used to extract custom metadata from PDF documents.
<i>instal</i> \XML Export\programs\tempout	The default output directory for converted files. Contains the KeyView DTD, sample style sheets, and character entity files. These files are required for viewing the converted XML files.
<i>instal</i> \XML Export\programs\tstxtract	Contains the C source code and supporting files for a sample program that demonstrates the File Extraction interface.
<i>instal</i> \XML Export\programs\xmlcallback	Contains the C source code and supporting files for a sample program that demonstrates how user callbacks can dynamically shape the XML conversion.
<i>instal</i> \XML Export\programs\xmlindex	Contains the C source code and supporting files for a sample program that produces text-only XML.
<i>instal</i> \XML Export\programs\xmlini	Contains the C source code and supporting files for a sample program that uses template files to set the conversion options.
<i>instal</i> \XML	Contains the C source code and supporting files for a sample

XML Export Installed Directory Structure, continued

Directory	Contents
	program that creates multiple XML files from a source document. The main file contains the table of contents. Each H1 heading is contained within its own file.
<i>instal</i> \XML Export\programs\xmlonefile	Contains the C source code and supporting files for a sample program that converts a source document into a single, formatted XML file.
<i>instal</i> \XML Export\rel_notes	Contains the <i>XML Export Release Notes</i> in HTML and PDF format.

Definition of Terms

The following are specialized terms used throughout the guide.

anchor	XML 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 XML anchor tags (<code><a xmlns:xlink= xlink href=> </code>) to facilitate navigation within a document. The major browsers do not currently support linking in XML documents.
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 XML 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 XML 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 347 .

Chapter 2: Getting Started

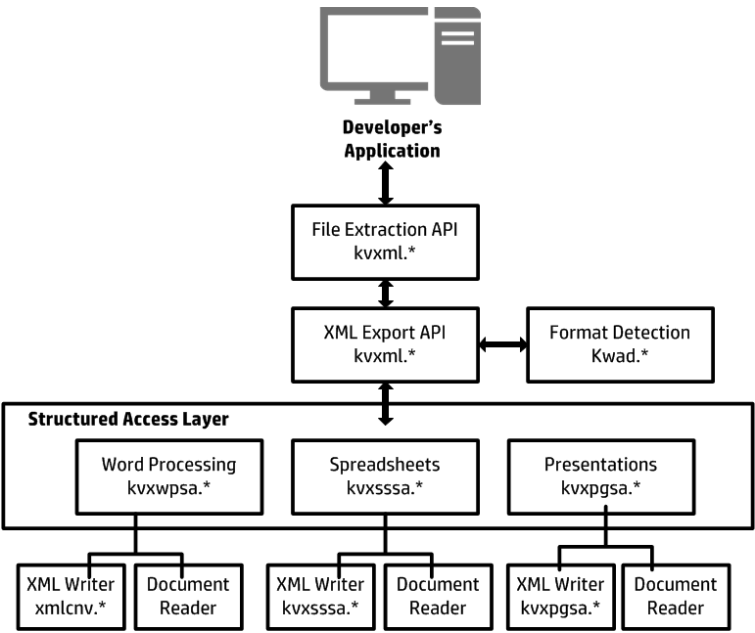
This section provides an overview of the XML Export SDK and describes how to use the C implementations of the API.

- [Architectural Overview](#) 23
- [Memory Abstraction](#) 25
- [Enhance Performance](#) 25
- [Convert Files Out of Process](#) 25
- [Convert Files](#) 32
- [Subfile Extraction](#) 33
- [Set Conversion Options](#) 33
- [Use the Export Demo Program](#) 36
- [Use the C-Language Implementation of the API](#) 39
- [Use the Verity Document Type Definition \(DTD\)](#) 42

Architectural Overview

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

XML 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 XML Export API through either a C-language, Java 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 45 .
XML Export API	The XML Export API exposes the functionality of XML Export and controls all other XML Export modules during the conversion process.
Format Detection Module	The format detection module determines the file type of the source file, which enables the XML Export interface to load the appropriate structured access layer module and document reader. See File Format Detection, on page 350 .
Structured Access Layer	<p>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:</p> <ol style="list-style-type: none">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 XML writer.5. Accepts the XML stream from the XML writer.6. Generates the XML output file with a table of contents, metadata, and the document's contents, and sends it to the XML 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 <i>token stream</i> 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 401 for a complete list of document readers.
HTML Writers	Each XML writer accepts a text stream or token stream from the structured access layer and generates an equivalent XML 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 401 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 181](#). 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 `xmlcallback` sample program demonstrates Export memory management features.

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 ([fplnit\(\)](#), on [page 153](#)). 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

Other Export API functions and the File Extraction functions 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 <i>file-size threshold</i> . 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 unit = seconds default = 180 range = 15~600	The length of time to wait for the Servant to connect to the application 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	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

Parameters for Out-of-Process Conversion, continued

Parameter	Description
default = 10 range = 5~30	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	<p>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.</p> <p>To calculate the <i>total retry interval</i>, the value set here is added to the platform-specific TCP retry value (on Windows, this is 1 second).</p>
ConnectRetry type = integer default = 120 range = 30~600	<p>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:</p> $\text{ConnectRetryInterval} + \text{platform-specific_TCP_retry_value} * \text{ConnectRetry}$ <p>For example, if the <code>ConnectRetryInterval</code> 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:</p> $2 + 1 * 120 = 360$ <p>The Servant would attempt to connect to the application every 3 seconds for 120 attempts for a total of 360 seconds.</p>
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 26](#).
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 `KVXMLEndOOPSession()` 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 in the C API

The `cnv2xmlloop` 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 26](#).
2. Declare instances of the following types and assign values to the members as required:

```
KVXMLTemplateEx  
KVXMLOptionsEx  
KVXMLHeadingInfo  
KVXMLTOCOptions
```

See [XML Export API Structures, on page 179](#) for more information.

3. Load the KVXML library and obtain the `KVXMLInterface` entry point by calling `KVXMLGetInterface()`.
See [KVXMLGetInterface\(\), on page 137](#).
4. Initialize an Export session by calling `fpInit()`. See [fpInit\(\), on page 153](#).
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 142.
 - b. Create an output stream (KVOutputStream) by calling `fpFileToOutputStreamCreate()`. See [fpFileToOutputStreamCreate\(\)](#), on page 143.
 - c. Proceed to [Step 6](#).
6. Set up an out-of-process session by calling `KVXMLStartOOPSession()`.

[KVXMLStartOOPSession\(\)](#), on page 170. 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`.
- Sets conversion options in the `KVXMLTemplate`, `KVXMLOptions`, and `KVXMLTOCOptions` 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—KVXMLStartOOPSession](#), below, and [KVXMLStartOOPSession\(\)](#), on page 170.

7. Convert the input and generate the output files by calling `KVXMLConvertFile()` or `fpConvertStream()`. The `KVXMLTemplate`, `KVXMLOptions`, and `KVXMLTOCOptions` structures are defined in the call to `KVXMLStartOOPSession()`, and should be `NULL` in the conversion call. A conversion function can be called only once in a single out-of-process session. See [KVXMLConvertFile\(\)](#), on page 164, and [fpConvertStream\(\)](#), on page 139.
8. Terminate the out-of-process session by calling `KVXMLEndOOPSession()`. The Servant ends the current conversion session, and releases the source data and session resources. See sample code in [Example—KVXMLEndOOPSession](#), on the next page, and [KVXMLEndOOPSession\(\)](#), on page 167.
9. If you used streams, free the memory allocated for the input stream and output stream by calling the `fpFileToInputStreamFree()` and `fpFileToOutputStreamFree()` functions. See [fpFileToInputStreamFree\(\)](#), on page 143 and [fpFileToOutputStreamFree\(\)](#), on page 144.
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 `KVXMLEndOOPSession()` 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 156.

Example—KVXMLStartOOPSession

The following sample code is from the `cnv2xmlloop` sample program:

```
/* declare OOP startsession function pointer */
KVXML_START_OOP_SESSION fpKVXMLStartOOPSession;
/* assign OOP startsession function pointer */
fpKVXMLStartOOPSession = (KVXML_START_OOP_SESSION)mpGetProcAddress
                        (hKVXML, "KVXMLStartOOPSession");
if(!fpKVXMLStartOOPSession)
{
    printf("Error assigning KVXMLStartOOPSession pointer\n");
    (*KVXMLInt.fpFileToInputStreamFree)(pKVXML, &Input);
    (*KVXMLInt.fpFileToOutputStreamFree)(pKVXML, &Output);
    mpFreeLibrary(hKVXML);
    return 7;
}
/*****START OOP SESSION *****/
if(!(*fpKVXMLStartOOPSession)(pKVXML,
    &Input,
    NULL,
    &XMLTemplates,      /* Markup and related variables */
    &XMLOptions,         /* Options */
    NULL,               /* TOC options */
    &oopServantPID,
    &error,
    0,
    NULL,
    NULL))
{
    printf("Error calling fpKVXMLStartOOPSession \n");
    (*KVXMLInt.fpShutDown)(pKVXML);
    mpFreeLibrary(hKVXML);
    return 9;
}
```

Example—KVXMLEndOOPSession

The following sample code is from the `cnv2xmlloop` sample program:

```
/* declare endsession function pointer */
KVXML_END_OOP_SESSION fpKVXMLEndOOPSession;
/* assign OOP endsession function pointer */
fpKVXMLEndOOPSession = (KVXML_END_OOP_SESSION)mpGetProcAddress
                        (hKVXML, "KVXMLEndOOPSession");
if(!fpKVXMLEndOOPSession)
{
    printf("Error assigning KVXMLEndOOPSession pointer\n");
    (*KVXMLInt.fpFileToInputStreamFree)(pKVXML, &Input);
    (*KVXMLInt.fpFileToOutputStreamFree)(pKVXML, &Output);
    mpFreeLibrary(hKVXML);
    return 8;
}
```

```
/******END OOP SESSION, DO NOT KEEP SERVANT ALIVE *****/
if(!(*fpKVXMLEndOOPSession)(pKVXML,
    FALSE,
    &error,
    0,
    NULL,
    NULL))
{
    printf("Error calling fpKVXMLEndOOPSession \n");
    (*KVXMLInt.fpShutDown)(pKVXML);
    mpFreeLibrary(hKVXML);
    return 10;
}
```

Convert Files

KeyView Export SDK enables you to *convert* many different types of documents to XML. Converting is the process of extracting the text from a document without the application-specific markup, and applying XML 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 XML output. See [Set Conversion Options, on the next page](#).
- 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 71](#).
- Extracting metadata—extracts selected metadata (document properties) from a file. See [Extract Metadata, on page 68](#).
- Converting character sets—controls the character set of both the input and the output text. See [Convert Character Sets, on page 71](#).
- Implementing callbacks—controls the conversion while it is in progress. See [XML Export API Callback Functions, on page 174](#).

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 36](#).
- 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 39](#).
- Use the C sample programs. See [Sample Programs, on page 102](#).

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 25.](#)

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 XML Export API to convert the file.

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

See [Use the File Extraction API, on page 45](#) 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 XML output. For example, you can specify the markup inserted at the beginning and end of specific XML 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:

- [KVXMLTemplate, on page 203](#)
- [KVXMLOptions, on page 195](#)

- [KVXMLHeadingInfo](#), on page 188
- [KVXMLTOCOptions](#), on page 207

The conversion options are described in [XML Export API Structures](#), on page 179.

Set Conversion Options by Using the API

Set conversion options by using any of the following functions:

- [fpConvertStream\(\)](#), on page 139
- [KVXMLConvertFile\(\)](#), on page 164
- [KVXMLStartOOPSession\(\)](#), on page 170

Set Conversion Options by Using the Template Files

XML 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:

```
[KVXMLOptions]
eOutputCharSet=KVCS_SJIS
bForceOutputCharSet=TRUE
```

To create valid XML, a template file *must* contain two structures: KVXMLTemplateEx and KVXMLOptionsEx.

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

An application must then read the template file and write the data to the appropriate Export structures. In the sample program `xmlini`, a template file is supplied as a command-line argument (see [xmlini](#), on page 107).

Templates

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

Template	Description
Cascading style	This template writes style sheet information to an external CSS file. This makes

Template	Description
)	<p>the XML output significantly smaller because the information is not stored in the output file.</p> <p>See Use Style Sheets, on page 78 and Use Style Sheets with xslini, on page 108 for more information on using an external CSS file.</p>
Index (xml_index.ini)	<p>Converts a source document into a single, largely unformatted XML file that is appropriate for use with an indexing engine.</p>
Single file (xml1file.ini)	<ul style="list-style-type: none"> Creates a single XML file. Does not define an XSL style sheet. A default XSL style sheet that is appropriate to the source document type is used. The defaults supplied are wp.xsl (for word processing documents), ss.xsl (for spreadsheets), pg.xsl (for presentations). Forces the output character set to UTF-8. Maintains the source document's fonts and styles. Does not create a table of contents.
Single file for presentations (xml1file_pg.ini)	<p>This template is designed specifically for presentation formats.</p> <ul style="list-style-type: none"> Creates a single XML file. Defines an XSL style sheet for presentations (pg.xsl). Forces the output character set to UTF-8. Because XML Export only extracts textual components from presentations, the bRasterizeFiles member of KVXMLOptions is set to FALSE. KVXMLOptions, on page 195. Only the szMainTop, szMainBottom, and szUserSummary parameters of the KVXMLTemplate structure are relevant to presentations and are set in the presentations template. A template file for presentations must not include any other parameters in the KVXMLTemplate structure. KVXMLTemplate, on page 203.
Single file with table of contents (xml1filetoc.ini)	<ul style="list-style-type: none"> Creates a single XML file. Creates a table of contents at the top of the XML document. Uses the Verity.dtd. Uses an XSL style sheet (wp.xsl). Forces the output character set to UTF-8. Lists all metadata (Title, Subject, Author, Comments, Created, Modified, Last Saved By, and Revision Number). Uses the name of the worksheets for spreadsheets.

Template	Description
	<ul style="list-style-type: none">Uses the slide titles for presentations. If no titles are available in the source document, it uses "slide 1," "slide 2," "slide 3," and so on.

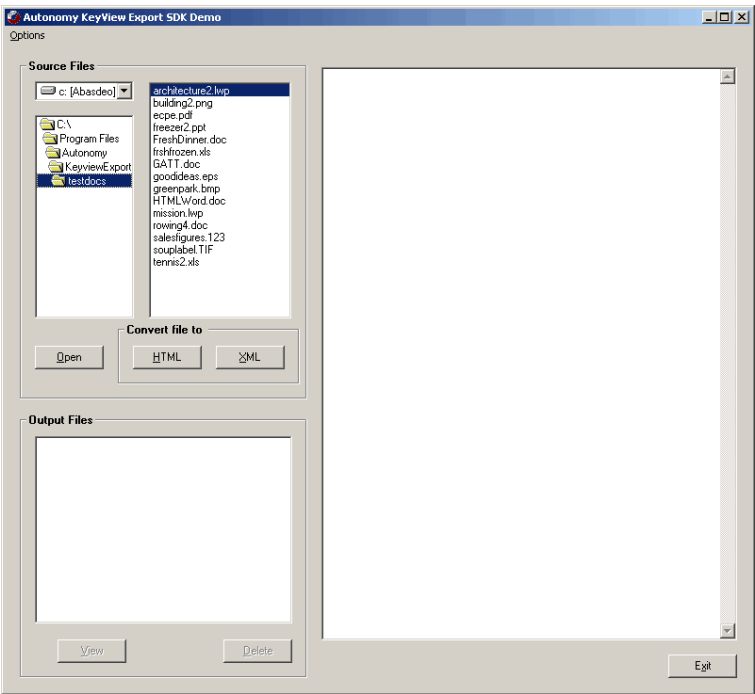
Use the Export Demo Program

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\xmlexport\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.

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

Export Demo: Launching



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

Change Input/Output Directories

If XML 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*.

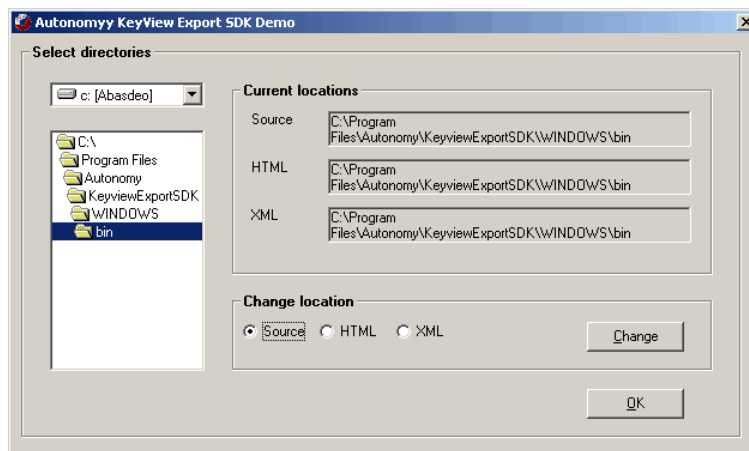
The default location for output files is the directory *install\xmlexport\programs\tempout*.

If XML 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 **XML**.
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 XML Export, you can configure options prior to the document conversion by using the `XMLConfig()` function. Export Demo demonstrates this function, and allows you to:

- Generate output with verbose markup and without images.
- Include position information in the markup generated for a PDF document.

Suppress Images

Export Demo provides an option to generate output with verbose markup and without images. For more information, see [KVXMLConfig\(\)](#), on page 156.

To specify that images are suppressed in the XML output, select **Options | XML Config | Suppress Images**.

Use PDF Position Information

Export Demo provides an option to include position information in the markup generated for a PDF document. For more information, see [KVXMLConfig\(\)](#), on page 156.

To specify that PDF position information be included in the XML output, select **Options | XML Config | Enable Position Token**.

Convert Files

To convert a single file

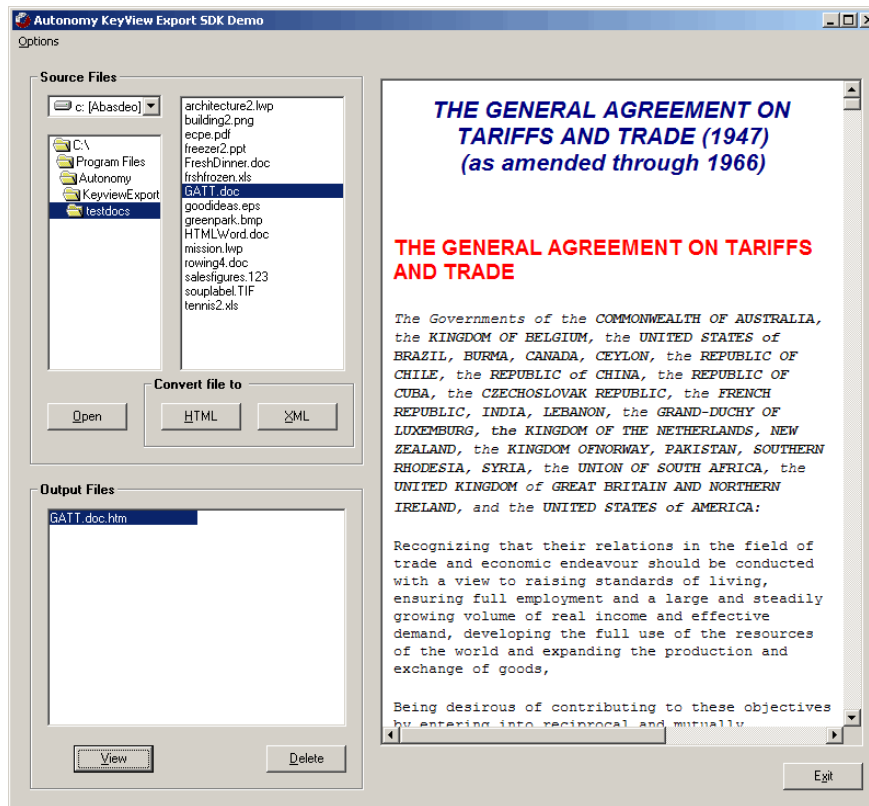
1. Select **Options | Convert | Single file**.
2. Select the document from the file list, and then click **XML** in the **Convert file to** pane.

To convert files in a directory

1. Select **Options | Convert | Entire directory**.
2. Click **XML** 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 111](#)—Open and extract subfiles in a container file. These functions also extract metadata and file format information, and control character set conversion on extraction.
- [XML Export API Functions, on page 137](#)—Extract format information (metadata, character set, and format), create an input/output stream from a file, and open, convert, and close the stream.
- [XML Export API Callback Functions, on page 174](#)—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 180](#) and [KVOutputStream, on page 181](#).

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 142](#) and [fpFileToOutputStreamCreate\(\), on page 143](#).

Convert Files

To use the C-language implementation of the API

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

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

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

```
KVXMLTemplateEx  
KVXMLOptionsEx  
KVXMLHeadingInfo  
KVXMLTOCOptions
```

See [XML Export API Structures, on page 179](#) for more information.

3. Load the `KVXML` library and obtain the `KVXMLInterface` entry point by calling `KVXMLGetInterface()`. See [KVXMLGetInterface\(\), on page 137](#).
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 153](#).
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 111](#).
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 142](#).
 - 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 143](#).
 - c. Proceed to [Step 7](#).
7. Declare the input stream or file name in the `KVOpenFileArg` structure. See [KVOpenFileArg, on page 129](#).

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 119](#).
9. Determine whether the source file is a container file (contains subfiles) by calling `fpGetMainFileInfo()`. See [fpGetMainFileInfo\(\), on page 115](#).
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 116](#).
12. Extract the subfile by calling `fpExtractSubFile()`. See [fpExtractSubFile\(\), on page 112](#).
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 `KVXMLStartOOPSession()`. See [KVXMLStartOOPSession\(\), on page 170](#).
15. Convert the input and generate the output files by calling `KVXMLConvertFile()` or `fpConvertStream()`. The structures `KVXMLTemplate`, `KVXMLOptions`, and `KVXMLTOCOptions` are defined in the call to `KVXMLStartOOPSession()`, and should be `NULL` in the conversion call. A conversion function can be called only once in a single out-of-process session. See [fpConvertStream\(\), on page 139](#) or [KVXMLConvertFile\(\), on page 164](#).

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 [XML Export API Callback Functions, on page 174](#).
16. If you are converting additional files, terminate the out-of-process session by calling `KVXMLEndOOPSession()` 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 `KVXMLEndOOPSession()` and setting the Boolean to `FALSE`. See [KVXMLEndOOPSession\(\), on page 167](#)
17. Close the file by calling `fpCloseFile()`. See [fpCloseFile\(\), on page 112](#).
18. If you used streams, free the memory allocated for the input stream and output stream by calling the functions `fpFileToInputStreamFree()` and `fpFileToOutputStreamFree()`. See [fpFileToInputStreamFree\(\), on page 143](#) and [fpFileToOutputStreamFree\(\), on page 144](#).
19. Repeat [Step 6](#) through [Step 18](#) for additional source files.
20. Shutdown the Export session by calling `fpShutDown()`. See [fpShutDown\(\), on page 156](#).

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
        fpGetSubFileMetadata()
        KVXMLStartOOPSession()
        fpConvertStream()
        KVXMLEndOOPSession(bKeepServantAlive TRUE)
        fpCloseFile()
    fpFileToInputStreamFree()
    fpFileToOutputStreamFree()
        set input/output file
        fpOpenFile()
        fpGetMainFileInfo()          /* not a container file */
        KVXMLStartOOPSession()
        KVXMLConvertFile()
        KVXMLEndOOPSession(bKeepServantAlive TRUE)
        fpCloseFile()
    ...
fpShutdown()
```

Use the Verity Document Type Definition (DTD)

XML Export produces well-formed, valid XML documents. Document validity is based on a Document Type Definition (DTD) called the `Verity.dtd`. The `Verity.dtd` is in the default output directory `tempout`. If the DTD is in a different directory, the full path must be specified in `pszVerityDTDPATH`.

The elements in the `Verity.dtd` are based on those defined in the W3C XHTML 1.0 specification and the attributes are based on those defined in the W3C CSS 2 specification.

The root element of each document is `"VerityXMLExport"`. Character entities are imported by using the three XHTML DTDs defined at the beginning of the `Verity.dtd`.

```
<!-- Character entities -->
<!ENTITY % HTMLlat1x SYSTEM "HTMLlat1x.ent">
%HTMLlat1x;
<!ENTITY % HTMLspecialx SYSTEM "HTMLspecialx.ent">
%HTMLspecialx;
<!ENTITY % HTMLsymbolx SYSTEM "HTMLsymbolx.ent">
%HTMLsymbolx;
```

Use XML Style Language Transformation (XSLT)

XML Export is designed to generate XML documents based on the Verity DTD. You can convert the XML produced by XML Export to other XML vocabularies, such as Wireless Markup Language (WML), by using XSLT.

Add Elements and Attributes to the DTD

XML Export can only generate XML that conforms to the Verity DTD. You can create your own DTD based on the Verity DTD. You cannot rename the Verity DTD, so make sure you back up the original Verity DTD to another name before making changes.

If you create your own DTD and add elements or attributes that are not defined in the original Verity DTD, you must ensure that the new markup is defined in the XML Export API classes. You can define the markup either by entering the markup directly in the styles, or by populating the styles by using the template files. See [Map Styles, on page 75](#) for more information on mapping styles to user-defined markup.

Move the DTD

The default output directory for the Verity DTD is `programs\tempout`. If you move the Verity DTD to another output directory, you must set the string value of `pszVerityDTDPath` to the new location. This path is added to the document type declaration in the XML file. [pszVerityDTDPath, on page 196](#).

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:

- [Use the File Extraction API, on page 45](#)
- [Use the XML Export API, on page 68](#)
- [Sample Programs, on page 102](#)

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	45
• Extract Subfiles	46
• Extract Images	48
• Recreate a File's Hierarchy	48
• Extract Mail Metadata	49
• Extract Subfiles from Outlook Files	56
• Extract Subfiles from Outlook Express Files	56
• Extract Subfiles from Mailbox Files	57
• Extract Subfiles from Outlook Personal Folders Files	57
• Extract Subfiles from Lotus Domino XML Language Files	61
• Extract Subfiles from Lotus Notes Database Files	61
• Extract Subfiles from PDF Files	64
• Extract Embedded OLE Objects	65
• Extract Subfiles from ZIP Files	65
• Default File Names for Extracted Subfiles	66

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.

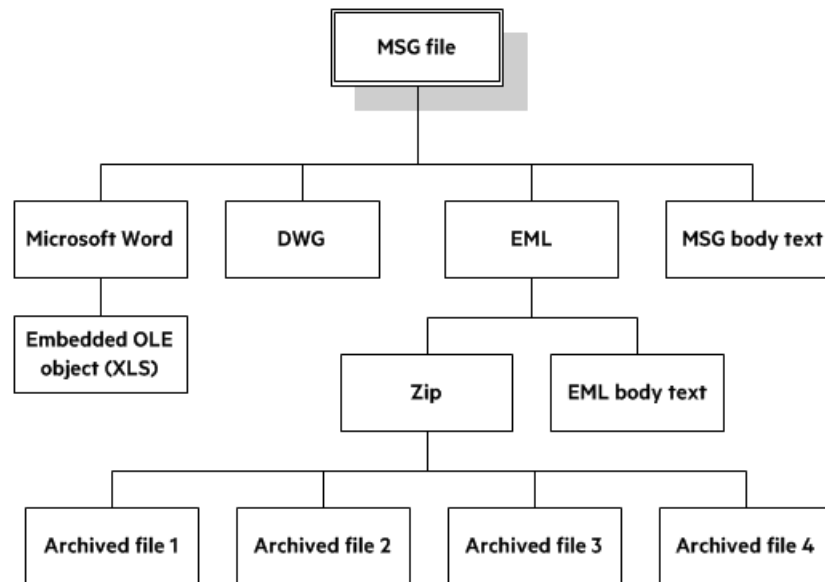
The following are examples of container files:

- 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 230](#) 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 116. However, if the path is an absolute path, the file could be created outside the directory you have chosen as the extract directory. Your application might then contain a vulnerability that could be exploited to write files to unexpected locations in the file system. This section discusses some KeyView features that can help you secure your application by sanitizing paths.

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

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

The following table shows the effect on some example paths.

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

To sanitize absolute paths

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

To sanitize absolute paths (through configuration)

- In the `formats_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.

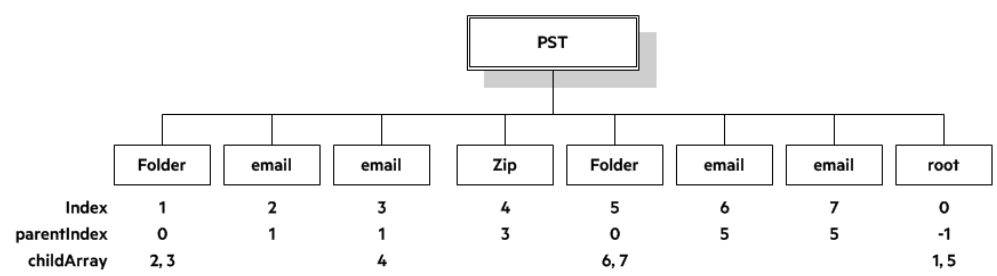
Create a Root Node

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.

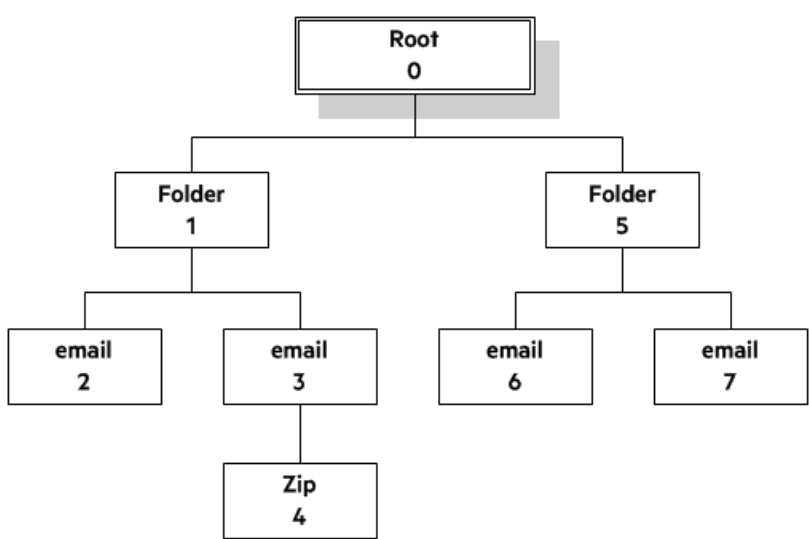
To create the root node, set `openFlag` to `KVOpenFileFlag_CreateRootNode` in the call to `fpOpenFile()`. When you create a root node, the value of `numSubFiles` in `KVMainFileInfo` includes the root node. For example, when you call `fpGetMainFileInfo()` on a Microsoft Word document with three embedded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

Default Metadata Set

KeyView internally defines a set of common mail metadata fields that you can extract as a group from mail formats. This default metadata set is listed in the following table. When you retrieve *all* metadata for a file—that is, pass `NULL` for the array of metadata—the complete set of default metadata, not all available metadata in the file, is returned.

Default Mail Metadata List

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

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

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

NOTE: You cannot pass the field names listed in the table individually for PST files. However, you can pass either the MAPI tag number or the MAPI tag name as integers. See [Microsoft Personal Folders File \(PST\) Metadata, on page 54](#).

Extract the Default Metadata Set

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

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

```
metaArg.index = subFileIndex;
```

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

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

Microsoft Outlook (MSG) Metadata

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

MSG-specific Metadata List

Field Name (string to specify)	Description
AttachFileName	An attachment's long file name and extension, excluding the path.
ConversationTopic	The topic of the first message in a conversation thread. A conversation thread is a series of messages and replies. This is the first message's subject with any prefix removed.
CreationTime	The time that the message or attachment was created. This value is displayed in the Sent field in the message's Properties dialog in Outlook.
InternetMessageID	The identifier for messages that come in over the Internet. This is the MAPI property <code>PR_INTERNET_MESSAGE_ID</code> . This property is not in the MAPI headers or MAPI documentation.
LastModificationTime	The time that the message or attachment was last modified. This value is displayed in the Modified field in the message's Properties dialog in Outlook.
Location	The physical location of the event specified in the Outlook calendar entry.
MessageID	The message transfer system (MTS) identifier for the message transfer agent (MTA). This value is displayed on the Message ID tab in the message's Properties dialog in Outlook.
Received	The date and time a message was delivered. This value is displayed in the Received field in the message's Properties dialog in Outlook.
Sender	<p>The name and email address of the message sender. This value is a concatenation of two MAPI properties in the following format:</p> <p>"PR_SENDER_NAME" <PR_SENDER_EMAIL_ADDRESS></p> <p>The Sender value might be the same as or different than the default</p>

MSG-specific Metadata List, continued

Field Name (string to specify)	Description
	metadata From value (see Default Metadata Set, on page 50), 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 50](#) 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, above](#).) If the Outlook file contains a non-mail attachment, the attachment is extracted in its native format to the same directory as the message text file. If the Outlook file contains a mail attachment, the complete attachment (including message text and attachments), the message text file, and any non-mail attachments are extracted to the same directory as the main message.

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

Extract Subfiles from Mailbox Files

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

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

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

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

The Mailbox (MBX) reader is an advanced feature and is sold and licensed separately. To enable this reader in a KeyView SDK, you must obtain the appropriate license key from Micro Focus. See [Update License Information, on page 19](#) for information on adding a new license key to an existing installation.

Extract Subfiles from Outlook Personal Folders Files

KeyView can extract Outlook items such as messages, appointments, contacts, tasks, notes, and journal entries from a PST file. When a PST file is extracted to disk, the text and header information (To, From, Sent, and so on) from each Outlook item is extracted to a text file. (If you do not want the header information to appear in the text file, see [Exclude Metadata from the Extracted Text File, on the previous page.](#))

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

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

NOTE: The Microsoft Outlook Personal Folders (PST) reader is an advanced feature and is sold and licensed separately. To enable this reader in a KeyView SDK, you must obtain the appropriate license key from Micro Focus. See [Update License Information, on page 19](#) for information on adding a new license key to an existing installation.

Use the Native or MAPI-based Reader

KeyView accesses PST files in one of two ways:

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

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

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

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

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

```
297=pst
```

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

```
297=pstn
```

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

Use the Native PST Reader (pstnsr)

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

Use the MAPI Reader (pstsr)

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

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

System Requirements

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

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

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

NOTE: The bit edition of Microsoft Outlook must match that of the KeyView software. For example, if 32-bit KeyView is used, 32-bit Outlook must be installed. If 64-bit KeyView is used, 64-bit Outlook must be installed.

If the bit editions do not match, an error message from Microsoft Office Outlook is displayed:

Either there is a no default mail client or the current mail client cannot fulfill the messaging request. Please run Microsoft Outlook and set it as the default mail client.

Additionally, KeyView displays the following return code:

Error 32: `KVError_PSTAccessFailed`.

MAPI Attachment Methods

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

`ATTACH_BY_VALUE`

`ATTACH_EMBEDDED_MSG`

`ATTACH_OLE`

`ATTACH_BY_REFERENCE`

`ATTACH_BY_REF_ONLY`

`ATTACH_BY_REF_RESOLVE`

Attachments using the `ATTACH_BY_VALUE`, `ATTACH_EMBEDDED_MSG`, or `ATTACH_OLE` attachment methods are extracted automatically when the PST file is extracted. An "attach by reference" method

means that the attachment is not in Outlook, but Outlook contains an absolute path to the attachment. Before you can extract these types of attachments, you must retrieve the path to access the attachment.

To extract "attach by reference" attachments

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 reader (`pstnsr`) returns the error message `KVERR_PasswordProtected` if a PST is encrypted with high encryption.

Detect PST Files While the Outlook Client is Running

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

- Close Outlook when using the Extraction API.
- Detect PST files by extension only and bypass the format detection module. To enable this option, add the following lines to the `formats_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 56](#).

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]
LNParser=2
```

Extract Subfiles from Lotus Notes Database Files

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

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

When a mail message in an NSF file is extracted to disk, the body text and header information (such as the values from the `SendTo`, `From`, and `DeliveredDate` fields) in each message is extracted to a text file. (If you do not want the header information to appear in the message text file, see [Exclude Metadata from the Extracted Text File, on page 56.](#))

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 19](#) 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_LIBRARY_PATH
```

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

AIX 5.x

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

[Notes]

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

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

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

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

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

Linux

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

```
[Notes]
```

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

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

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

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

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

Open Secured NSF Files

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

Format Note Subfiles

The KeyView NSF reader uses XML templates to format note subfiles. You can customize the templates to approximate the look and feel of the original notes as closely as possible. For more information, see [Extract and Format Lotus Notes Subfiles, on page 334](#).

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 XML Export API, the main file is converted to XML and the OLE object is converted to a graphics file that is referenced in the XML 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 46](#)). 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 `_kvn`, 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	.jrn1
distribution list	.dist
posting note	.post

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

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

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

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

Default File Name for Embedded OLE Objects

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

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

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

Chapter 4: Use the XML Export API

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

• Extract Metadata	68
• Extract File Format Information	71
• Convert Character Sets	71
• Map Styles	75
• Use Style Sheets	78
• Display Vector Graphics on UNIX and Linux	79
• Convert Revision Tracking Information	80
• Convert PDF Files	81
• Convert Spreadsheet Files	88
• Convert Presentation Files	91
• Convert XML Files	92
• Show Hidden Data	97
• Exclude Japanese Guide Text	99
• Obtain Image Info	99
• Source Code Identification	100

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 49](#).

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 230](#) 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 185.
2. Call the `fpGetSummaryInfo()` function. See [fpGetSummaryInfo\(\)](#), on page 152.

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 `KVXMLTemplate` structure in the template file. This defines the point at which the metadata appears in the XML output.
2. If you are using the `$USERSUMMARY` or `$SUMMARY` token, define the `szUserSummary` member of the `KVXMLTemplate` 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 `KVXMLTemplate` structure.
See [xmlini](#), on page 107.

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

Token	Description
<code>\$SUMMARYNN</code>	Inserts the data from a <i>specified</i> metadata field. <i>NN</i> is a number from 00 through 42 enumerated in <code>KVSumType</code> in <code>kvtypes.h</code> .
<code>\$SUMMARY</code>	Inserts the data from valid metadata fields in the range of 0 to 33 using the markup provided in <code>pszUserSummary</code> .
<code>\$USERSUMMARY</code>	Inserts the data from <i>every</i> valid non-standard metadata field using the markup provided in <code>pszUserSummary</code> .
<code>\$CONTENT</code>	Inserts the content of the metadata field specified by the <code>\$NAME</code> token.
<code>\$NAME</code>	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

\$SUMMARYNN

The following markup displays the contents of the "Title" field at the top of the main XML 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 H1 XML block:

```
szFirstH1Start=$SUMMARY
```

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

This example extracts the field name (\$NAME) and field content (\$CONTENT) for standard metadata and includes it at the beginning of the first heading level 1 XML block.

The generated XML might look like this:

```
<MetaData name="CodePage" content="1252" \>
<MetaData name="Title" content="My design document" \>
<MetaData name="Subject" content="design specifications" \>
<MetaData name="Author" content="John Doe" \>
<MetaData name="Keywords" content="" \>
<MetaData name="Comments" content="" \>
<MetaData name="Template" content="Normal.dot" \>
<MetaData name="LastAuthor" content="lchapman" \>
<MetaData name="RevNumber" content="6" \>
<MetaData name="EditTime" content="01/01/1601, 0:08" \>
<MetaData name="LastPrinted" content="14/01/2002, 14:06" \>
<MetaData name="Create_DTM" content="27/08/2003, 10:31" \>
<MetaData name="LastSave_DTM" content="29/08/2003, 14:07" \>
<MetaData name="PageCount" content="1" \>
<MetaData name="WordCount" content="4062" \>
<MetaData name="CharCount" content="23159" \>
<MetaData name="AppName" content="Microsoft Word 9.0" \>
<MetaData name="Security" content="0" \>
<MetaData name="Category" content="software" \>
<MetaData name="LineCount" content="192" \>
<MetaData name="ParCount" content="46" \>
<MetaData name="ScaleCrop" content="FALSE" \>
<MetaData name="Manager" content="" \>
<MetaData name="Company" content="Autonomy" \>
<MetaData name="LinksDirty" content="FALSE" \>
```

\$USERSUMMARY

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

```
szMainBottom=$USERSUMMARY
```

```
szUserSummary=<MetaData 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 XML file.

The generated XML might look like this:

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

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 350](#) 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 182](#).
2. Call the `fpGetStreamInfo()` function. [fpGetStreamInfo\(\), on page 151](#).

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 328](#) 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 230](#) 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 `KVXMLOptions` structure to `TRUE`.

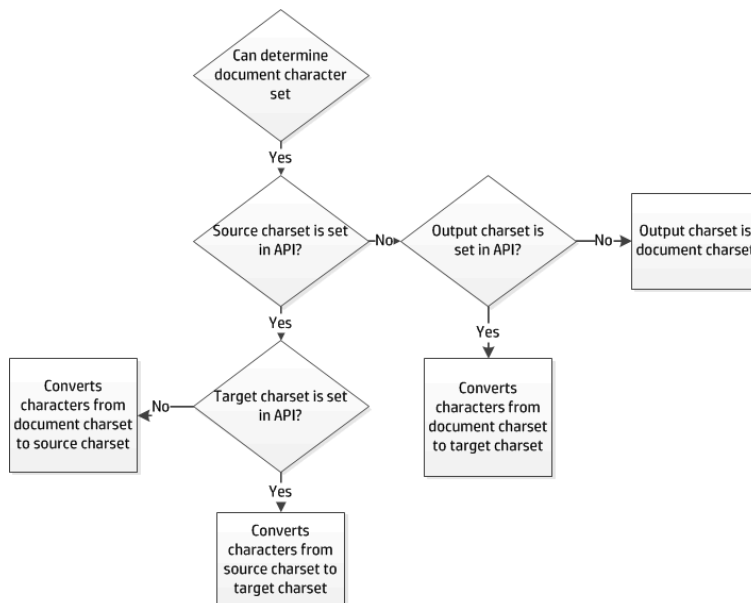
- Whether a target character set is set in the API.

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

Guidelines for Character Set Conversion

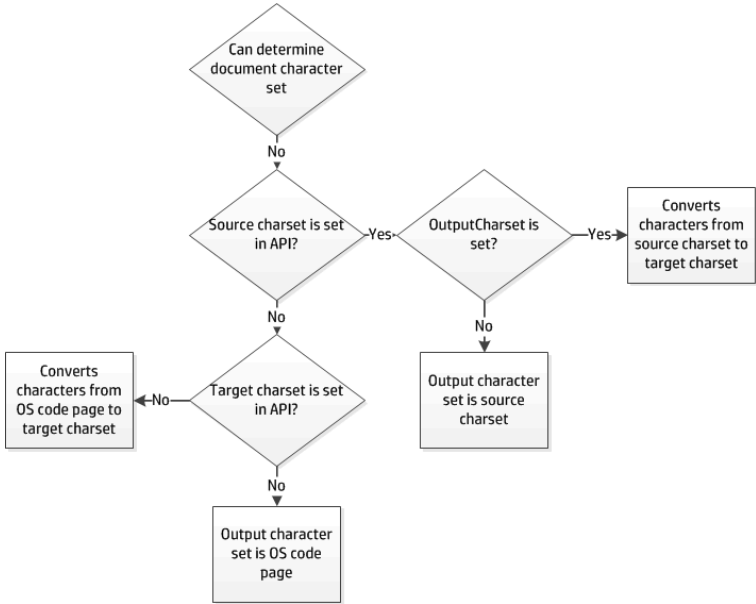
The following diagram shows how the output character set is determined when the document character set can be determined:

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 253](#) 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.

Document character set can be determined, continued

Source charset set	Target charset set	Output charset
--	--	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 253](#) 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.
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 `KVXMLOptions` structure to one of the character sets enumerated in `KVCharSet` in `kvtypes.h`. See [KVXMLOptions, on page 195](#).
2. Set the `bForceSrcCharSet` member of the `KVXMLOptions` structure to `TRUE`. See [KVXMLOptions, on page 195](#).

To specify the target character set

1. Set the `eOutputCharSet` member of the `KVXMLOptions` structure to one of the character sets enumerated in `KVCharSet` in `kvtypes.h`. See [KVXMLOptions, on page 195](#).
2. Set the `bForceOutputCharSet` member of the `KVXMLOptions` structure to `TRUE`. See [KVXMLOptions, on page 195](#).

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 XML. This is most often used to set the output character set of a mail message's body text. See [Use the File Extraction API, on page 45](#).

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 112](#).

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. 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 XML.

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 XML 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 XML containing:
`<recipe>Brownies</recipe>`.

You can also use style mapping to control the look of the XML 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 XML output.

If you map styles to elements or attributes that are not defined in the DTD, you must add the new elements or attributes to the DTD. You must also ensure the new markup is defined in the API, either by entering the markup directly in the classes, or by populating the classes using the template files.

Use the C API

To map styles by using the C API

1. Define the `KVStyle` structure. See [KVStyle](#), on page 184. 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 the next page.
2. Call the `fpSetStyleMapping()` function. See [fpSetStyleMapping\(\)](#), on page 155.

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 184.

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 184. 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
```

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]	<p>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.</p> <p>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 <i>all</i> heading levels. Export uses heading levels to define the overall structure of the XML output.</p>
KVSTYLE_ORDERLIST	The KVSTYLE_ORDERLIST flag specifies that the style should be tagged as an ordered list. Currently not implemented.
1. KVSTYLE_UNORDEREDLIST	The KVSTYLE_UNORDEREDLIST flag specifies that the style should

Flags for Defining Styles, continued

Flag	Description
	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.
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 XML output with a selected character. The default replacement character is "X," but you can specify a different replacement character by setting <code>cRedact</code> .

Use Style Sheets

XML is a content-based metalanguage designed to structure data. XML does not include information about how a document should be displayed in a browser. To view an XML document in a browser, information about how its displayed must be provided by style sheets. These are coded by using either Cascading Style Sheets (CSS) or Extensible Style Sheet Language (XSL).

The style sheet options are enumerated in `KVXMLStyleSheetType`.

Use Extensible Style Sheet Language (XSL)

You can use XSL style sheets to specify how XML data is displayed in a browser. Existing XSL style sheets can be used, but unlike CSS, style sheet information cannot be written to an external XSL file during the conversion.

Both CSS and XSL style sheets can be used to format XML documents. However, XSL can also transform XML documents. For example, list items can be transformed to display in alphabetical order, words can be replaced by other words, or empty elements can be replaced by text.

To use an existing XSL style sheet

1. Set `eStyleSheetType` to `XML_XSL` to enable XSL style sheet mapping.
2. Set `bUseExistingStyleSheet` to `TRUE` to apply a pre-existing style sheet to an XML document. Pre-existing style sheets are not validated.
3. Specify the path and file name of the style sheet file in `pszStyleSheet`.

If you set `bUseExistingStyleSheet` to `TRUE` and do not specify `pszStyleSheet`, a default XSL style sheet that is appropriate for the source document type is used.

The following are default XSL style sheets:

- `wp.xsl` (for word processing documents)
- `ss.xsl` (for spreadsheets)
- `pg.xsl` (for presentation graphics)

Use Cascading Style Sheets (CSS)

In addition to XSL style sheets, Export can write style sheet information to an external CSS file. The C sample program `xmlini` provides an example of how to use an existing style sheet, and output formatting data to an external file. See [xmlini](#), on page 107.

To enable CSS mapping and output the resulting formatting data in an external file

1. Set `eStyleSheetType` to `XML_CSS`.
2. Use the `KVXMLSetStyleSheet()` function to set the path and file name of the external style sheet. [KVXMLSetStyleSheet\(\)](#), on page 168.

To enable CSS mapping and use an existing CSS file

1. Set `eStyleSheetType` to `XML_CSS`.
2. Set `bUseExistingStyleSheet` to `TRUE` to specify a pre-existing style sheet for an XML document.
3. Specify the path and file name of the style sheet file in `pszStyleSheet`.

If you set `bUseExistingStyleSheet` to `TRUE` and do not specify `pszStyleSheet` or `SetExternalStyleFile`, a CSS style sheet is created.

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

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 `eOutputVectorGraphicType` member of the `KVXMLOptions` structure, and the options are enumerated in `KVXMLGraphicType` in `kvxml.h`. [KVXMLOptions](#), on page 195 and [KVXMLGraphicType](#), on page 218.

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 XML. The graphics in the document are converted to JPEG and stored in the output directory.

NOTE:

`kvvector.jar` must reside in the output directory.

Convert Revision Tracking 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, Export can be configured to convert the deleted text and graphics and include revision tracking information in the XML output. (The deleted content and revision tracking information is excluded from the XML output by default.)

Content that was added to the document is identified by `<ins>` tags. Content that was deleted from the document is identified by `` tags. The `<ins>` and `` tags include `cite` and `datetime` attributes

which define the name of the reviewer who made the change and the date the change was made respectively. (The date is in ISO-8601 format: YYYY-MM-DDThh:mm:ss.) The tags also include a `title` attribute which allows you to display the author and date information in a browser. These elements are included in the `verity.dtd`.

The following markup is generated for inserted text:

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

The following markup is generated for deleted text:

```
<del title="Deleted: JohnD, 2006-04-24T14:56:00" cite="mailto:JohnD"
datetime="2006-04-24T14:56:00">This text was deleted</del> in a previous version.
```

To convert deleted text and graphics and include revision tracking information

1. Call the `fpInit()` function. See [fpInit\(\)](#), on page 153.
2. Call the `fpXMLConfig()` function with the following arguments (See [KVXMLConfig\(\)](#), on page 156):

Argument	Parameter
<code>nType</code>	<code>KVCFG_INCLREVISIONMARK</code>
<code>nValue</code>	<code>TRUE</code> (non-zero)
<code>pData</code>	<code>NULL</code>

For example:

```
(*fpXMLConfig)(pKVXML, KVCFG_INCLREVISIONMARK, TRUE, NULL);
```

The `xmlini` sample program demonstrates this function. See [xmlini](#), on page 107.

3. Call the `fpConvertStream()` or `KVXMLConvertFile()` function. See [fpConvertStream\(\)](#), on page 139 or [KVXMLConvertFile\(\)](#), on page 164.

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 XML 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**.

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

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 153.
2. Call the `fpXMLConfig()` function with the following arguments (See [KVXMLConfig\(\)](#), on page 156):

Argument	Parameter
nType	KVCFG_LOGICALPDF
nValue	<p>Set to one of the following flags which are defined in <code>kvtypes.h</code>. (see LPDF_DIRECTION, on page 228):</p> <ul style="list-style-type: none">• <code>LPDF_LTR</code>—Logical reading order and left-to-right paragraph direction.• <code>LPDF_RTL</code>—Logical reading order and right-to-left paragraph direction.• <code>LPDF_AUTO</code>—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.• <code>LPDF_RAW</code>—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:

```
(*fpXMLConfig)(pKVXML, KVCFG_LOGICALPDF, LPDF_RTL, NULL);
```

The `cnv2xml` sample program demonstrates this function. See [cnv2xml](#), on page 104.

3. Call the `fpConvertStream()` or `KVXMLConvertFile()` function. See [fpConvertStream\(\)](#), on page 139 or [KVXMLConvertFile\(\)](#), on page 164.

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=1pdf
```

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
--------------	-----------------------------------

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.
--------------	--

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 XML 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 XML 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 XML 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 XML output

1. Call the `fpInit()` function. See [fpInit\(\)](#), on page 153.
2. Call the `KVXMLConfig()` function, with the following arguments (see [KVXMLConfig\(\)](#), on page 156):

Argument	Parameter
<code>nType</code>	<code>KVCFG_DELSOFTHYPHEN</code>
<code>nValue</code>	<code>TRUE</code> (non-zero)
<code>pData</code>	<code>NULL</code>

For example:

```
(*fpXMLConfig)(pKVXML, KVCFG_DELSOFTHYPHEN, TRUE, NULL);
```
3. Call the `fpConvertStream()` or `KVXMLConvertFile()` function. See [fpConvertStream\(\)](#), on page 139 or [KVXMLConvertFile\(\)](#), on page 164.

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 <code>KVSumInfoType</code> . See KVSumInfoType , on page 223.

For example:

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

```
/customer          STRING  
</META>
```

Configure the Size of Exported Images

When you use the `pdf2sr` reader to export images of the pages in a PDF file, you can configure the size of the images produced by KeyView.

NOTE:

When a page in a PDF document consists of a single embedded image (such as when the PDF is a scanned document), the page image is output at the original size of the embedded image and the following settings have no effect.

To configure the size of images produced by `pdf2sr`

1. Open the configuration file `formats_e.ini`.
2. Find the section `[pdf2sr]`, or create the section if it does not exist.
3. Set the configuration parameters `XMLXRes` and `XMLYRes`. `XMLXRes` specifies the width of the output image and `XMLYRes` specifies the height.
 - To specify an absolute size, in pixels, use positive values. The aspect ratio is always maintained, so you can set one of the dimensions and set the other parameter to 0. For example, to output images of PDF pages where the height of each image is 400 pixels, use the following configuration:

```
[pdf2sr]  
XMLXRes=0  
XMLYRes=400
```

If you set both `XMLXRes` and `XMLYRes` to positive values, KeyView produces the largest image that fits within the specified dimensions (the width or height will be as requested, and the other dimension is smaller than requested if required to preserve the aspect ratio).

- To specify a relative size, set `XMLXRes` to a negative value and `XMLYRes` to 0 (a negative value for `XMLYRes` is ignored). The aspect ratio is always maintained. For example, to output images of PDF pages where the size of each image is 150% of the original size, use the following configuration.

```
[pdf2sr]  
XMLXRes=-150  
XMLYRes=0
```

NOTE:

The default values for `XMLXRes` and `XMLYRes` are shown below. These values produce an image at 113% of the original page size:

```
[pdf2sr]  
XMLXRes=-113  
XMLYRes=0
```

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 `KVXMLOptions`. See [KVXMLOptions](#), on page 195.

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 `KVXMLOptions` 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 350](#) 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 153.
2. Define the `KVXConfigInfo` structure. See [KVXConfigInfo](#), on page 186.
3. Call the `KVXMLConfig()` function with the following arguments (see [KVXMLConfig\(\)](#), on page 156):

Argument	Parameter
----------	-----------

<code>nType</code>	<code>KVCFG_SETXMLCONFIGINFO</code>
<code>nValue</code>	<code>0</code>
<code>pData</code>	address of the <code>KVXConfigInfo</code> structure

For example:

```
KVXConfigInfo xinfo; /* populate xinfo */
(*fpXMLConfig)(pKVXML, 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 function `fpConvertStream()` or `KVXMLConvertFile()`. See [fpConvertStream\(\)](#), on page 139 or [KVXMLConvertFile\(\)](#), on page 164.

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 sample program (`xmlini`) demonstrates how to use a template file during the conversion process. See [xmlini](#), on page 107.

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 350 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 96.
szInMetaElement	The elements extracted from the file as metadata. All other elements are

Configuration Option	Description
	<p>extracted as text.</p> <p>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.</p>
szExMetaElement	<p>The child elements in the included metadata elements that are not extracted from the file as metadata. For example, the default extraction settings for the Visio XML format extract the <code>DocumentProperties</code> element as metadata. This element includes child elements such as <code>Title</code>, <code>Subject</code>, <code>Author</code>, <code>Description</code>, and so on. However, the child element <code>PreviewPicture</code> is defined in <code>szExMetaElement</code> because it is binary data and should not be extracted.</p> <p>You cannot exclude any metadata elements from the output for StarOffice files. All metadata is extracted regardless of this setting.</p> <p>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.</p>
szInContentElement	<p>The elements extracted from the file as content text. Enter an asterisk (*) to extract all elements including child elements.</p> <p>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.</p>
szExContentElement	<p>The child elements in the included content elements that are not extracted from the file as content text.</p> <p>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.</p>
szInAttribute	<p>The attribute values extracted from the file. If attributes are not defined here, attribute values are not extracted.</p> <p>Enter the namespace (if used), element name, and attribute name in the following format:</p> <pre>namespace:elementname@attributename</pre> <p>For example:</p> <pre>keyview:division@name</pre> <p>Multiple entries must be separated by commas.</p>

Specify an Element's Namespace and Attribute

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

```
ns_prefix: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 `KVXMLConfig()` 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 [KVXMLConfig\(\)](#), on page 156 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_SHOWDATEFIELDPCODE
File name field codes	Document file name	KVCFG_WP_SHOWFILENAMEFIELDPCODE
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

¹Word 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](#).

²Shown by default in Microsoft Word 97 to 2003 documents.

³Shown 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_SHOWCOMMENTSSSLIDE 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 [KVXMLConfig\(\)](#), on page 156.

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.

¹This setting affects PowerPoint 2003 and 2007 only.

²PowerPoint 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`.

Obtain Image Info

When exporting from presentation graphics files, and when using the `pdf2sr` reader to export from PDF files, KeyView can obtain information about the number of images that it would create during export, without having to run a full export. This option uses function pointers that are part of the [KVXMLInterfaceEx](#), on page 193 structure.

To extract image information

1. Initialize an image information session by calling the `fpGetOutputImageInfos()` function. You must pass the return value for this function to the other image functions.
2. Call the `fpGetOutputImageCount()` function to get the number of images identified.
3. (Optional) For each image, call `fpGetOutputImageInfo()` to obtain the image dimensions.
4. Free the internal resources associated with the image information session by calling `fpFreeImageInfos()`.

Example

```
int numberOfImages = 0;
void* pImageInfoContext = (*KVXMLInt.fpGetOutputImageInfos)(pContext, &inputStream,
&options);
if (pImageInfoContext == NULL)
{
    // Error handling code would go here.
    // fpGetKvErrorCode() could be called here to investigate.
}
(*KVXMLInt.fpGetOutputImageCount)(pImageInfoContext, &numberOfImages);
for (int i = 0; i < numberOfImages; i++)
{
    KVXMLImageInfo imageInfo;
    KVStructInit(&imageInfo);
    if ((*KVXMLInt.fpGetOutputImageInfo)(pImageInfoContext, i, &imageInfo))
    {
        // imageInfo.nWidth and imageInfo.nHeight
        // contain the dimensions at this point.
    }
}
(*KVXMLInt.fpFreeImageInfos)(pContext, pImageInfoContext);
pImageInfoContext = NULL;
```

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 260](#).

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 XML Export.

• Introduction	102
• tstxtract	103
• cnv2xml	104
• cnv2xmlloop	105
• metadata	106
• xmlindex	106
• xmlini	107
• xmlcallback	108
• xmlonefile	108
• xmlmulti	109
• Export Demo	109

Introduction

The sample programs demonstrate how to use the C and Visual Basic implementations of XML 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\xmlexport\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 XML Export. The following sample programs are provided:

- [tstxtract](#), on the next page
- [cnv2xml](#), on page 104
- [cnv2xmlloop](#), on page 105
- [metadata](#), on page 106
- [xmlindex](#), on page 106
- [xmlini](#), on page 107

- [xmlcallback](#), on page 108
- [xmlonefile](#), on page 108
- [xmlmulti](#), on page 109

You can use the `tstxtract`, `cnv2xml`, `cnv2xmloop`, and `xmlini` 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 XML 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 XML 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\xmlexport\programs\ExportDemo`, where `install` is the path name of the Export installation 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 <code>KVCS_SJIS</code> . See Coded Character Sets , on page 328 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.

Option	Description
-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.
-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.

cnv2xml

The *cnv2xml* sample program creates a single, formatted XML output file. It is called by the Export Demo sample program, but can also be used on its own. This program runs on both Windows and UNIX platforms.

To run *cnv2xml*, type the following at the command line:

```
cnv2xml [options] inputfile outputfile
```

where:

options is one or more of the options listed in [Options for cnv2xml](#), below.

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

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

The following options are available.

Options for cnv2xml

Option	Description
-c KVCFG_SUPPRESSIMAGES	This option specifies that XML output includes verbose markup, but no images. If this option is not set, embedded images in a document are regenerated as separate files and stored in the output directory. KVXMLConfig() , on page 156.
-c KVCFG_ENABLEPOSITIONINFO	This option specifies that a position element is included in the markup for PDF documents. The position element defines the absolute position of the text relative to the bottom left corner of the page, and includes additional information such as font and color. KVXMLConfig() , on page 156.
-c KVCFG_DELSOFTHYPHEN	This option specifies that soft hyphens in PDF files are deleted from the converted output. See Control Hyphenation , on page 85.
-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 82.
-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 82.
-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 82.
-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 82.

cnv2xmlloop

The `cnv2xmlloop` sample program creates a single, formatted XML output file, but unlike `cnv2xml`, it converts the file out of process. See [Convert Files Out of Process](#), on page 25 for more information on out of process conversions. This program runs on both Windows and UNIX platforms.

To run `cnv2xmlloop`, type the following at the command line:

```
cnv2xmlloop [options] inputfile outputfile
```

where:

options is one or more of the options listed in [Options for cnv2xmloop, below](#).

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

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

The following options are available.

Options for cnv2xmloop

Option	Description
-c KVCFG_SUPPRESSIMAGES	This option specifies that XML output includes verbose markup, but no images. If you do not set this option, embedded images in a document are regenerated as separate files and stored in the output directory. See KVXMLConfig(), on page 156 .
-c KVCFG_ENABLEPOSITIONINFO	This option specifies that a position element is included in the markup for PDF documents. The position element defines the absolute position of the text relative to the bottom left corner of the page, and includes additional information such as font and color. See KVXMLConfig(), on page 156 .

metadata

The `metadata` sample program converts a source document into a single XML file that contains only the document metadata (Author, Subject, Title, and so on). This program runs on both Windows and UNIX platforms.

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

```
metadata 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 XML output file.

xmlindex

The `xmlindex` sample program produces stripped-down XML output suitable for use with indexing engines. It converts a source document into a single, largely unformatted XML file. This program runs on both Windows and UNIX platforms.

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

```
xmlindex 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 XML output file.

xmllini

The `xmllini` sample program is used in conjunction with template files to produce well-formed XML documents. For more information, see [Set Conversion Options by Using the Template Files, on page 34](#). Sample template files are in the `programs\ini` directory. This program runs on both Windows and UNIX platforms.

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

```
xmllini [options] inifile inputfile outputfile
```

where:

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

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 XML output file.

The following options are available.

Options for xmllini

Option	Description
-s stylesheetfile	Reads style sheet information from an existing style sheet file, or writes the information to an external CSS file. See Use Style Sheets with xmllini, on the next page .
-x xmlconfig_ filename	Converts an XML file by using customized element extraction settings defined in the <code>kvxconfig.ini</code> file. If you do not enter the full path to the template file, the program looks for the file in the current working directory (<code>install\OS\bin</code> , where <code>install</code> is the path name of the Export installation directory and <code>OS</code> is the name of the operating system). See Convert XML Files, on page 92 .
-rm	If you set this flag, text and graphics that were deleted from a document with a revision tracking feature enabled are converted, and revision tracking information is included in the XML output. See Convert Revision Tracking Information, on page 80 .
-oop	Runs the conversion out of process.
-f1	Prints a list of converted files in the console.

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

Use Style Sheets with `xmlini`

The `xmlini` sample program provides an option that allows XML Export to read Cascading Style Sheet (CSS) or Extensible Style Sheet Language (XSL) style sheet information from an existing style sheet file, or to write CSS information to an external CSS file. If the CSS does not exist, it is created. The style sheet name is referenced in the output XML, for example:

```
<?xml-stylesheet href="c:\mystyle.css" type="text/css"?>
```

This type of conversion makes the XML output document significantly smaller and enables you to use the same style sheet for many conversions.

To apply an existing style sheet to a conversion by using the `xmlini` sample program

1. In the template file, set `eStyleSheetType` to either `XML_CSS` or `XML_XSL`. This specifies that the formatting data is stored in either a CSS or XSL style sheet.
2. At the command prompt, type:

```
xmlini -s stylesheetname infile inputfile outputfile
```

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

`xmlcallback`

The `xmlcallback` 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. This program runs on Windows platforms only.

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

```
xmlcallback 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 XML output file.

`xmlonefile`

The `xmlonefile` sample program converts a source document into a single, formatted XML file. This program runs on Windows platforms only.

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

```
xmlonefile 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 XML output file.

xmlmulti

The `xmlmulti` sample program creates multiple XML files from a source document. The main file contains the table of contents. Each H1 heading is contained within its own file. The main file contains hyperlinks to each H1 block; each H1 file contains navigation to the table of contents, as well as to the previous and next blocks. This program runs on Windows platforms only.

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

```
xmlmulti 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 XML output file.

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 XML, 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 `cnv2xml`. To adapt the sample program to your needs, modify the GUI by using Visual Basic, and modify the `cnv2xml` program by using C. See [cnv2xml, on page 104](#).

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

The source code for the program is in the directory `install\xmlexport\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 36](#) for more information.

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](#)
- [XML Export API Functions](#)
- [XML Export API Callback Functions](#)
- [XML 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()	111
• fpCloseFile()	112
• fpExtractSubFile()	112
• fpFreeStruct()	114
• fpGetMainFileInfo()	115
• fpGetSubFileInfo()	116
• fpGetSubFileMetaData()	117
• fpOpenFile()	119

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 131.

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,  
    KVMMainFileInfo     *fileInfo);
```

Arguments

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

fileInfo A pointer to the structure [KVMMainFileInfo](#). 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 `KVMMainFileInfoFlag_HasContent`), pass the file directly to the conversion functions.

Example

```
KVMMainFileInfo  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 48](#).
- 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 48](#).
- 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 132](#).
- 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 112.

Example

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

fpGetSubFileMetaData()

This function extracts metadata from mail stores, mail messages, and non-mail items in an NSF file. See [Extract Mail Metadata](#), on page 49.

Syntax

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

Arguments

- | | |
|-----------------|--|
| pFile | The identifier of the file. This is a file handle returned from fpOpenFile() . |
| metaArg | A pointer to the KVGetSubFileMetaArg structure, which defines metadata tags whose values are retrieved.

Before you initialize the KVGetSubFileMetaArg structure, use the KVStructInit macro to initialize the KVStructHead structure. |
| metaData | A pointer to the KVSubFileMetaData structure, which contains the retrieved metadata values. |

Returns

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

Discussion

- When you pass in 0 for metaArg->metaNameCount, and NULL for metaArg->metaNameArray, a set of default metadata is retrieved. See [Extract Mail Metadata, on page 49](#).
- 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 `kvextract.h`.

• KVCredential	121
• KVCredentialComponent	122
• KVExtractInterface	122
• KVExtractSubFileArg	123
• KVGetSubFileMetaArg	126
• KVMainFileInfo	127
• KVMetadataElem	128
• KVMetaName	129
• KVOpenFileArg	129
• KVOutputStream	131
• KVSubFileExtractInfo	131
• KVSubFileInfo	132
• KVSubFileMetaData	135

KVCredential

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

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

Member Descriptions

<code>itemCount</code>	The number of credentials defined for this file.
<code>items</code>	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 111](#). 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 111](#).

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 112](#). The structure is defined in `kvextract.h`.

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

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead, on page 183 .
<code>index</code>	The index number of the subfile to be extracted.
<code>srcCharset</code>	Specifies the source character set of the subfile when the file format's reader cannot determine the character set. The character sets are enumerated in <code>KVCharSet</code> of <code>kvtypes.h</code> . See Discussion, on page 125 .
<code>trgCharset</code>	If the file type is <code>KVFileType_Main</code> , this is the target character set of the extracted file. Otherwise, this is ignored. The character sets are enumerated in <code>KVCharSet</code> in <code>kvtypes.h</code> . See Discussion, on page 125 .

isMSLSB	This flag indicates whether the byte order for Unicode text is Big Endian (MSLSB) or Little Endian (LSBMSB).
extractionFlag	<p>A bitwise flag that defines additional parameters for file extraction. The following flags are available:</p> <ul style="list-style-type: none">• <code>KVExtractionFlag_CreateDir</code><p>This flag indicates whether the directory structure of a subfile should be created. If you set this flag, the path defined in <code>filePath</code> is created if it does not already exist. If you do not set this flag, the path is not created, and the function returns <code>FALSE</code>.</p>• <code>KVExtractionFlag_Overwrite</code><p>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 <code>KVError_OutputFileExists</code> is generated.</p>• <code>KVExtractionFlag_ExcludeMailHeader</code><p>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 56.</p>• <code>KVExtractionFlag_GetFormattedBody</code><p>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.</p><div><p>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.</p></div>• <code>KVExtractionFlag_SaveAsMSG</code><p>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.</p><div><p>NOTE: In file mode, when the application sets this flag in fpExtractSubFile(), it must also check the KVSubFileExtractInfo structure's <code>filePath</code> parameter to verify the file name used for extraction.</p></div>• <code>KVExtractionFlag_SanitizeAbsolutePath</code><p>If you set this flag, KeyView ensures that the file is extracted to a location</p>

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 47](#).

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

Discussion

- If the document character set is detected and is also specified in `srcCharset`, the detected character set is overridden by the specified character set. If the source character set is *not* detected and is *not* specified, character set conversion does not occur. The [Supported Formats, on page 230](#) 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 47](#).
 - 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 129](#).
 - If `filePath` is a full path or partial path, and `createDir` is `TRUE`, the directory is created if it does not already exist.
 - If `filePath` is not specified, a default name and the target directory specified in either `KVExtractSubFileArg->extractDir` or `KVOpenFileArg->extractDir` are used to create a full path.
 - If both `filePath` and `extractDir` are not specified or are invalid, an error is returned.
 - If `filePath` is valid, but `extractDir` is not valid, an error is returned.
- The following applies when the output is to a stream:

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

KVGetSubFileMetaArg

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

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

Member Descriptions

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead , on page 183.
<code>index</code>	The index number of the subfile for which metadata is extracted.
<code>metaNameCount</code>	The number of metadata fields to be extracted.
<code>metaNameArray</code>	A pointer to the KVMetaName structure that contains an array of metadata tags whose values are retrieved.
<code>srcCharset</code>	Specifies the source character set of the metadata when the format's reader cannot determine the character set. The character sets are enumerated in <code>KVCharSet</code> of <code>kvtypes.h</code> . See Discussion , below.
<code>trgCharset</code>	The target character set of the extracted metadata. The character sets are enumerated in <code>KVCharSet</code> in <code>kvtypes.h</code> .
<code>isMSBLSB</code>	This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).

Discussion

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

specified, character set conversion does not occur. The section [Supported Formats, on page 230](#) lists the formats for which the source character set can be determined.

- To retrieve a predefined list of metadata, pass 0 for metaNameCount and NULL for metaNameArray. The metadata in [Extract Mail Metadata, on page 49](#) is extracted.

KVMainFileInfo

This structure contains information about a main file that is open for extraction. It is initialized by calling [fpGetMainFileInfo\(\)](#). This structure is defined in `kvextract.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 183 .
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 179 .
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	<p>A bitwise flag that provides additional information about the main file. The following flag is available:</p> <p>KVMainFileInfoFlag_HasContent—The main file contains text that can be converted. Below are some examples of how this flag is used:</p> <ul style="list-style-type: none">• 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 [XML Export API Functions, on page 137](#).
- 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 the next page](#).

KVMetadataElem

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

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

Member Descriptions

<code>isValid</code>	Specifies whether the metadata returned from the API is valid data.
<code>dataID</code>	The integer name of the extracted metadata field.
<code>dataType</code>	The data type of the metadata field. The types are defined in KVMetadataType in <code>kvtypes.h</code> .
<code>strType</code>	A pointer to the string name of the metadata field.
<code>data</code>	<p>The contents of the metadata field.</p> <p>If the type member is <code>KVMetadata_Int4</code> or <code>KVMetadata_Bool</code>, this member contains the actual value. Otherwise, this member is a pointer to the actual value.</p> <p><code>KVMetadata_DateTime</code> points to an 8-byte value.</p> <p><code>KVMetadata_String</code> and <code>KVMetadata_Unicode</code> point to the beginning of the string that contains the text. The strings are NULL terminated.</p> <p><code>KVMetadata_Binary</code> points to the first element of a byte array.</p>

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 183 .
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	<p>A pointer to the developer-assigned instance of KVInputStream. The KVInputStream structure defines the input stream that contains the source. See KVInputStream, on page 180.</p> <p>If you are using a file as input, this is <code>NULL</code>.</p>
filePath	<p>A pointer to the full file path to the source file.</p> <p>If you are using a stream as input, this is <code>NULL</code>.</p>
extractDir	<p>A pointer to the default directory to which subfiles are extracted. This directory must exist.</p> <p>You can use this in conjunction with <code>KVExtractSubFileArg->filePath</code> to create the full output path. See KVExtractSubFileArg, on page 123.</p>
openFlag	<p>A bitwise flag that defines additional parameters for opening the file. The following flag is available:</p> <p><code>KVOpenFileFlag_CreateRootNode</code>—If you set this flag, KeyView creates a root object when extracting this file's subfiles. This root node does not have a parent and is at the highest level of the file's tree structure. It is used internally to provide a reference point from which all other child nodes are determined, and the file's hierarchy is created.</p> <p>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 48 for more information.</p> <p>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.</p>

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 `kvextract.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 183.
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	<p>The original path, file name, or path and file name of the subfile.</p> <p>If the subfile is embedded in the main file as a link, this is the external path to the subfile.</p>
infoFlag	<p>A bitwise flag that provides additional information about the extracted subfile. The following flags are available:</p> <ul style="list-style-type: none"> • <code>KVSubFileExtractInfoFlag_NeedsExtraction</code>—The file might contain subfiles and should be extracted further. • <code>KVSubFileExtractInfoFlag_FileCreated</code>—The file was created on disk. • <code>KVSubFileExtractInfoFlag_CharsetConverted</code>—The subfile's character set was converted. • <code>KVSubFileExtractInfoFlag_External</code>—The subfile is embedded in the main file as a link and is stored externally. For example, the subfile might be an object that was embedded in a Word document using "Link to File," or an attachment that is referenced in an MBX message. This type of file cannot be extracted. You must write code to access the subfile based on the path in the member <code>filePath</code> or <code>fileName</code>. • <code>KVSubFileExtractInfoFlag_FolderCreated</code>—A folder was created. • <code>KVSubFileExtractInfoFlag_NonFormattedBodyExtracted</code>—Indicates that a plain text version of the message was extracted due to an error extracting the formatted version of the message.
docInfo	<p>The file's major format (such as Microsoft Word or Corel Presentation), as defined by the structure <code>ADDOCINFO</code>. See ADDOCINFO, on page 179.</p> <p>If you output the data to a stream, the file format is not returned.</p>

KVSubFileInfo

This structure contains information about a subfile in a container file. It is initialized by calling [fpGetSubFileInfo\(\)](#). This structure is defined in `kvextract.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 183 .
subFileName	<p>The path, file name, or path and file name of the subfile.</p> <p>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 66.</p>
subFileType	<p>The subfile's position in the container file's hierarchy.</p> <ul style="list-style-type: none">• KVSubFileType_Main—The subfile is at the top level of the main file. This is the default subfile type. See Discussion, on page 135.• 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 48).• 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. <div><p>NOTE:</p><p>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).</p></div>
subFileSize	<p>The size of the subfile in bytes. This information might be useful if you do not want to extract very large files.</p> <p>This value is approximate and is the maximum size of the subfile. The subfile is usually smaller than this value when it is extracted.</p>
infoFlag	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.

<code>charset</code>	If the subfile is not an attachment, this is the character set of the subfile. If the subfile is an attachment, the character set is <code>KVCS_UNKNOWN</code> .
<code>isMSBLSB</code>	This flag indicates whether the byte order for Unicode text is Big Endian (MSBLSB) or Little Endian (LSBMSB).
<code>fileTime</code>	<p>When the subfile is a mail message, this is the file's <code>Sent</code> time. Otherwise, it is the last modified time. The file time is not available for the following file types:</p> <ul style="list-style-type: none">• EML attachments• OLE objects in a Microsoft Office document• Embedded images
<code>parentIndex</code>	The index number of this file's parent. For example, the index of a folder in which the subfile is stored, or the file to which the subfile is attached. If a file does not have a parent, the <code>parentIndex</code> is -1.
<code>childCount</code>	The number of first-level children in the subfile.
<code>childArray</code>	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 119](#) and [fpExtractSubFile\(\)](#), [on page 112](#).
- 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 48](#).

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

<code>KVStructHeader</code>	The KeyView version of the structure. See KVStructHead , on page 183 .
<code>nElem</code>	The number of metadata fields contained in the array.
<code>ppElem</code>	A pointer to an array of pointers that are the memory addresses of metadata field

values in the [KVMetadataElem](#) structure.

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: XML Export API Functions

This section describes the functions in the XML 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.

• KVXMLGetInterface()	137
• KVXMLGetInterfaceEx()	138
• fpConvertStream()	139
• fpFileToInputStreamCreate()	142
• fpFileToInputStreamFree()	143
• fpFileToOutputStreamCreate()	143
• fpFileToOutputStreamFree()	144
• fpFreeImageInfos()	145
• fpGetAnchor()	146
• fpGetConvertFileList()	147
• fpGetKvErrorCode	148
• fpGetKvErrorCodeEx	148
• fpGetOutputImageCount()	149
• fpGetOutputImageInfo()	150
• fpGetOutputImageInfos()	150
• fpGetStreamInfo()	151
• fpGetSummaryInfo()	152
• fpInit()	153
• fpSetStyleMapping()	155
• fpShutDown()	156
• fpValidateTemplate()	156
• KVXMLConfig()	156
• KVXMLConvertFile()	164
• KVXMLEndOOPSession()	167
• KVXMLSetStyleSheet()	168
• KVXMLStartOOPSession()	170

KVXMLGetInterface()

NOTE:

This function has been superseded by [KVXMLGetInterfaceEx\(\)](#); [KVXMLGetInterfaceEx\(\)](#) should be used instead of [KVXMLGetInterface\(\)](#).

This function is exported by the Export definition file. It supplies function pointers to other Export functions. When `KVXMLGetInterface()` is called, it assigns the function pointers in the structure `KVXMLInterface` to other functions described in this chapter. For example, `KVXMLInterface.fpInit` is assigned to point to `KVXMLInit()`.

Syntax

```
void pascal KVXMLGetInterface (KVXMLInterface *pInterface);
```

Arguments

`pInterface` A pointer to the structure `KVXMLInterface`. See [KVXMLInterfaceEx, on page 193](#).

Returns

None.

Discussion

- One of the initial steps in using the XML Export API is to create an instance of a `KVXMLInterface` structure and use this function to gain access to other functions.
- The functions can be called directly. For example, you can call `KVXMLGetSummaryInfo()` instead of using `fpGetSummaryInfo()` in `KVXMLInterface`. However, Micro Focus recommends that you assign the function pointers in `KVXMLInterface` to the functions for efficiency.

KVXMLGetInterfaceEx()

This function is exported by the Export definition file. It supplies function pointers to other Export functions. When `KVXMLGetInterfaceEx()` is called, it assigns the function pointers in the structure `KVXMLInterfaceEx` to other functions described in this chapter. For example, `KVXMLInterfaceEx.fpInit` is assigned to point to `KVXMLInit()`.

Syntax

```
BOOL pascal KVXMLGetInterfaceEx (KVXMLInterfaceEx *pInterface);
```

Arguments

`pInterface` A pointer to the structure `KVXMLInterfaceEx`. See [KVXMLInterfaceEx, on page 193](#).

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 `KVXMLGetInterfaceEx`. If you do not do this, the function fails.

Discussion

- One of the initial steps in using the XML Export API is to create an instance of a `KVXMLInterfaceEx` structure and use this function to gain access to other functions.
- The functions can be called directly. For example, you can call `KVXMLGetSummaryInfo()` instead of using `fpGetSummaryInfo()` in `KVXMLInterfaceEx`. However, Micro Focus recommends that you assign the function pointers in `KVXMLInterfaceEx` to the functions for efficiency.
- `KVXMLInterfaceEx` must be initialised by calling `KVStructInit` prior to passing it to `KVXMLGetInterfaceEx`, otherwise `KVXMLGetInterfaceEx` fails.

Example

```
KVXMLInterfaceEx KVXMLInt;  
BOOL (pascal *fpGetInterfaceEx)(KVXMLInterfaceEx *);  
...  
KVStructInit(&KVXMLInt);  
(*fpGetInterfaceEx)(&KVXMLInt);
```

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,  
    KVXMLTemplate        *pTemplates,  
    KVXMLOptions         *pOptions,  
    KVXMLTOCOptions      *pTOCCreateOptions,  
    KVXMLCallbacks      *pCallbacks,
```

```

    BOOL
    KVErrorCode

    bIndex,
    *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 180 .
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 181 .
pTemplates	<p>A pointer to the KVXMLTemplate 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 KVXMLTemplate, on page 203.</p> <p>If this pointer is NULL, the default values for the structure are used.</p>
pOptions	<p>A pointer to the KVXMLOptions 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 KVXMLOptions, on page 195.</p> <p>If this pointer is NULL, the default values for the structure are used.</p>
pCallbacks	<p>A pointer to the KVXMLCallbacks data structure. It is a structure of functions that Export calls for specific, user-defined purposes. See KVXMLCallbacks, on page 187.</p> <p>If callbacks are not used, this can be NULL.</p>
pTOCCreateOptions	<p>A pointer to the KVXMLTOCOptions data structure. It specifies whether a heading is included in the table of contents. See KVXMLTOCOptions, on page 207.</p> <p>If this pointer is NULL, the default values for the structure are used.</p>
bIndex	<p>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.</p> <p>You can set this option through the bIndexOnly member of the KVXMLOptions structure. See KVXMLOptions, on page 195.</p> <p>To generate output with verbose markup and without images, set the nType</p>

argument of the `KVXMLConfig()` function to `KVCFG_SUPPRESSIMAGES`. See [KVXMLSetStyleSheet\(\)](#), on page 168.

`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 25.
- When converting out of process, this function must be called after the call to `KVXMLStartOOPSession()` and before the call to `KVXMLEndOOPSession()`. See [KVXMLStartOOPSession\(\)](#), on page 170 and [KVXMLEndOOPSession\(\)](#), on page 167.
- When converting out of process, the values for the `KVXMLTemplate`, `KVXMLOptions`, and `KVXMLTOCOptions` structures should be set to `NULL`. These structures are already passed in the call to `KVXMLStartOOPSession()`. See [KVXMLStartOOPSession\(\)](#), on page 170.

Example

The following sample code is from the `cnv2xml` sample program:

```
if(!(*KVXMLInt.fpConvertStream)(
    pKVXML,          /* A pointer returned by fpInit() */
    NULL,            /* A pointer for callback functions */
    &Input,           /* Input stream */
    &Output,          /* Output stream */
    NULL,            /* Markup and related variables */
    &XMLOptions,      /* Options */
    NULL,            /* TOC options */
    NULL,            /* A pointer to callback functions */
    FALSE,           /* Index mode */
    &error))          /* Error return value */
{
    printf("Error converting %s to XML %d\n", argv[i - 1], error);
}
else
{
```

```
    printf("Conversion of %s to XML completed.\n\n", argv[i - 1]);  
}
```

fpFileToInputStreamCreateO

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 180 . |

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 `cnv2xml` sample program:

```
if(!(*KVXMLInt.fpFileToInputStreamCreate)(pKVXML, argv[i++], &Input))  
{  
    printf("Error creating input stream\n");  
    (*KVXMLInt.fpShutDown)(pKVXML);  
    mpFreeLibrary(hKVXML);  
    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 180.

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

<code>pContext</code>	A pointer returned from <code>fpInit()</code> .
<code>pszFileName</code>	A pointer to the name of the output file to create.
<code>pOutput</code>	A pointer to the developer-assigned instance of <code>KVOutputStream</code> . The <code>KVOutputStream</code> structure defines the output stream to which Export writes the generated XML. See KVOutputStream , on page 181.

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 `cnv2xml` sample program:

```
if (!(*KVXMLInt.fpFileToOutputStreamCreate)(pKVXML, argv[i], &Output))
{
    printf("Error creating output stream\n");
    (*KVXMLInt.fpFileToInputStreamFree)(pKVXML, &Input);
    (*KVXMLInt.fpShutDown)(pKVXML);
    mpFreeLibrary(hKVXML);
    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 XML. See [KVOutputStream, on page 181](#).

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()`.

fpFreeImageInfos()

This function frees the memory associated with an image info context. Call this function when you have finished using the image info context for calls to `fpGetOutputImageCount()` and `fpGetOutputImageInfo()` (see [fpGetOutputImageCount\(\), on page 149](#) and [fpGetOutputImageInfo\(\), on page 150](#)).

Syntax

```
BOOL pascal fpFreeImageInfos (  
    void* const pContext,  
    void* const pImageInfos)
```

Arguments

- pContext** A pointer returned from `fpInit()` and the pointer originally passed to `fpGetOutputImageInfo()` to create the image info context that you want to free. See [fpInit\(\), on page 153](#) and [fpGetOutputImageInfo\(\), on page 150](#).
- pImageInfos** A pointer returned from `fpGetOutputImageInfos()`. See [fpGetOutputImageInfos\(\), on page 150](#).

Returns

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

Discussion

- It is safe to call `fpFreeImageInfos()` with `pImageInfos()` set to `NULL`. The function returns `TRUE` in this case.
- You must call `fpFreeImageInfos()` before you call `fpShutdown()` (see [fpShutdown\(\)](#), on page 156).
- You must not call `fpGetOutputImageCount()`, `fpGetOutputImageInfo()`, and `fpFreeImageInfos()` using an image info context pointer for which the associated system resources have already been successfully freed by using `fpFreeImageInfos()`.

fpGetAnchor()

This function gets the file name automatically generated by Export and used for external graphics referenced with `<a xmlns:xlink= xlink href=>` tags and for heading-level table of contents entries.

Syntax

```
BOOL pascal fpGetAnchor(  
    void *pCallingContext,  
    KVHTMLAnchorTypeEx eAnchorTypeEx,  
    KVXMLAnchorType eAnchorType,  
    char *pszAnchor,  
    int cbAnchorMax,  
    BYTE *pHTML,  
    UINT cbHTML);
```

Arguments

<code>pCallingContext</code>	A pointer passed back to the callback functions.
<code>eAnchorTypeEx</code>	The graphic or block anchor type for the output stream. It must be one of the enumerated types defined in <code>KVXMLAnchorType</code> . See KVXMLAnchorType , on page 217.
<code>pszAnchor</code>	A pointer to the location in which the new anchor is stored.
<code>cbAnchorMax</code>	The maximum number of bytes to place in <code>pszAnchor</code> .
<code>pHTML</code>	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 pHTML.

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, pHTML, and cbHTML values that are also provided.
- pHTML 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 XML during a call to fpConvertStream() or KVXMLConvertFile().

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 25](#).
- When converting out of process, this function must be called after the call to `KVXMLStartOOPSession()` and before the call to `KVXMLEndOOPSession()`. See [KVXMLStartOOPSession\(\), on page 170](#) and [KVXMLEndOOPSession\(\), on page 167](#).

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 153](#).

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 [fpInit\(\)](#), on page 153.

Returns

The current extended error code.

fpGetOutputImageCount()

This function returns the number of images that would be obtained by the XML Export, and for which you can obtain image information by using `fpGetOutputImageInfo()`.

Syntax

```
BOOL pascal fpGetOutputImageCount (  
    const void* const pImageInfos,  
    int* const pnImages)
```

Arguments

pImageInfos A pointer returned from `fpGetOutputImageInfos()`.

pnImages A pointer to an integer to use to store the number of images found.

Returns

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

Discussion

If the function call is unsuccessful, it does not modify the value of the integer that `pnImages` points to.

fpGetOutputImageInfo()

This function returns the dimensions of the images that would be obtained during the XML Export process.

Syntax

```
BOOL pascal fpGetOutputImageInfo (  
    const void* const pImageInfos,  
    const int nImage,  
    KVXMLImageInfo* const ptImageInfo)
```

Arguments

- pImageInfos** A pointer returned from `fpGetOutputImageInfos()`.
- nImage** The zero-based index of the image to retrieve dimensions for.
- ptImageInfo** The pointer to a [KVXMLImageInfo](#), on page 190 structure, initialized with the `KVStructInit` macro, which the function fills with the dimensions of the image with index `nImage`.

Returns

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

fpGetOutputImageInfos()

This function returns the image information context that must be supplied to the `fpGetOutputImageCount()` or `fpGetOutputImageInfo()` functions. See [fpGetOutputImageCount\(\)](#), on the previous page and [fpGetOutputImageInfo\(\)](#), above.

You must free the system resources associated with this context after you use it, by using the `fpFreeImageInfos()` function.

Syntax

```
void* pascal fpGetOutputImageInfos (  
    void* const pContext,  
    KVInputStream* const pInput)
```

Arguments

- pContext** A pointer returned from `fpInit()`. See [fpInit\(\)](#), on page 153.
- pInput** The pointer to a `KVInputStream` instance. This instance defines the input stream that the function processes to extract the images.

Returns

- If the call is successful, the return value is the pointer to an image info context object.
- If the call is unsuccessful, the return value is `NULL`.

Discussion

- To obtain image information out of process, call `KVXMLStartOOPSession()` before you call `fpGetOutputImageInfos()`. You must open a new, second OOP session for any subsequent XML export from the input stream.
- If this function fails, you can call `fpGetKvErrorCode()` and `fpGetKvErrorCodeEx()` to help identify the cause of the failure.
- You can use multiple image info contexts at any one time.

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 [KVInputStream](#), on page 180.
- pStreamInfo** A pointer to the developer-assigned instance of `KVStreamInfo`. The `KVStreamInfo` structure defines the input stream document type and character set. See

[KVStreamInfo](#), on page 182.

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 68 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 180.
- 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 185.
- 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 230](#) 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 230](#) 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 25](#).
- You can call this function at any time after the call to `KVXMLInit()`.
- When converting out of process, this function must be called after the call to `KVXMLStartOOPSession()` and before the call to `KVXMLEndOOPSession()`. [KVXMLStartOOPSession\(\), on page 170](#) and [KVXMLEndOOPSession\(\), on page 167](#).
- 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

- | | |
|----------------------------|--|
| <code>pMemAllocator</code> | A pointer to a developer-defined memory allocator. If <code>NULL</code> is passed, the default C run-time memory allocation is used. |
| <code>pszKeyViewDir</code> | A pointer to the directory where the Export components are located. This is normally the directory <code>install\OS\bin</code> , where <code>install</code> is the path name of the Export installation directory and <code>OS</code> is the name of the operating system. |
| <code>pszDataFile</code> | A pointer to the directory and file name of the Export data file, <code>formats_e.ini</code> . |

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 350](#) for more information.

<code>pError</code>	A pointer to an error code defined in <code>KVErrorCode</code> or <code>KVErrorCodeEx</code> in <code>kvtypes.h</code> . See KVErrorCode, on page 211 and KVErrorCodeEx, on page 213 .
<code>dWord</code>	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 page 156](#).

Example

The following sample code is from the `cnv2xml` sample program:

```
pKVXML = (*KVXMLInt.fpInit)(NULL, ".", NULL, &error, 0);
if(!pKVXML)
{
    printf("Error initializing KVXML: %d\n", error);
    mpFreeLibrary(hKVXML);
    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 184. 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 25.
- When converting out of process, this function must be called after the call to `KVXMLStartOOPSession()` and before the call to `KVXMLEndOOPSession()`. See [KVXMLStartOOPSession\(\)](#), on page 170 and [KVXMLEndOOPSession\(\)](#), on page 167.
- 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

<code>pContext</code>	A pointer returned from <code>fpInit()</code> .
-----------------------	---

Returns

None.

Discussion

After this function is called, the `pContext` pointer must not be passed to any XML Export API.

fpValidateTemplate()

This function is used to ensure that the markup is well-formed and valid according to the DTD. It is currently not implemented.

KVXMLConfig()

This function is called directly and provides a way to configure options prior to the document conversion. Currently, the function is used for the following configurations:

- **Generate output without images**

Generate output with *verbose* markup and *without* images. To generate output with *minimal* markup (ID and style paragraph attributes) and *without* images, set the `bIndexOnly` member of the `KVXMLOptions` structure. See [KVXMLOptions](#), on page 195.

- **Enable PDF position information**

Include position information in the markup generated for a PDF document.

- **Configure PDF bookmarks**

Specify whether bookmarks in a PDF file are converted to simple XLinks in the XML output.

- **Configure Word bookmarks**

Disable the conversion of Microsoft Word bookmarks to zone elements.

- **Designate temporary directory**

Specify a directory in which temporary files created during XML conversion processes are stored.

NOTE: 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 the files document type.

- **Enable PDF logical reading order**

Convert paragraphs in PDF files in the order in which they appear on the page and with left-to-right or right-to-left paragraph direction. See [Convert PDF Files to a Logical Reading Order, on page 82](#).

- **Configure PDF soft hyphens**

Specify whether soft hyphens are removed from the XML output. See [Control Hyphenation, on page 85](#).

- **Enable Revision Marks**

Convert text and graphics that were deleted from a document with revision tracking enabled and include revision tracking information in the XML output. [Convert Revision Tracking Information, on page 80](#).

- **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()`.

- **Include position and invisible text tokens (with bounding boxes) in the output**

Add top, left, height, width, and rotation attributes to `<p>` elements.

Syntax

```
KVErrorCode pascal KVXMLConfig(  
    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, below](#).
- nValue** The integer value defined for the flags above.
- This is `TRUE` or `FALSE` for all flags except `KVCFG_LOGICALPDF`, `KVCFG_SETMETADATACHARSET`, `KVCFG_SETTEMPDIRECTORY`, and `KVCFG_SETXMLCONFIGINFO`.
- For `KVCFG_LOGICALPDF`, this is one of the paragraph direction options defined in the `LPDF_DIRECTION` enumerated type in `kvtypes.h`. See [LPDF_DIRECTION, on page 228](#).
- For `KVCFG_SETTEMPDIRECTORY` and `KVCFG_SETXMLCONFIGINFO`, this is not set.
- For `KVCFG_SETMETADATACHARSET`, `nValue` is a character set enumerated in `KVCharSet` of `kvtypes.h`. See [Convert Character Sets, on page 71](#).
- p** The data for the configuration flag.
- This is `NULL` for all flags except `KVCFG_SETTEMPDIRECTORY` and `KVCFG_SETXMLCONFIGINFO`.
- For `KVCFG_SETTEMPDIRECTORY`, this is path to the directory where temporary files are stored.
- For `KVCFG_SETXMLCONFIGINFO`, this is a pointer to the `KVXConfigInfo` structure. See [KVXConfigInfo, on page 186](#).
- For `KVCFG_SETPASSWORD`, this is the source file password.

Configuration Flags

The following flags are available for the `nType` argument in `KVXMLConfig()`. These flags are defined in `kvtypes.h`.

Flag	Description
<code>KVCFG_SUPPRESSIMAGES</code>	If you set <code>KVCFG_SUPPRESSIMAGES</code> , the XML output includes verbose markup, but no images. If you do not set this option, embedded images in a document are regenerated as separate files and stored in the output directory. To generate output with minimal markup (ID and style paragraph attributes) and without images, set the <code>bIndexOnly</code> member of the <code>KVXMLOptions</code> structure to <code>TRUE</code> . KVXMLOptions, on page 195 .
<code>KVCFG_ENABLEPOSITIONINFO</code>	If you set <code>KVCFG_ENABLEPOSITIONINFO</code> , a position element is included in the markup for PDF documents. The position element defines the absolute position of the text relative to the bottom left corner of the page, and includes additional information such as font and color.

Flag	Description
KVCFG_SETMETADATACHARSET	This option enables you to specify the output character set for metadata when using <code>fpGetSummaryInfo()</code> . <code>nValue</code> is a character set enumerated in <code>KVCharSet</code> of <code>kvtypes.h</code> . See Convert Character Sets, on page 71 . This function should be called before <code>fpGetSummaryInfo()</code> .
KVCFG_SUPPRESSTOCPRINTIMAGE	<p>If you set <code>KVCFG_SUPPRESSTOCPRINTIMAGE</code>, bookmarks in a PDF file are <i>not</i> converted to simple XLinks in the XML output. By default, PDF bookmarks are converted to source and destination anchors. For example,</p> <pre><a xmlns:xlink="http://www.w3.org/TR/xlink" xlink:href="#bmk1">Highlight File Format <a xmlns:xlink="http://www.w3.org/TR/xlink" name="bmk1"></pre>
KVCFG_DISABLEZONE	<p>If you set <code>KVCFG_DISABLEZONE</code>, the conversion of Microsoft Word bookmarks to zone elements (<code><zone name="xxx"></code>) in the output XML is disabled.</p> <p>A bookmark in Microsoft Word documents is a name given to a selected area of the document. The bookmark might enclose words, paragraphs, tables, table cells, lists, list items, or the entire document. In XML Export, bookmarks are converted to zone elements (<code><Zone name="xxx"></code>) by using the KeyView <code>KVT_ZONE</code> token.</p> <p>Depending on how bookmarks are defined in the original document, the creation of zone elements might result in malformed XML. In this case, you can disable zone creation to avoid these validity errors. Zone element creation is enabled by default.</p>
KVCFG_SETTEMPDIRECTORY	<p>The <code>KVCFG_SETTEMPDIRECTORY</code> 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.</p> <p>To define a directory for temporary files generated during an out-of-process conversion, set the <code>tempfilepath</code> parameter in the <code>formats_e.ini</code> file. See Convert Files Out of Process, on page 25.</p> <p>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.</p>
KVCFG_SETXMLCONFIGINFO	<p>The <code>KVCFG_SETXMLCONFIGINFO</code> flag enables you to define which elements and attributes are extracted from XML documents with a specified format ID or root element. You can use this to override the default settings for the supported XML formats (see Convert XML Files, on page 92), or to define settings for custom XML document types.</p> <p>The settings are defined in the <code>KVXConfigInfo</code> structure (see KVXConfigInfo, on page 186). To set custom settings for more than one document type, call the <code>KVXMLConfig()</code> function once for each type.</p>

Flag	Description
	You can also modify element extraction settings by using the <code>kvxconfig.ini</code> file. See Configure Element Extraction for XML Documents , on page 92.
KVCFG_LOGICALPDF	The KVCFG_LOGICALPDF flag converts paragraphs in a PDF file in the order in which they appear on the page (logical reading order). The <code>nValue</code> argument specifies the paragraph direction. See Convert PDF Files to a Logical Reading Order , on page 82.
KVCFG_DELSOFTHYPHEN	<p>If you set KVCFG_DELSOFTHYPHEN, soft hyphens in the source document are removed, and the hyphenated words are joined in the XML output. By default, soft hyphens are maintained. See Control Hyphenation, on page 85.</p> <p>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 139 or KVXMLConvertFile(), on page 164 for more information on running Export in index mode.</p>
KVCFG_INCLREVISIONMARK	<p>If you set this flag to TRUE, text and graphics that were deleted from a document with a revision tracking feature enabled are converted, and revision tracking information is included in the XML output.</p> <p>To reset the flag and exclude deleted content and revision tracking information from the XML output, set the flag to FALSE. See Convert Revision Tracking Information, on page 80. The default is FALSE.</p>
KVCFG_WP_NOCOMMENTS	<p>Set KVCFG_WP_NOCOMMENTS 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.</p> <p>You can also toggle comment output by modifying the <code>formats_e.ini</code> file. See Show Hidden Data, on page 97.</p>
KVCFG_WP_SHOWHIDDENTEXT	Set KVCFG_WP_SHOWHIDDENTEXT to TRUE to export hidden text from Microsoft Word documents.
KVCFG_WP_SHOWDATEFIELDCODE	Set KVCFG_WP_SHOWDATEFIELDCODE to TRUE to export date field codes from Microsoft Word documents.
KVCFG_WP_SHOWFILENAMEFIELDCODE	Set KVCFG_WP_SHOWFILENAMEFIELDCODE to TRUE to export the file name field code from Microsoft Word documents.
KVCFG_SS_SHOWHIDDENINFOR	Set KVCFG_SS_SHOWHIDDENINFOR to TRUE to export hidden information from Microsoft Excel files.
KVCFG_SS_SHOWCOMMENTS	Set KVCFG_SS_SHOWCOMMENTS to TRUE to export comments from Microsoft Excel files.
KVCFG_SS_	Set KVCFG_SS_SHOWFORMULA to TRUE to export formulas from Microsoft Excel

Flag	Description
	files.
KVCFG_PG_HIDEHIDDENSLIDE	Set KVCFG_PG_HIDEHIDDENSLIDE to TRUE not to export hidden slides from Microsoft PowerPoint files.
KVCFG_PG_HIDECOMMENT	Set KVCFG_PG_HIDECOMMENT 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 KVCFG_PG_SHOWCOMMENTSSLIDE to TRUE to export comments slides from Microsoft PowerPoint 2003 and 2007 files.
KVCFG_PG_SHOWSLIDNOTES	<p>Set KVCFG_PG_SHOWSLIDNOTES to TRUE to export slide notes from Microsoft PowerPoint files.</p> <p>You can also toggle slide note output by modifying the <code>formats_e.ini</code> file. See Show Hidden Data, on page 97.</p>
KVCFG_SETPASSWORD	<p>This flag enables you to define a password used to open a password-protected file for export. See Export Password Protected Files, on page 410.</p> <p>nValue is TRUE.</p> <p>p is the source file password, which can have a maximum length of 255 characters (the final byte is null).</p>
KVCFG_POSITIONINFOOUTPUTTYPE	This flag enables you to extend the existing <p> tags to include bounding box information.

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 `KVXMLConvertFile()`.
- This function runs in-process or out of process. See [Convert Files Out of Process, on page 25](#).
- When converting out of process, you must call this function after the call to `KVXMLStartOOPSession()` and before the call to `KVXMLEndOOPSession()`. See [KVXMLStartOOPSession\(\), on page 170](#) and [KVXMLEndOOPSession\(\), on page 167](#).

Examples

- To generate verbose markup, but no images:

```
(*fpXMLConfig)(pKVXML, KVCFG_SUPPRESSIMAGES, TRUE, NULL);
```

- To produce summary information in UTF8:

```
(*fpXMLConfig)(pKVXML, KVCFG_SETMETADATACHARSET, KVCS_UTF8, NULL);
```

- To specify bookmarks in a PDF file are not converted to XLinks in the XML output:

```
(*fpXMLConfig)(pKVXML, KVCFG_SUPPRESSTOCPRINTIMAGE, TRUE, NULL);
```

- To disable the conversion of zone elements:

```
(*fpXMLConfig)(pKVXML, KVCFG_DISABLEZONE, TRUE, NULL);
```

- To set a directory for temporary files:

```
char    tmpDir[250];  
strcpy (tmpDir, "c:\\temp\\xmlexport");  
(*fpXMLConfig)(pKVXML, KVCFG_SETTEMPDIRECTORY, 0, tmpDir);
```

- To specify custom extraction settings for conversion of an XML file:

```
KVXConfigInfo xinfo; /* populate xinfo */  
(*fpXMLConfig)(pKVXML, KVCFG_SETXMLCONFIGINFO, 0, &xinfo);
```

- To specify PDF files are converted to a logical reading order, and the paragraph direction for the PDF output is left to right:

```
(*fpXMLConfig)(pKVXML, KVCFG_LOGICALPDF, LPDF_LTR, NULL);
```

- To specify PDF files are converted to a logical reading order, and the paragraph direction for the PDF output is right to left:

```
(*fpXMLConfig)(pKVXML, KVCFG_LOGICALPDF, LPDF_RTL, NULL);
```

- To specify PDF files are converted to a logical reading order, and the paragraph direction for the PDF output is determined on the fly for each page:

```
(*fpXMLConfig)(pKVXML, KVCFG_LOGICALPDF, LPDF_AUTO, NULL);
```

- To specify soft hyphens are removed from the XML output:

```
(*fpXMLConfig)(pKVXML, KVCFG_DELSOFTHYPHEN, TRUE, NULL);
```

- To convert text and graphics that are identified by revision marks:

```
(*fpXMLConfig)(pKVXML, KVCFG_INCLREVISIOMARK, TRUE, NULL);
```

- To toggle hidden data output from Microsoft Word documents, use one of the KVCFG_WP flags:

```
(*fpXMLConfig)(pKVXML, KVCFG_WP_NOCOMMENTS, TRUE, NULL);
```

- To toggle hidden data output from Microsoft Excel documents, use one of the KVCFG_SS flags:

```
(*fpXMLConfig)(pKVXML, KVCFG_SS_SHOWHIDDENINFOR, TRUE, NULL);
```

- To toggle hidden data output from Microsoft PowerPoint documents, use one of the KVCFG_PG flags:

```
(*fpXMLConfig)(pKVXML, KVCFG_PG_HIDEHIDDENSLIDE, TRUE, NULL);
```

- To specify a password to open a password-protected file for export:

```
(*fpXMLConfig)(pKVXML, KVCFG_SETPASSWORD, TRUE, password);
```

where password is a null-terminated string of 255 or fewer characters.

- To include a position element in the markup for PDF documents:

```
(*fpXMLConfig)(pKVXML, KVCFG_ENABLEPOSITIONINFO, TRUE, NULL);
```

Using the PDF position element significantly changes the generated markup. For example, without the option, the XML output from a section of a PDF document looks like this:

```
<?xml version="1.0" encoding="utf-8" ?>
  <!DOCTYPE VerityXMLExport (View Source for full doctype...)>
- <VerityXMLExport>
- <WP>
- <p id="p1" font-size="33pt">
  
  Economic Fiscal Update
  <font size="18pt" color="#777777">Theand</font>
  <font size="14pt" color="#ffffff">October 30, 2002</font>
  <font size="29pt" color="#a4a4a4">Overview</font>
</p>
```

With the option enabled, the same section of the PDF document looks like this:

```
<?xml version="1.0" encoding="utf-8" ?>
  <!DOCTYPE VerityXMLExport (View Source for full doctype...)>
- <VerityXMLExport>
- <WP>
  <Position style="position:absolute;top:534px;left:254px;font-family:'Times New
Roman';font-size:33pt;white-space:nowrap;" />
  <Position style="position:absolute;top:393px;left:254px;white-space:nowrap;" />
  
  <Position style="position:absolute;top:308px;left:256px;font-family:'Times New
Roman';font-size:33pt;white-space:nowrap;" />
  Economic
  <Position style="position:absolute;top:346px;left:256px;font-family:'Times New
Roman';font-size:33pt;white-space:nowrap;" />
  Fiscal Update
  <Position style="position:absolute;top:298px;left:281px;font-family:'Times New
Roman';font-size:18pt;color:#777777;background-color:#ffffff;white-space:nowrap;"
/>
  The
  <Position style="position:absolute;top:336px;left:299px;font-family:'Times New
Roman';font-size:18pt;color:#777777;background-color:#ffffff;white-space:nowrap;"
/>
  and
```

```
<Position style="position:absolute;top:543px;left:397px;font-family:'Times New
Roman';font-size:14pt;color:#ffffff;background-color:#000000;white-space:nowrap;"
/>
October 30, 2004
<Position style="position:absolute;top:627px;left:382px;font-family:'Times New
Roman';font-size:29pt;color:#a4a4a4;background-color:#ffffff;white-space:nowrap;"
/>
Overview
```

- To include position information in attributes of <p> tags:

```
(*fpXMLConfig)(pKVXML, KVCFG_ENABLEPOSITIONINFO, TRUE, NULL);
(*fpXMLConfig)(pKVXML, KVCFG_POSITIONINFOOUTPUTTYPE, KVPIOT_ATTRIBUTES, NULL);
```

In this mode, each piece of content output by the reader with a position is put in its own <p> element. Line break (
) tags are not included in the output.

The <p> tags have position information, when this information is available from the reader. These are included in new attributes of the <p> tag: top, left, height, width, and rotation.

The top, left, width, and height attributes are all expressed in pixels. The top and left attributes give the coordinates of the top left corner of the content (an image, text box, and so on) relative to the top left corner of the page. The width and height attributes are the width and height of the content.

Rotation is expressed in degrees, and gives the clockwise rotation of the content about the top left corner. If the rotation attribute is not present, the rotation is assumed to be zero.

NOTE:

Not all readers output all these attributes for all pieces of content. Only pdf2sr outputs width, height and rotation information for text. pdf2sr does not put height and width attributes on <p> tags that enclose images; rather, the tags themselves have the height and width. For example:

```
<p id="p1" font-size="12pt" top="0px" left="0px"></p>
<p id="p2" font-family="MyriadPro-It" font-size="16pt" top="59px"
left="129px" height="21px" width="447px"><i>Aufforderung zur Einreichung
von Vorschlägen 2005:
</i></p>
```

KVXMLConvertFile()

This function is called directly and converts a source file to an output file.

Syntax

```
BOOL pascal KVXMLConvertFile (
    void                *pContext,
    void                *pCallingContext,
    char                *pInFileName,
```

```

char                *pOutFileName,
KVXMLTemplate       *pTemplates,
KVXMLOptions        *pOptions,
KVXMLTOCOptions     *pTOCCreateOptions,
KVXMLCallbacks      *pCallbacks,
BOOL               bIndex,
KVErrCode           *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.
pTemplates	<p>A pointer to the data structure KVXMLTemplate 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 KVXMLTemplate, on page 203.</p> <p>If this pointer is NULL, the default values for the structure are used.</p>
pOptions	<p>A pointer to the data structure KVXMLOptions. 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 KVXMLOptions, on page 195.</p> <p>If this pointer is NULL, the default values for the structure are used.</p>
pTOCCreateOptions	<p>A pointer to the KVXMLTOCOptions data structure. It specifies whether a heading is included in the table of contents. See KVXMLTOCOptions, on page 207.</p> <p>If this pointer is NULL, the default values for the structure are used.</p>
pCallbacks	<p>A pointer to the KVXMLCallbacks data structure. It is a structure of functions that Export calls for specific, user-defined purposes. See KVXMLCallbacks, on page 187.</p> <p>If callbacks are not used, this can be NULL.</p>
bIndex	<p>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 bIndex is set to FALSE, embedded images in a document are regenerated as separate files and stored in the output directory.</p> <p>This can also be set through the bNoPictures member in the template files.</p>
pError	A pointer to an error code if the call to KVXMLConvertFile() 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 pCallbacks is NULL, pOptions->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 25](#).
- When converting out of process, this function must be called after the call to KVXMLStartOOPSession() and before the call to KVXMLEndOOPSession(). See [KVXMLStartOOPSession\(\), on page 170](#) and [KVXMLEndOOPSession\(\), on the next page](#).
- When converting out of process, the values for the KVXMLTemplate, KVXMLOptions, and KVXMLTOCOptions structures should be set to NULL. These structures are already passed in the call to KVXMLStartOOPSession(). See [KVXMLStartOOPSession\(\), on page 170](#).

Example

```
if(!(*KVXMLInt.KVXMLConvertFile)(
    pKVXML,          /* Pointer returned by fpInit() */
    NULL,            /* Pointer for callback functions */
    &InputFile,       /* Input file */
    &OutputFile,      /* Output file */
    &XMLTemplates,    /* Markup and related variables */
    &XMLOptions,      /* Options */
    NULL,            /* TOC options */
    NULL,            /* A pointer to callback functions */
    FALSE,           /* Index mode */
    &error))          /* Error return value */
{
    printf("Error converting %s to XML %d\n", argv[i - 1], error);
}
else
{
    printf("Conversion of %s to XML completed.\n\n", argv[i - 1]);
}
```

KVXMLEndOOPSession()

This function terminates the current out-of-process conversion session, and releases the source data and resources related to the session.

Syntax

```
BOOL pascal KVXMLEndOOPSession(  
    void          *pContext,  
    BOOL          bKeepServantAlive,  
    KVErrCodeEx   *pError  
    DWORD         dwOptions,  
    void          *pReserved1,  
    void          *pReserved2 );
```

Arguments

pContext	A pointer returned from fpInit().
bKeepServantAlive	<p>Set bKeepServantAlive 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.</p> <p>Set bKeepServantAlive to FALSE to terminate the Export out-of-process session and the associated Servant process.</p>
pError	A pointer to an error code defined in KVErrCodeEx 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 `cnv2xmlloop` sample program:

```

/* declare endsession function pointer */
BOOL (pascal *fpKVXMLEndOOPSession)( void *,
        BOOL
        ,
        KVErrorCode
        *,
        DWORD
        ,
        void
        *,
        void
        *);
/* assign OOP endsession function pointer */
fpKVXMLEndOOPSession = (BOOL (pascal *) ( void *,
        BOOL
        ,
        KVErrorCode
        *,
        DWORD
        ,
        void
        *,
        void
        * ))mpGetProcAddress(hKVXML,
"KVXMLEndOOPSession");
if(!fpKVXMLEndOOPSession)
{
    printf("Error assigning KVXMLEndOOPSession() pointer\n");
    (*KVXMLInt.fpFileToInputStreamFree)(pKVXML, &Input);
    (*KVXMLInt.fpFileToOutputStreamFree)(pKVXML, &Output);
    mpFreeLibrary(hKVXML);
    return 8;
}
/*****END OOP SESSION, DO NOT KEEP SERVANT ALIVE *****/
if(!(*fpKVXMLEndOOPSession)(pKVXML,
        FALSE,
        &error,
        0,
        NULL,
        NULL))
{
    printf("Error calling fpKVXMLEndOOPSession \n");
    (*KVXMLInt.fpFileToInputStreamFree)(pKVXML, &Input);
    (*KVXMLInt.fpFileToOutputStreamFree)(pKVXML, &Output);
    (*KVXMLInt.fpShutDown)(pKVXML);
    mpFreeLibrary(hKVXML);
    return 10;
}

```

KVXMLSetStyleSheet()

This function is called directly and is used to specify the full path and file name of an external Style Sheet (XSL or CSS).

Syntax

```
BOOL pascal KVXMLSetStyleSheet(  
    void      *pContext,  
    char      *pszStyleSheetName,  
    char      *pszRef);
```

Arguments

<code>pContext</code>	A pointer returned from <code>fpInit()</code> .
<code>pszStyleSheetName</code>	A pointer to the full path and file name of the style sheet.
<code>pszUrlRef</code>	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 `KVXMLOptions` is set to `XML_XSL` or `XML_CSS`, an external style sheet is referenced by a processing instruction of the form:

```
<?xml-stylesheet href="pszRef" type="text/xsl"?>
```

or

```
<?xml-stylesheet href="pszRef" type="text/css"?>
```

- If the value for `pszStyleSheetName` includes the output directory, the `href` only consists of the file name since the XML output resides in the same directory as the style sheet 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.
- Style sheet information cannot be written to an external XSL file. XML Export can only reference an existing XSL style sheet.
- When `XML_CSS` is specified, a CSS file can be created based on `pszStyleSheetName`.
- If the name of the CSS is not specified by using this function, a CSS style file is created with an automatically-generated file name.
- If this function is used to specify the name of the style file, that file is referenced in the processing instruction.

- If the CSS file does not exist in the specified location, it is created.
- If it exists, but is empty, CSS styles are written to it.
- If the CSS file exists and is not empty, the file is not altered. There is no attempt made to validate the file.
- If there are multiple calls made to `fpConvertStream()` or `KVXMLConvertFile()`, and the name of the style sheet has been set with `KVXMLSetStyleSheet`, the file name can be disabled by calling `KVXMLSetStyleSheet` again with the `pszStyleSheetName` and `pszRef` set to `NULL`. The file name can then be set to a different value by calling `KVXMLSetStyleSheet` with the new file name prior to the next call to `fpConvertStream()` or `KVXMLConvertFile()`.
- This function runs in-process or out of process. See [Convert Files Out of Process, on page 25](#).
- When converting out of process, this function must be called after the call to `KVXMLStartOOPSession()` and before the call to `KVXMLEndOOPSession()`. See [KVXMLStartOOPSession\(\), below](#) and [KVXMLEndOOPSession\(\), on page 167](#).

KVXMLStartOOPSession()

This function performs the following:

- Initializes the out-of-process session.
- Specifies the input stream or file.
- Sets conversion options in the `KVXMLTemplate`, `KVXMLOptions`, and `KVXMLTOCOptions` 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 KVXMLStartOOPSession(  
    void                *pContext,  
    KVInputStream        *pInputStream,  
    char                *pFileName,  
    KVXMLTemplate        *pTemplates,  
    KVXMLOptions         *pOptions,  
    KVXMLTOCOptions      *pTOCCreateOptions  
    DWORD               *pPID,  
    KVErrCode           *pError  
    DWORD               dwOptions,  
    void                *pReserved1,  
    void                *pReserved2 );
```

Arguments

<code>pContext</code>	A pointer returned from <code>fpInit()</code> .
<code>pInputStream</code>	<p>A pointer to the developer-assigned instance of <code>KVInputStream</code>. The <code>KVInputStream</code> structure defines the input stream containing the source for the conversion.</p> <p>If <code>pInput</code> is defined, <code>pFileName</code> must be <code>NULL</code>. The input data can be defined as a data stream or file, but not both.</p>
<code>pFileName</code>	<p>A pointer to the file to be converted. The file must exist on the same file system as the Servant.</p> <p>If <code>pFileName</code> is defined, <code>pInput</code> must be <code>NULL</code>. The input data can be defined as a data stream or file, but not both.</p>
<code>pTemplatesEx</code>	<p>A pointer to the <code>KVXMLTemplate</code> 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 KVXMLTemplate, on page 203.</p> <p>If this pointer is <code>NULL</code>, the default values for the structure are used.</p>
<code>pOptionsEx</code>	<p>A pointer to the <code>KVXMLOptions</code> 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 KVXMLOptions, on page 195.</p> <p>If this pointer is <code>NULL</code>, the default values for the structure are used.</p>
<code>pTOCCreateOptions</code>	<p>A pointer to the <code>KVXMLTOCOptions</code> data structure. It specifies whether a heading is included in the table of contents. See KVXMLTOCOptions, on page 207.</p> <p>If this pointer is <code>NULL</code>, the default values for the structure are used.</p>
<code>pPID</code>	The address of a <code>DWORD</code> into which the Servant process ID is returned.
<code>pError</code>	A pointer to an error code defined in <code>KVErrorCode</code> in <code>kvtypes.h</code> .
<code>dwOptions</code>	Reserved for future use.
<code>pReserved1</code>	Reserved for future use.
<code>pReserved2</code>	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 [KVXMLEndOOPSession\(\)](#), on page 167.
- All functions that can run out of process must be called within the out-of-process session, that is, after the call to `KVXMLStartOOPSession()`, and before the call to `KVXMLEndOOPSession()`.
- The `KVXMLConvertFile()`, and `fpGetSummary()` functions can be called only once in a single out-of-process session.
- Because the `KVXMLTemplate`, `KVXMLOptions`, and `KVXMLTOCOptions` data structures are passed by this function, the same pointers in the call to `KVXMLConvertFile()` are ignored.

Example

The following sample code is from the `cnv2xmlloop` sample program:

```
/* declare OOP startsession function pointer */
BOOL (pascal *fpKVXMLStartOOPSession)( void *,
    KVInputStream *,
    char *,
    KVXMLTemplate *,
    KVXMLOptions *,
    KVXMLTOCOptions *,
    DWORD *,
    KVErrCode *,
    DWORD,
    void *,
    void * );

/* assign OOP startsession function pointer */
fpKVXMLStartOOPSession = (BOOL (pascal *) ( void *,
    KVInputStream *,
    char *,
    KVXMLTemplate *,
    KVXMLOptions *,
    KVXMLTOCOptions *,
    DWORD *,
    KVErrCode *,
    DWORD,
    void *,
    void * ))mpGetProcAddress(hKVXML,
"KVXMLStartOOPSession");
if(!fpKVXMLStartOOPSession)
{
    printf("Error assigning KVXMLStartOOPSession() pointer\n");
    (*KVXMLInt.fpFileToInputStreamFree)(pKVXML, &Input);
    (*KVXMLInt.fpFileToOutputStreamFree)(pKVXML, &Output);
}
```

```
    mpFreeLibrary(hKVXML);
    return 7;
}
/*****START OOP SESSION *****/
if(!(*fpKVXMLStartOOPSession)(pKVXML,
    &Input,
    NULL,
    &XMLTemplates,      /* Markup and related variables */
    &XMLOptions,         /* Options */
    NULL,               /* TOC options */
    &oopServantPID,
    &error,
    0,
    NULL,
    NULL))
{
    printf("Error calling fpKVXMLStartOOPSession \n");
    (*KVXMLInt.fpFileToInputStreamFree)(pKVXML, &Input);
    (*KVXMLInt.fpFileToOutputStreamFree)(pKVXML, &Output);
    (*KVXMLInt.fpShutDown)(pKVXML);
    mpFreeLibrary(hKVXML);
    return 9;
}
```

Chapter 9: XML Export API Callback Functions

This section describes the XML Export API callback functions.

• Introduction	174
• Continue()	174
• GetAnchor()	175
• GetAuxOutput()	177
• UserCB()	178

Introduction

The `fpConvertStream()` and `KVXMLConvertFile()` 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 `KVXMLCallbacks` 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 `KVXMLCallbacks` can define unique callback functions. Alternatively, the functions can be common to all instances of `KVXMLCallbacks`; these functions take appropriate action, depending on the value of the pointer `pCallingContext`.

The second parameter (`pCallingContext`) of the call to `fpConvertStream()` and `KVXMLConvertFile()` provides a void pointer used to identify the context of this call. If more than one call to `fpConvertStream()` or `KVXMLConvertFile()` 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 `KVXMLConvertFile()` must be set to the address of the `KVXMLCallbacks` structure to be used for this call.

For sample code, see the sample program `xmlcallback.c`. It creates an XML stream and demonstrates the use of the callback functions.

Continue()

When `fpConvertStream()` or `KVXMLConvertFile()` 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 `KVXMLConvertFile()`.
- 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 `<a xmlns:xlink= xlink href=>` 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                *pHTML,
    UINT                cbHTML);
```

Arguments

<code>pCallingContext</code>	A pointer that gets passed back to the caller-provided callback functions. This pointer, which can be <code>NULL</code> , is specified as the second parameter of the call to <code>fpConvertStream()</code> .
<code>eAnchorType</code>	The anchor type for the output stream. It must be one of the enumerated types defined in <code>KVXMLAnchorType</code> .
<code>pszAnchor</code>	A pointer to the location where the new anchor is stored.
<code>cbAnchorMax</code>	The maximum number of bytes to place in <code>pszAnchor</code> .
<code>pHTML</code>	This is either <code>NULL</code> or a pointer to one of the following: <ul style="list-style-type: none">• markup defining the contents of a table of contents entry• the external graphic file name• the external file name
<code>cbHTML</code>	The number of valid bytes in <code>pHTML</code> .

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 `eAnchorType` is `VectorPictureAnchor` or `RasterPictureAnchor`. For all other anchor types, call with the same parameters you were passed.
- `pszAnchor` must be assigned. It can be derived from the `cbAnchorMax`, `pHTML`, and `cbHTML` values, which are also provided.
- `pHTML` 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

<code>pCallingContext</code>	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 <code>fpConvertStream()</code> .
<code>eAnchorType</code>	A graphic or block anchor as defined by the enumerated types in <code>KVXMLAnchorType</code> .
<code>pszAnchor</code>	A pointer to location where a new anchor is stored. <code>pszAnchor</code> is based on the call to <code>GetAnchor()</code> .
<code>pNewOutput</code>	A pointer to a <code>KVOutputStream</code> 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 `KVXMLOptions` 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 `$USERCB` token in a member of `KVXMLTemplate`. For example, placing "`$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

<code>pCallingContext</code>	A pointer that gets passed back to the caller-provided callback function. This pointer, which can be <code>NULL</code> , is specified as the second parameter of the call to <code>fpConvertStream()</code> .
<code>pszUserCBid</code>	A pointer to a string that identifies the source of the callback. The identifier must be delimited by a trailing white space. For example, " <code>my_callback</code> ".
<code>pNewOutput</code>	A pointer to a <code>KVOutputStream</code> structure that can be used to write data to the current block.
<code>pReserved</code>	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: XML Export API Structures

This section provides information on the structures used by the XML Export API. These structures are defined in `kvxml.h`, `kvtypes.h`, and `adinfo.h`.

• ADDOCINFO	179
• KVInputStream	180
• KVMemoryStream	181
• KVOutputStream	181
• KVSTR	182
• KVStreamInfo	182
• KVStructHead	183
• KVStyle	184
• KVSumInfoElemEx	185
• KVSummaryInfoEx	185
• KVXConfigInfo	186
• KVXMLCallbacks	187
• KVXMLHeadingInfo	188
• KVXMLImageInfo	190
• KVXMLInterface	191
• KVXMLInterfaceEx	193
• KVXMLOptions	195
• KVXMLTemplate	203
• KVXMLTOCOptions	207

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 151](#).

```
typedef struct
{
    ENdocClass      eClass;
    ENdocFmt        eFormat;
    long            lVersion;
    unsigned long    ulAttributes;
}
ADDOCINFO, *ADDOCINFOPTR;
```

Member Descriptions

<code>eClass</code>	The file class of the source document (for example, spreadsheet, word processor, or encapsulation format) as defined by the <code>ENdocClass</code> enumerated type in <code>adinfo.h</code> .
<code>eFormat</code>	The major format of the source document (such as Microsoft Word or Corel Presentation) as defined by the <code>ENdocFmt</code> enumerated type in <code>adinfo.h</code> .
<code>lVersion</code>	The version number of the file format. The number is multiplied by 1000. For example, 1.02 is represented by 1020.
<code>ulAttributes</code>	Other attributes of the document as defined by the <code>ENdocAttributes</code> enumerated type in <code>adinfo.h</code> .

Discussion

When format detection is enhanced in future releases, new format IDs might be added to the `ENdocFmt` enumerated type. When 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 XML 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 XML Export. It is initialized by calling `fpInit()`. See [fpInit\(\)](#), on page 153.

```
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 XML 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 184.

```
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 151.

```
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 71.

`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 179.

- `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

<code>version</code>	The current KeyView version number. This is a symbolic constant (<code>KeyviewVersion</code>) defined in <code>kvxtract.h</code> . This constant is updated for each KeyView release.
<code>size</code>	The size of the <code>KVStructHeadRec</code> .
<code>reserved</code>	Reserved for internal use.
<code>internal</code>	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 182](#)). 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 155](#) and [Map Styles, on page 75](#).

XML 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 XML 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 182 . 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 182 .
MarkupEnd	The markup added to the end of a paragraph or character style. A pointer to the KVSTR structure. See KVSTR, on page 182 .
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 77 . 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, XML 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 XML 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

<code>isValid</code>	Specifies whether the data value is present in the document. The setting 1 specifies that the value is valid and exists.
<code>type</code>	The data type of the metadata element. The types are defined in the <code>KVSumInfoType</code> structure in <code>kvtypes.h</code> . See KVSumInfoType, on page 223 .
<code>data</code>	The content of the metadata field. If the <code>type</code> member is <code>KV_Int4</code> or <code>KV_Bool</code> , this member contains the actual value. Otherwise, this member is a pointer to the actual value. <code>KV_DateTime</code> and <code>KV_IEEE8</code> point to an 8-byte value. <code>KV_String</code> and <code>KV_Unicode</code> point to the beginning of the string that contains the text.
<code>pcType</code>	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 152](#).

```
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 92](#).

```
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 350](#) 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 96.](#)

pszInMeta	<p>The elements extracted from the file as metadata. All other elements are extracted as text. Multiple entries must be separated by commas.</p> <p>To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 96.</p>
pszExMeta	<p>The child elements in the included metadata elements that are not extracted from the file as metadata. For example, the default extraction settings for the Visio XML format extract the <code>DocumentProperties</code> element as metadata. This element includes child elements such as <code>Title</code>, <code>Subject</code>, <code>Author</code>, <code>Description</code>, and so on. However, the child element <code>PreviewPicture</code> is defined in <code>pszExMeta</code> because it is binary data and should not be extracted.</p> <p>You cannot exclude any metadata elements from the output for StarOffice files. All metadata is extracted regardless of this setting.</p> <p>To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 96.</p>
pszInContent	<p>The elements extracted from the file as content text. An asterisk (*) extracts all elements including child elements.</p> <p>To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 96.</p>
pszExContent	<p>The child elements in the included content elements that are not extracted from the file as content text.</p> <p>To further qualify the element, specify its namespace, its attributes, or both. See Specify an Element's Namespace and Attribute, on page 96.</p>
pszInAttribute	<p>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:</p> <p><i>namespace:elementname@attributename</i></p> <p>For example:</p> <p><code>microfocus:division@name</code></p>

KVXMLCallbacks

This structure provides all callbacks that can result from a call to `fpConvertStream()` or `KVXMLConvertFile()`. See [fpConvertStream\(\), on page 139](#) and [KVXMLConvertFile\(\), on page 164](#). Any and all of the function pointers can be NULL.

```
typedef BOOL (pascal *KVXMLCB_CONTINUE)(
    void                *pcallingContext,
    int                 nPercentDone);
typedef BOOL (pascal *KVXMLCB_GETANCHOR)(
    void                *pCallingContext,
```

```
KVXMLAnchorType    eAnchorType,
char               *pszAnchor,
Int                cbAnchorMax,
BYTE               *pHTML,
UINT               cbHTML);
typedef BOOL (pascal *KVXMLCB_GETAUXOUTPUT)(
    void            *pCallingContext,
    KVXMLAnchorType eAnchorType,
    char            *pszAnchor,
    KVOutputStream *pNewOutput);
typedef BOOL (pascal *KVXMLCB_USERCB) (
    void            *pCallingContext,
    char            *psUserCBid,
    KVOutputStream *pOutput,
    void            *pReserved);
typedef struct tag_KVXMLCallbacks
{
    KVXMLCB_CONTINUE      fpContinue;
    KVXMLCB_GETANCHOR     fpGetAnchor;
    KVXMLCB_GETAUXOUTPUT  fpGetAuxOutput;
    KVXMLCB_USERCB        fpUserCB;
}
KVXMLCallbacks;
```

Member Descriptions

- The members of this structure are function pointers to the functions described in [XML Export API Callback Functions](#), on page 174.
- If `fpGetAuxOutput()` is NULL, the `pszDefaultOutputDirectory` member of the instance of `KVXMLOptions` 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 [KVXMLOptions](#), on page 195.

KVXMLHeadingInfo

This structure defines how XML 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
- you set the `headingCreateType` member of `KVXMLTOCOptions` to allow automatic heading generation.

XML Export evaluates the text against each member in the order in which the members appear below.

See [KVXMLTOCOptions](#), on page 207 for more information on automatic generation of headings.

```
typedef struct tag_KVXMLHeadingInfo
{
```

```
int    minParaLen;  
int    maxParaLen;  
int    fontSizeMin;  
int    fontSizeMax;  
BOOL   bMustBeBold;  
BOOL   bMustBeItalic;  
BOOL   bMustBeUnderlined;  
BOOL   bNonZeroIndent;  
BOOL   bNoTabs;  
BOOL   bNoMultiSpaces;  
int    nSpaceBefore;  
int    nSpaceAfter;  
}  
KVXMLHeadingInfo;
```

Member Descriptions

<code>minParaLen</code>	<p>The minimum number of characters that a paragraph in the source document can contain for the text to meet the criteria for heading conversion.</p> <p>This option applies to word processing documents only.</p> <p>The default is 3 for heading levels 1 to 3.</p>
<code>maxParaLen</code>	<p>The maximum number of characters that a paragraph in the source document can contain for the text to meet the criteria for heading conversion.</p> <p>This option applies to word processing documents only.</p> <p>The default is 80 for heading levels 1 to 3.</p>
<code>fontSizeMin</code>	<p>The minimum font size of text in the source document for the text to meet the criteria for heading conversion.</p> <p>The default is 14 for heading level 1, and 12 for heading levels 2 and 3.</p>
<code>fontSizeMax</code>	<p>The maximum font size of text in the source document for the text to meet the criteria for heading conversion.</p> <p>The default is 20 for heading level 1, and 14 for heading levels 2 and 3.</p>
<code>bMustBeBold</code>	<p>If you set <code>bMustBeBold</code> to <code>TRUE</code>, the text in the source document must be bold to meet the criteria for heading conversion.</p> <p>The default is <code>TRUE</code> for heading levels 1 and 2, and <code>FALSE</code> for heading level 3.</p>
<code>bMustBeItalic</code>	<p>If you set <code>bMustBeItalic</code> to <code>TRUE</code>, the text in the source document must be italic to meet the criteria for heading conversion.</p> <p>The default is <code>FALSE</code>.</p>
<code>bMustBeUnderlined</code>	<p>If you set <code>bMustBeUnderlined</code> to <code>TRUE</code>, the text in the source document must be underlined to meet the criteria for heading conversion.</p> <p>The default is <code>FALSE</code>.</p>

bNonZeroIndent	<p>If you set <code>bNonZeroIndent</code> to <code>TRUE</code>, the text in the source document must be indented to meet the criteria for heading conversion. If you set <code>bNonZeroIndent</code> to <code>FALSE</code>, the text must be aligned left.</p> <p>The default is <code>FALSE</code>.</p>
bNoTabs	<p>If you set <code>bNoTabs</code> to <code>TRUE</code>, the text in the source document must <i>not</i> contain tabs to meet the criteria for heading conversion.</p> <p>The default is <code>FALSE</code>.</p>
bNoMultiSpaces	<p>If you set <code>bNoMultiSpaces</code> to <code>TRUE</code>, the text in the source document must <i>not</i> contain two or more contiguous white spaces to meet the criteria for heading conversion.</p> <p>The default is <code>FALSE</code>.</p>
nSpaceBefore	<p>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 <code>-1</code> is used, the amount of space before the paragraph is not considered in the heading generation.</p> <p>The default is <code>0</code>.</p>
nSpaceAfter	<p>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 <code>-1</code> is used, the amount of space after the paragraph is not considered in the heading generation.</p> <p>The default is <code>0</code>.</p>

KVXMLImageInfo

This structure contains the dimensions of an image in pixels. It is defined in `kvxml.h`. You must initialize it by calling `KVStructInit()` before you obtain image dimensions by using the `fpGetOutputImageInfo()` function.

```
typedef struct tag_KVXMLImageInfo{
    KVStructHeader;
    int nWidth;
    int nHeight;
}
KVXMLImageInfo;
```

Member Descriptions

KVStructHeader	The KeyView version of the structure. See KVStructHead , on page 183.
nWidth	The image width in pixels.
nHeight	The image height in pixels.

KVXMLInterface

The members of this structure are pointers to the API functions described in [XML Export API Functions, on page 137](#).

NOTE:

This structure has been superseded by [KVXMLInterfaceEx](#); KVXMLInterfaceEx should be used instead of KVXMLInterface.

```
typedef void* (pascal *KVXML_INIT) (
    KVMemoryStream      *pMemAllocator,
    char                 *pszKeyViewDir,
    char                 *pszDataFile,
    KVErrCode            *pError,
    DWORD                dWord);
typedef void (pascal *KVXML_SHUTDOWN)(void*);
typedef BOOL (pascal *KVXML_CONVERT_STREAM) (
    void *pContext,
    void                *pCallingContext,
    KVInputStream        *pInput,
    KVOutputStream      *pOutput,
    KVXMLTemplate        *pTemplates,
    KVXMLOptions         *pOptions,
    KVXMLTOCOptions      *pTOCCreateOptions,
    KVXMLCallbacks       *pCallbacks,
    BOOL                bIndex,
    KVErrCode            *pError);
typedef char** (pascal *KVXML_GET_FILE_LIST)(
    void                *pContext,
    int                 *pnSize );
typedef BOOL (pascal *KVXML_GET_STREAM_INFO)(
    void                *pContext,
    KVInputStream        *pInput,
    KVStreamInfo         *pStreamInfo );
typedef BOOL (pascal *KVXML_GET_ANCHOR) (
    void                *pCallingContext,
    KVXMLAnchorType      eAnchorType,
    char                 *pszAnchor,
    int                  cbAnchorMax,
    BYTE                 *pcHTML,
    UINT                 cbHTML);
typedef BOOL (pascal *KVXML_INPUTSTREAM_CREATE) (
    void                *pContext,
    char                 *pszFileName,
    KVInputStream        *pInput);
typedef BOOL (pascal *KVXML_INPUTSTREAM_FREE) (
    void                *pContext,
    KVInputStream        *pInput);
```

```
typedef BOOL (pascal *KVXML_OUTPUTSTREAM_CREATE) (
    void                *pContext,
    char                *pszFileName,
    KVOutputStream      *pOutput );
typedef BOOL (pascal *KVXML_OUTPUTSTREAM_FREE)(
    void                *pContext,
    KVOutputStream      *pOutput );
typedef KVLanguageID (pascal *KVXML_LANGUAGE_ID)(void *pContext);
typedef BOOL (pascal *KVXML_GET_SUMMARY_INFO)(
    void                *pContext,
    KVInputStream       *pInput,
    KVSummaryInfoEx     *pSummary,
    BOOL                bFree );
typedef BOOL (pascal *KVXML_SET_STYLE_MAPPING) (
    void                *pContext,
    KVStyle             *pStyles,
    int                 iStyles,
    BOOL                bCopy);
typedef BOOL (pascal *KVXML_VALIDATE_TEMPLATE)(
    void *pContext,
    KVOutputStream      *pOutput,
    KVXMLTemplate       *pTemplate,
    KVXMLOptions        *pOptions,
    KVXMLTOCOptions     *pTOCOptions,
    KVXMLCallbacks      *pCallBalls,
    KVMemoryStream      *pMemStream)
typedef struct tag_KVXMLInterface
{
    KVXML_INIT                fpInit;
    KVXML_SHUTDOWN            fpShutDown;
    KVXML_CONVERT_STREAM     fpConvertStream;
    KVXML_GET_FILE_LIST      fpGetConvertFileList;
    KVXML_GET_STREAM_INFO    fpGetStreamInfo;
    KVXML_GET_ANCHOR         fpGetAnchor;
    KVXML_INPUTSTREAM_CREATE fpFileToInputStreamCreate;
    KVXML_INPUTSTREAM_FREE   fpFileToInputStreamFree;
    KVXML_OUTPUTSTREAM_CREATE fpFileToOutputStreamCreate;
    KVXML_OUTPUTSTREAM_FREE  fpFileToOutputStreamFree;
    KVXML_GET_SUMMARY_INFO   fpGetSummaryInfo;
    KVXML_SET_STYLE_MAPPING  fpSetStyleMapping;
    KVXML_VALIDATE_TEMPLATE  fpValidateTemplate;
}
                                KVXMLInterface;
```


Member Descriptions

The members of this structure are function pointers to the functions described in [XML Export API Functions, on page 137](#).

KVXML_VALIDATE_TEMPLATE is currently not implemented.

KVXMLInterfaceEx

The members of this structure are pointers to the API functions described in [XML Export API Functions, on page 137](#).

This structure supersedes KVXMLInterface. KVXMLInterfaceEx should be used instead of KVXMLInterface.

Compared to KVXMLInterface, KVXMLInterfaceEx adds two functions for checking error codes, and allows for binary compatible extensibility in future releases.

```
typedef void* (pascal *KVXML_INIT) (
    KVMemoryStream *pMemAllocator,
    char *pszKeyViewDir,
    char *pszDataFile,
    KVErrorCode *,
    DWORD dWord);
typedef void (pascal *KVXML_SHUTDOWN)(void*);
typedef BOOL (pascal *KVXML_CONVERT_STREAM) (
    void *pContext,
    void *pCallingContext,
    KVInputStream *pInput,
    KVOutputStream *pOutput,
    KVXMLTemplate *pTemplates,
    KVXMLOptions *pOptions,
    KVXMLTOCOptions *pTOCCreateOptions,
    KVXMLCallbacks *pCallbacks,
    BOOL bIndex,
    KVErrorCode *pError);
typedef char** (pascal *KVXML_GET_FILE_LIST)(
    void *pContext,
    int *pnSize );
typedef BOOL (pascal *KVXML_GET_STREAM_INFO)(
    void *pContext,
    KVInputStream *pInput,
    KVStreamInfo *pStreamInfo );
typedef BOOL (pascal *KVXML_GET_ANCHOR) (
    void *pCallingContext,
    KVXMLAnchorType eAnchorType,
    char *pszAnchor,
    int cbAnchorMax,
    BYTE *pHTML,
```

```

        UINT cbHTML);
typedef BOOL (pascal *KVXML_INPUTSTREAM_CREATE) (
    void *pContext,
    char *pszFileName,
    KVInputStream *pInput);
typedef BOOL (pascal *KVXML_INPUTSTREAM_FREE) (
    void *pContext,
    KVInputStream *pInput);
typedef BOOL (pascal *KVXML_OUTPUTSTREAM_CREATE) (
    void *pContext,
    char *pszFileName,
    KVOutputStream *pOutput );
typedef BOOL (pascal *KVXML_OUTPUTSTREAM_FREE)(
    void *pContext,
    KVOutputStream *pOutput );
typedef KVLanguageID (pascal *KVXML_LANGUAGE_ID)(void *pContext);
typedef BOOL (pascal *KVXML_GET_SUMMARY_INFO)(
    void *pContext,
    KVInputStream *pInput,
    KVSummaryInfoEx *pSummary,
    BOOL bFree );
typedef BOOL (pascal *KVXML_SET_STYLE_MAPPING) (
    void *pContext,
    KVStyle *pStyles,
    int iStyles,
    BOOL bCopy);
typedef BOOL (pascal *KVXML_VALIDATE_TEMPLATE)(
    void *pContext,
    KVOutputStream *pOutput,
    KVXMLTemplate *pTemplate,
    KVXMLOptions *pOptions,
    KVXMLTOCOptions *pTOCOptions,
    KVXMLCallbacks *pCallBalls,
    KVMemoryStream *pMemStream);
typedef KVErrCode(pascal *KVXML_GET_KV_ERROR_CODE) (void *);
typedef KVErrCodeEx(pascal *KVXML_GET_KV_ERROR_CODE_EX) (void *);

typedef struct tag_KVXMLInterfaceEx
{
    KVStructHeader;
    KVXML_INITEX fpInit;
    KVXML_SHUTDOWN fpShutDown;
    KVXML_CONVERT_STREAMEX fpConvertStream;
    KVXML_GET_FILE_LIST fpGetConvertFileList;
    KVXML_GET_STREAM_INFO fpGetStreamInfo;
    KVXML_GET_ANCHOREX fpGetAnchor;
    KVXML_INPUTSTREAM_CREATE fpFileToInputStreamCreate;
    KVXML_INPUTSTREAM_FREE fpFileToInputStreamFree;

```

```

        KVXML_OUTPUTSTREAM_CREATE  fpFileToOutputStreamCreate;
        KVXML_OUTPUTSTREAM_FREE    fpFileToOutputStreamFree;
        KVXML_GET_SUMMARY_INFO     fpGetSummaryInfo;
        KVXML_SET_STYLE_MAPPING    fpSetStyleMapping;
        KVXML_VALIDATE_TEMPLATE    fpValidateTemplate;
        KVXML_GET_KV_ERROR_CODE    fpGetKvErrorCode;
        KVXML_GET_KV_ERROR_CODE_EX fpGetKvErrorCodeEx;
    }
    KVXMLInterfaceEx;

```

KVXMLOptions

This structure defines the options that control the XML 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 `fpConvertStream()` or `KVXMLConvertFile()` function. See [fpConvertStream\(\), on page 139](#) or [KVXMLConvertFile\(\), on page 164](#).

```

typedef struct tag_KVXMLOptions
{
    BOOL                bUseVerityDTD;
    char                *pszVerityDTDPath;
    KVXMLStyleSheetType eStyleSheetType
    BOOL                bUseExistingStyleSheet;
    char                *pszStyleSheet;
    BOOL                bIndexOnly;
    KVCharSet           eOutputCharSet;
    BOOL                bForceOutputCharSet;
    KVCharSet           eSrcCharSet;
    BOOL                bForceSrcCharSet;
    KVLanguageID        eOutputLanguageID;
    BOOL                bUseDocumentColors;
    BOOL                bUseDocumentFontInfo;
    BOOL                bNbspEmptyCells;
    ENSATableBorder     eSATableBorder;
    int                 nTableBorderWidth;
    char                *pszBaseURL;
    char                *pszMainURL;
    char                *pszDefaultOutputDirectory;
    char                *pszPicPath;
    char                *pszPicURL;
    char                *pszJavaURL;
    BOOL                bRemoveFileNameSpaces;
    BOOL                bRasterizeFiles
    KVXMLGraphicType    eOutputRasterGraphicType;
    KVXMLGraphicType    eOutputVectorGraphicType;
    int                 cxVectorToRasterXRes;
    int                 cyVectorToRasterYRes;
    int                 nCompressionQuality;

```

```

        BOOL                bGenerateURLs;
        long                lcbMaxMemUsage;
        BYTE                cReplaceChar;
        BYTE                cRedact;
        KVXMLEmptyParaType  eEmptyParaType;
        KVXMLHardPageBreakType eHardPageBreakType;
        BOOL                bSupportColumnHeadings;
        BOOL                bSupportRowHeadings;
        BOOL                bSupportCellSpan;
        BOOL                bSupportRowSpan;
        BOOL                bSupportColumnWidth;
        BOOL                bRemoveEmptyColumns;
        BOOL                bRemoveEmptyRows;
        BOOL                bEnableEmptyRows;
        int                 nRowsBeforeSplit;
    }
    KVXMLOptions;

```

Member Descriptions

bUseVerityDTD

Set bUseVerityDTD to TRUE to generate XML based on the Verity DTD. For more information, see [Use the Verity Document Type Definition \(DTD\), on page 42](#). This generates a valid XML document suitable as a general interchange format. If you set bUseVerityDTD to FALSE, the XML is based on the source document's paragraph structure.

The default is TRUE.

pszVerityDTDPath

If you move the Verity DTD from the default tempout directory to another output directory, set the string value of pszVerityDTDPath to the new location. This path is added to the document type declaration in the XML file.

The default is no path, that is, the DTD is assumed to be in the same directory as the generated XML files.

eStyleSheetType

One of the enumerated options for processing style sheet information. The options are defined in KVXMLStyleSheetType in kvxml.h. See [KVXMLStyleSheetType, on page 216](#).

- **STYLESHEET_DISABLED**—Disables style sheet formatting. This is the default option.
- **XML_CSS**—Enables Cascading Style Sheet (CSS) formatting, and outputs the generated formatting data in an external CSS file referenced in the XML output as a tag.
- **XML_XSL**—Enables Extensible Style Sheet Language (XSL) formatting, and uses an external XSL file referenced in a `<?xml-stylesheet...?>` processing instruction.

<code>bUseExistingStyleSheet</code>	<p>Set <code>bUseExistingStyleSheet</code> to <code>TRUE</code> to apply an existing XSL style sheet or a CSS file to an XML document. The style sheet file name is inserted into the type declaration at the beginning of the XML file. The location of the external style sheet file is set by <code>pszStyleSheet</code>. If <code>pszStyleSheet</code> is not specified and the style sheet type is XSL, a default XSL style sheet appropriate for the source document type is used. The default XSL style sheets are:</p> <ul style="list-style-type: none">• <code>wp.xls</code> (for word processing documents)• <code>ss.xls</code> (for spreadsheets)• <code>pg.xls</code> (for presentations) <p>If <code>pszStyleSheet</code> is not specified and the style sheet type is CSS, a CSS file is created.</p> <p>Existing style sheets are not validated.</p> <p>The default is <code>FALSE</code>.</p>
<code>pszStyleSheet</code>	<p>The path and file name of an external style sheet.</p> <p>The default is no path.</p>
<code>bIndexOnly</code>	<p>Set <code>bIndexOnly</code> to <code>TRUE</code> to generate output with minimal markup (ID and style paragraph attributes) and without images. Because the generated output is minimized to textual content, it is suitable for an indexing engine. If you set <code>bIndexOnly</code> to <code>FALSE</code>, embedded images in a document are regenerated as separate files and stored in the output directory.</p> <p>The template file named <code>xml_index.ini</code> and the <code>xmlindex</code> sample program demonstrate the effect of setting <code>bIndexOnly</code>.</p> <p>To generate output with verbose markup and without images, set the <code>nType</code> argument of the <code>KVXMLConfig()</code> function to <code>KVCFG_SUPPRESSIMAGES</code>. See KVXMLConfig(), on page 156.</p> <p>The default is <code>FALSE</code>.</p>
<code>eOutputCharSet</code>	<p>The character set to use for textual output. To ensure that the character set defined here is used, you must set <code>bForceOutputCharSet</code> to <code>TRUE</code>. The available character sets are enumerated in <code>KVCharSet</code> in <code>kvtypes.h</code>. See Convert Character Sets, on page 71.</p> <p>Supported Formats, on page 230 lists the file formats for which character set information can be determined.</p> <p>The default is <code>KVCS_UNKNOWN</code>.</p>
<code>bForceOutputCharSet</code>	<p>Set <code>bForceOutputCharSet</code> to <code>TRUE</code> to use the output character set specified in <code>eOutputCharSet</code>, regardless of the internal document information or the source character set specified by <code>eSrcCharSet</code>. See Convert Character Sets, on page 71.</p>

	<p>Forcing a character set to KVCS_UNKNOWN is always ignored.</p> <p>The default is FALSE.</p>
eSrcCharSet	<p>This option specifies the character set of the document. To ensure that the character set defined here is used, you must set bForceSrcCharSet to TRUE. The available character sets are enumerated in KVCharSet in kvtypes.h. See Convert Character Sets, on page 71. Supported Formats, on page 230 lists the file formats for which character set information can be determined.</p> <p>The default is KVCS_UNKNOWN.</p>
bForceSrcCharSet	<p>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 71.</p> <p>Forcing a character set to KVCS_UNKNOWN is always ignored.</p> <p>The default is FALSE.</p>
eOutputLanguageID	<p>The language for the textual output of language-specific data such as time and date. eOutputLanguageID must be in the system locale. If eOutputLanguageID is invalid or not supplied, the system default is used. Language IDs are defined in KVLanguageID in kvtypes.h.</p> <p>The default is Language_UNKNOWN.</p>
bUseDocumentColors	<p>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 tags of the output.</p> <p>The default is FALSE.</p>
bUseDocumentFontInfo	<p>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 tags in the output.</p> <p>The default is FALSE.</p>
bNbspEmptyCells	<p>Set bNbspEmptyCells to TRUE to include a non-breaking space (<td>&nbsp;</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.</p> <p>This option applies to word processing documents and spreadsheets only.</p> <p>The default is TRUE.</p>
eSatableBorder	<p>This option 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 210.</p>

	<p>This option applies to word processing documents only.</p> <p>The default is <code>SA_BaseOnDocument</code>.</p>
<code>nTableBorderWidth</code>	<p>This option sets the width of the table border in pixels.</p> <p>This option applies to word processing documents only.</p> <p>The default is 1.</p>
<code>pszBaseURL</code>	<p>The base URL that replaces the <code>\$BASE</code> token in the XML output.</p> <p>The default is <code>NULL</code>.</p>
<code>pszMainURL</code>	<p>The main URL that replaces the <code>\$MAIN</code> token in the XML output.</p> <p>The default is <code>NULL</code>.</p>
<code>pszDefaultOutputDirectory</code>	<p>The default output directory for auxiliary files created during the conversion.</p> <p>The default is <code>NULL</code>, and the files are placed in the directory in which your application is running.</p>
<code>pszPicPath</code>	<p>The output directory for graphic files created during the conversion. If specified, this member can also be used by the callback functions <code>KVXMLGetAnchor</code> and <code>KVXMLGetAuxOutput</code>.</p> <p>This option applies to word processing documents only.</p> <p>The default is <code>NULL</code>, and the files are placed in the directory in which your application is running.</p>
<code>pszPicURL</code>	<p>The URL of the graphic files created from embedded graphics in the source document. To specify a complete image source, this element must be combined with <code>pszAnchor</code> of the <code>fpGetAnchor</code> callback function. See GetAnchor(), on page 175.</p> <p>For example, setting <code>pszPicURL</code> to <code>../cgi-bin/</code> and setting <code>pszAnchor</code> to <code>pic.jpg</code> results in the following markup:</p> <pre><a xmlns:xlink= xlink href="../cgi-bin/pic.jpg"></pre> <p>This option applies to word processing documents only.</p> <p>The default is <code>NULL</code>.</p>
<code>pszJavaURL</code>	<p>The URL where the Java rasterizer (<code>kvvector.jar</code>) is located.</p> <p>The Java rasterizer is not currently enabled.</p> <p>The default is <code>NULL</code>.</p>
<code>bRemoveFileNameSpaces</code>	<p>Set <code>bRemoveFileNameSpaces</code> to <code>TRUE</code> to remove spaces from generated output file names.</p> <p>The default is <code>FALSE</code>.</p>
<code>bRasterizeFiles</code>	<p>Set <code>bRasterizeFiles</code> to <code>TRUE</code> to rasterize slides from presentations into single images. Set <code>bRasterizeFiles</code> to <code>FALSE</code> to only extract text from presentation files. When you set this member to <code>FALSE</code>,</p>

graphics do not appear in the 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 91](#).

<code>eOutputRasterGraphicType</code>	<p>The output format of rasterized embedded graphics. There are six options enumerated in <code>KVXMLGraphicType</code> in <code>kvxml.h</code>. See KVXMLGraphicType, on page 218.</p> <p>The default is <code>KVGFX_JPEG</code>.</p>
<code>eOutputVectorGraphicType</code>	<p>The output format of vector graphics. The options are enumerated in <code>KVXMLGraphicType</code> in <code>kvxml.h</code>. The default is <code>JPEG</code>. See KVXMLGraphicType, on page 218. For more information on converting vector graphics on UNIX or Linux, see Display Vector Graphics on UNIX and Linux, on page 79.</p> <p>The default is <code>KVGFX_JPEG</code>.</p>
<code>cxVectorToRasterXRes</code>	<p>Specifies the horizontal resolution when converting presentation files and vector graphics. This is set in conjunction with <code>cyVectorToRasterYRes</code>. For more information, see Set the Resolution of Presentations and Vector Graphics, on page 203.</p> <p>The default value is 0, which means the original resolution is retained.</p>
<code>cyVectorToRasterYRes</code>	<p>Specifies the vertical resolution when converting presentation files and vector graphics. This is set in conjunction with <code>cxVectorToRasterXRes</code>. For more information, see Set the Resolution of Presentations and Vector Graphics, on page 203.</p> <p>The default value is 0, which means the original resolution is retained.</p>
<code>nCompressionQuality</code>	<p>This option 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).</p> <p>This option applies to word processing documents only.</p> <p>The default is 0.</p>
<code>bGenerateURLs</code>	<p>Set <code>bGenerateURLs</code> to TRUE to add anchor tags (<code><a xmlns:xlink=xlink href=> </code>) to text starting with "www", "http:" or "file:".</p> <p>This option applies to word processing documents only.</p>

	<p>The default is FALSE.</p>
<code>lcbMaxMemUsage</code>	<p>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.</p> <p>This option applies to word processing or text documents only.</p> <p>The default is LONG_MAX. The unit is in bytes.</p>
<code>cReplaceChar</code>	<p>The character used when a character in the source document's character set cannot be mapped to the output character set.</p> <p>The default replacement character is a question mark (?).</p>
<code>cRedact</code>	<p>The character that replaces tagged text that has been designated, through style mapping, to be omitted from the output. This functionality is useful when you need to hide confidential or sensitive information.</p> <p>The specified character is used for all text that has been mapped to a style processed with the KVSTYLE_REDACT flag (defined in <code>kvtypes.h</code>). See Map Styles, on page 75.</p> <p>This option applies to word processing documents only.</p> <p>The default replacement character is "X".</p>
<code>eEmptyParaType</code>	<p>This option determines if paragraphs without content generate markup or ID attributes in the output file. There are three options enumerated in <code>KVXMLEmptyParaType</code> in <code>kvxml.h</code>. See KVXMLEmptyParaType, on page 220.</p> <p>This option applies to word processing documents only.</p> <p>The default is KVEPT_SUPPRESS.</p>
<code>eHardPageBreakType</code>	<p>This option determines if hard page breaks generate markup or ID attributes in the output file. There are four options enumerated in <code>KVXMLEmptyParaType</code> in <code>kvxml.h</code>. See KVXMLHardPageBreakType, on page 221.</p> <p>This option applies to word processing documents only.</p> <p>The default is KVHPBT_SUPPRESS.</p>
<code>bSupportColumnHeadings</code>	<p>Set <code>bSupportColumnHeadings</code> to TRUE to include column headings from the source spreadsheet in the output.</p> <p>This option applies to spreadsheets only.</p> <p>The default is FALSE.</p>
<code>bSupportRowHeadings</code>	<p>Set <code>bSupportRowHeadings</code> to TRUE to include row headings from the source spreadsheet in the output.</p> <p>This option applies to spreadsheets only.</p>

	<p>The default is FALSE.</p>
<code>bSupportCellSpan</code>	<p>Set <code>bSupportCellSpan</code> to <code>TRUE</code> to include <code>colspan="n"</code> markup in the output.</p> <p>This option applies to spreadsheets only.</p> <p>The default value is FALSE.</p>
<code>bSupportRowSpan</code>	<p>Set <code>bSupportRowSpan</code> to <code>TRUE</code> to include row span data from the source spreadsheet in the output.</p> <p>This option applies to spreadsheets only.</p> <p>The default value is FALSE. Currently not supported.</p>
<code>bSupportColumnWidth</code>	<p>Set <code>bSupportColumnWidth</code> to <code>TRUE</code> to include column width data from the source spreadsheet in the output.</p> <p>This option applies to spreadsheets only.</p> <p>The default value is FALSE.</p>
<code>bRemoveEmptyColumns</code>	<p>Set <code>bRemoveEmptyColumns</code> to <code>TRUE</code> to remove spreadsheet columns that do not contain data and to disable cell merging.</p> <p>This option applies to spreadsheets only.</p> <p>The default is FALSE.</p>
<code>bRemoveEmptyRows</code>	<p>Set <code>bRemoveEmptyRows</code> to <code>TRUE</code> to remove spreadsheet rows that do not contain data or color, and to disable cell merging.</p> <p>This option applies to spreadsheets only.</p> <p>The default is FALSE.</p>
<code>bEnableEmptyRows</code>	<p>Set <code>bEnableEmptyRows</code> to <code>TRUE</code> to display empty rows in a spreadsheet format. If you set <code>bEnableEmptyRows</code> to <code>FALSE</code>, empty rows are not displayed. This applies only to 20 or more consecutive empty rows.</p> <p>This option applies to spreadsheets only.</p> <p>The default is FALSE.</p>
<code>nRowsBeforeSplit</code>	<p>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.</p> <p>This option applies to spreadsheets only.</p> <p>The default is 0.</p>

Discussion

A pointer to this structure is passed as an argument to `fpConvertStream()` and `KVXMLConvertFile()`. 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.

Set the Resolution of Presentations and Vector Graphics

The members `cxVectorToRasterXRes` and `cyVectorToRasterYRes` 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 XML 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
```

KVXMLTemplate

This structure defines the overall framework of the XML output. Members in this structure define the XML markup written at specific points in the output stream. The pointers contain XML markup that might include embedded `KeyView`-defined tokens. The XML markup contained in these strings should be well-formed. For the generated document to be valid, the markup must conform to the Verity DTD. The structure is initialized by calling the `fpConvertStream()` or `KVXMLConvertFile()` function. See [fpConvertStream\(\)](#), on page 139 or [KVXMLConvertFile\(\)](#), on page 164.

```
typedef struct tag_KVXMLTemplate
{
    char    *pszMainTop;
    char    *pszMainBottom;
    char    *pszFirstH1Start;
    char    *pszFirstH1End;
    char    *pszMiddleH1Start;
    char    *pszMiddleH1End;
    char    *pszLastH1Start;
    char    *pszLastH1End;
    char    *pszH[2..6]XML;
    char    *pszTOCH[1..6]Start;
    char    *pszTOC_H[1..6];
    char    *pszTOCH[1..6]End;
    char    *pszXFile;
    char    *pszXStartBlock;
    char    *pszXEndBlock;
    char    *pszStartBlock;
    char    *pszEndBlock;
    BOOL    bPutBlocksInSeparateFiles;
    BOOL    bHardPageMakesNewBlock
    long    lcbBlockSize;
    char    *pszChunkTemplate;
    char    *pszUserSummary;
    char    *pszTOCH[1..6]LeafNode;
}
KVXMLTemplate;
```

Member Descriptions

<code>pszMainTop</code>	The markup and tokens inserted at the beginning of the main XML file. Most of the sample template files feature <MetaData> tags with tokens that store the metadata of the input document. This member does not include the processing instructions or document type declarations that appears at the beginning of an XML document. The document type declaration <?xml version= ...> is automatically generated by XML Export. If you are using style sheets or the Verity DTD, the <?xml stylesheet= ...> and <!DOCTYPE ...> processing instructions are also automatically generated by XML Export.
<code>pszMainBottom</code>	The markup and tokens inserted at the end of the main XML file.
<code>pszFirstH1Start</code>	The markup and tokens inserted at the beginning of the first created H1 XML block (that is, the block associated with the first H1 table of contents entry).
<code>pszFirstH1End</code>	The markup and tokens inserted at the end of the first created H1 XML block (that is, the block associated with the first H1 table of

	contents entry).
pszMiddleH1Start	The markup and tokens inserted at the beginning of those H1 XML 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).
pszMiddleH1End	The markup and tokens inserted at the end of those H1 XML 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).
pszLastH1Start	The markup and tokens inserted at the beginning of the last created H1 XML block (that is, the block associated with the last H1 table of contents entry).
pszLastH1End	The markup and tokens inserted at the end of the last created H1 XML block (that is, the block associated with the last H1 table of contents entry).
pszH[2..6]XML	The markup and tokens inserted in an XML block for heading levels 2 through 6.
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: <code><ol list-style-type="upper-roman"></code>
pszTOC_H[1..6]	The markup and tokens required to process the table of contents entries for heading levels 1 through 6. For example: <code><a xmlns:xlink="http://www.w3.org/TR/xlink" xlink href="#\$ANCHOR"> \$TOC_E</code> If the table of contents heading contains special characters, such as an ampersand (&) or parentheses, you must use the \$TOCPE token in the pszTOC_H[1..6] markup. This token retains character entities and prevents validity errors. See Export Tokens, on page 347 for more information on table of contents tokens.
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: <code></code>
pszXFile	The markup and tokens generated and placed in an extra XML file. This file holds content from the source document. To process this file, you would use the \$XANCHOR token. See Export Tokens, on page 347 for more information on Export tokens.
pszXStartBlock	The markup and tokens inserted at the beginning of each XML 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 347 for more information on Export tokens.
pszXEndBlock	The markup and tokens to include in the output at the end of

	each XML block generated by the <code>\$XANCHOR</code> token. If either this member or <code>pszXStartBlock</code> is defined, both <code>pszStartBlock</code> and <code>pszEndBlock</code> are ignored. See Export Tokens, on page 347 for more information on Export tokens.
<code>pszStartBlock</code>	The markup and tokens inserted at the beginning of each block created as a result of <code>lcbBlockSize</code> or <code>bHardPageMakesNewBlock</code> .
<code>pszEndBlock</code>	The markup and tokens inserted at the end of each block created as a result of <code>lcbBlockSize</code> or <code>bHardPageMakesNewBlock</code> .
<code>bPutBlocksInSeparateFiles</code>	Set <code>bPutBlocksInSeparateFiles</code> to <code>TRUE</code> to create a separate XML file for each heading level 1 block. Each new block uses the markup defined in <code>pszStartBlock</code> and <code>pszEndBlock</code> . If you set <code>bPutBlocksInSeparateFiles</code> to <code>FALSE</code> , each heading level 1 block is placed sequentially in the same file, after the initial markup is written.
<code>bHardPageMakesNewBlock</code>	Set <code>bHardPageMakesNewBlock</code> to <code>TRUE</code> to have hard page breaks in the source document generate new XML files during the conversion process. The member <code>pszchunktemplate</code> provides the appropriate table of contents entry for the new block. This option applies to word processing documents and spreadsheets only.
<code>lcbBlockSize</code>	The maximum size (in bytes) of heading level 1 XML 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. Setting <code>lcbBlockSize</code> to 0 indicates that there is no maximum size.
<code>pszChunkTemplate</code>	If an H1 XML block is subdivided into separate files as a result of the size limitations specified in <code>lcbBlockSize</code> , this member provides a template for creating a table of contents entry for the new file. The block number can be made a part of this template by inserting the <code>\$SPLITBLOCKNUMBER</code> token. For example: Page <code>\$SPLITBLOCKNUMBER</code>
<code>pszUserSummary</code>	The markup and tokens generated when the <code>\$USERSUMMARY</code> or <code>\$SUMMARY</code> tokens are used. For example: <MetaData name=" <code>\$NAME</code> " content=" <code>\$CONTENT</code> "/>
<code>pszTOCH[1..6]LeafNode</code>	The markup that replaces <code>pszTOC_H[1..6]</code> entries for leaf nodes in the table of contents. A leaf node is a node that has no children.

Discussion

A pointer to this structure is passed as an argument to `fpConvertStream()` and `KVXMLConvertFile()`. 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.

KVXMLTOCOptions

This structure defines whether a heading is included in the table of contents. Source text is converted to a heading in the XML output if

- it meets **all** the criteria defined by the members of `KVXMLHeadingInfo`, and
- the `headingCreateType` member of `KVXMLTOCOptions` is set to allow automatic heading generation.

The structure is initialized by calling the `fpConvertStream()` or `KVXMLConvertFile()` function. See [fpConvertStream\(\), on page 139](#) or [KVXMLConvertFile\(\), on page 164](#).

See [KVXMLOptions, on page 195](#) for more information on the criteria used to determine whether a heading is included in the table of contents.

```
typedef struct tag_KVXMLTOCOptions
{
    BOOL                bAllowHeadingsInTables;
    KVHeadingCreateOptions headingCreateType;
    KVXMLHeadingInfo    *pH1;
    KVXMLHeadingInfo    *pH2;
    KVXMLHeadingInfo    *pH3;
    KVXMLHeadingInfo    *pH4;
    KVXMLHeadingInfo    *pH5;
    KVXMLHeadingInfo    *pH6;
}
KVXMLTOCOptions;
```

Member Descriptions

<code>bAllowHeadingsInTables</code>	<p>This option determines whether the text in tables is considered for automatic heading generation. If you set <code>bAllowHeadingsInTables</code> to <code>TRUE</code>, the text in tables is included in the determination of headings and table of contents entries.</p> <p>This option applies to word processing documents and spreadsheets only.</p> <p>The default is <code>FALSE</code>.</p>
<code>headingCreateType</code>	<p>This option determines how XML Export subdivides the source document into table of contents entries. You can set this option to one of the two options enumerated in <code>KVHeadingCreateOptions</code> in <code>kvxml.h</code>. See KVHeadingCreateOptions, on page 219.</p> <p>The determination of table of contents entries is based on whether the source document contains <i>heading styles</i> or whether <i>text attributes</i> conform to the criteria defined in the <code>KVXMLHeadingInfo</code> structure. See KVXMLHeadingInfo, on page 188.</p>

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`.

`KVXMLHeadingInfo`

A pointer to the `KVXMLHeadingInfo` structure. See [KVXMLHeadingInfo](#), on page 188.

When the table of contents entries are not based on the heading styles of the source document, 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 `KVXMLHeadingInfo`.

Discussion

A pointer to this structure is passed as an argument to `fpConvertStream()` and `KVXMLConvertFile()`. 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.

Chapter 11: Enumerated Types

This section provides information on some of the enumerated types used by the XML Export API.

- [Introduction](#) 209
- [ENSATableBorder](#) 210
- [KVCredKeyType](#) 211
- [KVErrorCode](#) 211
- [KVErrorCodeEx](#) 213
- [KVXMLStyleSheetType](#) 216
- [KVXMLAnchorType](#) 217
- [KVXMLGraphicType](#) 218
- [KVHeadingCreateOptions](#) 219
- [KVXMLEmptyParaType](#) 220
- [KVXMLHardPageBreakType](#) 221
- [KVMetadataType](#) 221
- [KVMetaNameType](#) 223
- [KVSumInfoType](#) 223
- [KVSumType](#) 224
- [LPDF_DIRECTION](#) 228

Introduction

The enumerated types are in `adinfo.h`, `kvtypes.h`, `kvxml.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 122. This enumerated type is defined in `kvextract.h`.

Definition

```
typedef enum tag_KVCredKeyType
{
    KVCredKeyType_UserName,
    KVCredKeyType_UserIdFile,
    KVCredKeyType_Password,
}
KVCredKeyType;
```

Enumerators

KVCredKeyType_UserName	The credential in KVCredentialComponent is a user name.
KVCredKeyType_UserIdFile	The credential in KVCredentialComponent is a path to a file that contains user IDs.
KVCredKeyType_Password	The credential in KVCredentialComponent is a password.

KVErrorCode

This enumerated type defines the type of error generated if 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 <code>fpContinue()</code> returns <code>FALSE</code> .
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 error creating a child process.
KVERR_WaitForChildFailed	The wait for child process failed.
KVERR_ChildTimeOut	The child process hung or timed out.
KVERR_ArchiveFileNotFound	Attempt to extract nonexistent file.
KVERR_ArchiveFatalError	A fatal error occurred processing an archive file.

KVErrorCodeEx

This enumerated type defines extended error codes. The type is defined in `kvtypes.h`.

Definition

```
typedef enum tag_KVErrorCodeEx
{
    KVErrror_OpenStreamFailure = KVERR_ArchiveFatalError + 1, /* 22 KVOpen stream
failure */
    KVErrror_InterfaceFunctionNotFound, /* 23 Interface function not found */
    KVErrror_InputFileNotFound, /* 24 Cannot find input file*/
    KVErrror_OpenOutputFileFailed, /* 25 Cannot open output file*/
    KVErrror_MemoryLeak, /* 26 Memory leak*/
    KVErrror_MemoryOverwrite, /* 27 Memory overwrite*/
    KVErrror_GPF, /* 28 Exception during oop filtering*/
    KVErrror_OopCore, /* 29 Core dump in child process*/
    KVErrror_KVoopLogFailed, /* 30 Creation of oop error log failed*/
    KVErrror_OverNestedFileLimit, /* 31 File exceeds nested file limit*/
    KVErrror_PSTAccessFailed, /* 32 Access failed on PST files*/
    KVErrror_PasswordRequired, /* 33 Password required to access file*/
    KVErrror_InvalidArgs /* 34 Input argument/structure is invalid*/
    KVErrror_ReaderUsageDenied, /* 35 Reader requires a valid license*/
    KVErrror_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 */
}
KVErrCodeEx;
```

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 <code>KVERR_General</code> code. This enumerator is used by KeyView Filter.
<code>KVError_KVoopLogFailed</code>	The creation of the out-of-process error log failed. This is an extended error for the <code>KVERR_General</code> code. This enumerator is used by KeyView Filter.
<code>KVError_OverNestedFileLimit</code>	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.
<code>KVError_PSTAccessFailed</code>	<p>The PST file could not be converted. This error might be returned when a call to <code>fpOpenFile()</code> returns <code>NULL</code> for one of the following reasons:</p> <ul style="list-style-type: none"> • A Microsoft Outlook client is not installed. • A Microsoft Outlook client is installed, but is not the default email client. • A Microsoft Outlook client is installed, but is not configured correctly. • The PST file is corrupt. • The PST file is read-only (PST files must allow read and write access). • The MAPI call fails. • The bit editions of Microsoft Outlook do not match the bit editions of the KeyView software. <p>For example, if 32-bit KeyView is used, 32-bit Outlook must be installed. If 64-bit KeyView is used, 64-bit Outlook must be installed.</p>
<code>KVError_PasswordRequired</code>	To open the file, you must provide credentials. This error might be returned when a call to <code>fpOpenFile()</code> returns <code>NULL</code> .
<code>KVError_InvalidArgs</code>	The input argument or structure is invalid. This error is generated by the File Extraction APIs.
<code>KVError_ReaderUsageDenied</code>	<p>The current license key does not enable the document reader required to convert the file. This error might be returned when a call to <code>fpOpenFile()</code> returns <code>NULL</code>.</p> <p>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.</p>
<code>KVError_OopBadConfig</code>	Information in the <code>kvxconfig.ini</code> 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 <code>fpClose()</code> , 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 210](#).
- If an extended error code is called for a format to which the error does not apply, the `KVError_Last` code is returned.

KVXMLStyleSheetType

This enumerated type defines the options for processing style sheet information. This enumerated type is defined in `kvxml.h`.

Definition

```
typedef enum tag_KVXMLStyleSheetType{    STYLESHEET_DISABLED = 0,
    XML_CSS,
    XML_XSL,
}
KVXMLStyleSheetType;
```

Enumerators

STYLESHEET_ DISABLED Disables Cascading Style Sheet (CSS) formatting.

XML_CSS Enables Cascading Style Sheet (CSS) formatting and generates an external file or uses an existing external file which is referenced in a `<?xml-stylesheet...?>` processing instruction.

XML_XSL Enables Extensible Style Sheet Language (XSL) formatting and uses an external XSL file which is referenced in a `<?xml-stylesheet...?>` processing instruction.

KVXMLAnchorType

This enumerated type defines the anchor types for the output stream. This enumerated type is defined in `kvxml.h`.

Definition

```
typedef enum tag_KVXMLAnchorType
{
    VectorPictureAnchor = 0,
    RasterPictureAnchor,
    H1Anchor,
    H2Anchor,
    H3Anchor,
    H4Anchor,
    H5Anchor,
    H6Anchor,
    XAnchor,
```

```
    AnimatedGIFAnchor,  
    CSSAnchor,  
    XSLAnchor,  
    GeneralAnchor,  
    DBAnchor,  
    JPEGAnchor  
}  
KVXMLAnchorType;
```

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.
XSLAnchor	An anchor for an external XSL file.
GeneralAnchor	Reserved for future use.
DBAnchor	Used internally.
JPEGAnchor	An anchor for an embedded JPEG graphic.

KVXMLGraphicType

This enumerated type defines graphic formats to which embedded graphics and presentations are converted. This enumerated type is defined in `kvxml.h`.

Definition

```
typedef enum tag_KVXMLGraphicType
{
    KVGFX_GIF,
    KVGFX_JPEG,
    KVGFX_PNG,
    KVGFX_CGM,
    KVGFX_WMF,
    KVGFX_JAVA
}
KVXMLGraphicType;
```

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.

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 KVXMLHeadingInfo). This enumerated type is defined in kvxml.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.

KVXMLEmptyParaType

This enumerated type defines the options for paragraphs that do not contain content. This enumerated type is defined in `kvxml.h`.

Definition

```
typedef enum tag_KVXMLEmptyParaType
{
    KVEPT_SUPPRESS,    /* No markup generated      */
    KVEPT_EMPTY,       /* Use <p/>                  */
    KVEPT_VERBOSE      /* Use <p id="..."&nbsp;</p> */
}
KVXMLEmptyParaType;
```

Enumerators

KVEPT_SUPPRESS	paragraphs without content are ignored. They do not contribute white space and do not affect the ID number of subsequent paragraphs. This is the default value.
KVEPT_EMPTY	paragraphs without content are represented by an "empty" paragraph element <code><p/></code> . These contribute minimal white space, but do not affect the ID number of subsequent paragraphs.
KVEPT_VERBOSE	paragraphs without content contain a fully-defined start tag <code><p id="..."></code> with all non-default attributes, a <code>&nbsp;</code> character entity, and end tag <code></p></code> . These contribute additional white space and affect the ID number of subsequent paragraphs.

KVXMLHardPageBreakType

This enumerated type defines the options for hard page breaks. This enumerated type is defined in `kvxml.h`.

Definition

```
typedef enum tag_KVXMLHardPageBreakType
{
    KVHPBT_SUPPRESS, /* No markup generated          */
    KVHPBT_EMPTY,    /* Use <Page/>                      */
    KVHPBT_EMPTYID,  /* Use <Page id="n"/>                */
    KVHPBT_ID        /* Use <Page id="n"> ... </Page> */
}
KVXMLHardPageBreakType;
```

Enumerators

KVHPBT_SUPPRESS	No markup is generated for hard page breaks. This is the default value.
KVHPBT_EMPTY	An empty page element, <code><Page/></code> , without ID attributes is generated for hard page breaks.
KVHPBT_EMPTYID	An empty page element, <code><Page id="n"/></code> , with ID attributes is generated for hard page breaks. The ID is incremented for each subsequent hard page break.
KVHPBT_ID	A "non-empty" "Page" element is generated for hard page breaks. The page tags enclose the contents immediately after the <code><WP></code> tag. When subsequent hard page breaks are encountered, the previous "Page" element is closed with a <code></Page></code> tag, and a <code><Page id="..."></code> opening tag is added. The final "Page" element is closed immediately before the closing <code></WP></code> tag.

KVMetadataType

This enumerated type defines the data type of metadata that can be extracted from a subfile in a mail message or mail store. If a metadata field has a corresponding KeyView type in `KVMetadataType`, the metadata is converted to the [KVMetadataElem](#) structure, and the structure member `isValid` is 1. This enumerated type is defined in `kvtypes.h`.

Definition

```
typedef enum
{
    KVMetadata_Unknown = 0,
```

```
KVMetadata_Bool          = 1,  
KVMetadata_Binary        = 2,  
KVMetadata_Int4          = 3,  
KVMetadata_UInt4         = 4,  
KVMetadata_Int8          = 5,  
KVMetadata_UInt8         = 6,  
KVMetadata_String        = 7,  
KVMetadata_Unicode       = 8,  
KVMetadata_DateTime      = 9,  
KVMetadata_Float         = 10,  
KVMetadata_Double        = 11,  
KVMetadata_Last  
}  
KVMetadataType;
```

Enumerators

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 210](#).

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 129](#). This enumerated type is defined in `kvextract.h`.

Definition

```
typedef enum
{
    KVMetaNameType_Integer = 0,
    KVMetaNameType_String  = 1
}
KVMetaNameType;
```

Enumerators

`KVMetaNameType_Integer` The metadata field is an integer.

`KVMetaNameType_String` The metadata field is a string.

KVSumInfoType

This enumerated type defines the data type of the metadata field extracted from a document. This enumerated type is defined in `kvtypes.h`.

Definition

```
typedef enum tag_KVSumInfoType
{
    KV_String      = 0x1,
    KV_Int4        = 0x2,
    KV_DateTime    = 0x3,
    KV_ClipBoard   = 0x4,
    KV_Bool        = 0x5,
    KV_Unicode     = 0x6,
    KV_IEEE8       = 0x7,
    KV_Other       = 0x8
}
KVSumInfoType;
```

Enumerators

KV_String	The value in the metadata field is a string.
KV_Int4	The value in the metadata field is an integer.
KV_DateTime	The value in the metadata field is a date and time. This type is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (Windows FILETIME EPOCH). You might need to convert this value into another format.
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 <i>Draft</i> , <i>Reviewed</i> , or <i>Final</i> .
KV_UserDefined	The contents of the first entry in the array of non-standard metadata. This could be user-defined metadata, or metadata unique to a file type.

LPDF_DIRECTION

This enumerated type defines the paragraph direction of extracted paragraphs from a PDF file when logical order is enabled. This enumerated type is defined in `kvtypes.h`.

Definition

```
typedef enum{  
    LPDF_RAW = 0,  
    LPDF_LTR,  
    LPDF_RTL,  
    LPDF_AUTO  
} LPDF_DIRECTION ;
```

Enumerators

`LPDF_RAW` Unstructured paragraph flow. This is the default behavior.

`LPDF_LTR` Logical reading order and left-to-right paragraph direction.

`LPDF_RTL` Logical reading order and right-to-left paragraph direction.

`LPDF_AUTO` Logical reading order. The PDF reader determines the paragraph direction for each PDF page, and then sets the direction accordingly. This is the default when logical order is enabled.

Part IV: 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](#)230

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 259](#).
- 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
Expert Witness Compression Format (EnCase)	6	encasesr	E01, L01	N	N	Y	Y	N	n/a	N
	7	encase2sr	Lx01	N	N	Y	Y	N	n/a	N
GZIP	2	kvgzsr	GZ	N	N	N	Y	N	n/a	N
		kvgz	GZ	N	N	Y	N	N	n/a	N
ISO	n/a	isosr	ISO	N	N	Y	Y	N	n/a	N
Java Archive	n/a	unzip	JAR	N	N	Y	Y	N	n/a	N
Legato EMailXtender Archive	n/a	emxsr	EMX	N	N	Y	Y	N	n/a	N

¹7zip is supported with the multiarcsr reader on some platforms for Extract.

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
MacBinary	n/a	macbinsr	BIN	N	N	Y	Y	N	n/a	N
Mac Disk Copy Disk Image	n/a	dmgsr	DMG	N	N	Y	Y	N	n/a	N
Microsoft Backup File	n/a	bkfsr	BKF	N	N	Y	Y	N	n/a	N
Microsoft Cabinet format	1.3	cabsr	CAB	N	N	Y	Y	N	n/a	N
Microsoft Compiled HTML Help	3	chmsr	CHM	N	N	Y	Y	N	n/a	N
Microsoft Compressed Folder	n/a	lzhsr	LZH LHA	N	N	N	Y	N	n/a	N
Microsoft Power BI Desktop format	n/a	unzip	PBIX	N	N	N	Y	N	n/a	N
PKZIP	through 9.0	unzip	ZIP	N	N	Y	Y	N	n/a	N
RAR archive	2.0 through 3.5	rarsr	RAR	N	N	N	Y	N	n/a	N
RAR5 archive	5	multiarcsr	RAR5	N	N	N	Y	N	n/a	N
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

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
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 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
Zipped Keyhole Markup Language	n/a	unzip	ZIP	N	N	N	Y	N	n/a	N

Binary Format**Supported Binary Formats**

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Executable	n/a	exesr	EXE	N	N	Y	N	N	n/a	N
Link Library	n/a	exesr	DLL	N	N	Y	N	N	n/a	N

Computer-Aided Design Formats

Supported CAD Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
AutoCAD Drawing	R13, R14, R15/2000, 2004, 2007, 2010, 2013, 2018	kpODArdr kpDWGrdr ¹	DWG	Y	Y	Y	N	Y	Y	N
AutoCAD Drawing Exchange	R13, R14, R15/2000, 2004, 2007, 2010, 2013	kpODArdr kpDXFrdr ²	DXF	Y	Y	Y	N	Y	Y	N
CATIA formats	5	kpCATrdr	CAT ³	Y	N	N	N	Y	N	N
Microsoft Visio	4, 5, 2000, 2002, 2003, 2007, 2010 ⁴	vsdsr	VSD	Y	Y	Y	Y ⁵	Y	Y	N
		kpVSD2rdr	VSD, VSS VST	Y	Y	Y	N	Y	Y	N

¹The kpODArdr reader can filter, export, and view all versions but is supported only on Windows, Linux, and 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.

²The 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.

³All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.

⁴Viewing and Export use the graphic reader, kpVSD2rdr for Microsoft Visio 2003, 2007, and 2010, and vsdsr for all earlier versions. Image fidelity in Viewing and Export is therefore only supported for versions 2003 and above. Filter uses the graphic reader kpVSD2rdr for Microsoft Visio 2003, 2007, and 2010, and vsdsr for all earlier versions.

⁵Extraction of embedded OLE objects is supported for Filter on Windows platforms only.

Supported CAD Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
	2013	ActiveX components	VSDM VSSM VSTM VSDX VSSX VSTX	N	N	Y ¹	N	Y	N	N
		kpVSDXrdr	VSDM VSSM VSTM VSDX VSSX VSTX	Y	Y	Y	Y	Y	Y	N
Unigraphics (UG) NX		kpUGrdr	PRT	Y	N	N	N	N	N	N

Database Formats**Supported Database Formats**

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
dBase Database	III+, IV	dbfsr	DBF	Y	Y	Y	N	N	N	N

¹Visio 2013 is supported in Viewing only, with the support of ActiveX components from the Microsoft Visio 2013 Viewer. Image fidelity is supported but other features, such as highlighting, are not.

Supported Database Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Access	95, 97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	mdbsr	MDB, ACCDB	Y	T	T	N	N	Y ¹	N
Microsoft Project	2000, 2002, 2003, 2007, 2010, 2013, 2016	mppsr	MPP	Y	Y	Y	Y	Y	Y	N

Desktop Publishing

Supported Desktop Publishing Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Publisher	98 to 2016	mspubsr	PUB	Y	T	T	Y	Y	Y	N

Display Formats

Supported Display Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe PDF	1.1 to 1.7	pdfsr	PDF	Y	Y	N	Y ²	Y	Y	N
		pdf2sr	PDF	N	Y	N	N	N	N	N

¹Charset is not supported for Microsoft Access 95 or 97.²Includes support for extraction of subfiles from PDF Portfolio documents.

Supported Display Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
		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	kpcdrdr	CDR	N	Y	Y	N	N	N	N
DCX Fax System	n/a	kpcxdr	DCX	N	Y	Y	N	N	N	N
Digital Imaging & Communications in	n/a	dcmsr	DCM	M	N	N	N	Y	N	N

¹kppdf2rdr is an alternate graphic-based reader that produces high-fidelity output but does not support other features such as highlighting or text searching.

²Files with non-partitioned data are supported.

³CDR/CDR with TIFF header.

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Medicine (DICOM)										
Encapsulated PostScript (raster)	TIFF header	kpepsrdr	EPS	N	Y	Y	N	N	N	N
Enhanced Metafile	n/a	kpemfrdr	EMF	Y	Y	Y	N	Y	N	N
GIF	87, 89	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	kpjpgdrdr	JPEG	N	Y	Y	N	N	N	N
		jpgsr		M	M	N	N	Y	N	N
JPEG 2000	n/a	kpjp2000rdr	JP2, JPF, J2K, JPWL, JPX, PGX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
JPEG 2000 PGX Verification Model image	n/a	kjpg2000rdr	PGX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
Lotus AMIDraw Graphics	n/a	kpsdwrdr	SDW	N	Y	Y	N	N	N	N
Lotus Pic	n/a	kppicrdr	PIC	Y	Y	Y	N	N	N	N
Macintosh Raster	2	kppctrdr	PIC PCT	N	Y	Y	N	N	N	N
MacPaint	n/a	kpmacrdr	PNTG	N	Y	Y	N	N	N	N
Microsoft Office Drawing	n/a	kpmsohdr	MSO	N	Y	Y	N	N	N	N
Omni Graffiti	n/a	kpGFLrdr	GRAFFLE	Y	N	N	N	Y	Y	N
PC PaintBrush	3	kppcxrdr	PCX	N	Y	Y	N	N	N	N
Portable Network Graphics	n/a	kppngrdr	PNG	N	Y	Y	N	N	N	N
		pngsr	PNG	M	M	N	N	Y	N	N
Scalable Vector Graphics	n/a	xmlsr	SVG	Y	T	T	N	Y	Y	N
SGI RGB Image	n/a	kpsgirdr	RGB	N	Y	Y	N	N	N	N
Sun Raster Image	n/a	kpsunrdr	RS	N	Y	Y	N	N	N	N

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Tagged Image File	through 6.0 ¹	tifsr	TIFF	M	M	N	N	Y	N	N
		kptifdr	TIFF	N	Y	Y	N	N	N	N
Truevision Targa	2	kpTGArdr	TGA	N	Y	Y	N	N	N	N
Windows Animated Cursor	n/a	kpanirdr	ANI	N	Y	Y	N	N	N	N
Windows Bitmap	n/a	kpbmprdr	BMP	N	Y	Y	N	N	N	N
		bmpsr	BMP	M	M	N	N	Y	N	N
Windows Icon Cursor	n/a	kpicordr	ICO	N	Y	Y	N	N	N	N
Windows Metafile	3	kpwmfrdr	WMF	Y ²	Y	Y	N	N	N	N
WordPerfect Graphics 1	1	kpwpgrdr	WPG	N	Y	Y	N	N	N	N
WordPerfect Graphics 2	2, 7	kpwg2rdr	WPG	N	Y	Y	N	N	N	N

¹The following compression types are supported: no compression, CCITT Group 3 1-Dimensional Modified Huffman, CCITT Group 3 T4 1-Dimensional, CCITT Group 4 T6, LZW, JPEG (only Gray, RGB and CMYK color space are supported), and PackBits.

²Windows Metafiles can contain both raster images (KeyView file class 4) and vector graphics (KeyView file class 5). Filtering is supported only for vector graphics (class 5).

Mail Formats

Supported Mail Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Documentum EMCMF	n/a	msgsr	EMCMF	N	N	Y	Y	Y	Y	N
Domino XML Language ¹	n/a	dxlsr	DXL	N	N	Y	Y	Y	N	N
GroupWise FileSurf	n/a	gwfssr	GWFS	N	N	Y	Y	Y	N	N
Legato Extender	n/a	onmsr	ONM	N	N	Y	Y	Y	N	N
Lotus Notes database	4, 5, 6.0, 6.5, 7.0, 8.0	nsfsr	NSF	N	N	Y	Y	Y	N	N
Mailbox ²	Thunderbird 1.0, Eudora 6.2	mbxsr ³	MBX	N	N	T	Y	Y	Y	N
Microsoft Entourage	2004	entsr	various	N	N	Y	Y	Y	Y	N

¹Supports non-encrypted embedded files only.

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

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

Supported Mail Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Database										
Microsoft Outlook	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	msgsr ¹	MSG, OFT	Y	T	T	Y	Y	Y ²	N
Microsoft Outlook DBX	5.0, 6.0	dbxsr	DBX	N	N	Y	Y	Y	Y	N
Microsoft Outlook Express	Windows 6 Macintosh 5	emlsr ³	EML	Y	T	T	Y	Y	Y	N
		mbxsr ⁴	EML	N	N	T	Y	Y	Y	N
Microsoft Outlook iCalendar	1.0, 2.0	icssr	ICS, VCS	N	N	Y	Y	Y	Y	N
Microsoft Outlook for Macintosh	2011	olmsr	OLM	N	N	Y	Y	N	Y	N
Microsoft Outlook Offline Storage File	97, 2000, 2002, 2003, 2007, 2010, 2013	pffsr ⁵	OST	N	N	Y	Y	Y	Y	N

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

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

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

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

⁵The reader pffsr is available only on Windows and Linux.

Supported Mail Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Outlook Personal Folder	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	pstsr ¹²	PST	N	N	Y	Y	Y	N	N
	97, 2000, 2002, 2003, 2007, 2010, 2013	pstnsr	PST	N	N	Y	Y	Y	Y	N
Microsoft Outlook vCard Contact	2.1, 3.0, 4.0	vcfsr	VCF	Y	Y	T	N	Y	N	N
Text Mail (MIME)	n/a	emlsr ³	various	Y	T	T	Y	Y	Y	N
		mbxsr ⁴	various	Y	T	T	Y	Y	Y	N
Transport Neutral Encapsulation Format	n/a	tnfsr	various	N	N	Y	Y	Y	Y	N

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

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

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

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

Multimedia Formats

Viewing SDK plays some multimedia files using the Windows Media Control Interface (MCI). MCI is a set of Windows APIs that communicate with multimedia devices.

Supported Multimedia Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
3GPP video file	n/a	mpeg4sr	3GP	M	N	N	N	Y	N	N
3GPP2 video file	n/a	mpeg4sr	3G2	M	N	N	N	Y	N	N
Adobe Flash Player audio	n/a	mpeg4sr	F4A	M	N	N	N	Y	N	N
Adobe Flash Player audio book	n/a	mpeg4sr	F4B	M	N	N	N	Y	N	N
Adobe Flash Player protected video	n/a	mpeg4sr	F4P	M	N	N	N	Y	N	N
Adobe Flash Player video	n/a	mpeg4sr	F4V	M	N	N	N	Y	N	N
Apple ISO-BMFF QuickTime video	n/a	MCI	QT MOV	N	N	Y	N	N	N	N
Apple MPEG-4 Part 14 audio	n/a	mpeg4sr	M4A	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14 audio book	n/a	mpeg4sr	M4B	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14 protected audio	n/a	mpeg4sr	M4P	M	N	N	N	Y	N	N
Apple MPEG-4 Part 14	n/a	mpeg4sr	M4V	M	N	N	N	Y	N	N

Supported Multimedia Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
video										
Audible Enhanced Audiobook	n/a	mpeg4sr	AAX	M	N	N	N	Y	N	N
KDDI video file	n/a	MCI		N	N	Y	N	N	N	N
Advanced Systems Format	1.2	asfsr	ASF WMA WMV	N	N	N	N	Y	N	N
Audio Interchange File Format	n/a	MCI	AIFF	N	N	Y	N	N	N	N
		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
		riffr	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	kplWPGGrdr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16 iCloud 2018	kplWPG13rdr 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

¹This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Macromedia Flash	through 8.0	swfsr	SWF	Y	Y	Y	N	N	Y ¹	N
Microsoft OneNote	2007, 2010, 2013, 2016	kpONErdr	ONE ONETOC2	Y	Y	Y	Y	N	Y	N
Microsoft PowerPoint Macintosh	98	kpp40rdr	PPT	Y	Y	Y	N	N	N	N
	2001, v.X, 2004	kpp97rdr	PPT PPS POT	Y	Y	Y	N	P	Y	N
Microsoft PowerPoint PC	4	kpp40rdr	PPT	Y	Y	Y	N	P	N	N
Microsoft PowerPoint Windows	95	kpp95rdr	PPT	Y	Y	Y	N	P	Y	N
Microsoft PowerPoint Windows	97, 2000, 2002, 2003	kpp97rdr	PPT PPS POT	Y	Y	Y	Y	P	Y	Y ²
Microsoft PowerPoint Windows XML	2007, 2010, 2013, 2016	kpppxrdr	PPTX PPTM POTX POTM PPSX PPSM	Y	Y	Y	Y	Y	Y	Y

¹The character set cannot be determined for versions 5.x and lower.²Slide footers are supported for Microsoft PowerPoint 97 and 2003.

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
			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

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

²Supported 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 iCloud 2018	iwss13sr 1	NUMBERS	Y	T	T	N	N	Y	N
Applix Spreadsheets	4.2, 4.3, 4.4	assr	AS	Y	Y	Y	N	N	Y	N
Comma Separated Values	n/a	csvsr	CSV	Y	Y	Y	N	N	N	N
Corel Quattro Pro	5, 6, 7, 8	qpssr	WB2 WB3	Y	Y	Y	N	P	Y	N
	X4	qpwsr	QPW	Y	N	Y	N	P	Y	N
Data Interchange Format	n/a	difsr		Y	Y	Y	N	N	N	N
Lotus 1-2-3	96, 97, R9, 9.8	l123sr	123	Y	Y	Y	N	P	Y	N
Lotus 1-2-3	2, 3, 4, 5	wkssr	WK4	Y	Y	Y	N	N	Y	N
Lotus 1-2-3 Charts	2, 3, 4, 5	kpchtrdr	123	N	Y	Y	N	N	N	N
Microsoft Excel Charts	2, 3, 4, 5, 6, 7	kpchtrdr	XLS	N	Y	Y	N	N	N	N

¹This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Excel Macintosh	98, 2001, v.X, 2004	xlssr	XLS	Y	Y	Y	Y ¹	Y	Y	N
Microsoft Excel Windows	2.2 through 2003	xlssr	XLS XLW XLT XLA	Y	Y	Y	Y ²	Y	Y	Y
Microsoft Excel Windows XML	2007, 2010, 2013, 2016	xlxsxr	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

¹Supported using the embedded objects reader `olesr`.²Supported for versions 97 and higher using the embedded objects reader `olesr`.³Generated by OpenOffice Calc 2.0, StarOffice 8 Calc, and IBM Lotus Symphony Spreadsheet 3.0.⁴Supported 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 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

¹This reader is available only on Windows (32-bit and 64-bit), Linux (32-bit and 64-bit), and Solaris x86-64.²This reader is only supported on Windows 32-bit platforms.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
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

¹Supported 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	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

¹Supported 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

¹Generated by OpenOffice Writer 2.0, StarOffice 8 Writer, and IBM Lotus Symphony Documents 3.0.²Supported 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

¹To 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.

- [Detected Formats](#)260

Detected Formats

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

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

Format Name	Number	Format Description	File Extension
AES_Multiplus_Comm_Fmt	1	Multiplus (AES)	PTF
ASCII_Text_Fmt	2	Text	
MSDOS_Batch_File_Fmt	3	MS-DOS Batch File	BAT
Applix_Alis_Fmt	4	APPLIX ASTERIX	AX
BMP_Fmt	5	Windows Bitmap Image (BMP)	BMP
CT_DEF_Fmt	6	Convergent Technologies DEF Comm. Format	
Corel_Draw_Fmt	7	Corel Draw	CDR
CGM_ClearText_Fmt	8	Computer Graphics Metafile (CGM)	CGM
CGM_Binary_Fmt	9	Computer Graphics Metafile (CGM)	CGM
CGM_Character_Fmt	10	Computer Graphics Metafile (CGM)	CGM
Word_Connection_Fmt	11	Word Connection	CN
COMET_TOP_Word_Fmt	12	COMET TOP	
CEOwrite_Fmt	13	CEOwrite	CW
DSA101_Fmt	14	DSA101 (Honeywell Bull)	

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

Format Name	Number	Format Description	File Extension
IBM_Display_Write_Fmt	34	Display Write	IP
Sun_Raster_Fmt	35	Sun Raster	RAS
Ami_Pro_Fmt	36	Lotus Ami Pro	SAM
Ami_Pro_StyleSheet_Fmt	37	Lotus Ami Pro Style Sheet	
MORE_Fmt	38	MORE Database MAC	
Lyrix_Fmt	39	Lyrix Word Processing	
MASS_11_Fmt	40	MASS-11	M1
MacPaint_Fmt	41	MacPaint	PNTG
MS_Word_Mac_Fmt	42	Microsoft Word for Macintosh	DOC
SmartWare_II_Comm_Fmt	43	SmartWare II	
MS_Word_Win_Fmt	44	Microsoft Word for Windows	DOC, WPS
Multimate_Fmt	45	MultiMate	
Multimate_Fnote_Fmt	46	MultiMate Footnote File	
Multimate_Adv_Fmt	47	MultiMate Advantage	
Multimate_Adv_Fnote_Fmt	48	MultiMate Advantage Footnote File	
Multimate_Adv_II_Fmt	49	MultiMate Advantage II	
Multimate_Adv_II_Fnote_Fmt	50	MultiMate Advantage II Footnote File	
Multiplan_PC_Fmt	51	Multiplan (PC)	
Multiplan_Mac_Fmt	52	Multiplan (Mac)	

Format Name	Number	Format Description	File Extension
MS_RTF_Fmt	53	Rich Text Format (RTF)	RTF
MS_Word_PC_Fmt	54	Microsoft Word for PC	MW
MS_Word_PC_StyleSheet_Fmt	55	Microsoft Word for PC Style Sheet	
MS_Word_PC_Glossary_Fmt	56	Microsoft Word for PC Glossary	
MS_Word_PC_Driver_Fmt	57	Microsoft Word for PC Driver	
MS_Word_PC_Misc_Fmt	58	Microsoft Word for PC Miscellaneous File	
NBI_Async_Archive_Fmt	59	NBI Async Archive Format	
Navy_DIF_Fmt	60	Navy DIF	ND
NBI_Net_Archive_Fmt	61	NBI Net Archive Format	NN
NIOS_TOP_Fmt	62	NIOS TOP	
FileMaker_Mac_Fmt	63	Filemaker MAC	FP5, FP7
ODA_Q1_11_Fmt	64	ODA / ODIF	
ODA_Q1_12_Fmt	65	ODA / ODIF	
OLIDIF_Fmt	66	OLIDIF (Olivetti)	
Office_Writer_Fmt	67	Office Writer	OW
PC_Paintbrush_Fmt	68	PC Paintbrush Graphics (PCX)	PCX
CPT_Comm_Fmt	69	CPT	
Lotus_PIC_Fmt	70	Lotus PIC	PIC
Mac_PICT_Fmt	71	QuickDraw Picture	PCT

Format Name	Number	Format Description	File Extension
Philips_Script_Word_Fmt	72	Philips Script	
PostScript_Fmt	73	PostScript	PS
PRIMEWORD_Fmt	74	PRIMEWORD	
Quadratron_Q_One_v1_Fmt	75	Q-One V1.93J	
Quadratron_Q_One_v2_Fmt	76	Q-One V2.0	
SAMNA_Word_IV_Fmt	77	SAMNA Word	SAM
Ami_Pro_Draw_Fmt	78	Lotus Ami Pro Draw	SDW
SYLK_Spreadsheet_Fmt	79	SYLK	
SmartWare_II_WP_Fmt	80	SmartWare II	
Symphony_Fmt	81	Symphony	WR1
Targa_Fmt	82	Targa	TGA
TIFF_Fmt	83	Tag Image File Format (TIFF)	TIF, TIFF
Targon_Word_Fmt	84	Targon Word	TW
Uniplex_Ucalc_Fmt	85	Uniplex Ucalc	SS
Uniplex_WP_Fmt	86	Uniplex	UP
MS_Word_UNIX_Fmt	87	Microsoft Word UNIX	
WANG_PC_Fmt	88	WANG PC	
WordERA_Fmt	89	WordERA	
WANG_WPS_Comm_Fmt	90	WANG WPS	WF

Format Name	Number	Format Description	File Extension
WordPerfect_Mac_Fmt	91	WordPerfect MAC	
WordPerfect_Fmt	92	WordPerfect	WP, WP4, WPD, WOP
WordPerfect_VAX_Fmt	93	WordPerfect VAX	
WordPerfect_Macro_Fmt	94	WordPerfect Macro	
WordPerfect_Dictionary_Fmt	95	WordPerfect Spelling Dictionary	
WordPerfect_Thesaurus_Fmt	96	WordPerfect Thesaurus	
WordPerfect_Resource_Fmt	97	WordPerfect Resource File	
WordPerfect_Driver_Fmt	98	WordPerfect Driver	
WordPerfect_Cfg_Fmt	99	WordPerfect Configuration File	
WordPerfect_Hyphenation_Fmt	100	WordPerfect Hyphenation Dictionary	
WordPerfect_Misc_Fmt	101	WordPerfect Miscellaneous File	
WordMARC_Fmt	102	WordMARC	WM, PW
Windows_Metafile_Fmt	103	Windows Metafile	WMF
Windows_Metafile_NoHdr_Fmt	104	Windows Metafile (no header)	WMF
SmartWare_II_DB_Fmt	105	SmartWare II	
WordPerfect_Graphics_Fmt	106	WordPerfect Graphics	WPG, QPG
WordStar_Fmt	107	WordStar	WS, WSD
WANG_WITA_Fmt	108	WANG WITA	WT
Xerox_860_Comm_Fmt	109	Xerox 860	

Format Name	Number	Format Description	File Extension
Xerox_Writer_Fmt	110	Xerox Writer	
DIF_SpreadSheet_Fmt	111	Data Interchange Format (DIF)	DIF
Enable_Spreadsheet_Fmt	112	Enable Spreadsheet	SSF
SuperCalc_Fmt	113	Supercalc	CAL
UltraCalc_Fmt	114	UltraCalc	
SmartWare_II_SS_Fmt	115	SmartWare II	
SOF_Encapsulation_Fmt	116	Serialized Object Format (SOF)	SOF
PowerPoint_Win_Fmt	117	Microsoft PowerPoint PC	
PowerPoint_Mac_Fmt	118	Microsoft PowerPoint MAC	
PowerPoint_95_Fmt	119	Microsoft PowerPoint 95	
PowerPoint_97_Fmt	120	Microsoft PowerPoint 97	
PageMaker_Mac_Fmt	121	PageMaker for Macintosh	
PageMaker_Win_Fmt	122	PageMaker for Windows	
MS_Works_Mac_WP_Fmt	123	Microsoft Works Word Processor for MAC	MWK
MS_Works_Mac_DB_Fmt	124	Microsoft Works Database for MAC	
MS_Works_Mac_SS_Fmt	125	Microsoft Works Spreadsheet for MAC	
MS_Works_Mac_Comm_Fmt	126	Microsoft Works Communication for MAC	
MS_Works_DOS_WP_Fmt	127	Microsoft Works Word Processor for DOS	WPS
MS_Works_DOS_DB_Fmt	128	Microsoft Works Database for DOS	WDB

Format Name	Number	Format Description	File Extension
MS_Works_DOS_SS_Fmt	129	Microsoft Works Spreadsheet for DOS	
MS_Works_Win_WP_Fmt	130	Microsoft Works Word Processor for Windows	WPS, W40
MS_Works_Win_DB_Fmt	131	Microsoft Works Database for Windows	
MS_Works_Win_SS_Fmt	132	Microsoft Works Spreadsheet for Windows	S30, S40
PC_Library_Fmt	133	DOS/Windows Object Library	LIB, A
MacWrite_Fmt	134	MacWrite	
MacWrite_II_Fmt	135	MacWrite II	
Freehand_Fmt	136	Freehand MAC	
Disk_Doubler_Fmt	137	Disk Doubler	
HP_GL_Fmt	138	HP Graphics Language	HPGL
FrameMaker_Fmt	139	FrameMaker	FM, FRM
FrameMaker_Book_Fmt	140	FrameMaker	BOOK
Maker_Markup_Language_Fmt	141	Maker Markup Language	
Maker_Interchange_Fmt	142	Maker Interchange Format (MIF)	MIF
JPEG_File_Interchange_Fmt	143	JPEG Interchange Format	JPG, JPEG
Reflex_Fmt	144	Reflex	
Framework_Fmt	145	Framework	
Framework_II_Fmt	146	Framework II	FW3
Paradox_Fmt	147	Paradox	DB

Format Name	Number	Format Description	File Extension
MS_Windows_Write_Fmt	148	Microsoft Windows Write	WRI
Quattro_Pro_DOS_Fmt	149	Quattro Pro for DOS	WQ1
Quattro_Pro_Win_Fmt	150	Quattro Pro for Windows	WB1, WB2, WB3
Persuasion_Fmt	151	Persuasion	
Windows_Icon_Fmt	152	Windows Icon Format	ICO
Windows_Cursor_Fmt	153	Windows Cursor	CUR
MS_Project_Activity_Fmt	154	Microsoft Project	
MS_Project_Resource_Fmt	155	Microsoft Project	
MS_Project_Calc_Fmt	156	Microsoft Project	
PKZIP_Fmt	157	ZIP Archive	ZIP
Quark_Xpress_Fmt	158	Quark Xpress MAC	
ARC_PAK_Archive_Fmt	159	PAK/ARC Archive	ARC, PAK
MS_Publisher_Fmt	160	Microsoft Publisher	PUB
PlanPerfect_Fmt	161	PlanPerfect	
WordPerfect_Auxiliary_Fmt	162	WordPerfect auxiliary file	WPW
MS_WAVE_Audio_Fmt	163	Microsoft Wave	WAV
MIDI_Audio_Fmt	164	MIDI	MID, MIDI
AutoCAD_DXF_Binary_Fmt	165	AutoCAD DXF	DXF
AutoCAD_DXF_Text_Fmt	166	AutoCAD DXF	DXF

Format Name	Number	Format Description	File Extension
dBase_Fmt	167	dBase	DBF, VCX
OS_2_PM_Metafile_Fmt	168	OS/2 PM Metafile	MET
Lasergraphics_Language_Fmt	169	Lasergraphics Language	
AutoShade_Rendering_Fmt	170	AutoShade Rendering	
GEM_VDI_Fmt	171	GEM VDI	VDI
Windows_Help_Fmt	172	Windows Help File	HLP
Volkswriter_Fmt	173	Volkswriter	VW4
Ability_WP_Fmt	174	Ability	
Ability_DB_Fmt	175	Ability	
Ability_SS_Fmt	176	Ability	
Ability_Comm_Fmt	177	Ability	
Ability_Image_Fmt	178	Ability	
XyWrite_Fmt	179	XYWrite / Nota Bene	XY4
CSV_Fmt	180	CSV (Comma Separated Values)	CSV
IBM_Writing_Assistant_Fmt	181	IBM Writing Assistant	IWA
WordStar_2000_Fmt	182	WordStar 2000	WS2
HP_PCL_Fmt	183	HP Printer Control Language	PCL
UNIX_Exe_PreSysV_VAX_Fmt	184	Unix Executable (PDP-11/pre-System V VAX)	
UNIX_Exe_Basic_16_Fmt	185	Unix Executable (Basic-16)	

Format Name	Number	Format Description	File Extension
UNIX_Exe_x86_Fmt	186	Unix Executable (x86)	
UNIX_Exe_iAPX_286_Fmt	187	Unix Executable (iAPX 286)	
UNIX_Exe_MC68k_Fmt	188	Unix Executable (MC680x0)	
UNIX_Exe_3B20_Fmt	189	Unix Executable (3B20)	
UNIX_Exe_WE32000_Fmt	190	Unix Executable (WE32000)	
UNIX_Exe_VAX_Fmt	191	Unix Executable (VAX)	
UNIX_Exe_Bell_5_Fmt	192	Unix Executable (Bell 5.0)	
UNIX_Obj_VAX_Demand_Fmt	193	Unix Object Module (VAX Demand)	
UNIX_Obj_MS8086_Fmt	194	Unix Object Module (old MS 8086)	
UNIX_Obj_Z8000_Fmt	195	Unix Object Module (Z8000)	
AU_Audio_Fmt	196	NeXT/Sun Audio Data	AU
NeWS_Font_Fmt	197	NeWS bitmap font	
cpio_Archive_CRChdr_Fmt	198	cpio archive (CRC Header)	
cpio_Archive_CHRhdr_Fmt	199	cpio archive (CHR Header)	
PEX_Binary_Archive_Fmt	200	SUN PEX Binary Archive	
Sun_vfont_Fmt	201	SUN vfont Definition	
Curses_Screen_Fmt	202	Curses Screen Image	
UUEncoded_Fmt	203	UU encoded	UUE
WriteNow_Fmt	204	WriteNow MAC	

Format Name	Number	Format Description	File Extension
PC_Obj_Fmt	205	DOS/Windows Object Module	OBJ
Windows_Group_Fmt	206	Windows Group	
TrueType_Font_Fmt	207	TrueType Font	TTF
Windows_PIF_Fmt	208	Program Information File (PIF)	PIF
MS_COM_Executable_Fmt	209	PC (.COM)	COM
StuffIt_Fmt	210	StuffIt (MAC)	HQX
PeachCalc_Fmt	211	PeachCalc	CAL
Wang_GDL_Fmt	212	WANG Office GDL Header	
Q_A_DOS_Fmt	213	Q & A for DOS	
Q_A_Win_Fmt	214	Q & A for Windows	JW
WPS_PLUS_Fmt	215	WPS-PLUS	WPL
DCX_Fmt	216	DCX FAX Format(PCX images)	DCX
OLE_Fmt	217	OLE Compound Document	OLE
EBCDIC_Fmt	218	EBCDIC Text	
DCS_Fmt	219	DCS	
UNIX_SHAR_Fmt	220	SHAR	SHAR
Lotus_Notes_BitMap_Fmt	221	Lotus Notes Bitmap	
Lotus_Notes_CDF_Fmt	222	Lotus Notes CDF	CDF
Compress_Fmt	223	Unix Compress	Z

Format Name	Number	Format Description	File Extension
GZ_Compress_Fmt	224	GZ Compress	GZ
TAR_Fmt	225	TAR	TAR
ODIF_FOD26_Fmt	226	ODA / ODIF	F26
ODIF_FOD36_Fmt	227	ODA / ODIF	F36
ALIS_Fmt	228	ALIS	
Envoy_Fmt	229	Envoy	EVY
PDF_Fmt	230	Portable Document Format	PDF
BinHex_Fmt	231	BinHex	HQX
SMTP_Fmt	232	SMTP	SMTP
MIME_Fmt	233	MIME ¹	EML, MBX
USENET_Fmt	234	USENET	
SGML_Fmt	235	SGML	SGML
HTML_Fmt	236	HTML	HTM, HTML
ACT_Fmt	237	ACT	ACT
PNG_Fmt	238	Portable Network Graphics (PNG)	PNG
MS_Video_Fmt	239	Video for Windows (AVI)	AVI
Windows_Animated_Cursor_Fmt	240	Windows Animated Cursor	ANI
Windows_CPP_Obj_Storage_Fmt	241	Windows C++ Object Storage	
Windows_Palette_Fmt	242	Windows Palette	PAL

Format Name	Number	Format Description	File Extension
RIFF_DIB_Fmt	243	RIFF Device Independent Bitmap	
RIFF_MIDI_Fmt	244	RIFF MIDI	RMI
RIFF_Multimedia_Movie_Fmt	245	RIFF Multimedia Movie	
MPEG_Fmt	246	MPEG Movie	
QuickTime_Fmt	247	QuickTime Movie	MOV, QT, MP4
AIFF_Fmt	248	Audio Interchange File Format (AIFF)	AIF, AIFF
Amiga_MOD_Fmt	249	Amiga MOD	MOD
Amiga_IFF_8SVX_Fmt	250	Amiga IFF (8SVX) Sound	IFF
Creative_Voice_Audio_Fmt	251	Creative Voice (VOC)	VOC
AutoDesk_Animator_FLI_Fmt	252	AutoDesk Animator FLIC	FLI
AutoDesk_AnimatorPro_FLC_Fmt	253	AutoDesk Animator Pro FLIC	FLC
Compactor_Archive_Fmt	254	Compactor / Compact Pro	
VRML_Fmt	255	VRML	WRL
QuickDraw_3D_Metafile_Fmt	256	QuickDraw 3D Metafile	
PGP_Secret_Keyring_Fmt	257	PGP Secret Keyring	
PGP_Public_Keyring_Fmt	258	PGP Public Keyring	
PGP_Encrypted_Data_Fmt	259	PGP Encrypted Data	
PGP_Signed_Data_Fmt	260	PGP Signed Data	
PGP_SignedEncrypted_Data_Fmt	261	PGP Signed and Encrypted Data	

Format Name	Number	Format Description	File Extension
PGP_Sign_Certificate_Fmt	262	PGP Signature Certificate	SIG
PGP_Compressed_Data_Fmt	263	PGP Compressed Data	
PGP_ASCII_Public_Keyring_Fmt	264	ASCII-armored PGP Public Keyring	PGP
PGP_ASCII_Encoded_Fmt	265	ASCII-armored PGP encoded	
PGP_ASCII_Signed_Fmt	266	ASCII-armored PGP signed	
OLE_DIB_Fmt	267	OLE DIB object	
SGL_Image_Fmt	268	SGL Image	RGB
Lotus_ScreenCam_Fmt	269	Lotus ScreenCam	SCM
MPEG_Audio_Fmt	270	MPEG Audio	MPEGA, MPG, MP3
FTP_Software_Session_Fmt	271	FTP Session Data	STE
Netscape_Bookmark_File_Fmt	272	Netscape Bookmark File	
Corel_Draw_CMX_Fmt	273	Corel CMX	CMX
AutoDesk_DWG_Fmt	274	AutoDesk Drawing (DWG)	DWG
AutoDesk_WHIP_Fmt	275	AutoDesk WHIP	WHP
Macromedia_Director_Fmt	276	Macromedia Director	DCR
Real_Audio_Fmt	277	Real Audio	RM, RA
MSDOS_Device_Driver_Fmt	278	MSDOS Device Driver	SYS
Micrografx_Designer_Fmt	279	Micrografx Designer	DSF
SVF_Fmt	280	Simple Vector Format (SVF)	SVF

Format Name	Number	Format Description	File Extension
Applix_Words_Fmt	281	Applix Words	AW
Applix_Graphics_Fmt	282	Applix Graphics	AG
MS_Access_Fmt	283	Microsoft Access	MDB
MS_Access_95_Fmt	284	Microsoft Access 95	MDB
MS_Access_97_Fmt	285	Microsoft Access 97	MDB
MacBinary_Fmt	286	MacBinary	BIN
Apple_Single_Fmt	287	Apple Single	
Apple_Double_Fmt	288	Apple Double	AD
Enhanced_Metafile_Fmt	289	Enhanced Metafile	EMF
MS_Office_Drawing_Fmt	290	Microsoft Office Drawing	
XML_Fmt	291	XML	XML
DeVice_Independent_Fmt	292	DeVice Independent file (DVI)	DVI
Unicode_Fmt	293	Unicode	UNI
Lotus_123_Worksheet_Fmt	294	Lotus 1-2-3	WKS, WK1, WK3, WK4
Lotus_123_Format_Fmt	295	Lotus 1-2-3 Formatting	FM3
Lotus_123_97_Fmt	296	Lotus 1-2-3 97	123
Lotus_Word_Pro_96_Fmt	297	Lotus Word Pro 96	
Lotus_Word_Pro_97_Fmt	298	Lotus Word Pro 97	LWP, MWP
Freelance_DOS_Fmt	299	Lotus Freelance for DOS	

Format Name	Number	Format Description	File Extension
Freelance_Win_Fmt	300	Lotus Freelance for Windows	PRE
Freelance_OS2_Fmt	301	Lotus Freelance for OS/2	PRS
Freelance_96_Fmt	302	Lotus Freelance 96	PRZ
Freelance_97_Fmt	303	Lotus Freelance 97	
MS_Word_95_Fmt	304	Microsoft Word 95	DOC
MS_Word_97_Fmt	305	Microsoft Word 97	DOC, WPS
Excel_Fmt	306	Microsoft Excel	XLS
Excel_Chart_Fmt	307	Microsoft Excel	XLC
Excel_Macro_Fmt	308	Microsoft Excel	XLM
Excel_95_Fmt	309	Microsoft Excel 95	
Excel_97_Fmt	310	Microsoft Excel 97	
Corel_Presentations_Fmt	311	Corel Presentations	XFD, XFDL
Harvard_Graphics_Fmt	312	Harvard Graphics	PR4
Harvard_Graphics_Chart_Fmt	313	Harvard Graphics Chart	CH3, CHT
Harvard_Graphics_Symbol_Fmt	314	Harvard Graphics Symbol File	SY3
Harvard_Graphics_Cfg_Fmt	315	Harvard Graphics Configuration File	
Harvard_Graphics_Palette_Fmt	316	Harvard Graphics Palette	
Lotus_123_R9_Fmt	317	Lotus 1-2-3 Release 9	123
Applix_Spreadsheets_Fmt	318	Applix Spreadsheets	AS

Format Name	Number	Format Description	File Extension
MS_Pocket_Word_Fmt	319	Microsoft Pocket Word	PWD
MS_DIB_Fmt	320	Microsoft Device Independent Bitmap	DIB
MS_Word_2000_Fmt	321	Microsoft Word 2000	DOC
Excel_2000_Fmt	322	Microsoft Excel 2000	XLS
PowerPoint_2000_Fmt	323	Microsoft PowerPoint 2000	PPT
MS_Access_2000_Fmt	324	Microsoft Access 2000	MDB
MS_Project_4_Fmt	325	Microsoft Project 4	
MS_Project_41_Fmt	326	Microsoft Project 4.1	MPP
MS_Project_98_Fmt	327	Microsoft Project 98	MPP
Folio_Flat_Fmt	328	Folio Flat File	FFF
HWP_Fmt	329	HWP(Arae-Ah Hangul)	HWP
ICHITARO_Fmt	330	ICHITARO	JTD
IS_XML_Fmt	331	Extended or Custom XML	XML
Oasys_Fmt	332	Oasys	OAS, OA2, OA3
PBM_ASC_Fmt	333	Portable Bitmap Utilities ASCII format (PBM)	PBM
PBM_BIN_Fmt	334	Portable Bitmap Utilities BINARY format (PBM)	PBM
PGM_ASC_Fmt	335	Portable Greymap Utilities ASCII format (PGM)	PGM
PGM_BIN_Fmt	336	Portable Greymap Utilities BINARY format (PGM)	PGM
PPM_ASC_Fmt	337	Portable Pixmap Utilities ASCII format (PPM)	PPM

Format Name	Number	Format Description	File Extension
PPM_BIN_Fmt	338	Portable Pixmap Utilities BINARY format (PPM)	PPM
XBM_Fmt	339	X Bitmap format (XBM)	XBM
XPM_Fmt	340	X Pixmap format (XPM)	XPM
FPX_Fmt	341	FlashPix FPX Image format	FPX
PCD_Fmt	342	PCD Image format	PCD
MS_Visio_Fmt	343	Microsoft Visio	VSD
MS_Project_2000_Fmt	344	Microsoft Project 2000	MPP
MS_Outlook_Fmt	345	Microsoft Outlook	MSG, OFT
ELF_Relocatable_Fmt	346	ELF Relocatable	O
ELF_Executable_Fmt	347	ELF Executable	
ELF_Dynamic_Lib_Fmt	348	ELF Dynamic Library	SO
MS_Word_XML_Fmt	349	Microsoft Word 2003 XML	XML
MS_Excel_XML_Fmt	350	Microsoft Excel 2003 XML	XML
MS_Visio_XML_Fmt	351	Microsoft Visio 2003 XML	VDX
SO_Text_XML_Fmt	352	StarOffice Text XML	SXW
SO_Spreadsheet_XML_Fmt	353	StarOffice Spreadsheet XML	SXC, STC
SO_Presentation_XML_Fmt	354	StarOffice Presentation XML	SXD, SXI
XHTML_Fmt	355	XHTML	XML, ASP
MS_OutlookPST_Fmt	356	Microsoft Outlook Personal Folders File (.pst)	PST

Format Name	Number	Format Description	File Extension
RAR_Fmt	357	RAR	RAR
Lotus_Notes_NSF_Fmt	358	IBM Lotus Notes Database NSF/NTF	NSF
Macromedia_Flash_Fmt	359	Macromedia Flash (.swf)	SWF
MS_Word_2007_Fmt	360	Microsoft Word 2007 XML - Docx	DOCX, DOTX
MS_Excel_2007_Fmt	361	Microsoft Excel 2007 XML	XLSX, XLTX
MS_PPT_2007_Fmt	362	Microsoft PowerPoint 2007 XML	PPTX, POTX, PPSX
OpenPGP_Fmt	363	OpenPGP Message Format (with new packet format)	PGP
Intergraph_V7_DGN_Fmt	364	Intergraph Standard File Format (ISFF) V7 DGN (non-OLE)	
MicroStation_V8_DGN_Fmt	365	MicroStation V8 DGN (OLE)	DGN
MS_Word_Macro_2007_Fmt	366	Microsoft Word Macro 2007 XML	DOCM, DOTM
MS_Excel_Macro_2007_Fmt	367	Microsoft Excel Macro 2007 XML	XLSM, XLTM, XLAM
MS_PPT_Macro_2007_Fmt	368	Microsoft PPT Macro 2007 XML	PPTM, POTM, PPSM, PPAM
LZH_Fmt	369	LZH Archive	LZH, LHA
Office_2007_Fmt	370	Office 2007 document	XLSB
MS_XPS_Fmt	371	Microsoft XML Paper Specification (XPS)	XPS
Lotus_Domino_DXL_Fmt	372	IBM Domino Data in XML format (.dxl)	DXL
ODF_Text_Fmt	373	ODF Text	ODT
ODF_Spreadsheet_Fmt	374	ODF Spreadsheet	ODS

Format Name	Number	Format Description	File Extension
ODF_Presentation_Fmt	375	ODF Presentation	ODP
Legato_Extender_ONM_Fmt	376	Legato Extender Native Message ONM	ONM
bin_Unknown_Fmt	377	Bin unknown format (.xxx)	
TNEF_Fmt	378	Transport Neutral Encapsulation Format (TNEF)	various
CADAM_Drawing_Fmt	379	CADAM Drawing	CDD
CADAM_Drawing_Overlay_Fmt	380	CADAM Drawing Overlay	CDO
NURSTOR_Drawing_Fmt	381	NURSTOR Drawing	NUR
HP_GLP_Fmt	382	HP Graphics Language (Plotter)	HPG
ASF_Fmt	383	Advanced Systems Format (ASF)	ASF
WMA_Fmt	384	Windows Media Audio Format (WMA)	WMA
WMV_Fmt	385	Windows Media Video Format (WMV)	WMV
EMX_Fmt	386	Legato EMailXtender Archives Format (EMX)	EMX
Z7Z_Fmt	387	7 Zip Format (7z)	7Z
MS_Excel_Binary_2007_Fmt	388	Microsoft Excel Binary 2007	XLSB
CAB_Fmt	389	Microsoft Cabinet File (CAB)	CAB
CATIA_Fmt	390	CATIA Formats (CAT*)	CATPART, CATPRODUCT ²
YIM_Fmt	391	Yahoo Instant Messenger History	DAT
ODF_Drawing_Fmt	392	ODF Drawing	ODG

Format Name	Number	Format Description	File Extension
Founder_CEB_Fmt	393	Founder Chinese E-paper Basic (ceb)	CEB
QPW_Fmt	394	Corel Quattro Pro 9+ for Windows	QPW
MHT_Fmt	395	MHTML format (MHT) ¹	MHT, MHTML
MDI_Fmt	396	Microsoft Document Imaging Format	MDI
GRV_Fmt	397	Microsoft Office Groove Format	GRV
IWWP_Fmt	398	Apple iWork Pages format	PAGES
IWSS_Fmt	399	Apple iWork Numbers format	NUMBERS
IWPG_Fmt	400	Apple iWork Keynote format	KEY
BKF_Fmt	401	Windows Backup File	BKF
MS_Access_2007_Fmt	402	Microsoft Access 2007	ACCDB
ENT_Fmt	403	Microsoft Entourage Database Format	
DMG_Fmt	404	Mac Disk Copy Disk Image File	DMG
CWK_Fmt	405	AppleWorks File	CWK
OO3_Fmt	406	Omni Outliner V3 File	OO3
OPML_Fmt	407	Omni Outliner OPML File	OPML
Omni_Graffle_XML_Fmt	408	Omni Graffle XML File	GRAFFLE
PSD_Fmt	409	Photoshop Document	PSD, PSB
Apple_Binary_PList_Fmt	410	Apple Binary Property List format	PLIST
Apple_iChat_Fmt	411	Apple iChat format	ICHAT

Format Name	Number	Format Description	File Extension
OOUTLINE_Fmt	412	OOutliner File	OOUTLINE
BZIP2_Fmt	413	Bzip 2 Compressed File	BZ2
ISO_Fmt	414	ISO-9660 CD Disc Image Format	ISO
DocuWorks_Fmt	415	DocuWorks Format	XDW
RealMedia_Fmt	416	RealMedia Streaming Media	RM, RA
AC3Audio_Fmt	417	AC3 Audio File Format	AC3
NEF_Fmt	418	Nero Encrypted File	NEF
SolidWorks_Fmt	419	SolidWorks Format Files	SLDASM, SLDPRT, SLDDRW, SLDDRT
XFDL_Fmt	420	Extensible Forms Description Language	XFDL, XFD
Apple_XML_PList_Fmt	421	Apple XML Property List format	PLIST
OneNote_Fmt	422	OneNote Note Format	ONE
Dicom_Fmt	424	Digital Imaging and Communications in Medicine (Dicom)	DCM
EnCase_Fmt	425	Expert Witness Compression Format (EnCase)	E01, L01, Lx01
Scrap_Fmt	426	Shell Scrap Object File	SHS
MS_Project_2007_Fmt	427	Microsoft Project 2007	MPP
MS_Publisher_98_Fmt	428	Microsoft Publisher from version 98	PUB
Skype_Fmt	429	Skype Log File	DBB
HL7_Fmt	430	Health level7 message	HL7

Format Name	Number	Format Description	File Extension
MS_OutlookOST_Fmt	431	Microsoft Outlook Offline Folders File (OST)	OST
Epub_Fmt	432	Electronic Publication	EPUB
MS_OEDBX_Fmt	433	Microsoft Outlook Express DBX	DBX
BB_Activ_Fmt	434	BlackBerry Activation File	DAT
DiskImage_Fmt	435	Disk Image	DMG
Milestone_Fmt	436	Milestone Document	MLS, ML3, ML4, ML5, ML6, ML7, ML8, ML9, MLA
E_Transcript_Fmt	437	RealLegal E-Transcript File	PTX
PostScript_Font_Fmt	438	PostScript Type 1 Font	PFB
Ghost_DiskImage_Fmt	439	Ghost Disk Image File	GHO, GHS
JPEG_2000_JP2_File_Fmt	440	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	JP2, JPF, J2K, JPWL, JPX, PGX
Unicode_HTML_Fmt	441	Unicode HTML	HTM, HTML
CHM_Fmt	442	Microsoft Compiled HTML Help	CHM
EMCMF_Fmt	443	Documentum EMCMF format	EMCMF
MS_Access_2007_Tmpl_Fmt	444	Microsoft Access 2007 Template	ACCDT
Jungum_Fmt	445	Samsung Electronics Jungum Global document	GUL
JBIG2_Fmt	446	JBIG2 File Format	JB2, JBIG2
EFax_Fmt	447	eFax file	EFX

Format Name	Number	Format Description	File Extension
AD1_Fmt	448	AD1 Evidence file	AD1
SketchUp_Fmt	449	Google SketchUp	SKP
GWFS_Email_Fmt	450	Group Wise File Surf email	GWFS
JNT_Fmt	451	Windows Journal format	JNT
Yahoo_yChat_Fmt	452	Yahoo! Messenger chat log	YCHAT
PaperPort_MAX_File_Fmt	453	PaperPort MAX image file	MAX
ARJ_Fmt	454	ARJ (Archive by Robert Jung) file format	ARJ
RPMSG_Fmt	455	Microsoft Outlook Restricted Permission Message	RPMSG
MAT_Fmt	456	MATLAB file format	MAT, FIG
SGY_Fmt	457	SEG-Y Seismic Data format	SGY, SEG Y
CDXA_MPEG_PS_Fmt	458	MPEG-PS container with CDXA stream	MPG
EVT_Fmt	459	Microsoft Windows NT Event Log	EVT
EVTX_Fmt	460	Microsoft Windows Vista Event Log	EVTX
MS_OutlookOLM_Fmt	461	Microsoft Outlook for Macintosh format	OLM
WARC_Fmt	462	Web ARChive	WARC
JAVACLASS_Fmt	463	Java Class format	CLASS
VCF_Fmt	464	Microsoft Outlook vCard file format	VCF
EDB_Fmt	465	Microsoft Exchange Server Database file format	EDB
ICS_Fmt	466	Microsoft Outlook iCalendar file format	ICS, VCS

Format Name	Number	Format Description	File Extension
MS_Visio_2013_Fmt	467	Microsoft Visio 2013	VSDX, VSTX, VSSX
MS_Visio_2013_Macro_Fmt	468	Microsoft Visio 2013 macro	VSDM, VSTM, VSSM
ICHITARO_Compr_Fmt	469	ICHITARO Compressed format	JTDC
IWWP13_Fmt	470	Apple iWork 2013 Pages format	IWA
IWSS13_Fmt	471	Apple iWork 2013 Numbers format	IWA
IWPG13_Fmt	472	Apple iWork 2013 Keynote format	IWA
XZ_Fmt	473	XZ archive format	XZ
Sony_WAVE64_Fmt	474	Sony Wave64 format	W64
Conifer_WAVPACK_Fmt	475	Conifer Wavpack format	WV
Xiph_OGG_VORBIS_Fmt	476	Xiph Ogg Vorbis format	OGG
MS_Visio_2013_Stencil_Fmt	477	MS Visio 2013 stencil format	VSSX
MS_Visio_2013_Stencil_Macro_Fmt	478	MS Visio 2013 stencil Macro format	VSSM
MS_Visio_2013_Template_Fmt	479	MS Visio 2013 template format	VSTX
MS_Visio_2013_Template_Macro_Fmt	480	MS Visio 2013 template Macro format	VSTM
Borland_Reflex_2_Fmt	481	Borland Reflex 2 format	R2D
PKCS_12_Fmt	482	PKCS #12 (p12) format	P12, PFX
B1_Fmt	483	B1 format	B1
ISO_IEC_MPEG_4_Fmt	484	ISO/IEC MPEG-4 (ISO 14496) format	MP4
RAR5_Fmt	485	RAR5 Format	RAR

Format Name	Number	Format Description	File Extension
Unigraphics_NX_Fmt	486	Unigraphics (UG) NX CAD Format	PRT
PTC_Creo_Fmt	487	PTC Creo CAD Format	ASM, PRT
KML_Fmt	488	Keyhole Markup Language	KML
KMZ_Fmt	489	Zipped Keyhole Markup Language	KMZ
WML_Fmt	490	Wireless Markup Language	WML
ODF_Formula_Fmt	491	ODF Formula	ODF
SO_Text_Fmt	492	Star Office Writer Text	SDW, SGL, VOR
SO_Spreadsheet_Fmt	493	Star Office Calc Spreadsheet	SDC
SO_Presentation_Fmt	494	Star Office Impress Presentation	SDD, SDA
SO_Math_Fmt	495	Star Office Math	SMF
STEP_Fmt	496	ISO 10303-21 STEP format	
STL_Fmt	497	3D Systems STL ASCII format	
AppleScript_Fmt	498	AppleScript Source Code ³	APPLESCRIPT
Assembly_Fmt	499	Assembly Code ³	
C_Fmt	500	C Source Code ³	C, H
Csharp_Fmt	501	C# Source Code ³	CS
CPlusPlus_Fmt	502	C++ Source Code ³	CPP, HPP
Css_Fmt	503	Cascading Style Sheet ³	CSS
Clojure_Fmt	504	Clojure Source Code ³	CLJ, CL2

Format Name	Number	Format Description	File Extension
CoffeeScript_Fmt	505	CoffeeScript Source Code ³	COFFEE, CAKE
Lisp_Fmt	506	Common Lisp Source Code ³	EL
Dockerfile_Fmt	507	Dockerfile ³	
Eiffel_Fmt	508	Eiffel Source Code ³	E
Erlang_Fmt	509	Erlang Source Code ³	ERL, ES
Fsharp_Fmt	510	F# Source Code ³	FS
Fortran_Fmt	511	Fortran Source Code ³	F
Go_Fmt	512	Go Source Code ³	GO
Groovy_Fmt	513	Groovy Source Code ³	GRT, GVY
Haskell_Fmt	514	Haskell Source Code ³	HS
Ini_Fmt	515	Initialization (INI) file ³	
Java_Fmt	516	Java Source Code ³	JAVA
Javascript_Fmt	517	Javascript Source Code ³	JS
Lua_Fmt	518	Lua Source Code ³	LUA
Makefile_Fmt	519	Makefile ³	MAKE
Mathematica_Fmt	520	Wolfram Mathematica Source Code ³	M
ObjC_Fmt	521	Objective-C Source Code ³	
ObjCpp_Fmt	522	Objective-C++ Source Code ³	
ObjJ_Fmt	523	Objective-J Source Code ³	J

Format Name	Number	Format Description	File Extension
PHP_Fmt	524	PHP Source Code ³	PHP
PLSQL_Fmt	525	PLSQL Source Code ³	
Pascal_Fmt	526	Pascal Source Code ³	PASCAL
Perl_Fmt	527	Perl Source Code ³	PL
Powershell_Fmt	528	PowerShell Source Code ³	PS1
Prolog_Fmt	529	Prolog Source Code ³	PRO, PROLOG
Puppet_Fmt	530	Puppet Source Code ³	PP
Python_Fmt	531	Python Source Code ³	PY
R_Fmt	532	R Source Code ³	R
Ruby_Fmt	533	Ruby Source Code ³	RB
Rust_Fmt	534	Rust Source Code ³	RS
Scala_Fmt	535	Scala Source Code ³	SC
Shell_Fmt	536	Shell Script ³	SH
Smalltalk_Fmt	537	Smalltalk Source Code ³	ST
ML_Fmt	538	Standard ML Source Code ³	ML
Swift_Fmt	539	Swift Source Code ³	SWIFT
Tcl_Fmt	540	Tool Command Language (Tcl) Source Code ³	TM
Tex_Fmt	541	TeX Typesetting File ³	
TypeScript_Fmt	542	TypeScript Source Code ³	TS

Format Name	Number	Format Description	File Extension
Verilog_Fmt	543	Verilog Source Code ³	V
YAML_Fmt	544	YAML File ³	YML
Wiki_Fmt	545	MediaWiki File	
MS_Word_2007_Flat_XML_Fmt	546	Microsoft Word 2007 XML - Flat xml	XML
Matroska_Fmt	547	Matroska video File	MKV
SVG_Fmt	548	Scalable Vector Graphics image	SVG
Shapefile_Fmt	549	Shapefile	SHP, SHX
Flash_Video_Fmt	550	Flash video File	FLV
Embedded_OpenType_Fmt	551	Embedded OpenType font	EOT
Web_Open_Font_Fmt	552	Web Open Font Format	WOFF, WOFF2
OpenType_Fmt	553	OpenType Font	OTF
MNG_Fmt	554	Multiple-image Network Graphics	MNG
JNG_Fmt	555	JPEG Network Graphics	JNG
AppleScript_Binary_Fmt	556	AppleScript Binary Source Code	SCPT
Maya_Binary_Fmt	557	Autodesk Maya binary file	MB
Jupiter_Tessellation_Fmt	558	UGS Jupiter Tessellation file	JT
OGV_Fmt	559	Ogg Theora Video format	OGV
OGG_Container_Fmt	560	General Ogg Container format	OGG
GNU_Message_Catalog_Fmt	561	GNU Message Catalog format	MO

Format Name	Number	Format Description	File Extension
Windows_Shortcut_Fmt	562	Windows shortcut file	LNK
Apple_Typedstream_Fmt	563	Apple/NeXT typedstream data format	
XCF_Fmt	564	GIMP XCF image	XCF
PaintShop_Pro_Fmt	565	PaintShop Pro image	PSP, PSPIMAGE
SQLite_Database_Fmt	566	SQLite database format	QHC
MySQL_Table_Fmt	567	MySQL table definition file	FRM
Microsoft_Program_DB_Fmt	568	Microsoft Program Database format	PDB
OpenEXR_Fmt	569	OpenEXR image format	EXR
XMV_Fmt	570	4X Movie File	
AMV_Fmt	571	AMV video file	AMV
NIFF_Fmt	572	Notation Interchange File Format	NIF
CuBase_Fmt	573	Steinberg CuBase file	
SoundFont_Fmt	574	SoundFont file	
WebP_Fmt	575	WebP image	WEBP
ICC_Fmt	576	International Color Consortium files	ICC, ICM
PCF_Fmt	577	X11 Portable Compiled Font file	PCF
WebM_Fmt	578	WebM video file	WEBM
AMFF_Fmt	579	Amiga Metafile	AMF
ANBM_Fmt	580	IFF Animated Bitmap	

Format Name	Number	Format Description	File Extension
ANIM_Fmt	581	IFF Amiga animated raster graphics format	
DEEP_Fmt	582	IFF-DEEP TVPaint image	DEEP
FAXX_Fmt	583	IFF-FAXX Facsimile image	
ICON_Fmt	584	IFF Glow Icon image	
ILBM_Fmt	585	Interleaved BitMap image	IFF
LWOB_Fmt	586	LightWave Object format	LWOB
MAUD_Fmt	587	IFF-MAUD MacroSystem audio format	
PBM_Fmt	588	IFF Planar BitMap	
TDDD_Fmt	589	IFF TDDD and Imagine Object animation format	TDD
DjVu_Fmt	590	AT&T DjVu format	DJVU
InDesign_Fmt	591	Adobe InDesign document	
Calamus_Fmt	592	Calamus Desktop Publishing	
Adaptive_MultiRate_Fmt	593	Adaptive Multi-Rate audio format	AMR
FLAC_Fmt	594	Free Lossless Audio Codec format	FLAC
Ogg_FLAC_Fmt	595	Ogg Container FLAC audio format	OGG
SAS7BDAT_Fmt	596	SAS7BDAT database storage format	SAS7BDAT
Design_Web_Format_Fmt	597	Autodesk Design Web Format	DWF
Adobe_Flash_Audio_Book_Fmt	598	Adobe Flash Player audio book	F4B
Adobe_Flash_Audio_Fmt	599	Adobe Flash Player audio	F4A

Format Name	Number	Format Description	File Extension
Adobe_Flash_Protected_Video_Fmt	600	Adobe Flash Player protected video	F4P
Adobe_Flash_Video_Fmt	601	Adobe Flash Player video	F4V
Audible_Audiobook_Fmt	602	Audible Enhanced Audiobook	AAX
Canon_Camera_Fmt	603	Canon Digital Camera image	
Canon_Raw_Fmt	604	Canon Raw image	CR3
Casio_Camera_Fmt	605	Casio Digital Camera image	
Convergent_Design_Fmt	606	Convergent Design file	
DMB_MAF_Audio_Fmt	607	DMB MAF audio	
DMB_MAF_Video_Fmt	608	DMB MAF video	
DMP_Content_Fmt	609	Digital Media Project Content Format	
DVB_Fmt	610	Digital Video Broadcast format	DVB
Dirac_Wavelet_Compression_Fmt	611	ISO-BMFF Dirac Wavelet compression	
HEICS_Image_Sequence_Fmt	612	High Efficiency Image Format HEVC image sequence	HEICS
HEIC_Image_Fmt	613	High Efficiency Image Format HEVC image	HEIC
HEIFS_Image_Sequence_Fmt	614	High Efficiency Image Format image sequence	HEIFS
HEIF_Image_Fmt	615	High Efficiency Image Format image	HEIF
ISMACryp_Fmt	616	ISMACryp 2.0 Encrypted format	
ISO_3GPP2_Fmt	617	3GPP2 video file	3G2
ISO_3GPP_Fmt	618	3GPP video file	3GP

Format Name	Number	Format Description	File Extension
ISO_JPEG2000_JP2_Fmt	619	ISO-BMFF JPEG 2000 image	JP2
ISO_JPEG2000_JPM_Fmt	620	ISO-BMFF JPEG 2000 compound image	JPM
ISO_JPEG2000_JPX_Fmt	621	ISO-BMFF JPEG 2000 with extensions	JPX
ISO_QuickTime_Fmt	622	Apple ISO-BMFF QuickTime video	QT, MOV
KDDI_Video_Fmt	623	KDDI Video file	
MAF_Photo_Player_Fmt	624	MAF Photo Player	
MPEG4_AVC_Fmt	625	ISO-BMFF MPEG-4 with AVC extension	
MPEG4_M4A_Fmt	626	Apple MPEG-4 Part 14 audio	M4A
MPEG4_M4B_Fmt	627	Apple MPEG-4 Part 14 audio book	M4B
MPEG4_M4P_Fmt	628	Apple MPEG-4 Part 14 protected audio	M4P
MPEG4_M4V_Fmt	629	Apple MPEG-4 Part 14 video	M4V
MPEG4_Sony_PSP_Fmt	630	Sony PSP MPEG-4	MP4
MPEG_21_Fmt	631	MPEG-21	
Mobile_QuickTime_Fmt	632	Mobile QuickTime video	MQV
Motion_JPEG_2000_Fmt	633	Motion JPEG 2000	MJ2, MJP2
NTT_MPEG4_Fmt	634	NTT MPEG-4	
Nero_MPEG4_AVC_Profile	635	Nero MPEG-4 profile with AVC extension	
Nero_MPEG4_Audio_Fmt	636	Nero AAC audio	
Nero_MPEG4_Profile	637	Nero MPEG-4 profile	

Format Name	Number	Format Description	File Extension
OMA_DRM_Fmt	638	OMA DRM Format	
Panasonic_Camera_Fmt	639	Panasonic Digital Camera image	
Ross_Video_Fmt	640	Ross video	
SDA_Video_Fmt	641	SDA SD Memory Card video	
Samsung_Stereoscopic_Fmt	642	Samsung stereoscopic stream	
Sony_XAVC_Fmt	643	Sony XAVC video	
JPEG_2000_PGX_Fmt	644	JPEG 2000 PGX Verification Model image	PGX
Apple_Desktop_Services_Store_Fmt	645	Apple Desktop Services Store file	DS_Store
Core_Audio_Fmt	646	Apple Core Audio Format	CAF
VICAR_Fmt	647	VICAR image format	IMG
FITS_Fmt	648	Flexible Image Transport System FITS image	FIT
DIF_Fmt	649	Digital Interface Format (DIF) DV video	DV
MPEG_Transport_Stream_Fmt	650	MPEG Transport Stream data	TS
MPEG_Sequence_Fmt	651	MPEG Sequence format	
Ogg_OGM_Fmt	652	Ogg OGM video format	OGM
Ogg_Speex_Fmt	653	Ogg Speex audio format	SPX
Ogg_Opus_Fmt	654	Ogg Opus audio format	OGG
Musepack_Audio_Fmt	655	Musepack audio format	MPC
ART_Image_Fmt	656	ART image format	ART

Format Name	Number	Format Description	File Extension
Vivo_Fmt	657	Vivo audio-video format	VIV
QCP_Fmt	658	Qualcomm QCP audio	QCP
CSP_Codec_Fmt	659	Creative Signal Processor codec	CSP
TwinVQ_Fmt	660	NTT TwinVQ audio format	VQF
Interplay_MVE_Fmt	661	Interplay MVE video format	MVE
IRIX_Moviemaker_Fmt	662	IRIX Silicon Graphics moviemaker video file	MV, MOVIE
Sega_FILM_Fmt	663	Sega FILM video format	CPK, CAK
SMAF_Fmt	664	Synthetic music Mobile Application Format	MMF
NIST_SPHERE_Fmt	665	NIST SPeech HEader Resources format	NIST
Chinese_AVS_Fmt	666	Chinese AVS video format	
VQA_Fmt	667	Westwood Studios Vector Quantized Animation video file	VQA
YAFA_Fmt	668	Wildfire YAFA animation	YAFA
Origin_MVE_Fmt	669	Origin Wing Commander III MVE movie format	MVE
BBC_Dirac_Fmt	670	BBC Dirac video format	DRC
Maya_ASCII_Fmt	671	Autodesk Maya ASCII file format	MA
RenderMan_Fmt	672	Pixar RenderMan Interface Bytestream file	RIB
NOFF_Binary_Fmt	673	NOFF 3D Object File Format	NOFF
VTK_ASCII_Fmt	674	Visualization Toolkit VTK ASCII format	VTK

Format Name	Number	Format Description	File Extension
VTK_Binary_Fmt	675	Visualization Toolkit VTK Binary format	VTK
Wolfram_CDF_Fmt	676	Wolfram Mathematica Computable Document Format	CDF
Wolfram_Notebook_Fmt	677	Wolfram Mathematica Notebook Format	NB
HDF4_Fmt	678	Hierarchical Data Format HDF4	HDF, H4
HDF5_Fmt	679	Hierarchical Data Format HDF5	HDF, H5
ARMovie_Fmt	680	Acom RISC ARMovie video format	RPL
Windows_TV_DVR_Fmt	681	Windows Television DVR format	WTV
InstallShield_Z_Fmt	682	InstallShield Z archive format	Z
MS_DirectDraw_Surface_Fmt	683	Microsoft DirectDraw Surface container format	DDS
Bink_Fmt	684	Bink audio-video container format	BIK, BK2
LZMA_Fmt	685	LZMA compressed data format	LZMA
True_Audio_Fmt	686	True Audio format	TTA
Keepass_Fmt	687	Keepass Password file	KDB, KDBX
RPM_Fmt	688	RPM Package Manager file	RPM
Printer_Font_Metrics_Fmt	689	Adobe Printer Font Metrics format	PFM
Adobe_Font_Metrics_Fmt	690	Adobe Font Metrics ASCII format	AFM
Printer_Font_ASCII_Fmt	691	Adobe Printer Font ASCII format	PFA
Netware_Loadable_Module_Fmt	692	Netware Loadable Module format	NLM
TCPdump_pcap_Fmt	693	TCPdump packet stream capture savefile format	PCAP

Format Name	Number	Format Description	File Extension
Multiple_Master_Font_Fmt	694	Adobe Multiple master font format	MMM
TrueType_Font_Collection_Fmt	695	TrueType font collection format	TTC
Shapefile_Spatial_Index_Fmt	696	Shapefile binary spatial index format	SBX, SBN
Java_Key_Store_Fmt	697	Java Key Store format	KS
Java_JCE_Key_Store_Fmt	698	Java JCE Key Store format	
Quark_Xpress_Intel_Fmt	699	QuarkXPress Intel format	QXB
Windows_Imaging_Fmt	700	Microsoft Windows Imaging Format WIM	WIM
VMware_Virtual_Disk_Fmt	701	VMware Virtual Disk Format 5.0	VMDK
XPCConnect_Typelib_Fmt	702	XPCConnect Typelib Format	XPT
MS_DOS_Compression_Fmt	703	Microsoft MS-DOS installation compression	EX_
DLS_Fmt	704	DLS Downloadable Sounds format	DLS
MS_Windows_Registry_Fmt	705	Microsoft Windows Registry format	
Microsoft_Help_2_Fmt	706	Microsoft Help 2.0 format	HXD, HXW, HXH
Qt_Translation_Fmt	707	Qt binary translation file format	QM
PEM_SSL_Certificate_Fmt	708	PEM-encoded SSL certificate	CRT, PEM, CER, KEY
PostScript_Printer_Description_Fmt	709	Adobe PostScript Printer Description file	PPD
Speedo_Font_Fmt	710	Speedo Font format	SPD
InstallShield_Cabinet_Fmt	711	InstallShield Cabinet Archive format	CAB, HDR
InstallShield_Uninstall_Fmt	712	InstallShield Uninstall format	ISU

Format Name	Number	Format Description	File Extension
MS_OEDBX_Folder_Fmt	713	Outlook Express DBX folder database format	DBX
LabVIEW_Fmt	714	National Instruments LabVIEW file format	VI
SAP_Archive_SAR_Fmt	715	SAP compression archive SAR format	SAR
Netscape_Address_Book_Fmt	716	Netscape Address Book format	NAB
Universal_3D_Fmt	717	Universal 3D file format	U3D
Open_Inventor_ASCII_Fmt	718	Open Inventor ASCII format	IV
Open_Inventor_Binary_Fmt	719	Open Inventor Binary format	IV
X_Window_Dump_Fmt	720	X Window Dump image	XWD
Git_Packfile_Fmt	721	Git Packfile format	PACK
Xara_Xar_Fmt	722	Xara X Xar image format	XAR
Internet_Archive_ARC_Fmt	723	Internet Archive ARC format	ARC
Applix_Builder_Fmt	724	Applix Builder format	AB
Applix_Bitmap_Fmt	725	Applix Bitmap image format	IM
PEM_RSA_Private_Key_Fmt	726	PEM-encoded RSA private key	PEM
MIFF_Fmt	727	Magick Image File Format	MIFF
Subversion_Dump_Fmt	728	Subversion Dump format	
Virtual_Hard_Disk_Fmt	729	Microsoft Virtual Hard Disk format	VHD
Direct_Access_Archive_Fmt	730	PowerISO Direct Access Archive format	DAA
Debian_Binary_Fmt	731	Debian binary package format	DEB

Format Name	Number	Format Description	File Extension
XUL_Fastload_Fmt	732	Mozilla XUL Fastload format	MFL
Nastran_OP2_Fmt	733	Nastran OP2 format	OP2
Binary_Logging_Fmt	734	CAD Binary Logging Format	BLF
Measurement_Data_Fmt	735	CAD Measurement Data Format	MDF
Abaqus_ODB_Fmt	736	Abaqus ODB Format	ODB
Open_Diagnostic_Data_Exchange_Fmt	737	Vector Open Diagnostic Data Exchange format	ODX
Vector_ASCII_Fmt	738	Vector CAD ASCII ASC format	ASC
LSDYNA_State_Database_Fmt	739	LS-DYNA State Database format	
LSDYNA_Binary_Output_Fmt	740	LS-DYNA binary output (binout) format	
MS_Power_BI_Fmt	741	Microsoft Power BI Desktop format	PBIX
Tableau_Workbook_Fmt	742	Tableau Workbook format	TWB
Tableau_Packaged_Workbook_Fmt	743	Tableau Packaged Workbook format	TWBX
Tableau_Extract_Fmt	744	Tableau Extract format	TDE
Tableau_Data_Source_Fmt	745	Tableau Data Source format	TDS
Tableau_Packaged_Data_Source_Fmt	746	Tableau Packaged Data Source format	TDSX
Tableau_Preferences_Fmt	747	Tableau Preferences format	TPS
Tableau_Map_Source_Fmt	748	Tableau Map Source format	TMS
ABAP_Fmt	749	ABAP Source Code ⁴	ABAP
AMPL_Fmt	750	AMPL Source Code ⁴	AMPL

Format Name	Number	Format Description	File Extension
APL_Fmt	751	APL Source Code ⁴	APL
ASN1_Fmt	752	ASN.1 Source Code ⁴	ASN
ATS_Fmt	753	ATS Source Code ⁴	
Agda_Fmt	754	Agda Source Code ⁴	AGDA
Alloy_Fmt	755	Alloy Source Code ⁴	ALS
Apex_Fmt	756	Apex Source Code ⁴	CLS
Arduino_Fmt	757	Arduino Source Code ⁴	INO
AsciiDoc_Fmt	758	AsciiDoc Source Code ⁴	ASC
AspectJ_Fmt	759	AspectJ Source Code ⁴	AJ
Awk_Fmt	760	Awk Source Code ⁴	AWK
BlitzMax_Fmt	761	BlitzMax Source Code ⁴	BMX
Bluespec_Fmt	762	Bluespec Source Code ⁴	BSV
Brainfuck_Fmt	763	Brainfuck Source Code ⁴	B, BF
Brightscript_Fmt	764	Brightscript Source Code ⁴	BRS
CLIPS_Fmt	765	CLIPS Source Code ⁴	CLP
CMake_Fmt	766	CMake Source Code ⁴	CMAKE
COBOL_Fmt	767	COBOL Source Code ⁴	CBL, CCP, COB, CPY
CWeb_Fmt	768	CWeb Source Code ⁴	W
CartoCSS_Fmt	769	CartoCSS Source Code ⁴	MSS

Format Name	Number	Format Description	File Extension
Ceylon_Fmt	770	Ceylon Source Code ⁴	CEYLON
Chapel_Fmt	771	Chapel Source Code ⁴	CHPL
Clarion_Fmt	772	Clarion Source Code ⁴	CLW
Clean_Fmt	773	Clean Source Code ⁴	DCL, ICL
Component_Pascal_Fmt	774	Component Pascal Source Code ⁴	CP
Cool_Fmt	775	Cool Source Code ⁴	CL
Coq_Fmt	776	Coq Source Code ⁴	V
Creole_Fmt	777	Creole Source Code ⁴	CREOLE
Crystal_Fmt	778	Crystal Source Code ⁴	CR
Csound_Fmt	779	Csound Source Code ⁴	ORC
Csound_Document_Fmt	780	Csound Document Source Code ⁴	CSD
Cuda_Fmt	781	Cuda Source Code ⁴	CU
D_Fmt	782	D Source Code ⁴	DCL, ICL
DIGITAL_Command_Language_Fmt	783	DIGITAL Command Language Source Code ⁴	COM
DTrace_Fmt	784	DTrace Source Code ⁴	D
Dart_Fmt	785	Dart Source Code ⁴	DART
E_Fmt	786	E Source Code ⁴	E
ECL_Fmt	787	ECL Source Code ⁴	ECL
Elm_Fmt	788	Elm Source Code ⁴	ELM

Format Name	Number	Format Description	File Extension
Emacs_Lisp_Fmt	789	Emacs Lisp Source Code ⁴	EL
EmberScript_Fmt	790	EmberScript Source Code ⁴	EM
Fantom_Fmt	791	Fantom Source Code ⁴	FAN
Forth_Fmt	792	Forth Source Code ⁴	FOR, FORTH
FreeMarker_Fmt	793	FreeMarker Source Code ⁴	FTL
Frege_Fmt	794	Frege Source Code ⁴	FR
G_code_Fmt	795	G-code Source Code ⁴	G
GAMS_Fmt	796	GAMS Source Code ⁴	GMS
GAP_Fmt	797	GAP Source Code ⁴	
GDScript_Fmt	798	GDScript Source Code ⁴	GD
GLSL_Fmt	799	GLSL Source Code ⁴	GLSL
Game_Maker_Language_Fmt	800	Game Maker Language Source Code ⁴	GML
Gnuplot_Fmt	801	Gnuplot Source Code ⁴	GNU, GP
Golo_Fmt	802	Golo Source Code ⁴	GOLO
Gosu_Fmt	803	Gosu Source Code ⁴	GS
Gradle_Fmt	804	Gradle Source Code ⁴	GRADLE
GraphQL_Fmt	805	GraphQL Source Code ⁴	GRAPHQL
Graphviz_DOT_Fmt	806	Graphviz (DOT) Source Code ⁴	DOT
HLSL_Fmt	807	HLSL Source Code ⁴	HLSL

Format Name	Number	Format Description	File Extension
Hack_Fmt	808	Hack Source Code ⁴	
Haml_Fmt	809	Haml Source Code ⁴	HAML
Handlebars_Fmt	810	Handlebars Source Code ⁴	HBS
Hy_Fmt	811	Hy Source Code ⁴	HY
IDL_Fmt	812	IDL Source Code ⁴	PRO
IGOR_Pro_Fmt	813	IGOR Pro Source Code ⁴	IPF
Idris_Fmt	814	Idris Source Code ⁴	IDR
Inform_7_Fmt	815	Inform 7 Source Code ⁴	I7X
Ioke_Fmt	816	Ioke Source Code ⁴	IK
Isabelle_Fmt	817	Isabelle Source Code ⁴	
J_Fmt	818	J Source Code ⁴	IJS
JSONiq_Fmt	819	JSONiq Source Code ⁴	JQ
JSX_Fmt	820	JSX Source Code ⁴	JSX
Jasmin_Fmt	821	Jasmin Source Code ⁴	J
Jolie_Fmt	822	Jolie Source Code ⁴	
Julia_Fmt	823	Julia Source Code ⁴	JL
KiCad_Layout_Fmt	824	KiCad Layout Source Code ⁴	
KiCad_Schematic_Fmt	825	KiCad Schematic Source Code ⁴	SCH
Kotlin_Fmt	826	Kotlin Source Code ⁴	KT

Format Name	Number	Format Description	File Extension
LFE_Fmt	827	LFE Source Code ⁴	LFE
LOLCODE_Fmt	828	LOLCODE Source Code ⁴	LOL
Lasso_Fmt	829	Lasso Source Code ⁴	LAS, LASSO
Limbo_Fmt	830	Limbo Source Code ⁴	
LiveScript_Fmt	831	LiveScript Source Code ⁴	LS
M_Fmt	832	M Source Code ⁴	M
MAXScript_Fmt	833	MAXScript Source Code ⁴	MS
Markdown_Fmt	834	Markdown Source Code ⁴	MD
Matlab_Fmt	835	Matlab Source Code ⁴	M
Max_Code_Fmt	836	Max Source Code ⁴	MXT
Mercury_Fmt	837	Mercury Source Code ⁴	
Modelica_Fmt	838	Modelica Source Code ⁴	MO
Modula_2_Fmt	839	Modula-2 Source Code ⁴	MOD
Monkey_Fmt	840	Monkey Source Code ⁴	MONKEY
Moocode_Fmt	841	Moocode Source Code ⁴	MOO
NL_Fmt	842	NL Source Code ⁴	NL
NSIS_Fmt	843	NSIS Source Code ⁴	NSI
NetLogo_Fmt	844	NetLogo Source Code ⁴	NLOGO
NewLisp_Fmt	845	NewLisp Source Code ⁴	NL

Format Name	Number	Format Description	File Extension
Nginx_Fmt	846	Nginx Source Code ⁴	VHOST
Nix_Fmt	847	Nix Source Code ⁴	NIX
Nu_Fmt	848	Nu Source Code ⁴	NU
OCaml_Fmt	849	OCaml Source Code ⁴	
OpenCL_Fmt	850	OpenCL Source Code ⁴	CL
OpenEdge_ABL_Fmt	851	OpenEdge ABL Source Code ⁴	
OpenSCAD_Fmt	852	OpenSCAD Source Code ⁴	SCAD
Ox_Fmt	853	Ox Source Code ⁴	OX
Oxygene_Fmt	854	Oxygene Source Code ⁴	OXYGENE
Oz_Fmt	855	Oz Source Code ⁴	OZ
PAWN_Fmt	856	PAWN Source Code ⁴	PWN
PLpgSQL_Fmt	857	PLpgSQL Source Code ⁴	PLSQL
Pan_Fmt	858	Pan Source Code ⁴	PAN
Parrot_Assembly_Fmt	859	Parrot Assembly Source Code ⁴	PASM
PicoLisp_Fmt	860	PicoLisp Source Code ⁴	
Pike_Fmt	861	Pike Source Code ⁴	PIKE
Pony_Fmt	862	Pony Source Code ⁴	PONY
Processing_Fmt	863	Processing Source Code ⁴	PDE
PureBasic_Fmt	864	PureBasic Source Code ⁴	PB

Format Name	Number	Format Description	File Extension
QMake_Fmt	865	QMake File ⁴	
RAML_Fmt	866	RAML Source Code ⁴	RAML
RDoc_Fmt	867	RDoc Source Code ⁴	RDOC
REXX_Fmt	868	REXX Source Code ⁴	REXX
Racket_Fmt	869	Racket Source Code ⁴	
Ragel_Fmt	870	Ragel Source Code ⁴	
Rascal_Fmt	871	Rascal Source Code ⁴	RSC
Rebol_Fmt	872	Rebol Source Code ⁴	REB, REBOL
Red_Fmt	873	Red Source Code ⁴	RED
RenPy_Fmt	874	Ren'Py Source Code ⁴	RPY
RenderScript_Fmt	875	RenderScript Source Code ⁴	RS
Ring_Fmt	876	Ring Source Code ⁴	RING
RobotFramework_Fmt	877	RobotFramework Source Code ⁴	ROBOT
SAS_Fmt	878	SAS Source Code ⁴	SAS
SPARQL_Fmt	879	SPARQL Source Code ⁴	
SQL_Fmt	880	SQL Source Code ⁴	
SQLPL_Fmt	881	SQLPL Source Code ⁴	
SaltStack_Fmt	882	SaltStack Source Code ⁴	SLS
Scheme_Fmt	883	Scheme Source Code ⁴	

Format Name	Number	Format Description	File Extension
Scilab_Fmt	884	Scilab Source Code ⁴	SCI
Squirrel_Fmt	885	Squirrel Source Code ⁴	NUT
Stan_Fmt	886	Stan Source Code ⁴	STAN
Stata_Fmt	887	Stata Source Code ⁴	
Stylus_Fmt	888	Stylus Source Code ⁴	STYL
SuperCollider_Fmt	889	SuperCollider Source Code ⁴	SC
SystemVerilog_Fmt	890	SystemVerilog Source Code ⁴	SV
TXL_Fmt	891	TXL Source Code ⁴	TXL
Turing_Fmt	892	Turing Source Code ⁴	T
Turtle_Fmt	893	Turtle Source Code ⁴	TTL
UrWeb_Fmt	894	UrWeb Source Code ⁴	UR, URS
Vim_script_Fmt	895	Vim script File ⁴	VIM
Visual_Basic_Fmt	896	Visual Basic Source Code ⁴	VB
WebAssembly_Fmt	897	WebAssembly Source Code ⁴	WAT
WebIDL_Fmt	898	WebIDL Source Code ⁴	WEBIDL
X10_Fmt	899	X10 Source Code ⁴	X10
XQuery_Fmt	900	XQuery Source Code ⁴	XQM
Xojo_Fmt	901	Xojo Source Code ⁴	
Xtend_Fmt	902	Xtend Source Code ⁴	XTEND

Format Name	Number	Format Description	File Extension
YANG_Fmt	903	YANG Source Code ⁴	YANG
Zephir_Fmt	904	Zephir Source Code ⁴	ZEP
eC_Fmt	905	eC Source Code ⁴	EC
reStructuredText_Fmt	906	reStructuredText Source Code ⁴	
xBASE_Fmt	907	xBASE Source Code ⁴	
Windows_Installer_Fmt	908	MSI Windows Installer format	MSI
Autodesk_3ds_Max_Fmt	909	Autodesk 3ds Max format	MAX
PhotoDraw_Mix_Fmt	910	PhotoDraw MIX image	MIX
Softimage_SCN_Fmt	911	Softimage Scene SCN format	SCN
Parasolid_XT_Fmt	912	Parasolid ascii XT format	X_T
Parasolid_XB_Fmt	913	Parasolid binary XB format	X_B
IGES_Fmt	914	Initial Graphics Exchange Specification format	IGS
ACE_Archive_Fmt	915	ACE archive format	ACE
Grasshopper_GHX_Fmt	916	Grasshopper GHX format	GHX
MS_FrontPage_Macro_Fmt	917	Microsoft FrontPage macro file format	FPM
MS_AtWork_Fax_Fmt	918	Microsoft AtWork Fax format	AWD
MS_Image_Composer_Fmt	919	Microsoft Image Composer format	MIC
MS_Visual_InterDev_Fmt	920	Microsoft Visual InterDev web project items file	WDM
Macromedia_Flash_FLA_OLE_Fmt	921	Macromedia Flash FLA Project File OLE format	FLA

Format Name	Number	Format Description	File Extension
Corel_Draw_X4_Fmt	922	CorelDRAW version X4 onwards	CDRX
Ogg_Daala_Fmt	923	Ogg Daala video format	OGV
Ogg_BBC_Dirac_Fmt	924	Ogg BBC Dirac video format	OGV
PKCS_7_Fmt	925	PKCS #7 cryptographic format	P7S
Time_Stamped_Data_Fmt	926	Time-stamped data format	TSD
Sereal_Fmt	927	Sereal data serialization format	SRL
Associated_Signature_Simple_Fmt	928	Associated Signature Container Simple format	ASICS
Associated_Signature_Extended_Fmt	929	Associated Signature Container Extended format	ASICE
iBooks_Fmt	930	Apple iBooks format	IBOOKS
PDF_Forms_Data_Fmt	931	PDF Forms Data Format	FDF
PDF_XML_Forms_Data_Fmt	932	PDF XML Forms Data Format	XFDF
AxCrypt_Fmt	933	AxCrypt encrypted document	AXX
Unix_Archive_Fmt	934	Unix Archive ar format	AR
Berkeley_Btree_Database_Fmt	935	Berkeley DB btree database format	DB
Berkeley_Hash_Database_Fmt	936	Berkeley DB hash database format	DB
Berkeley_Log_Database_Fmt	937	Berkeley DB log database format	
Berkeley_Queue_Database_Fmt	938	Berkeley DB queue database format	
BitTorrent_Fmt	939	BitTorrent file format	TORRENT
Chrome_Extension_Fmt	940	Google Chrome Extension format	CRX

Format Name	Number	Format Description	File Extension
Dalvik_Executable_Fmt	941	Dalvik Executable dex format	DEX
Foxmail_Fmt	942	Foxmail email format	BOX
GRIB_Fmt	943	General Regularly-distributed Information in Binary form GRIB format	GRB, GRIB2
Zstandard_Fmt	944	Zstandard compression format	ZSTD
LZ4_Fmt	945	LZ4 compressed file	LZ4
MS_Money_Fmt	946	Microsoft Money format	MNY
NetCDF_Fmt	947	Network Common Data Form NetCDF format	NC
SAS6_Data_Fmt	948	SAS 6 Data storage format	SD2
SAS_Transport_Fmt	949	SAS Transport File XPORT format	XPT, XPORT
Snappy_Framed_Fmt	950	Snappy Framed compression format	SZ
Stata_Data_Fmt	951	Stata Data Format	DTA
SPSS_SAV_Fmt	952	SPSS Statistics Data File Format	SAV
Zoo_Archive_Fmt	953	Zoo Compressed Archive Format	ZOO
CDX_Fmt	954	ChemDraw CDX format	CDX
CDXML_Fmt	955	ChemDraw CDXML format	CDXML
BPG_Fmt	956	Better Portable Graphics BPG format	BPG
Apple_Icon_Fmt	957	Apple Icon image format	ICNS
NITF_Fmt	958	National Imagery Transmission Format NITF image	NTF, NITF

Format Name	Number	Format Description	File Extension
ERDAS_Imagine_Fmt	959	ERDAS Imagine image format	HFA
MS_Office_Temporary_Owner_Fmt	960	Microsoft Office temporary owner file	
EAC3_Audio_Fmt	961	Enhanced-AC3 (EAC3) Audio File format	AC3
COFF_Relocatable_Fmt	962	Common Object File Format (COFF) relocatable object	O
COFF_Executable_Fmt	963	Common Object File Format (COFF) executable	
COFF_Dynamic_Lib_Fmt	964	Common Object File Format (COFF) dynamic library	
ELF_Core_Fmt	965	ELF Core file	
Purify_Fmt	966	Rational Purify data file	PFY
Kryptel_Fmt	967	Kryptel encrypted file	EDC
Windows_Core_Dump_Fmt	968	Windows heap or mini core dump file	DMP
Qt_Prerendered_Font_Fmt	969	Qt Prerendered Font format	QPF2
AIX_Relocatable_Fmt	970	AIX/RISC COFF relocatable object	
AIX_Executable_Fmt	971	AIX/RISC COFF executable	
AIX_Dynamic_Lib_Fmt	972	AIX/RISC COFF dynamic library	A
HPUX_Relocatable_Fmt	973	HPUX/PA-RISC COFF relocatable object	
HPUX_Executable_Fmt	974	HPUX/PA-RISC COFF executable	
HPUX_Dynamic_Lib_Fmt	975	HPUX/PA-RISC COFF dynamic library	SL
XML_EBCDIC_Fmt	976	EBCDIC-encoded XML file	XML
MPEG_JVT_H264_Fmt	977	MPEG JVT-NAL sequence H264 video	264

Format Name	Number	Format Description	File Extension
Material_Exchange_Fmt	978	Material Exchange Format audio-video container format	MXF
MS_Agent_Character_Fmt	979	Microsoft Agent Character file	ACS
Quicken_Fmt	980	Quicken data file	QDF
MS_Outlook_Address_Fmt	981	Microsoft Outlook address file	WAB
MS_Answer_Wizard_Fmt	982	Microsoft Answer Wizard file	
ADX_Fmt	983	ADX audio file	ADX
System_Deployment_Image_Fmt	984	Microsoft System Deployment Image SDI format	SDI
Free_Lossless_Image_Fmt	985	Free Lossless Image Format (FLIF)	FLIF
DPX_Fmt	986	Digital Picture Exchange (DPX) image format	DPX
Avro_Fmt	987	Apache Avro binary format	AVRO
InstallShield_Archive_Fmt	988	InstallShield archive (early versions) format	EX_
Mac_Executable_Fmt	989	Mac OS-X (Mach-O) executable format	
GDSII_Fmt	990	GDSII data format	GDS
ActiveMime_Fmt	991	Microsoft ActiveMime (mso) documents	MSO
SmartCharts_Fmt	992	BizInt SmartCharts data format	CHP, CHRR
Webex_ARF_Fmt	993	Webex advanced network ARF recordings	ARF
Webex_WRF_Fmt	994	Webex local WRF recordings	WRF
PGP_NetShare_Fmt	995	Symantec PGP NetShare encrypted file	
Ability_WP_OLE_Fmt	996	Ability Write later versions format	AWW

Format Name	Number	Format Description	File Extension
Ability_SS_OLE_Fmt	997	Ability Spreadsheet later versions format	AWS
InDesign_IDML_Fmt	998	Adobe InDesign IDML format	IDML
Executable_JAR_Fmt	999	Executable Java Archive (jar) file	JAR
IDOL_IDX_Fmt	1000	IDOL Server IDX file	IDX
Android_Package_Kit_Fmt	1001	Android Package Kit (APK) format	APK
Android_Binary_XML_Fmt	1002	Android Binary XML (compressed by aapt) format	XML
Java_WAR_Fmt	1003	Java WAR file format	WAR
Java_EAR_Fmt	1004	Java EAR file format	EAR
Atom_Syndication_Fmt	1005	Atom Syndication Format	ATOM
RSS_Fmt	1006	RSS syndication XML format	RSS
SMIL_Fmt	1007	Synchronized Multimedia Integration Language (SMIL) XML format	SMIL
XSLT_Fmt	1008	Extensible Stylesheet Language Transformations (XSLT) format	XSL, XSLT
XML_Shareable_Playlist_Fmt	1009	XML Shareable Playlist Format (XSPF)	XSPF
FictionBook_Fmt	1010	FictionBook e-book XML format	FB2
Adobe_Premiere_Project_Fmt	1011	Adobe Premiere project format	PPJ
RDF_XML_Fmt	1012	RDF/XML format	RDF
Really_Simple_Discovery_Fmt	1013	Really Simple Discovery (RSD) XML format	RSD
SBML_Fmt	1014	Systems Biology Markul Language (SBML) XML format	SBML

Format Name	Number	Format Description	File Extension
SRU_Fmt	1015	Search/Retrieve via URL (SRU) XML format	SRU
SSML_Fmt	1016	Speech Synthesis Markup Language (SSML) XML format	SSML
PLS_Fmt	1017	Pronunciation Lexicon Specification (PLS) XML format	PLS
TEI_Fmt	1018	Text Encoding Initiative (TEI) XML format	TEI
METS_Fmt	1019	Metadata Encoding and Transmission Standard (METS) XML format	METS
MODS_Fmt	1020	Metadata Object Description Schema (MODS) XML format	MODS
Metalink_Fmt	1021	Metalink XML format	METALINK
Open_eBook_Fmt	1022	Open eBook (OEBPS) XML format	OPF
SRGS_Fmt	1023	Speech Recognition Grammar Specification (SRGS) XML format	SRGS
SPARQL_Results_Fmt	1024	SPARQL Query Results XML format	SRX
Adobe_XML_Data_Package_Fmt	1025	Adobe XML Data Package format	XDP
ESzigno_Fmt	1026	e-Szigno signed xml document	ES3
Mozilla_XUL_Fmt	1027	Mozilla XML User Interface Language (XUL) XML format	XUL
SyncML_Fmt	1028	Synchronization Markup Language (SyncML) XML format	XML
VoiceXML_Fmt	1029	VoiceXML (VXML) XML format	VXML
TI_Target_Configuration_Fmt	1030	Texas Instruments CCXML target configuration XML format	CCXML

Format Name	Number	Format Description	File Extension
LZFSE_Fmt	1031	Lempel-Ziv Finite State Entropy (LZFSE) compression format	LZFSE
Kindle_eBook_Fmt	1032	Amazon Kindle or Mobipocket eBook format	AZW, PRC
Oasis_Stream_Fmt	1033	Open Artwork System Interchange Standard (OASIS) format	OAS
Amazon_KFX_Fmt	1034	Amazon KFX eBook format	KFX
KTX_Fmt	1035	KTX image format	KTX
GMSH_Mesh_Fmt	1036	GMSH Mesh polygon format	MSH
Collada_DAE_Fmt	1037	Collada Digital Asset Exchange (DAE) format	DAE
YIN_Fmt	1038	YIN XML format	YIN
MPEG_Playlist_Fmt	1039	MPEG audio playlist format	M3U
Windows_Audio_Playlist_Fmt	1040	Windows Audio playlist format	WAX
DTS_Audio_Fmt	1041	DTS Coherent Acoustics audio format	DTS
Chemical_Markup_Language_Fmt	1042	Chemical Markup Language (CML) XML format	CML
CrystalMaker_Fmt	1043	CrystalMaker chemical format	CMDF
VTK_XML_Fmt	1044	Visualization Toolkit VTK XML format	VTU
IPFIX_Fmt	1045	IP Flow Information Export (IPFIX) format	IPFIX
Portable_Font_Resource_Fmt	1046	Portable Font Resource font format	PFR
MARC_Fmt	1047	Machine-Readable Cataloging (MARC21) format	MARC
MARC_XML_Fmt	1048	Machine-Readable Cataloging (MARC) XML format	XML

Format Name	Number	Format Description	File Extension
XAR_Fmt	1049	Extensible Archive (XAR) format	
Symbian_Installer_Fmt	1050	Symbian installer format	SIS
SO_Drawing_XML_Fmt	1051	OpenDocument format (OpenOffice 1/StarOffice 6.7) Drawing XML	SXD
SO_Text_Global_XML_Fmt	1052	OpenDocument format (OpenOffice 1/StarOffice 6.7) Writer Master document XML	SXG
ODF_Chart_Fmt	1053	ODF Chart	ODC
ODF_Database_Fmt	1054	ODF Database	ODB
ODF_Image_Fmt	1055	ODF Image	ODI
ODF_Text_Master_Fmt	1056	ODF Text Master	ODM
ODF_Text_Web_Fmt	1057	ODF Text Web	OTH
ODF_Chart_Template_Fmt	1058	ODF Chart Template	OTC
ODF_Formula_Template_Fmt	1059	ODF Formula Template	OTF
ODF_Drawing_Template_Fmt	1060	ODF Drawing/Graphics Template	OTG
ODF_Image_Template_Fmt	1061	ODF Image Template	OTI
ODF_Presentation_Template_Fmt	1062	ODF Presentation Template	OTP
ODF_Spreadsheet_Template_Fmt	1063	ODF Spreadsheet Template	OTS
ODF_Text_Template_Fmt	1064	ODF Text Template	OTT
ODF_Chart_XML_Fmt	1065	ODF Chart flat XML format	FODC
ODF_Drawing_XML_Fmt	1066	ODF Drawing/Graphics flat XML format	FODG

Format Name	Number	Format Description	File Extension
ODF_Formula_XML_Fmt	1067	ODF Formula flat XML format	FODF
ODF_Image_XML_Fmt	1068	ODF Image flat XML format	FODI
ODF_Presentation_XML_Fmt	1069	ODF Presentation flat XML format	FODP
ODF_Spreadsheet_XML_Fmt	1070	ODF Spreadsheet flat XML format	FODS
ODF_Text_XML_Fmt	1071	ODF Text flat XML format	FODT
ODF_Extension_Fmt	1072	ODF Extension format	OXT
StarView_Metafile_Fmt	1073	OpenOffice StarView MetaFile format	SVM
BBeB_LRF_eBook_Fmt	1074	Broad Band eBook (BBeB) in LRF format	LRF
GPG_Trust_DB_Fmt	1075	GPG trust database format	GPG
VICE_Emulator_Fmt	1076	VICE (Versatile Commodore Emulator) format	VSF
Portable_Game_Notation_Fmt	1077	Portable Game Notation chess format	PGN
Doom_WAD_Fmt	1078	Doom IWAD/PWAD format	WAD
Device_Tree_Blob_Fmt	1079	Linux Device Tree Blob format	DTB
BDF_Font_Fmt	1080	Glyph Bitmap Distribution Format	BDF
PC_Screen_Font_Fmt	1081	PC Screen Font format	PSF
JNLP_Fmt	1082	Java Network Launching Protocol	JNLP
XAML_Browser_Application_Fmt	1083	XAML Browser Application (XBAP) format	XBAP
MS_Binder_Fmt	1084	Microsoft Office Binder format	OBP
XAP_Fmt	1085	Microsoft Silverlight application (XAP) format	XAP

Format Name	Number	Format Description	File Extension
StuffIt_X_Fmt	1086	StuffIt X (SITX) archive format	SITX
FIG_Fmt	1087	Facility for Interactive Generation of figures (FIG) image format	FIG
XPIInstall_Fmt	1088	XPIInstall Cross-Platform Installer Module (XPI) format	XPI
XDF_Fmt	1089	Extensible Data Format (XDF) XML format	XDF
MXML_Fmt	1090	MXML UI markup language XML format	MXML
MusicXML_Fmt	1091	MusicXML format	MXL
Finale_Fmt	1092	Finale audio format	MUS
Spotfire_DXP_Fmt	1093	TIBCO Spotfire DXP data format	DXP
MS_Office_Theme_2007_Fmt	1094	Microsoft Office theme format	THMX
Adobe_AIR_Installer_Fmt	1095	Adobe AIR application installer package	AIR
Flex_Project_Fmt	1096	Adobe Flash Flex project file format	FXP
FoxPro_Fmt	1097	FoxPro compiled source format	FXP
VST_Preset_Fmt	1098	Virtual Studio Technology (VST) preset format	FXP
Mischief_Image_Fmt	1099	Mischief vector graphics image format	ART
FreeArc_Fmt	1100	FreeArc archive format	ARC
Autodesk_3ds_Fmt	1101	Autodesk 3ds format	3DS
Monkeys_Audio_Fmt	1102	Monkey's Audio format	APE
CALS_Fmt	1103	CALS raster image format	CAL

Format Name	Number	Format Description	File Extension
Dr_Halo_PAL_Fmt	1104	Dr Halo raster image PAL file format	PAL
DPG_Fmt	1105	Nintendo DS DPG video format	DPG
JPEG_XR_Fmt	1106	JPEG XR (extended range) image format	JXR, HDP
TCR_eBook_Fmt	1107	TCR (Text Compression for Reader) eBook format	TCR

¹MHT, EML, and MBX files might return either format 2, 233, or 395, depending on the text in the file. In general, files that contain fields such as **To**, **From**, **Date**, or **Subject** are considered to be email messages; files that contain fields such as **content-type** and **mime-version** are considered to be MHT files; and files that do not contain any of those fields are considered to be text files.

²All CAT file extensions, for example CATDrawing, CATProduct, CATPart, and so on.

³This format is returned only if you enable source code identification. See [Source Code Identification, on page 100](#).

⁴This format is returned only if you enable extended source code identification. See [Source Code Identification, on page 100](#).

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](#) 320
- [Coded Character Sets](#) 328

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 (HGX)	n/a	n/a	n/a
Bzip2 (BZ2)	n/a	n/a	n/a
EnCase – Expert Witness Compression Format (E01)	n/a	n/a	n/a
GZIP (GZ)	n/a	n/a	n/a
ISO (ISO)	n/a	n/a	n/a
Java Archive (JAR)	n/a	n/a	n/a
Legato EMailXtender Archive (EMX)	n/a	n/a	n/a
MacBinary (BIN)	n/a	n/a	n/a
Mac Disk Copy Disk Image (DMG)	n/a	n/a	n/a
Microsoft Backup File (BKF)	n/a	n/a	n/a

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Microsoft Cabinet format (CAB)	n/a	n/a	n/a
Microsoft Compiled HTML Help (CHM)	n/a	n/a	n/a
Microsoft Compressed Folder (LZH)	n/a	n/a	n/a
PKZip (ZIP)	n/a	n/a	n/a
Microsoft Outlook DBX (DBX)	Y	Y	Y
Microsoft Outlook Offline Storage File (OST)	Y	Y	Y
RAR Archive (RAR)	n/a	n/a	n/a
Tape Archive (TAR)	n/a	n/a	n/a
UNIX Compress (Z)	n/a	n/a	n/a
UUEncoding (UUE)	n/a	n/a	n/a
Windows Scrap File (SHS)	n/a	n/a	n/a
WinZip (ZIP)	n/a	n/a	n/a
Binary			
Executable (EXE)	n/a	n/a	n/a
Link Library (DLL)	n/a	n/a	n/a
Computer-aided Design			
AutoCAD Drawing (DWG)	Y	Y	Y
AutoCAD Drawing Exchange (DXF)	Y	Y	Y
CATIA formats (CAT)	Y	N	N
Microsoft Visio (VSD)	Y	Y	Y
Database			
dBase Database	Y	N	N
Microsoft Access (MDB)	Y	Y	N
Microsoft Project (MPP)	Y	Y	N
Desktop Publishing			
Microsoft Publisher	N	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Display			
Adobe Portable Document Format (PDF)	Y	Y ¹	Y
Graphics			
Computer Graphics Metafile (CGM)	Y	N	N
Corel DRAW (CDR)	n/a	n/a	n/a
DCX Fax System (DCX)	Y	N	N
DICOM – Digital Imaging and Communications in Medicine (DCM)	n/a	n/a	n/a
Encapsulated PostScript (EPS)	Y	N	N
Enhanced Metafile (EMF)	Y	Y	N
Graphic Interchange Format (GIF)	n/a	n/a	n/a
JBIG2	n/a	n/a	n/a
JPEG	n/a	n/a	n/a
JPEG 2000	n/a	n/a	n/a
Lotus AMIDraw Graphics (SDW)	n/a	n/a	n/a
Lotus Pic (PIC)	n/a	n/a	n/a
Macintosh Raster (PICT/PCT)	n/a	n/a	n/a
MacPaint (PNTG)	n/a	n/a	n/a
Microsoft Office Drawing (MSO)	n/a	n/a	n/a
Omni Graffle (GRAFFLE)	Y	N	N
PC PaintBrush (PCX)	n/a	n/a	n/a

¹Multibyte PDFs are supported, provided the PDF document is created by using either Character ID-keyed (CID) fonts, predefined CJK CMap files, or ToUnicode font encodings, and does not contain embedded fonts. See the Adobe website and the Adobe Acrobat documentation for more information. Any multibyte characters that are not supported are displayed using the replacement character. By default, the replacement character is a question mark (?).

To determine the type of font encodings that are used in a PDF, open the PDF in Adobe Acrobat, and select File > Document Info > Fonts. If the Encoding column lists Custom or Embedded encodings, you might encounter problems converting the PDF.

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Portable Network Graphics (PNG)	n/a	n/a	n/a
SGI RGB Image (RGB)	n/a	n/a	n/a
Sun Raster Image (RS)	n/a	n/a	n/a
Tagged Image File (TIFF)	Y	N	N
Truevision Targa (TGA)	n/a	n/a	n/a
Windows Animated Cursor (ANI)	n/a	n/a	n/a
Windows Bitmap (BMP)	n/a	n/a	n/a
Windows Icon Cursor (ICO)	n/a	n/a	n/a
Windows Metafile (WMF)	Y	Y	N
WordPerfect Graphics 1 (WPG)	Y	N	N
WordPerfect Graphics 2 (WPG)	Y	N	N
Mail			
Documentum EMCMP Format	Y	Y	Y
Domino XML Language (DXL)	Y	Y	N
GroupWise FileSurf	Y	N	N
Legato Extender (ONM)	Y	Y	N
Lotus Notes database (NSF)	Y	Y	Y
Mailbox (MBX)	Y	Y	Y
Microsoft Entourage Database	Y	Y	Y
Microsoft Outlook (MSG)	Y	Y	Y
Microsoft Outlook Express (EML)	Y	Y	Y
Microsoft Outlook iCalendar	Y	Y	Y
Microsoft Outlook for Macintosh	Y	Y	Y
Microsoft Outlook Offline Storage File	Y	Y	Y
Microsoft Outlook Personal File Folders (PST)	Y	Y	Y
Microsoft Outlook vCard Contact			
Text Mail (MIME)	Y	Y	Y

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Transport Neutral Encapsulation Format	Y	Y	Y
Multimedia			
Advanced Systems Format (ASF)	n/a	n/a	n/a
Audio Interchange File Format (AIFF)	n/a	n/a	n/a
Microsoft Wave Sound (WAV)	n/a	n/a	n/a
MIDI (MID)	n/a	n/a	n/a
MPEG 1 Audio Layer 3 (MP3)	n/a	n/a	n/a
MPEG 1 Video (MPG)	n/a	n/a	n/a
MPEG 2 Audio (MPEGA)	n/a	n/a	n/a
MPEG 4 Audio (MP4)	n/a	n/a	n/a
NeXT/Sun Audio (AU)	n/a	n/a	n/a
QuickTime Movie (QT/MOV)	n/a	n/a	n/a
Windows Video (AVI)	n/a	n/a	n/a
Presentations			
Apple iWork Keynote (GZ)	Y	Y	N
Applix Presents (AG)	character set 1252 only	N	N
Corel Presentations (SHW)	character set 1252 only	N	N
Extensible Forms Description Language (XFD)	Y	Y	N
Lotus Freelance Graphics 2 (PRE)	character set 850 only	N	N
Lotus Freelance Graphics (PRZ)	Y	Japanese, Simple Chinese, Traditional Chinese, Thai only	N
Macromedia Flash (SWF)	Y	Y	N
Microsoft OneNote	Y	Y	N
Microsoft PowerPoint PC (PPT)	character set 1252 only	Traditional Chinese only	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Microsoft PowerPoint Windows (PPT)	Y	Japanese, Simple Chinese, Traditional Chinese, Korean only	Hebrew only
Microsoft PowerPoint Macintosh (PPT)	Y	N	N
Microsoft PowerPoint Windows XML 2007 and 2010 (PPTX)	Y	Y	Y
OASIS Open Document (ODP)	Y	Y	N
OpenOffice Impress (ODP)	Y	Y	N
StarOffice Impress (ODP)	Y	Y	N
Spreadsheets			
Apple iWork Numbers (GZ)	Y	Y	N
Applix Spreadsheets (AS)	character set 1252 only	N	N
Comma Separated Values (CSV)	character set 1252 only	N	N
Corel Quattro Pro (QPW/WB3)	Y	N	N
Data Interchange Format (DIF)	Y	Y	Y ¹
Lotus 1-2-3 (123)	Y	Y	Y
Lotus 1-2-3 (WK4)	Y	Y	N
Lotus 123 Charts (123)	Y	Y	N
Microsoft Excel Charts (XLS)	Y	Y	N
Microsoft Excel Macintosh (XLS)	Y	N	N
Microsoft Excel Windows (XLS)	Y	Y	Y ²
Microsoft Excel Windows XML 2007 (XLSX)	Y	Y	N
Microsoft Office Excel Binary Format (XLSB)	Y	Y	N
Microsoft Works Spreadsheet (S30/S40)	Y	N	N
OASIS Open Document (ODS)	Y	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
OpenOffice Calc (ODS)	Y	Y	N
StarOffice Calc (ODS)	Y	Y	N
Text and Markup			
ANSI (TXT)	Y	Y	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
Fujitsu Oasys (OA2)	Y	Y	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Hangul (HWP)	Y	Y	N
Health level7 (HL7)	Y	Y	Y
IBM DCA/RTF (DC)	character sets 500, 1026 only	N	N
JustSystems Ichitaro (JTD)	Y	Y	N
Lotus AMI Pro (SAM)	Y	Simple Chinese, Traditional Chinese, Japanese, Thai only	Y
Lotus AMI Professional Write Plus (AMI)	Y	Simple Chinese, Traditional Chinese, Japanese, Thai only	N
Lotus Word Pro (LWP)	Y	Y	Y ³
Lotus SmartMaster (MWP)	Y	Y	N
Microsoft Word PC (DOC)	character set 1252 only	N	N
Microsoft Word Windows V1-2 (DOC)	Y	N	N
Microsoft Word Windows V6, 7, 8, 95 (DOC)	Y	Y	Hebrew only ³
Microsoft Word Windows V97 through 2003 (DOC)	Y	Y	Y ³
Microsoft Word Windows XML 2007 and 2010 (DOCX)	Y	Y	Y ³
Microsoft Word Macintosh (DOC)	Y	N	Y ³
Microsoft Works (WPS)	Y	Japanese only	N
Microsoft Write (WRI)	Y	Japanese only	N
OASIS Open Document (ODT)	Y	Y	N
Omni Outliner (OO3)	Y	Y	N
OpenOffice Writer (ODT)	Y	Y	N
Open Publication Structure eBook (EPUB)	Y	Y	Y
StarOffice Writer (ODT)	Y	Y	N
Skype Log (DBB)	Y	Y (null-terminated charsets)	N

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
WordPad (RTF)	Y	Y	Y
WordPerfect Linux (WPS)	Y	N	N
WordPerfect Macintosh (WPS)	Y	N	N
WordPerfect Windows (WO)	Y	N	N
XML Paper Specification (XPS)	Y	Y	N
XYWrite Windows (XY4)	character set 1252 only	N	N
Yahoo! Instant Messenger (DAT)	Y	Y (null-terminated charsets)	N

¹The text direction in the output file might not be correct.

²In Export SDK, a bidirectional right-to-left (RTL) tag is extracted from this format and included in the direction element (`<dir=RTL>`) of the output.

Coded Character Sets

This section lists which character set you can use to specify the target character set. The coded character sets are enumerated in `kvtypes.h` and defined in the 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)	Y1
KVCS_57005	ISCII Telugu (x-iscii-te)	Y1
KVCS_57006	ISCII Assamese (x-iscii-as)	Y1
KVCS_57007	ISCII Oriya (x-iscii-or)	Y1
KVCS_57008	ISCII Kannada (x-iscii-ka)	Y1
KVCS_57009	ISCII Malayalam (x-iscii-ma)	Y1
KVCS_57010	ISCII Gujarathi (x-iscii-gu)	Y1
KVCS_57011	ISCII Panjabi (x-iscii-pa)	Y 1
KVCS_GB18030b2	Reserved for internal use	n/a
KVCS_GB18030	GB18030 (Chinese 4-byte character set)	Y
KVCS_8859_11	ISO 8859-11 (Thai)	Y
KVCS_8859_16	ISO 8859-16 (Latin-10 South-Eastern Europe)	Y
KVCS_ARABICMAC	Arabic Mac (x-mac-arabic)	Y
KVCS_KOI8U	Cyrillic (KOI8U Ukrainian)	Y
KVCS_HZGB2312	The 7-bit representation of GB 2312 / RFC 1842	n/a

¹The character set cannot be forced as output in Export SDK and Viewing SDK because the character

set is not supported by the major browsers.

Appendix D: Extract and Format Lotus Notes Subfiles

This section describes how to create XML templates to alter the appearance of extracted Lotus mail note subfiles so that they maintain the look and feel of the original notes.

- [Overview](#) 334
- [Customize XML Templates](#) 334
- [Template Elements and Attributes](#) 336
- [Date and Time Formats](#) 341

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 336](#).
3. Extract the Lotus database file.

Use Demo Templates

For testing purposes, you can extract notes by using a set of demo templates, which are provided to demonstrate the proper usage of all the XML elements and attributes, because the default templates do not use all the XML elements.

The demo templates are available at:

install\OS\bin\NSFtemplates

To use the demo XML templates

1. In the `formats.ini` file, set the following parameter.

```
[nsfsr]
UseDemoTemplate=1
```

2. In the `main.xml` file, uncomment the following section.

```
<ifini name="UseDemoTemplate" text="1">
  <call file="demo.xml"/>
  <quit/>
</ifini>
```

Use Old Templates

For testing purposes, you can extract notes by using legacy templates, which produce MHTML output. You can generate similar output by disabling the XML templates, but using the old templates enables you to see the XML code and compare it to the standard and demo templates.

To use the old XML templates

1. In the `formats.ini` file, set the following parameter.

```
[nsfsr]
UseOldTemplate=1
```

2. In the `main.xml` file, uncomment the following section.

```
<ifini name="UseOldTemplate" text="1">
  <call file="default_old.xml">
  <quit>
</ifini>
```

Disable XML Templates

For testing purposes, you can disable XML templates; KeyView extracts the notes in MHTML format. You can compare the MHTML output directly by the NSF reader with the MHTML output indirectly by the NSF reader through the XML templates.

To disable XML templates

- 1. In the `formats.ini` file, set the following parameter.

```
[nsfsr]  
ExtractByTemplate=0
```

Template Elements and Attributes

This section lists the valid XML elements and attributes that you can use when creating or modifying templates. See the demo templates for examples.

Conditional Elements

The following table lists the valid conditional elements.

Conditional elements

Element	Description
<keyview>	The KeyView XML template container ("root") element
<if*>	<p>If the condition from the comparison is true, process the XML. Conditions can be nested up to 25 levels deep.</p> <p>Attributes</p> <ul style="list-style-type: none">• <code>name</code>. (Required) The name of the main item to compare to <code>item</code> or <code>text</code>.• <code>item</code>. (Required if no <code>text</code>) The name of the item to compare to the item specified by <code>name</code>.• <code>text</code>. (Required if no <code>item</code>) The text to compare to the item specified by <code>name</code>.
<ifex>, <ifnx>	<p>If <code>name</code> item exists and has a <code>text</code> value or not.</p> <p>The Notes item might have a value that cannot be converted to text, such as an image.</p>
<ifeq>, <ifne>, <iflt>, <ifle>, <ifgt>, <ifge>	<p>Respectively, if <code>text</code> ==, !=, <, >, <=, >, >=.</p> <p>Text comparison uses a case-insensitive string compare.</p>
<iftdeq>, <iftdne>, <iftdlt>, <iftdle>, <iftdgt>, <iftdge>	<p>Respectively, if time/date ==, !=, <, >, <=, >, >=.</p> <p>Time/date comparison converts dates to text in local time using the Notes default, <code>TZFMT_NEVER</code>, because Notes also sometimes converts fields to text internally. For example:</p> <p><code>text="06/30/2005 02:52:04 PM"</code></p>

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>	<p>If a name value exists, process XML.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required) The name of the main item to compare in <case> subelements.
<case>	<p>If the comparison condition is true, process XML, then stop processing the rest of <switch>.</p> <p>Attributes</p> <ul style="list-style-type: none"> text. (Required) The text to compare to the name item of <switch>.
<default>	If all <case> conditions were false, process XML. This element must be the last element in <switch>, after all the <case> elements. Any <case> elements after the <default> element are ignored.
<for>	<p>If a name value exists, process XML. Process for each part of the name item.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required) The name of the main item. max. (Optional) The maximum index to process. By default, all are processed.
<index>	Output <for> loop index (1-based). <index> is only valid within a <for> element.

Control Elements

The following table lists the valid control elements.

Control Elements

Element	Description
<call>	<p>Call another XML template. You can nest templates up to 10 levels deep.</p> <p>Attributes</p>

Control Elements, continued

Element	Description
	<ul style="list-style-type: none"> file. (Required) The template file name. This name must be unique.
<log>	<p>Log message to the NSF log file.</p> <p>Attributes</p> <ul style="list-style-type: none"> text. (Required) The text to log. type. (Optional) The type of log message. The following values are valid: <ul style="list-style-type: none"> ERROR WARN INFO DIAG (the default option) DEBUG DUMP
<quit>	<p>Stop processing the template. Exits without error.</p> <p>Attributes</p> <ul style="list-style-type: none"> text. (Optional) The text to log. type. (Optional) The type of log message. See <log>, above.
<stop>	<p>Stop processing the template. Exits with an ERROR log message.</p> <p>Attributes</p> <ul style="list-style-type: none"> text. (Required) The text to log.

Data Elements

The following table lists the valid data elements.

Data elements

Element	Description
<text>	<p>Output text.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.
<rich>	<p>Output rich text (MHTML). Images are output in the next part or parts of the MHTML, after the first <HTML> part.</p>

Data elements, continued

Element	Description
	Attributes <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.
<body>	Output the message body in rich text (MHTML). As with <rich> , on the previous page, images are output in the next part or parts of the MHTML.
<form>	Output the message form (usually \$Body field) in rich text (MHTML). Attributes <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.
<addr>	Output an address. Attributes <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output. type. (Optional) The type of address to output. Set this attribute to CN (Common Name), which is the only supported type.
<name>	Output the name of the last name item, or in other words the current main item. The item must exist.
<format>	Set the default format for <date> and <date_kv>. This element does not set the <text> format. See Date and Time Formats, on page 341 for a list of all Notes and KeyView date and time formats and integer values. Attributes <ul style="list-style-type: none"> format. (Optional. Omit to reset to defaults) The Notes and KeyView date and time format. You can set the following formats: <ul style="list-style-type: none"> TD=int. The Time Date format (TDFMT_*) TS=int. The Time Show format (TSFMT_*) TT=int. The Time Time format (TTFMT_*) TZ=int. The Time Zone format (TZFMT_*) KV=int. The KeyView date and time format <p>where int is an integer value that corresponds to the desired format.</p> <p>Separate multiple formats with commas. For example:</p> <pre>format="TD=0,TS=2,TT=1,TZ=1,KV=55"</pre>
<date>	Output a Notes date. Attributes <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.

Data elements, continued

Element	Description
	<ul style="list-style-type: none"> format. (Optional) See <format>, on the previous page. You can set the following values: <ul style="list-style-type: none"> TD TS TT TZ
<date_kv>	<p>Output a KeyView date.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output. format. (Optional) See <format>, on the previous page. You can set the following values: <ul style="list-style-type: none"> TZ KV
<time>	<p>Output a time range, for example 1 hour, 30 minutes.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The item name of the start date or time. item. (Required) The item name of the end date or time.
<zone>	<p>Output a Notes time zone mnemonic, for example MST.</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of date item to output.
<zone_utc>	<p>Output a time zone as UTC, for example (UTC-06:00).</p>
<logo>	<p>Output the mail header logo.</p> <p>The image link is included in the output; the actual image is output to a different part of the MHTML subfile.</p>
<image>	<p>Output an image.</p> <p>The image link is included in the output; the actual image is output to the MHTML next part, as with <rich>, on page 338 and <body>, on the previous page.</p>
<image_uri>	<p>Output an image URI, in quotation marks. The actual image is output to a different part of the MHTML subfile.</p> <p>Attributes</p>

Data elements, continued

Element	Description
	<ul style="list-style-type: none">• <code>link</code>. (Required if there is no <code>file</code>) The image link, such as a form or title name. For example:<ul style="list-style-type: none">• <code>link="StdNotesLtr0"</code>• <code>file</code>. (Required if there is no <code>link</code>) The name of the image file. The file must exist in the <code>.././templates/images</code> directory. For example:<ul style="list-style-type: none">• <code>file="boxcheck.gif"</code>

Date and Time Formats

This section lists the supported Notes and KeyView date and time formats for use with `<format>`, `<date>`, and `<date_kv>`.

Lotus Notes Date and Time Formats

This section lists supported Lotus Notes date and time formats, and the integer values that specify each one.

Lotus Notes date and time formats

Format	Integer Value	Description
TDFMT_FULL	0	(The Notes default) Year, month, and day
TDFMT_CPARTIAL	1	Month and day, year if not this year
TDFMT_PARTIAL	2	Month and day
TDFMT_DPARTIAL	3	Year and month
TDFMT_FULL4	4	Four-digit year, month, and day
TDFMT_CPARTIAL4	5	Month and day, four-digit year if not this year
TDFMT_DPARTIAL4	6	Four-digit year and month
TTFMT_FULL	0	(Notes default) Hour, minute, and second
TTFMT_PARTIAL	1	Hour and minute
TTFMT_HOUR	2	Hour

Lotus Notes date and time formats, continued

Format	Integer Value	Description
TZFMT_NEVER	0	(Notes default) All time zones are converted to the current time zone
TZFMT_SOMETIMES	1	Show only when outside the current time zone
TZFMT_ALWAYS	2	Show for all time zones
TSFMT_DATE	0	Date
TSFMT_TIME	1	Time
TSFMT_DATETIME	2	(The Notes default) Date and time
TSFMT_CDATETIME	4	Date and time, or time today or time yesterday

KeyView Date and Time Formats

This section lists KeyView date and time formats. The KeyView formats use the following syntax:

- Month

Month = full month name

Mon = abbreviated month name

m = month (number)

mm = two-digit month (leading 0)
- Weekday

Weekday = full weekday name

Wday = abbreviated weekday name
- Year

yy = two-digit year

yyyy = four-digit year
- >Day

d = day (number)

dd = two-digit day (leading 0)
- Time

h = 12-hour

H = 24-hour

m = minutes

s = seconds

P = AM/PM

p = am/pm

Separators _ = space
 c = comma
 s = slash
 a = dash
 o = dot

KeyView date and time formats

Format	Output	Integer Value
12-Hour and 24-Hour Time Formats		
KVDTF_P	P	1
KVDTF_P_hmm	P h:mm	2
KVDTF_hmm_P	h:mm P	3
KVDTF_P_hhmm	P hh:mm	4
KVDTF_hhmm_P	hh:mm P	5
KVDTF_P_hmmss	P h:mm:ss	6
KVDTF_hmmss_P	h:mm:ss P	7
KVDTF_P_hhmmss	P hh:mm:ss	8
KVDTF_hhmmss_P	hh:mm:ss P	9
KVDTF_Hmm	H:mm	10
KVDTF_HHmm	HH:mm	11
KVDTF_mmss	mm:ss	12
KVDTF_Hmmss	H:mm:ss	13
KVDTF_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_ddsmsyy	dd/mm/yy	20
KVDTF_ddsmsyy_Hmm	dd/mm/yy H:mm	21
KVDTF_ddsmm_P_hmm	dd/mm P h:mm	22
KVDTF_ddsmm_hmm_P	dd/mm h:mm P	23
KVDTF_ddsmm_P_hhmm	dd/mm P hh:mm	24
KVDTF_ddsmm_hhmm_P	dd/mm hh:mm P	25
KVDTF_ddsmsyy_P_hmm	dd/mm/yy P h:mm	26
KVDTF_ddsmsyy_hmm_P	dd/mm/yy h:mm P	27
KVDTF_ddsmsyy_P_hmmss	dd/mm/yy P h:mm:ss	28
KVDTF_ddsmsyy_hmmss_P	dd/mm/yy h:mm:ss P	29
KVDTF_ddsmsyy_P_hhmmss	dd/mm/yy P hh:mm:ss	30
KVDTF_ddsmsyy_hhmmss_P	dd/mm/yy hh:mm:ss P	31
KVDTF_yysmmsdd_P_hhmmss	yy/mm/dd P hh:mm:ss	32
KVDTF_yysmmsdd_hhmmss_P	yy/mm/dd hh:mm:ss P	33
KVDTF_msdsyy_Hmm	m/d/yy H:mm	34
KVDTF_mmsddsyy_Hmm	mm/dd/yy H:mm	35
KVDTF_msdsyy_P_hmm	m/d/yy P h:mm	36
KVDTF_msdsyy_hmm_P	m/d/yy h:mm P	37
KVDTF_mmsddsyy_hmm_P	mm/dd/yy h:mm P	38
KVDTF_mmsdd_P_hhmm	mm/dd P hh:mm	39
KVDTF_mmsdd_hhmm_P	mm/dd hh:mm P	40
KVDTF_mmsddsyy_P_hhmmss	mm/dd/yy P hh:mm:ss	41
KVDTF_mmsddsyy_hhmmss_P	mm/dd/yy hh:mm:ss P	42
KVDTF_msd	m/d	43
KVDTF_yysm	yy/m	44
KVDTF_yysmm	yy/mm	45

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_ysmsd	yy/m/d	46
KVDTF_ysmmsdd	yy/mm/dd	47
KVDTF_yysmmsdd	yyyy/mm/dd	48
Numerical Date Formats with Dashes		
KVDTF_ddammayy	dd-mm-yy	49
KVDTF_mmadd	mm-dd	50
KVDTF_mmayy	mm-yy	51
KVDTF_yyammadd	yy-mm-dd	52
KVDTF_yyyymmadd	yyyy-mm-dd	53
KVDTF_yyyymmaddaHHmmss	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 `KVXMLTemplate` structure, `XmlTemplateInfo` class, and template files. They are placeholders for markup that appears in the XML output. For example, the `$CHARSET` token marks the place in the XML output where the name of the source document's character set is inserted. It would be used in the tag `< charset=$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 XML file. Use in the <code><base href=xx></code> tag.
\$CHARSET	Inserts the character set of the source document, if that information is ascertainable. Supported Formats, on page 230 lists the file formats for which character set information can be determined.
\$CONTENT	Inserts the content of the metadata field specified by the <code>\$NAME</code> token. This token is used in conjunction with the <code>\$SUMMARY</code> , <code>\$USERSUMMARY</code> , and <code>\$NAME</code> tokens to insert source document metadata into the XML output. An example of this token's use is: <code>pszUserSummary=<MetaData name="\$NAME" content="\$CONTENT"></code> Supported Formats, on page 230 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 XML, that is, the main output stream.
\$NAME	<p>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 XML output. An example of this token's use is:</p> <pre>pszUserSummary=<MetaData name="\$NAME" content="\$CONTENT"></pre> <p>The section Supported Formats, on page 230 lists file formats that support metadata.</p>
\$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.
\$SUMMARY	<p>Inserts the data from standard metadata fields using the markup provided in the <code>pszUserSummary</code> member of the structure <code>KVXMLTemplate</code>. Standard fields are enumerated from 0 to 33 in <code>KVSumType</code> in <code>kvtypes.h</code>. See the tokens \$USERSUMMARY, on the next page, \$NAME, above, and \$CONTENT, on the previous page.</p> <p>The section Supported Formats, on page 230 lists file formats that support metadata.</p>
\$SUMMARY <i>NN</i>	<p>Inserts the data from a <i>specified</i> metadata field. <i>NN</i> is a number from 0 through 33 enumerated in the <code>KVSumType</code> structure in <code>kvtypes.h</code>. An example of this token's use is:</p> <pre>pszMainTop= <title> \$SUMMARY01 </head> <body></pre> <p>The section Supported Formats, on page 230 lists file formats that support metadata.</p>
\$SPLITBLOCKNUMBER	Inserts the page number for each block generated as a result of <code>bHardPageMakesNewBlock</code> or <code>lcbBlockSize</code> .
\$TOC	Inserts the table of contents at this point in the current output stream. This token is typically embedded in <code>pszMainTop</code> .
\$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 XML 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. XML 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 XML 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 <code>UserCB()</code> and identifies the callback used in the function.
\$USERSUMMARY	<p>Inserts the data from every valid non-standard metadata field using the markup provided in the <code>pszUserSummary</code> member of the <code>KVXMLTemplate</code> structure. Non-standard metadata are any fields not listed from 0 to 33 in <code>KVSumType</code>, 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 347.</p> <p>The section Supported Formats, on page 230 lists file formats that support metadata.</p>
\$XANCHOR	<p>Inserts the anchor to an extra file into the XML output.</p> <p>The contents of the extra file is defined by <code>pszXFile</code>, and the block generated by this token is defined by <code>pszXStartBlock</code> and <code>pszXEndBlock</code>.</p>

Appendix F: File Format Detection

This section describes how file formats are detected in the KeyView Export SDK.

• Introduction	350
• Extract Format Information	350
• Determine Format Support	350
• Translate Format Information	352
• Determine a Document Reader	354
• Category Values in formats_e.ini	354

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 230](#).

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 352](#).

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 352](#).
- The reader associated with each format. See [Determine a Document Reader, on page 354](#).

- 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 `ADDONCINFO` 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 `ADDONCINFO`, 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=1123

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, 1123sr.

[Files Required for Redistribution, on page 398](#) lists the document readers provided with KeyView.

Category Values in `formats_e.ini`

This section lists the possible category values for format information in the `formats_e.ini` file. The corresponding values for the format information extracted from a call to `fpGetStreamInfo()` are listed in the `adinfo.h` header file.

- [Major Formats](#)
- [File Classes](#)
- [Minor Formats](#)

Major Formats

Number	Format	File Class
1	Multiplus (AES)	adWORDPROCESSOR
2	MS-DOS Batch File	adEXECUTABLE
3	APPLIX ASTERIX	adWORDPROCESSOR
4	Windows Bitmap Image (BMP)	adRASTERIMAGE
5	Convergent Technologies DEF Comm. Format	adWORDPROCESSOR
6	Corel Draw	adVECTORGRAPHIC
7	Keyword COM.FILE (KSIF)	

Major Formats, continued

Number	Format	File Class
8	Computer Graphics Metafile (CGM)	adVECTORGRAPHIC
9	Word Connection	adWORDPROCESSOR
10	COMET TOP	adWORDPROCESSOR
11	CEOwrite	adWORDPROCESSOR
12	DSA101 (Honeywell Bull)	adWORDPROCESSOR
13	DCA-RFT (IBM Revisable Form)	adWORDPROCESSOR
14	CDA / DDIF	adWORDPROCESSOR
15	Dummy File (Internal)	
16	DG Common Data Stream (CDS)	adWORDPROCESSOR
17	Dummy Print File (Internal)	
18	Windows Draw (Micrografx)	adVECTORGRAPHIC
19	Vistaword	adWORDPROCESSOR
20	DECdx	adWORDPROCESSOR
21	Enable Word Processing	adWORDPROCESSOR
22	Encapsulated PostScript	AutoDetNoFormat
23	MSDOS/Windows Program	adEXECUTABLE
24	CCITT G3 1D	adRASTERIMAGE
25	Graphics Interchange Format (GIF89a)	adRASTERIMAGE
26	HP Word PC	adWORDPROCESSOR
27	IBM 1403 Line Printer	adWORDPROCESSOR
28	DCF Script	adWORDPROCESSOR
29	DCA-FFT (IBM Final Form)	adWORDPROCESSOR
30	Interleaf	adWORDPROCESSOR
31	GEM Bit Image	adRASTERIMAGE
32	Display Write	adWORDPROCESSOR
33	Sun Raster	adRASTERIMAGE

Major Formats, continued

Number	Format	File Class
34	Keywords PICL	
35	Lotus Ami Pro Style Sheet	adWORDPROCESSOR
36	MORE Database MAC	adOUTLINE
37	Lyrix Word Processing	adWORDPROCESSOR
38	MASS-11	adWORDPROCESSOR
39	MacPaint	adRASTERIMAGE
40	Microsoft Word Mac	adWORDPROCESSOR
41	SmartWare II	adCOMMUNICATION
42	Microsoft Word for Windows	adWORDPROCESSOR
43	MultiMate Advantage II Footnote File	adWORDPROCESSOR
44	Multiplan (Mac)	adSPREADSHEET
45	Microsoft Pocket Word	adWORDPROCESSOR
46	Microsoft Word for PC Miscellaneous File	adWORDPROCESSOR
47	NBI Async Archive Format	adWORDPROCESSOR
48	Navy DIF	adWORDPROCESSOR
49	NBI Net Archive Format	adWORDPROCESSOR
50	NIOS TOP	adWORDPROCESSOR
51	Filemaker MAC	adDATABASE
52	ODA / ODIF	adWORDPROCESSOR
53	OLIDIF (Olivetti)	adWORDPROCESSOR
54	Keyword OSM	
55	Office Writer	adWORDPROCESSOR
56	PC Paintbrush Graphics (PCX)	adRASTERIMAGE
57	CPT	adWORDPROCESSOR
58	Lotus PIC	adVECTORGRAPHIC
59	QuickDraw Picture	AutoDetNoFormat

Major Formats, continued

Number	Format	File Class
60	Philips Script	adWORDPROCESSOR
61	PostScript	adVECTORGRAPHIC
62	PRIMEWORD	adWORDPROCESSOR
63	Q-One V1.93J	adWORDPROCESSOR
64	Q-One V2.0	adWORDPROCESSOR
65	SAMNA Word	adWORDPROCESSOR
66	Lotus Ami Pro Draw	adVECTORGRAPHIC
67	SYLK	adSPREADSHEET
68	SmartWare II	adWORDPROCESSOR
69	Symphony	adSPREADSHEET
70	Targa	adRASTERIMAGE
71	Tag Image File Format (TIFF)	AutoDetNoFormat
72	Targon Word	adWORDPROCESSOR
73	Uniplex Ucalc	adSPREADSHEET
74	Uniplex	adWORDPROCESSOR
75	Microsoft Word UNIX	adWORDPROCESSOR
76	WANG PC	adWORDPROCESSOR
77	WordERA	adWORDPROCESSOR
78	WANG WPS	adWORDPROCESSOR
79	WordPerfect MAC	adWORDPROCESSOR
80	WordPerfect 5.2	adWORDPROCESSOR
81	Lotus 1-2-3 Release 9	adSPREADSHEET
82	WordMARC	adWORDPROCESSOR
83	Windows Metafile (no header)	adVECTORGRAPHIC
84	SmartWare II	adDATABASE
85	WordPerfect Graphics V1.0 (WPG)	adRASTERIMAGE

Major Formats, continued

Number	Format	File Class
86	WordPerfect	adWORDPROCESSOR
87	WordStar	adWORDPROCESSOR
88	WANG WITA	adWORDPROCESSOR
89	Xerox 860	adWORDPROCESSOR
90	Microsoft Excel	adSPREADSHEET
91	Xerox Writer	adWORDPROCESSOR
92	Data Interchange Format (DIF)	adSPREADSHEET
93	Enable Spreadsheet	adSPREADSHEET
94	Supercalc	adSPREADSHEET
95	UltraCalc	adSPREADSHEET
96	SmartWare II	adSPREADSHEET
97	Serialized Object Format (SOF)	adENCAPSULATION
98	Microsoft PowerPoint PC	adPRESENTATION
99	Microsoft PowerPoint MAC	adPRESENTATION
100	PageMaker for Macintosh	adDESKTOPPUBLSH
101	PageMaker for Windows	adDESKTOPPUBLSH
103	Microsoft Works Word Processor for MAC	adWORDPROCESSOR
104	Microsoft Works Database for MAC	adDATABASE
105	Microsoft Works Spreadsheet for MAC	adSPREADSHEET
106	Microsoft Works Communication for MAC	adCOMMUNICATION
107	Microsoft Works Word Processor for DOS	adWORDPROCESSOR
108	Microsoft Works Database for DOS	adDATABASE
109	Microsoft Works Spreadsheet for DOS	adSPREADSHEET
111	DOS/Windows Object Library	adLIBRARY
112	MacWrite	adWORDPROCESSOR
113	MacWrite II	adWORDPROCESSOR

Major Formats, continued

Number	Format	File Class
114	Freehand MAC	adVECTORGRAPHIC
115	Disk Doubler	adENCAPSULATION
116	HP Graphics Language	adVECTORGRAPHIC
117	Maker Interchange Format (MIF)	adWORDPROCESSOR
118	JPEG Interchange Format	adRASTERIMAGE
119	Reflex	adDATABASE
120	Framework II	adMIXED
121	Paradox	adDATABASE
123	Microsoft Windows Write	adWORDPROCESSOR
124	Quattro Pro for DOS	adSPREADSHEET
126	Persuasion	adPRESENTATION
127	Corel Presentations	adPRESENTATION
128	Windows Icon Format	adRASTERIMAGE
129	Microsoft Project	adSCHEDULE
131	Harvard Graphics Palette	adVECTORGRAPHIC
132	ZIP Archive	AutoDetNoFormat
133	Windows Cursor	adRASTERIMAGE
134	Quark Xpress MAC	adDESKTOPPUBLSH
135	PAK/ARC Archive	adENCAPSULATION
136	FrameMaker	adDESKTOPPUBLSH
137	Microsoft Publisher	adDESKTOPPUBLSH
138	PlanPerfect	adSCHEDULE
139	WordPerfect auxiliary file	adMISC
140	Lotus Freelance 97	adPRESENTATION
141	Microsoft Wave	adSOUND
142	MIDI	adSOUND

Major Formats, continued

Number	Format	File Class
143	AutoCAD DXF	adVECTORGRAPHIC
144	dBase	adDATABASE
145	OS/2 PM Metafile	adVECTORGRAPHIC
146	Lasergraphics Language	adVECTORGRAPHIC
147	AutoShade Rendering	adVECTORGRAPHIC
148	GEM VDI	adVECTORGRAPHIC
149	Windows Help File	adMISC
150	Volkswriter	adWORDPROCESSOR
151	Ability	adRASTERIMAGE
152	XYWrite / Nota Bene	adWORDPROCESSOR
153	CSV (Comma Separated Values)	adSPREADSHEET
154	IBM Writing Assistant	adWORDPROCESSOR
155	WordStar 2000	adWORDPROCESSOR
156	WordStar 6.0	adWORDPROCESSOR
157	HP Printer Control Language	adVECTORGRAPHIC
158	ELF Executable	adEXECUTABLE
159	ELF Relocatable	adOBJECTMODULE
160	ELF Dynamic Library	adLIBRARY
161	NeXT/Sun Audio Data	adSOUND
162	NeWS bitmap font	adFONT
163	cpio archive (CHR Header)	adENCAPSULATION
164	SUN PEX Binary Archive	adENCAPSULATION
165	SUN vfont Definition	adFONT
166	Curses Screen Image	adRASTERIMAGE
167	UU encoded	adENCAPSULATION
168	WriteNow MAC	adWORDPROCESSOR

Major Formats, continued

Number	Format	File Class
169	DOS/Windows Object Module	adOBJECTMODULE
170	Windows Group	adMISC
171	TrueType Font	adFONT
172	Program Information File (PIF)	adMISC
173	PC (.COM)	adEXECUTABLE
174	Maker Markup Language	adDESKTOPPUBLSH
175	StuffIt (MAC)	adENCAPSULATION
176	PeachCalc	adSPREADSHEET
177	WANG Office GDL Header	adENCAPSULATION
178	WordPerfect 6.0	adWORDPROCESSOR
179	Q & A for DOS	adWORDPROCESSOR
180	Q & A for Windows	adWORDPROCESSOR
181	WPS-PLUS	adWORDPROCESSOR
182	DCX FAX Format(PCX images)	adFAXFORMAT
183	OLE Compound Document	adENCAPSULATION
184	Quattro Pro for Windows	adSPREADSHEET
185	Keyword Viewer Markup Format	
186	EBCDIC Text	adWORDPROCESSOR
187	DCS	adWORDPROCESSOR
188	Microsoft Excel 2000	adSPREADSHEET
189	Microsoft Word 95	adWORDPROCESSOR
190	SHAR	adENCAPSULATION
191	Lotus Notes Bitmap	adRASTERIMAGE
192	Unix Compress	adENCAPSULATION
193	Lotus Notes CDF	adWORDPROCESSOR
194	TAR	adENCAPSULATION

Major Formats, continued

Number	Format	File Class
195	WordPerfect Graphics V2.0 (WPG2)	adWORDPROCESSOR
196	ODA / ODIF	adWORDPROCESSOR
197	ALIS	adWORDPROCESSOR
198	GZ Compress	adENCAPSULATION
199	Envoy	adWORDPROCESSOR
200	Portable Document Format	adWORDPROCESSOR
201	KW ODA Internal Raw Bitmap (RBM)	adRASTERIMAGE
202	KW ODA G4 (G4)	adRASTERIMAGE
203	KW ODA G31D (G31)	adRASTERIMAGE
204	KW ODA Internal G32D (G32)	adRASTERIMAGE
205	Microsoft Word for Mac V 4.x/5.x	adWORDPROCESSOR
206	BinHex	adENCAPSULATION
207	SMTP	adENCAPSULATION
208	MIME	adENCAPSULATION
209	SGML	adWORDPROCESSOR
210	Netscape Bookmark File	adWORDPROCESSOR
211	ACT	adWORDPROCESSOR
212	Microsoft PowerPoint 95	adPRESENTATION
213	Portable Network Graphics (PNG)	adRASTERIMAGE
214	Video for Windows (AVI)	adMOVIE
215	Windows Animated Cursor	adRASTERIMAGE
216	Windows C++ Object Storage	adMIXED
217	Windows Palette	adRASTERIMAGE
218	RIFF Device Independent Bitmap	adRASTERIMAGE
219	RIFF MIDI	adSOUND
220	RIFF Multimedia Movie	adMOVIE

Major Formats, continued

Number	Format	File Class
221	MPEG Movie	adMOVIE
222	QuickTime Movie	adMOVIE
223	Audio Interchange File Format (AIFF)	adSOUND
224	Amiga MOD	adSOUND
225	Amiga IFF (8SVX) Sound	adSOUND
226	Creative Voice (VOC)	adSOUND
227	Microsoft Works Word Processor for Windows	adWORDPROCESSOR
228	Microsoft Works Spreadsheet for Windows	adSPREADSHEET
229	AutoDesk Animator FLIC	adANIMATION
230	AutoDesk Animator Pro FLIC	adANIMATION
231	Microsoft Works Database for Windows	adDATABASE
232	Microsoft Works Communication (Windows)	adCOMMUNICATION
233	Compactor / Compact Pro	adENCAPSULATION
234	VRML	adVECTORGRAPHIC
235	QuickDraw 3D Metafile	adVECTORGRAPHIC
236	PGP Secret Keyring	adENCAPSULATION
237	PGP Public Keyring	adENCAPSULATION
238	PGP Encrypted Data	adENCAPSULATION
239	PGP Signed Data	adENCAPSULATION
240	PGP Signed and Encrypted Data	adENCAPSULATION
241	PGP Signature Certificate	adENCAPSULATION
242	ASCII-armored PGP Public Keyring	adENCAPSULATION
243	ASCII-armored PGP encoded	adENCAPSULATION
244	ASCII-armored PGP signed	adENCAPSULATION
245	OLE DIB object	adRASTERIMAGE
246	PGP Compressed Data	adENCAPSULATION

Major Formats, continued

Number	Format	File Class
247	SGI Image	adRASTERIMAGE
248	Lotus ScreenCam	adANIMATION
249	MPEG Audio	adSOUND
250	FTP Session Data	adCOMMUNICATION
251	Netscape Bookmark file	adWORDPROCESSOR
252	Corel CMX	adVECTORGRAPHIC
253	AutoDesk Drawing (DWG)	adVECTORGRAPHIC
254	AutoDesk WHIP	adVECTORGRAPHIC
255	Macromedia Director	adANIMATION
256	Real Audio	adSOUND
257	MSDOS Device Driver	adEXECUTABLE
258	Micrografx Designer	adVECTORGRAPHIC
259	Simple Vector Format (SVF)	adVECTORGRAPHIC
260	WordPerfect Office document (WPD)	
261	Applix Words	adWORDPROCESSOR
262	Applix Graphics	adPRESENTATION
263	Microsoft Access 2000	adDATABASE
264	USENET	adWORDPROCESSOR
265	MacBinary	adENCAPSULATION
266	Apple Single	adENCAPSULATION
267	Apple Double	adENCAPSULATION
268	Lotus Word Pro 97	adWORDPROCESSOR
269	Microsoft Word 2000	adWORDPROCESSOR
270	Enhanced Metafile	adVECTORGRAPHIC
271	Microsoft Office Drawing	adVECTORGRAPHIC
272	Microsoft PowerPoint 2000	adPRESENTATION

Major Formats, continued

Number	Format	File Class
273	Extended or Custom XML	adWORDPROCESSOR
274	DeVice Independent file (DVI)	adVECTORGRAPHIC
275	Unicode	adWORDPROCESSOR
276	Framework	adMIXED
277	KPIF Chart Stream	
278	Applix Spreadsheets	adSPREADSHEET
279	Microsoft Device Independent Bitmap	adRASTERIMAGE
280	KeyView GPF Filter	
281	Microsoft Project 2000	adSCHEDULE
282	Folio Flat File	adWORDPROCESSOR
283	HWP(Arae-Ah Hangul)	adWORDPROCESSOR
284	ICHITARO	adWORDPROCESSOR
285	Microsoft Visio 2003 XML	adWORDPROCESSOR
286	Oasys	adWORDPROCESSOR
287	Portable Bitmap Utilities BINARY format (PBM)	adRASTERIMAGE
288	Portable Greymap Utilities BINARY format (PGM)	adRASTERIMAGE
289	Portable Pixmap Utilities BINARY format (PPM)	adRASTERIMAGE
290	X Bitmap format (XBM)	adRASTERIMAGE
291	X Pixmap format (XPM)	adRASTERIMAGE
292	FlashPix FPX Image format	adRASTERIMAGE
293	PCD Image format	adRASTERIMAGE
294	Microsoft Visio	adPRESENTATION
295	Microsoft Outlook	adENCAPSULATION
296	XHTML	adWORDPROCESSOR
297	Microsoft Outlook Personal Folders File (.pst)	adENCAPSULATION
298	RAR	adENCAPSULATION

Major Formats, continued

Number	Format	File Class
299	IBM Lotus Notes Database NSF/NTF	adENCAPSULATION
300	Macromedia Flash (.swf)	adWORDPROCESSOR
301	Microsoft Word 2007 XML - Flat xml	adWORDPROCESSOR
302	Microsoft Excel 2007 XML	adSPREADSHEET
303	Microsoft PowerPoint 2007 XML	adPRESENTATION
304	OpenPGP Message Format (with new packet format)	adENCAPSULATION
305	Intergraph Standard File Format (ISFF) V7 DGN (non-OLE)	adVECTORGRAPHIC
306	MicroStation V8 DGN (OLE)	adVECTORGRAPHIC
307	Microsoft Word Macro 2007 XML	adWORDPROCESSOR
308	Microsoft Excel Macro 2007 XML	adSPREADSHEET
309	Microsoft PPT Macro 2007 XML	adPRESENTATION
310	LZH Archive	adENCAPSULATION
311	Office 2007 document	adMISC
312	Microsoft XML Paper Specification (XPS)	adWORDPROCESSOR
313	IBM Domino Data in XML format (.dxi)	adENCAPSULATION
314	ODF Text Template	adWORDPROCESSOR
315	ODF Spreadsheet Template	adSPREADSHEET
316	ODF Presentation Template	adPRESENTATION
317	Legato Extender Native Message ONM	adENCAPSULATION
318	Bin unknown format (.xxx)	adWORDPROCESSOR
319	Transport Neutral Encapsulation Format (TNEF)	adENCAPSULATION
320	CADAM Drawing	adVECTORGRAPHIC
321	CADAM Drawing Overlay	adVECTORGRAPHIC
322	NURSTOR Drawing	adVECTORGRAPHIC
323	HP Graphics Language (Plotter)	adVECTORGRAPHIC
324	Advanced Systems Format (ASF)	adMISC

Major Formats, continued

Number	Format	File Class
325	Windows Media Audio Format (WMA)	adSOUND
326	Windows Media Video Format (WMV)	adMOVIE
327	Legato EMailXtender Archives Format (EMX)	adENCAPSULATION
328	7 Zip Format (7z)	adENCAPSULATION
329	Microsoft Excel Binary 2007	adSPREADSHEET
330	Microsoft Cabinet File (CAB)	adENCAPSULATION
331	CATIA Formats (CAT*)	adVECTORGRAPHIC
332	Yahoo Instant Messenger History	adWORDPROCESSOR
333	Founder Chinese E-paper Basic (ceb)	adWORDPROCESSOR
334	Corel Quattro Pro 9+ for Windows	adSPREADSHEET
335	MHTML format (MHT)	adWORDPROCESSOR
336	Microsoft Document Imaging Format	adRASTERIMAGE
337	Microsoft Office Groove Format	adWORDPROCESSOR
338	Apple iWork Pages format	adWORDPROCESSOR
339	Apple iWork Numbers format	adSPREADSHEET
340	Apple iWork Keynote format	adPRESENTATION
341	Windows Backup File	adENCAPSULATION
342	Microsoft Access 2007	adDATABASE
343	Microsoft Entourage Database Format	adENCAPSULATION
344	Mac Disk Copy Disk Image File	adENCAPSULATION
345	AppleWorks File	adWORDPROCESSOR
346	Omni Outliner V3 File	adWORDPROCESSOR
347	Omni Outliner OPML File	adWORDPROCESSOR
348	Omni Graffle XML File	adVECTORGRAPHIC
349	Photoshop Document	adRASTERIMAGE
350	Apple Binary Property List format	adMISC

Major Formats, continued

Number	Format	File Class
351	Apple iChat format	adWORDPROCESSOR
352	OOutliner File	adWORDPROCESSOR
353	Bzip 2 Compressed File	adENCAPSULATION
354	ISO-9660 CD Disc Image Format	adENCAPSULATION
355	DocuWorks Format	adWORDPROCESSOR
356	RealMedia Streaming Media	adMOVIE
357	AC3 Audio File Format	adSOUND
358	Nero Encrypted File	adENCAPSULATION
359	SolidWorks Format Files	adVECTORGRAPHIC
360	I-DEAS Format	adVECTORGRAPHIC
361	I-DEAS Drawing Format	adVECTORGRAPHIC
362	Unigraphics (UG) NX CAD Format	adVECTORGRAPHIC
363	UGS Jupiter Tessellation file	adCAD
364	3D Systems STL ASCII format	adMISC
365	Parasolid XT	adVECTORGRAPHIC
366	Extensible Forms Description Language	adPRESENTATION
367	Apple XML Property List format	adMISC
368	OneNote Note Format	adPRESENTATION
369	iFilter	adWORDPROCESSOR
370	Digital Imaging and Communications in Medicine (Dicom)	adRASTERIMAGE
371	Expert Witness Compression Format (EnCase)	adENCAPSULATION
372	Shell Scrap Object File	adENCAPSULATION
373	Microsoft Project 2007	adSCHEDULE
374	Microsoft Publisher from version 98	adDESKTOPPUBLSH
375	Skype Log File	adWORDPROCESSOR
376	Lotus Notes Bitmap Format (DXL embedded images)	adRASTERIMAGE

Major Formats, continued

Number	Format	File Class
377	Health level7 message	adWORDPROCESSOR
378	Microsoft Outlook Offline Folders File (OST)	adENCAPSULATION
379	Electronic Publication	adWORDPROCESSOR
380	Microsoft Outlook Express DBX Message Database	adENCAPSULATION
381	BlackBerry Activation File	adWORDPROCESSOR
382	Disk Image	adENCAPSULATION
383	Milestone Document	adRASTERIMAGE
384	RealLegal E-Transcript File	adWORDPROCESSOR
385	PostScript Type 1 Font	adFONT
386	Ghost Disk Image File	adENCAPSULATION
387	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	adRASTERIMAGE
388	Unicode HTML	adWORDPROCESSOR
389	Microsoft Compiled HTML Help	adENCAPSULATION
390	Documentum EMCDF format	adENCAPSULATION
391	Microsoft Access 2007 Template	adDATABASE
392	Samsung Electronics Jungum Global document	adWORDPROCESSOR
393	JBIG2 File Format	adRASTERIMAGE
394	eFax file	adRASTERIMAGE
395	AD1 Evidence file	adENCAPSULATION
396	Google SketchUp	adVECTORGRAPHIC
397	Group Wise File Surf email	adENCAPSULATION
398	Windows Journal format	adWORDPROCESSOR
399	Yahoo! Messenger chat log	adWORDPROCESSOR
400	PaperPort MAX image file	adRASTERIMAGE
402	ARJ (Archive by Robert Jung) file format	adENCAPSULATION
403	Microsoft Outlook Restricted Permission Message	adENCAPSULATION

Major Formats, continued

Number	Format	File Class
404	MATLAB file format	adWORDPROCESSOR
405	SEG-Y Seismic Data format	adWORDPROCESSOR
406	MPEG-PS container with CDXA stream	adMOVIE
407	Microsoft Windows NT Event Log	adMISC
408	Microsoft Windows Vista Event Log	adMISC
409	Microsoft Outlook for Macintosh format	adENCAPSULATION
410	Web ARChive	adENCAPSULATION
411	Java Class format	adWORDPROCESSOR
412	Microsoft Outlook vCard file format	adWORDPROCESSOR
413	Microsoft Exchange Server Database file format	adENCAPSULATION
414	Microsoft Outlook iCalendar file format	adENCAPSULATION
415	MS Visio 2013 template Macro format	adPRESENTATION
417	ICHITARO Compressed format	adWORDPROCESSOR
418	Apple iWork 2013 Pages format	adWORDPROCESSOR
419	Apple iWork 2013 Numbers format	adSPREADSHEET
420	Apple iWork 2013 Keynote format	adPRESENTATION
421	XZ archive format	adENCAPSULATION
422	Sony Wave64 format	adSOUND
423	Conifer Wavpack format	adSOUND
424	Xiph Ogg Vorbis format	adSOUND
425	Borland Reflex 2 format	adDATABASE
426	PKCS #12 (p12) format	adWORDPROCESSOR
427	B1 format	adENCAPSULATION
428	ISO/IEC MPEG-4 (ISO 14496) format	adMOVIE
429	RAR5 Format	adENCAPSULATION
430	PTC Creo CAD Format	adVECTORGRAPHIC

Major Formats, continued

Number	Format	File Class
431	Keyhole Markup Language	adWORDPROCESSOR
432	Zipped Keyhole Markup Language	adWORDPROCESSOR
433	Wireless Markup Language	adWORDPROCESSOR
434	ODF Formula	adWORDPROCESSOR
435	Star Office Writer Text	adWORDPROCESSOR
436	Star Office Calc Spreadsheet	adSPREADSHEET
437	Star Office Impress Presentation	adPRESENTATION
438	Star Office Math	adMISC
439	ISO 10303-21 STEP format	adMISC
440	AppleScript Source Code	adSOURCECODE
441	Assembly Code	adSOURCECODE
442	C Source Code	adSOURCECODE
443	C# Source Code	adSOURCECODE
444	C++ Source Code	adSOURCECODE
445	Cascading Style Sheet	adSOURCECODE
446	Clojure Source Code	adSOURCECODE
447	CoffeeScript Source Code	adSOURCECODE
448	Common Lisp Source Code	adSOURCECODE
449	Dockerfile	adSOURCECODE
450	Eiffel Source Code	adSOURCECODE
451	Erlang Source Code	adSOURCECODE
452	F# Source Code	adSOURCECODE
453	Fortran Source Code	adSOURCECODE
454	Go Source Code	adSOURCECODE
455	Groovy Source Code	adSOURCECODE
456	Haskell Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
457	Initialization (INI) file	adSOURCECODE
458	Java Source Code	adSOURCECODE
459	Javascript Source Code	adSOURCECODE
460	Lua Source Code	adSOURCECODE
461	Makefile	adSOURCECODE
462	Wolfram Mathematica Source Code	adSOURCECODE
463	Matlab Source Code	adSOURCECODE
464	Objective-C Source Code	adSOURCECODE
465	Objective-C++ Source Code	adSOURCECODE
466	Objective-J Source Code	adSOURCECODE
467	PHP Source Code	adSOURCECODE
468	PLSQL Source Code	adSOURCECODE
469	Pascal Source Code	adSOURCECODE
470	Perl Source Code	adSOURCECODE
471	PowerShell Source Code	adSOURCECODE
472	Prolog Source Code	adSOURCECODE
473	Puppet Source Code	adSOURCECODE
474	Python Source Code	adSOURCECODE
475	R Source Code	adSOURCECODE
476	Ruby Source Code	adSOURCECODE
477	Rust Source Code	adSOURCECODE
478	Scala Source Code	adSOURCECODE
479	Shell Script	adSOURCECODE
480	Smalltalk Source Code	adSOURCECODE
481	Standard ML Source Code	adSOURCECODE
482	Swift Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
483	Tool Command Language (Tcl) Source Code	adSOURCECODE
484	TeX Typesetting File	adSOURCECODE
485	TypeScript Source Code	adSOURCECODE
486	Verilog Source Code	adSOURCECODE
487	YAML File	adSOURCECODE
488	MediaWiki File	adWORDPROCESSOR
489	Matroska video File	adMOVIE
490	Scalable Vector Graphics image	adVECTORGRAPHIC
491	Shapefile	adMISC
492	Flash video File	adMOVIE
493	Embedded OpenType font	adFONT
494	Web Open Font Format	adFONT
495	OpenType Font	adFONT
496	Multiple-image Network Graphics	adANIMATION
497	JPEG Network Graphics	adRASTERIMAGE
498	AppleScript Binary Source Code	adSOURCECODE
499	Autodesk Maya binary file	adCAD
500	Ogg Theora Video format	adMOVIE
501	General Ogg Container format	adMISC
502	GNU Message Catalog format	adMISC
503	Windows shortcut file	adMISC
504	Apple/NeXT typedstream data format	adMISC
505	GIMP XCF image	adRASTERIMAGE
506	PaintShop Pro image	adRASTERIMAGE
507	SQLite database format	adDATABASE
508	MySQL table definition file	adDATABASE

Major Formats, continued

Number	Format	File Class
509	Microsoft Program Database format	adDATABASE
510	OpenEXR image format	adRASTERIMAGE
511	4X Movie File	adMOVIE
512	AMV video file	adMOVIE
513	Notation Interchange File Format	adSOUND
514	Steinberg CuBase file	adSOUND
515	SoundFont file	adSOUND
516	WebP image	adRASTERIMAGE
517	International Color Consortium files	adMISC
518	X11 Portable Compiled Font file	adFONT
519	WebM video file	adMOVIE
520	Amiga Metafile	adVECTORGRAPHIC
521	IFF Animated Bitmap	adRASTERIMAGE
522	IFF Amiga animated raster graphics format	adRASTERIMAGE
523	IFF-DEEP TVPaint image	adRASTERIMAGE
524	IFF-FAXX Facsimile image	adRASTERIMAGE
525	IFF Glow Icon image	adRASTERIMAGE
526	Interleaved BitMap image	adRASTERIMAGE
527	LightWave Object format	adMISC
528	IFF-MAUD MacroSystem audio format	adSOUND
529	IFF Planar BitMap	adRASTERIMAGE
530	IFF TDDD and Imagine Object animation format	adRASTERIMAGE
531	AT&T DjVu format	adWORDPROCESSOR
532	Adobe InDesign document	adDESKTOPPUBLSH
533	Calamus Desktop Publishing	adDESKTOPPUBLSH
534	Adaptive Multi-Rate audio format	adSOUND

Major Formats, continued

Number	Format	File Class
535	Free Lossless Audio Codec format	adSOUND
536	Ogg Container FLAC audio format	adSOUND
537	SAS7BDAT database storage format	adDATABASE
538	Autodesk Design Web Format	adCAD
539	Adobe Flash Player audio book	adSOUND
540	Adobe Flash Player audio	adSOUND
541	Adobe Flash Player protected video	adMOVIE
542	Adobe Flash Player video	adMOVIE
543	Audible Enhanced Audiobook	adSOUND
544	Canon Digital Camera image	adRASTERIMAGE
545	Canon Raw image	adRASTERIMAGE
546	Casio Digital Camera image	adRASTERIMAGE
547	Convergent Design file	adRASTERIMAGE
548	DMB MAF audio	adSOUND
549	DMB MAF video	adMOVIE
550	Digital Media Project Content Format	adMISC
551	Digital Video Broadcast format	adMOVIE
552	ISO-BMFF Dirac Wavelet compression	adMISC
553	High Efficiency Image Format HEVC image sequence	adRASTERIMAGE
554	High Efficiency Image Format HEVC image	adRASTERIMAGE
555	High Efficiency Image Format image sequence	adRASTERIMAGE
556	High Efficiency Image Format image	adRASTERIMAGE
557	ISMACryp 2.0 Encrypted format	adENCAPSULATION
558	3GPP2 video file	adMOVIE
559	3GPP video file	adMOVIE
560	ISO-BMFF JPEG 2000 image	adRASTERIMAGE

Major Formats, continued

Number	Format	File Class
561	ISO-BMFF JPEG 2000 compound image	adRASTERIMAGE
562	ISO-BMFF JPEG 2000 with extensions	adRASTERIMAGE
563	Apple ISO-BMFF QuickTime video	adMOVIE
564	KDDI Video file	adMOVIE
565	MAF Photo Player	adMISC
566	ISO-BMFF MPEG-4 with AVC extension	adMOVIE
567	Apple MPEG-4 Part 14 audio	adSOUND
568	Apple MPEG-4 Part 14 audio book	adSOUND
569	Apple MPEG-4 Part 14 protected audio	adSOUND
570	Apple MPEG-4 Part 14 video	adMOVIE
571	Sony PSP MPEG-4	adSOUND
572	MPEG-21	adMISC
573	Mobile QuickTime video	adMOVIE
574	Motion JPEG 2000	adMOVIE
575	NTT MPEG-4	adMOVIE
576	Nero MPEG-4 profile with AVC extension	adMOVIE
577	Nero AAC audio	adSOUND
578	Nero MPEG-4 profile	adMOVIE
579	OMA DRM Format	adMISC
580	Panasonic Digital Camera image	adRASTERIMAGE
581	Ross video	adMOVIE
582	SDA SD Memory Card video	adMOVIE
583	Samsung stereoscopic stream	adMISC
584	Sony XAVC video	adMOVIE
585	JPEG 2000 PGX Verification Model image	adRASTERIMAGE
586	Apple Desktop Services Store file	adMISC

Major Formats, continued

Number	Format	File Class
587	Apple Core Audio Format	adSOUND
588	VICAR image format	adRASTERIMAGE
589	Flexible Image Transport System FITS image	adRASTERIMAGE
590	Digital Interface Format (DIF) DV video	adMOVIE
591	MPEG Transport Stream data	adMISC
592	MPEG Sequence format	adMISC
593	Ogg OGM video format	adMOVIE
594	Ogg Speex audio format	adSOUND
595	Ogg Opus audio format	adSOUND
596	Musepack audio format	adSOUND
597	ART image format	adRASTERIMAGE
598	Vivo audio-video format	adMOVIE
599	Qualcomm QCP audio	adSOUND
600	Creative Signal Processor codec	adMISC
601	NTT TwinVQ audio format	adSOUND
602	Interplay MVE video format	adMOVIE
603	IRIX Silicon Graphics moviemaker video file	adMOVIE
604	Sega FILM video format	adMOVIE
605	Synthetic music Mobile Application Format	adSOUND
606	NIST SPeech HEader Resources format	adSOUND
607	Chinese AVS video format	adMOVIE
608	Westwood Studios Vector Quantized Animation video file	adANIMATION
609	Wildfire YAFA animation	adANIMATION
610	Origin Wing Commander III MVE movie format	adMOVIE
611	BBC Dirac video format	adMOVIE
612	Autodesk Maya ASCII file format	adCAD

Major Formats, continued

Number	Format	File Class
613	Pixar RenderMan Interface Bytestream file	adVECTORGRAPHIC
614	NOFF 3D Object File Format	adVECTORGRAPHIC
615	Visualization Toolkit VTK ASCII format	adVECTORGRAPHIC
616	Visualization Toolkit VTK Binary format	adVECTORGRAPHIC
617	Wolfram Mathematica Computable Document Format	adMISC
618	Wolfram Mathematica Notebook Format	adMISC
619	Hierarchical Data Format HDF4	adMISC
620	Hierarchical Data Format HDF5	adMISC
621	Acom RISC ARMovie video format	adMOVIE
622	Windows Television DVR format	adMOVIE
623	InstallShield Z archive format	adENCAPSULATION
624	Microsoft DirectDraw Surface container format	adENCAPSULATION
625	Bink audio-video container format	adMOVIE
626	LZMA compressed data format	adENCAPSULATION
627	True Audio format	adSOUND
628	Keepass Password file	adMISC
629	RPM Package Manager file	adENCAPSULATION
630	Adobe Printer Font Metrics format	adFONT
631	Adobe Font Metrics ASCII format	adFONT
632	Adobe Printer Font ASCII format	adFONT
633	Netware Loadable Module format	adMISC
634	TCPdump packet stream capture savefile format	adMISC
635	Adobe Multiple master font format	adFONT
636	TrueType font collection format	adFONT
637	Shapefile binary spatial index format	adMISC
638	Java Key Store format	adMISC

Major Formats, continued

Number	Format	File Class
639	Java JCE Key Store format	adMISC
640	QuarkXPress Intel format	adDESKTOPPUBLISH
641	Microsoft Windows Imaging Format WIM	adMISC
642	VMware Virtual Disk Format 5.0	adMISC
643	XPCConnect Typelib Format	adMISC
644	Microsoft MS-DOS installation compression	adENCAPSULATION
645	DLS Downloadable Sounds format	adSOUND
646	Microsoft Windows Registry format	adMISC
647	Microsoft Help 2.0 format	adENCAPSULATION
648	Qt binary translation file format	adMISC
649	PEM-encoded SSL certificate	adENCAPSULATION
650	Adobe PostScript Printer Description file	adMISC
651	Speedo Font format	adFONT
652	InstallShield Cabinet Archive format	adENCAPSULATION
653	InstallShield Uninstall format	adENCAPSULATION
654	Outlook Express DBX folder database format	adENCAPSULATION
655	National Instruments LabVIEW file format	adMISC
656	SAP compression archive SAR format	adENCAPSULATION
657	Netscape Address Book format	adMISC
658	Universal 3D file format	adVECTORGRAPHIC
659	Open Inventor ASCII format	adVECTORGRAPHIC
660	Open Inventor Binary format	adVECTORGRAPHIC
661	X Window Dump image	adRASTERIMAGE
662	Git Packfile format	adENCAPSULATION
663	Xara X Xar image format	adVECTORGRAPHIC
664	Internet Archive ARC format	adENCAPSULATION

Major Formats, continued

Number	Format	File Class
665	Applix Builder format	adMISC
666	Applix Bitmap image format	adRASTERIMAGE
667	PEM-encoded RSA private key	adENCAPSULATION
668	Magick Image File Format	adRASTERIMAGE
669	Subversion Dump format	adENCAPSULATION
670	Microsoft Virtual Hard Disk format	adENCAPSULATION
671	PowerISO Direct Access Archive format	adENCAPSULATION
672	Debian binary package format	adENCAPSULATION
673	Mozilla XUL Fastload format	adMISC
674	Nastran OP2 format	adCAD
675	CAD Binary Logging Format	adCAD
676	CAD Measurement Data Format	adCAD
677	Abaqus ODB Format	adCAD
678	Vector Open Diagnostic Data Exchange format	adCAD
679	Vector CAD ASCII ASC format	adCAD
680	LS-DYNA State Database format	adCAD
681	LS-DYNA binary output (binout) format	adCAD
682	Microsoft Power BI Desktop format	adANALYTICS
683	Tableau Workbook format	adANALYTICS
684	Tableau Packaged Workbook format	adANALYTICS
685	Tableau Extract format	adANALYTICS
686	Tableau Data Source format	adANALYTICS
687	Tableau Packaged Data Source format	adANALYTICS
688	Tableau Preferences format	adANALYTICS
689	Tableau Map Source format	adANALYTICS
690	ABAP Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
691	AMPL Source Code	adSOURCECODE
692	APL Source Code	adSOURCECODE
693	ASN.1 Source Code	adSOURCECODE
694	ATS Source Code	adSOURCECODE
695	Agda Source Code	adSOURCECODE
696	Alloy Source Code	adSOURCECODE
697	Apex Source Code	adSOURCECODE
698	Arduino Source Code	adSOURCECODE
699	AsciiDoc Source Code	adSOURCECODE
700	AspectJ Source Code	adSOURCECODE
701	Awk Source Code	adSOURCECODE
702	BlitzMax Source Code	adSOURCECODE
703	Bluespec Source Code	adSOURCECODE
704	Brainfuck Source Code	adSOURCECODE
705	Brightscript Source Code	adSOURCECODE
706	CLIPS Source Code	adSOURCECODE
707	CMake Source Code	adSOURCECODE
708	COBOL Source Code	adSOURCECODE
709	CWeb Source Code	adSOURCECODE
710	CartoCSS Source Code	adSOURCECODE
711	Ceylon Source Code	adSOURCECODE
712	Chapel Source Code	adSOURCECODE
713	Clarion Source Code	adSOURCECODE
714	Clean Source Code	adSOURCECODE
715	Component Pascal Source Code	adSOURCECODE
716	Cool Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
717	Coq Source Code	adSOURCECODE
718	Creole Source Code	adSOURCECODE
719	Crystal Source Code	adSOURCECODE
720	Csound Source Code	adSOURCECODE
721	Csound Document Source Code	adSOURCECODE
722	Cuda Source Code	adSOURCECODE
723	D Source Code	adSOURCECODE
724	DIGITAL Command Language Source Code	adSOURCECODE
725	DTrace Source Code	adSOURCECODE
726	Dart Source Code	adSOURCECODE
727	E Source Code	adSOURCECODE
728	ECL Source Code	adSOURCECODE
729	Elm Source Code	adSOURCECODE
730	Emacs Lisp Source Code	adSOURCECODE
731	EmberScript Source Code	adSOURCECODE
732	Fantom Source Code	adSOURCECODE
733	Forth Source Code	adSOURCECODE
734	FreeMarker Source Code	adSOURCECODE
735	Frege Source Code	adSOURCECODE
736	G-code Source Code	adSOURCECODE
737	GAMS Source Code	adSOURCECODE
738	GAP Source Code	adSOURCECODE
739	GDScript Source Code	adSOURCECODE
740	GLSL Source Code	adSOURCECODE
741	Game Maker Language Source Code	adSOURCECODE
742	Gnuplot Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
743	Golo Source Code	adSOURCECODE
744	Gosu Source Code	adSOURCECODE
745	Gradle Source Code	adSOURCECODE
746	GraphQL Source Code	adSOURCECODE
747	Graphviz (DOT) Source Code	adSOURCECODE
748	HLSL Source Code	adSOURCECODE
749	Hack Source Code	adSOURCECODE
750	Haml Source Code	adSOURCECODE
751	Handlebars Source Code	adSOURCECODE
752	Hy Source Code	adSOURCECODE
753	IDL Source Code	adSOURCECODE
754	IGOR Pro Source Code	adSOURCECODE
755	Idris Source Code	adSOURCECODE
756	Inform 7 Source Code	adSOURCECODE
757	Ioke Source Code	adSOURCECODE
758	Isabelle Source Code	adSOURCECODE
759	J Source Code	adSOURCECODE
760	JSONiq Source Code	adSOURCECODE
761	JSX Source Code	adSOURCECODE
762	Jasmin Source Code	adSOURCECODE
763	Jolie Source Code	adSOURCECODE
764	Julia Source Code	adSOURCECODE
765	KiCad Layout Source Code	adSOURCECODE
766	KiCad Schematic Source Code	adSOURCECODE
767	Kotlin Source Code	adSOURCECODE
768	LFE Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
769	LOLCODE Source Code	adSOURCECODE
770	Lasso Source Code	adSOURCECODE
771	Limbo Source Code	adSOURCECODE
772	LiveScript Source Code	adSOURCECODE
773	M Source Code	adSOURCECODE
774	MAXScript Source Code	adSOURCECODE
775	Markdown Source Code	adSOURCECODE
776	Max Source Code	adSOURCECODE
777	Mercury Source Code	adSOURCECODE
778	Modelica Source Code	adSOURCECODE
779	Modula-2 Source Code	adSOURCECODE
780	Monkey Source Code	adSOURCECODE
781	Moocode Source Code	adSOURCECODE
782	NL Source Code	adSOURCECODE
783	NSIS Source Code	adSOURCECODE
784	NetLogo Source Code	adSOURCECODE
785	NewLisp Source Code	adSOURCECODE
786	Nginx Source Code	adSOURCECODE
787	Nix Source Code	adSOURCECODE
788	Nu Source Code	adSOURCECODE
789	OCaml Source Code	adSOURCECODE
790	OpenCL Source Code	adSOURCECODE
791	OpenEdge ABL Source Code	adSOURCECODE
792	OpenSCAD Source Code	adSOURCECODE
793	Ox Source Code	adSOURCECODE
794	Oxygene Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
795	Oz Source Code	adSOURCECODE
796	PAWN Source Code	adSOURCECODE
797	PLpgSQL Source Code	adSOURCECODE
798	Pan Source Code	adSOURCECODE
799	Parrot Assembly Source Code	adSOURCECODE
800	PicoLisp Source Code	adSOURCECODE
801	Pike Source Code	adSOURCECODE
802	Pony Source Code	adSOURCECODE
803	Processing Source Code	adSOURCECODE
804	PureBasic Source Code	adSOURCECODE
805	QMake File	adSOURCECODE
806	RAML Source Code	adSOURCECODE
807	RDoc Source Code	adSOURCECODE
808	REXX Source Code	adSOURCECODE
809	Racket Source Code	adSOURCECODE
810	Ragel Source Code	adSOURCECODE
811	Rascal Source Code	adSOURCECODE
812	Rebol Source Code	adSOURCECODE
813	Red Source Code	adSOURCECODE
814	Ren'Py Source Code	adSOURCECODE
815	RenderScript Source Code	adSOURCECODE
816	Ring Source Code	adSOURCECODE
817	RobotFramework Source Code	adSOURCECODE
818	SAS Source Code	adSOURCECODE
819	SPARQL Source Code	adSOURCECODE
820	SQL Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
821	SQLPL Source Code	adSOURCECODE
822	SaltStack Source Code	adSOURCECODE
823	Scheme Source Code	adSOURCECODE
824	Scilab Source Code	adSOURCECODE
825	Squirrel Source Code	adSOURCECODE
826	Stan Source Code	adSOURCECODE
827	Stata Source Code	adSOURCECODE
828	Stylus Source Code	adSOURCECODE
829	SuperCollider Source Code	adSOURCECODE
830	SystemVerilog Source Code	adSOURCECODE
831	TXL Source Code	adSOURCECODE
832	Turing Source Code	adSOURCECODE
833	Turtle Source Code	adSOURCECODE
834	UrWeb Source Code	adSOURCECODE
835	Vim script File	adSOURCECODE
836	Visual Basic Source Code	adSOURCECODE
837	WebAssembly Source Code	adSOURCECODE
838	WebIDL Source Code	adSOURCECODE
839	X10 Source Code	adSOURCECODE
840	XQuery Source Code	adSOURCECODE
841	Xojo Source Code	adSOURCECODE
842	Xtend Source Code	adSOURCECODE
843	YANG Source Code	adSOURCECODE
844	Zephir Source Code	adSOURCECODE
845	eC Source Code	adSOURCECODE
846	reStructuredText Source Code	adSOURCECODE

Major Formats, continued

Number	Format	File Class
847	xBase Source Code	adSOURCECODE
848	MSI Windows Installer format	adENCAPSULATION
849	Autodesk 3ds Max format	adCAD
850	PhotoDraw MIX image	adRASTERIMAGE
851	Softimage Scene SCN format	adCAD
852	Parasolid ascii XT format	adCAD
853	Parasolid binary XB format	adCAD
854	Initial Graphics Exchange Specification format	adCAD
855	ACE archive format	adENCAPSULATION
856	Grasshopper GHX format	adCAD
857	Microsoft FrontPage macro file format	adWORDPROCESSOR
858	Microsoft AtWork Fax format	adFAXFORMAT
859	Microsoft Image Composer format	adRASTERIMAGE
860	Microsoft Visual InterDev web project items file	adMISC
861	Macromedia Flash FLA Project File OLE format	adWORDPROCESSOR
862	CorelDRAW version X4 onwards	adVECTORGRAPHIC
863	Ogg Daala video format	adMOVIE
864	Ogg BBC Dirac video format	adMOVIE
865	PKCS #7 cryptographic format	adWORDPROCESSOR
866	Time-stamped data format	adENCAPSULATION
867	Sereal data serialization format	adMISC
868	Associated Signature Container Simple format	adENCAPSULATION
869	Associated Signature Container Extended format	adENCAPSULATION
870	Apple iBooks format	adWORDPROCESSOR
871	PDF Forms Data Format	adWORDPROCESSOR
872	PDF XML Forms Data Format	adWORDPROCESSOR

Major Formats, continued

Number	Format	File Class
873	AxCrypt encrypted document	adENCAPSULATION
874	Unix Archive ar format	adENCAPSULATION
875	Berkeley DB btree database format	adDATABASE
876	Berkeley DB hash database format	adDATABASE
877	Berkeley DB log database format	adDATABASE
878	Berkeley DB queue database format	adDATABASE
879	BitTorrent file format	adMISC
880	Google Chrome Extension format	adENCAPSULATION
881	Dalvik Executable dex format	adEXECUTABLE
882	Foxmail email format	adWORDPROCESSOR
883	General Regularly-distributed Information in Binary form GRIB format	adMISC
884	Zstandard compression format	adENCAPSULATION
885	LZ4 compressed file	adENCAPSULATION
886	Microsoft Money format	adSPREADSHEET
887	Network Common Data Form NetCDF format	adMISC
888	SAS 6 Data storage format	adDATABASE
889	SAS Transport File XPORT format	adDATABASE
890	Snappy Framed compression format	adENCAPSULATION
891	Stata Data Format	adDATABASE
892	SPSS Statistics Data File Format	adDATABASE
893	Zoo Compressed Archive Format	adENCAPSULATION
894	ChemDraw CDX format	adMISC
895	ChemDraw CDXML format	adMISC
896	Better Portable Graphics BPG format	adRASTERIMAGE
897	Apple Icon image format	adRASTERIMAGE

Major Formats, continued

Number	Format	File Class
898	National Imagery Transmission Format NITF image	adRASTERIMAGE
899	ERDAS Imagine image format	adRASTERIMAGE
900	Microsoft Office temporary owner file	adMISC
901	Enhanced-AC3 (EAC3) Audio File format	adSOUND
902	Common Object File Format (COFF) relocatable object	adOBJECTMODULE
903	Common Object File Format (COFF) executable	adEXECUTABLE
904	Common Object File Format (COFF) dynamic library	adLIBRARY
905	ELF Core file	adMISC
906	Rational Purify data file	adMISC
907	Kryptel encrypted file	adENCAPSULATION
908	Windows heap or mini core dump file	adMISC
909	Qt Prerendered Font format	adFONT
910	AIX/RISC COFF relocatable object	adOBJECTMODULE
911	AIX/RISC COFF executable	adEXECUTABLE
912	AIX/RISC COFF dynamic library	adLIBRARY
913	HPUX/PA-RISC COFF relocatable object	adOBJECTMODULE
914	HPUX/PA-RISC COFF executable	adEXECUTABLE
915	HPUX/PA-RISC COFF dynamic library	adLIBRARY
916	EBCDIC-encoded XML file	adWORDPROCESSOR
917	MPEG JVT-NAL sequence H264 video	adMOVIE
918	Material Exchange Format audio-video container format	adMOVIE
919	Microsoft Agent Character file	adMOVIE
920	Quicken data file	adMISC
921	Microsoft Outlook address file	adMISC
922	Microsoft Answer Wizard file	adMISC
923	ADX audio file	adSOUND

Major Formats, continued

Number	Format	File Class
924	Microsoft System Deployment Image SDI format	adMISC
925	Free Lossless Image Format (FLIF)	adRASTERIMAGE
926	Digital Picture Exchange (DPX) image format	adRASTERIMAGE
927	Apache Avro binary format	adMISC
928	InstallShield archive (early versions) format	adENCAPSULATION
929	Mac OS-X (Mach-O) executable format	adEXECUTABLE
930	GDSII data format	adMISC
931	Microsoft ActiveMime (mso) documents	adMISC
932	BizInt SmartCharts data format	adMISC
933	Webex advanced network ARF recordings	adMOVIE
934	Webex local WRF recordings	adMOVIE
935	Symantec PGP NetShare encrypted file	adENCAPSULATION
936	Ability Write later versions format	adWORDPROCESSOR
937	Ability Spreadsheet later versions format	adSPREADSHEET
938	Adobe InDesign IDML format	adDESKTOPPUBLSH
939	Executable Java Archive (jar) file	adENCAPSULATION
940	IDOL Server IDX file	adENCAPSULATION
941	Android Package Kit (APK) format	adEXECUTABLE
942	Android Binary XML (compressed by aapt) format	adWORDPROCESSOR
943	Java WAR file format	adENCAPSULATION
944	Java EAR file format	adENCAPSULATION
945	Atom Syndication Format	adWORDPROCESSOR
946	RSS syndication XML format	adWORDPROCESSOR
947	Synchronized Multimedia Integration Language (SMIL) XML format	adWORDPROCESSOR
948	Extensible Stylesheet Language Transformations (XSLT) format	adWORDPROCESSOR

Major Formats, continued

Number	Format	File Class
949	XML Shareable Playlist Format (XSPF)	adWORDPROCESSOR
950	FictionBook e-book XML format	adWORDPROCESSOR
951	Adobe Premiere project format	adMISC
952	RDF/XML format	adWORDPROCESSOR
953	Really Simple Discovery (RSD) XML format	adWORDPROCESSOR
954	Systems Biology Markup Language (SBML) XML format	adWORDPROCESSOR
955	Search/Retrieve via URL (SRU) XML format	adWORDPROCESSOR
956	Speech Synthesis Markup Language (SSML) XML format	adWORDPROCESSOR
957	Pronunciation Lexicon Specification (PLS) XML format	adWORDPROCESSOR
958	Text Encoding Initiative (TEI) XML format	adWORDPROCESSOR
959	Metadata Encoding and Transmission Standard (METS) XML format	adWORDPROCESSOR
960	Metadata Object Description Schema (MODS) XML format	adWORDPROCESSOR
961	Metalink XML format	adWORDPROCESSOR
962	Open eBook (OEBPS) XML format	adWORDPROCESSOR
963	Speech Recognition Grammar Specification (SRGS) XML format	adWORDPROCESSOR
964	SPARQL Query Results XML format	adWORDPROCESSOR
965	Adobe XML Data Package format	adWORDPROCESSOR
966	e-Szigno signed xml document	adWORDPROCESSOR
967	Mozilla XML User Interface Language (XUL) XML format	adWORDPROCESSOR
968	Synchronization Markup Language (SyncML) XML format	adWORDPROCESSOR
969	VoiceXML (VXML) XML format	adWORDPROCESSOR
970	Texas Instruments CCXML target configuration XML format	adWORDPROCESSOR
971	Lempel-Ziv Finite State Entropy (LZFSE) compression format	adENCAPSULATION
972	Amazon Kindle or Mobipocket eBook format	adWORDPROCESSOR
973	Open Artwork System Interchange Standard (OASIS) format	adMISC

Major Formats, continued

Number	Format	File Class
974	Amazon KFX eBook format	adWORDPROCESSOR
975	KTX image format	adRASTERIMAGE
976	GMSH Mesh polygon format	adCAD
977	Collada Digital Asset Exchange (DAE) format	adCAD
978	YIN XML format	adWORDPROCESSOR
979	MPEG audio playlist format	adSOUND
980	Windows Audio playlist format	adSOUND
981	DTS Coherent Acoustics audio format	adSOUND
982	Chemical Markup Language (CML) XML format	adWORDPROCESSOR
983	CrystalMaker chemical format	adMISC
984	Visualization Toolkit VTK XML format	adVECTORGRAPHIC
985	IP Flow Information Export (IPFIX) format	adMISC
986	Portable Font Resource font format	adFONT
987	Machine-Readable Cataloging (MARC21) format	adDATABASE
988	Machine-Readable Cataloging (MARC) XML format	adWORDPROCESSOR
989	Extensible Archive (XAR) format	adENCAPSULATION
990	Symbian installer format	adENCAPSULATION
991	OpenDocument format (OpenOffice 1/StarOffice 6.7) Writer Master document XML	adWORDPROCESSOR
992	ODF Chart	adVECTORGRAPHIC
993	ODF Database	adDATABASE
994	ODF Image	adRASTERIMAGE
995	ODF Text Master	adWORDPROCESSOR
996	ODF Text Web	adWORDPROCESSOR
997	ODF Chart Template	adVECTORGRAPHIC
998	ODF Formula Template	adWORDPROCESSOR

Major Formats, continued

Number	Format	File Class
999	ODF Image Template	adRASTERIMAGE
1000	ODF Chart flat XML format	adVECTORGRAPHIC
1001	ODF Drawing/Graphics flat XML format	adWORDPROCESSOR
1002	ODF Formula flat XML format	adVECTORGRAPHIC
1003	ODF Image flat XML format	adRASTERIMAGE
1004	ODF Presentation flat XML format	adPRESENTATION
1005	ODF Spreadsheet flat XML format	adSPREADSHEET
1006	ODF Text flat XML format	adWORDPROCESSOR
1007	ODF Extension format	adMISC
1008	OpenOffice StarView MetaFile format	adRASTERIMAGE
1009	Broad Band eBook (BBEB) in LRF format	adWORDPROCESSOR
1010	GPG trust database format	adMISC
1011	VICE (Versatile Commodore Emulator) format	adMISC
1012	Portable Game Notation chess format	adWORDPROCESSOR
1013	Doom IWAD/PWAD format	adMISC
1014	Linux Device Tree Blob format	adMISC
1015	Glyph Bitmap Distribution Format	adFONT
1016	PC Screen Font format	adFONT
1017	Java Network Launching Protocol	adWORDPROCESSOR
1018	XAML Browser Application (XBAP) format	adWORDPROCESSOR
1019	Microsoft Office Binder format	adENCAPSULATION
1020	Microsoft Silverlight application (XAP) format	adENCAPSULATION
1021	StuffIt X (SITX) archive format	adENCAPSULATION
1022	Facility for Interactive Generation of figures (FIG) image format	adVECTORGRAPHIC
1023	XPIInstall Cross-Platform Installer Module (XPI) format	adENCAPSULATION

Major Formats, continued

Number	Format	File Class
1024	Extensible Data Format (XDF) XML format	adWORDPROCESSOR
1025	MXML UI markup language XML format	adWORDPROCESSOR
1026	MusicXML format	adENCAPSULATION
1027	Finale audio format	adSOUND
1028	TIBCO Spotfire DXP data format	adANALYTICS
1029	Microsoft Office theme format	adMISC
1030	Adobe AIR application installer package	adENCAPSULATION
1031	Adobe Flash Flex project file format	adENCAPSULATION
1032	FoxPro compiled source format	adLIBRARY
1033	Virtual Studio Technology (VST) preset format	adSOUND
1034	Mischief vector graphics image format	adVECTORGRAPHIC
1035	FreeArc archive format	adENCAPSULATION
1036	Autodesk 3ds format	adCAD
1037	Monkey's Audio format	adSOUND
1038	CALS raster image format	adRASTERIMAGE
1039	Dr Halo raster image PAL file format	adRASTERIMAGE
1040	Nintendo DS DPG video format	adMOVIE
1041	JPEG XR (extended range) image format	adRASTERIMAGE
1042	TCR (Text Compression for Reader) eBook format	adWORDPROCESSOR

File Classes

Attribute Number	Description	File class
0	No file class	AutoDetNoFormat
01	Word processor	adWORDPROCESSOR
02	Spreadsheet	adSPREADSHEET
03	Database	adDATABASE

File Classes, continued

Attribute Number	Description	File class
04	Raster image	adRASTERIMAGE
05	Vector graphic	adVECTORGRAPHIC
06	Presentation	adPRESENTATION
07	Executable	adEXECUTABLE
08	Encapsulation	adENCAPSULATION
09	Sound	adSOUND
10	Desktop publishing	adDESKTOPPUBLSH
11	Outline/planning	adOUTLINE
12	Miscellaneous	adMISC
13	Mixed format	adMIXED
14	Font	adFONT
15	Time scheduling	adSCHEDULE
16	Communications	adCOMMUNICATION
17	Object module	adOBJECTMODULE
18	Library module	adLIBRARY
19	Fax	adFAXFORMAT
20	Movie	adMOVIE
21	Animation	adANIMATION
22	Source Code	adSOURCECODE
23	Computer-Aided Design	adCAD
24	BI and analysis tools	adANALYTICS

Minor Formats

Attribute Number	Minor Format
00	Minor format not defined
01	Standard

Minor Formats, continued

Attribute Number	Minor Format
02	Book
03	Chart
04	Macro
05	Text
06	Binary
07	PC
08	Windows
09	DOS
10	Macintosh
11	RGB
12	TIFF
13	IFF
14	Experimental
15	Format Information
16	RLE
17	Symbol
18	Old
19	Footnote
20	Style
21	Palette
22	Configuration
23	Activity
24	Resource
25	Calculation
26	Glossary
27	Spelling

Minor Formats, continued

Attribute Number	Minor Format
28	Thesaurus
29	Hyphenation
30	Miscellaneous
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](#) 398
- [Support Files](#) 399
- [Document Readers and Writers](#) 401
- [Document Type Definition Files](#) 407

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 350 .
htmlexport.	Required by the Java API.
KeyView.jar	Interface for Java support. NOTE: This file can be found at the path <i>install/javaapi/KeyView.jar</i> where <i>install</i> is the Export SDK installation directory.
kpifcnvt.*	Graphic conversion routines.
kpifutil.*	Graphic utility routines.
kvdecrypt.*	Decryption utility functions
kvxtract.*	File Extraction interface.
kvexport.*	Export C API. Interface to the HTML and XML Export C APIs.
kvexportdotnet.*	Interface for .NET support.
kvolefio.*	Embedded OLE object writer.
kvutil.*	Internal KeyView utility functions.

File	Description
kvxpgsa.*	Interface between presentations or graphic readers and the Export API.
kvxml.*	XML Export C API.
kvxssa.*	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.
xmlcnv.*	XML converter for the 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 1123sr.* and kpprzrdr.*.
cbmap.map	Character mappings for Adobe Portable Document Format (PDF).
CEBDLL.*	Required by cebssr.
chartbls.ux	Character mapping tables.
chmdll.*	Required by chmsr.
codeidentifierplugin	Required for source code identification.
DFECore.*	Required by cebssr.

File	Description
Filter.*	Required by <code>cebsr</code> .
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 <code>kvlangdetect.*</code>
libey32.dll	(Windows platforms only) SSL utility functions used by KeyView mail format readers.
libpff.*	Required by <code>pffsr</code> .
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
dx1sr.*	Domino XML Language reader
em1sr.*	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
kpcdrdr.*	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
kpiwpgdrdr.*	Apple iWork Keynote reader
kpjbig2rdr.*	JBIG2 reader
kpjp2000rdr.*	JPEG 2000 reader
kpjpgdrdr.*	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
kpsmordr.*	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
kppicrdr.*	Pictor PC Paint format (PIC) reader

File	Description
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
kptifdr.*	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
kwpgrdr.*	WordPerfect Graphics 1 reader
kpxfd1rdr.*	Extensible Forms Description Language reader
kvgzsr.*	GZIP reader
kvhqxsr.*	BinHex reader
kvzeesr.*	UNIX Compress reader
l123sr.*	Lotus 123 v96/97/98 reader
lasr.*	Lotus AMI Pro reader
1tbenn30.dll	Lotus Word Pro support (supported on Windows x86 platform only)

File	Description
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
mppsрr.*	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

¹This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

File	Description
nsfsrc.*	IBM Notes Database reader ¹
oa2src.*	Fujitsu Oasys reader
odfsssrc.*	Oasis Open Document Format spreadsheets (ODS) reader
odfwpsrc.*	Oasis Open Document Format word processing (ODT) reader
olesrc.*	Embedded OLE object reader.
olmsrc.*	Microsoft Outlook for Macintosh reader
onmsrc.*	Legato EMailXtender Native Message reader
oo3src.*	Omni Outliner reader
pdf2src.*	Alternative Adobe Portable Document Format file (PDF) reader
pdfsrc.*	Adobe Portable Document File (PDF) reader
pfsrc.*	Microsoft Outlook Offline Storage File reader
pngsrc.*	Portable Network Graphics (PNG) reader
pstsrc.dll	Microsoft Outlook Personal Folders file MAPI-based reader (supported on Windows platform only) ²
pstnsrc.*	Microsoft Outlook Personal Folders file native reader ³
qpssrc.*	Quattro Pro spreadsheet reader
rarsrc.*	RAR Archive reader
rtfsrc.*	Microsoft Rich Text Format reader
skypesrc.*	Skype log file reader
sosrc.*	StarOffice/OpenOffice reader
starcsrc.*	StarOffice Calc reader
starwsr.*	StarOffice Writer reader
swfsrc.*	Macromedia Flash reader
tarsrc.*	Tape archive reader

¹This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

²This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

³This reader is an advanced feature and is sold and licensed separately from KeyView Export SDK.

File	Description
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

Document Type Definition Files

The following files related to the `verity.dtd` can be redistributed with your application.

File	Description
Verity.dtd	The document type definition file that defines the structure of an XML document. XML document validity is based on the <code>Verity.dtd</code> . The <code>Verity.dtd</code> is required and must be in the same directory as the output XML file.

File	Description
HTMLlat1x.ent	The file defining Latin characters. This file is referenced in the <code>verity.dtd</code> . This file is required and must be in the same directory as the <code>Verity.dtd</code> .
HTMLspecialx.ent	The file defining special characters. This file is referenced in the <code>verity.dtd</code> . This file is required and must be in the same directory as the <code>Verity.dtd</code> .
HTMLsymbolx.ent	The file defining symbols. This file is referenced in the <code>verity.dtd</code> . This file is required and must be in the same directory as the <code>Verity.dtd</code> .
wp.xsl	The default style sheet for word processing documents. This file is optional and must be in the same directory as the output XML file.
pg.xsl	The default style sheet for presentation graphics. This file is optional and must be in the same directory as the output XML file.
ss.xsl	The default style sheet for spreadsheets. This file is optional and must be in the same directory as the output XML file.

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](#) 409
- [Open Password Protected Container Files](#) 410
- [Export Password Protected Files](#) 410

Supported Password Protected File Types

The following table lists the password-protected file types that KeyView supports.

Key to support table

Symbol	Description
Y	Format is supported.
N	Format is not supported.
S	Support for viewing subfiles.
V	Support for viewing content.
P	Password required.
C	Password and certificate or User ID file required.

Supported password-protected file types

File Type	Version	Filter	Export	Extract	View	Credentials
PST (Windows)	n/a	N	N	Y	S	P
PST (non-Windows) ¹	n/a	N	N	Y	S	N
ZIP	n/a	N	N	Y	S	P
7-Zip	n/a	N	N	Y	S	P
RAR	n/a	N	N	Y	S	P
SMIME in MSG, EML, MBX	n/a	N	N	Y	N	C

¹The native PST reader, `pstnsr`, does not require credentials to open password-protected PST files that use compressible encryption.

Supported password-protected file types, continued

File Type	Version	Filter	Export	Extract	View	Credentials
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.

- **IBM 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 reader (pstnsr) returns the error message `KVERR_PasswordProtected` if a PST is encrypted with High Encryption.

To open container files

1. Define the credential information in the `KVOpenFileArg` data structure.
2. Pass `KVOpenFileArg` to the `fpOpenFile()` function.
3. Call `fpCloseFile()`.

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 `KVXMLConfig()` function with the following arguments :

Argument	Parameter
----------	-----------

<code>nType</code>	<code>KVCFG_SETPASSWORD</code>
--------------------	--------------------------------

Argument	Parameter
----------	-----------

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:

```
(*fpXMLConfig)(pKVXML, KVCFG_SETPASSWORD, TRUE, password);
```

where password is a null-terminated string of 255 or fewer characters.

3. Call the `fpConvertStream()` or `KVXMLConvertFile()` 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 XML Export SDK C Programming Guide (Micro Focus KeyView 12.1)

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!