



Hewlett Packard
Enterprise

KeyView

Software Version: 11.5

Viewing SDK Programming Guide

Document Release Date: October 2017

Software Release Date: October 2017

Legal notices

Warranty

The only warranties for Hewlett Packard Enterprise Development LP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HPE shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted rights legend

Confidential computer software. Valid license from HPE required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright notice

© Copyright 2016-2017 Hewlett Packard Enterprise Development LP

Trademark notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation updates

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

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

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

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

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

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

Support

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

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

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

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

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

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

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

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

Contents

Part I: Overview of Viewing SDK	15
Chapter 1: Introduction to Viewing SDK	16
Overview	16
Features	16
Viewing SDK and Visual Basic	17
Platforms, Compilers, and Dependencies	17
Supported Platforms	17
Supported Compilers	17
Software Dependencies	17
Windows Installation	18
Package Contents	19
License Information	19
Enable Advanced Document Readers	20
Update License Information	20
Directory Structure	21
Chapter 2: Getting Started	23
Before You Begin	23
View Initialization Information	23
Use an Initialization File	23
Viewing API	24
ActiveX Control	24
Use the Windows Registry File	24
Viewing API	25
ActiveX Control	25
Remove Functionality from an Application	25
Deploy Viewing API Applications	26
Deploy ActiveX Control Applications	26
Develop .NET Applications	27
Method and Property Naming Conventions	28
Sample Code	28
Deploy .NET Applications	28
Part II: Viewing API	30
Chapter 3: Use the Viewing API	31
Overview of the Viewing API	31
Create a Viewing API Window	32
Get the Viewer Window of the Document	33
Open and View a Document	33
Notification Messages	34
Save a Document	34

Convert a Document	34
Print a Document	35
Change the Print Job Name	35
Determine the Document Format	36
Extract Document Metadata	36
Change Document Options	36
Annotate, Highlight, or Index a Document	37
Draw a Page	37
Draw a Page into a Supplied Device Context	37
Edit a Document	38
Search for Text	38
Copy Text	38
Modify the Document View	38
Change the Layout of a Document	38
Change the Aspect Ratio of a Document	38
Invert, Rotate, or Magnify a Document	39
Display or Hide Gridlines in a Document	39
Play a Multimedia Document	39
Change the Current Object in a Document	40
View Deleted Items and Document Revision Marks	40
View Container Files	41
Microsoft Outlook Personal Folders (PST) Files	41
Use the Native or MAPI-based Reader	42
Use the Native PST Reader (pstnsr)	43
Use the MAPI-Based Reader (psts)	43
System Requirements	43
Lotus Notes Database (NSF)	43
System Requirements	44
Installation and Configuration	44
Format Notes	44
View Mail Messages and Mail Stores	44
View Archive Files	46
Extract Subfiles to a Viewing Window or Disk	47
Display Subfiles in the Preview Pane	47
Set a Password for a Container File	48
View PDF Documents	48
Use the Acrobat ActiveX Control	48
Use the Microsoft WebBrowser ActiveX Control	48
Use a Graphic-based PDF Reader	49
Use the kppdfrdr Reader	49
Use the kppdf2rdr Reader	50
Specify the Graphic-based Reader	50
View Microsoft Visio Files	51
Extract Microsoft Excel Formulas	51
Chapter 4: Viewing API Sample Programs	54
Overview	54

Compile the Sample Programs	54
Run the Sample Programs	54
Viewing SDK Initialization Information	54
hellovapi	55
Load kvvapi.dll	55
Create the VAPI Window	55
Open a Document	56
hellovapi.c	57
hellovapi.h	61
hellovapi.rc	62
vapidemo	62
mfckv	62
rtfdemo	62
prntdemo	63
filetype	63
ihademo	63
drawdemo	64
uzipdemo	64
Chapter 5: Message Parameters	65
VAPIM_ANNOTATE	65
VAPIM_ENABLEINDEX	66
VAPIM_GETNEXTTEXTBUFFER	67
VAPIM_GETPAGEFROMLOGICAL	68
VAPIM_GETSUMMARYINFO	68
VAPIM_GETTEXT	69
VAPIM_GOTO_PAGE	70
VAPIM_HAVEHILITE	71
VAPIM_POSITION	71
VAPIM_POSITIONHILITE	72
VAPIM_SETCURSOR	73
VAPIM_SETHILITE	73
VAPIM_SETHILITEOPTIONS	74
VAPIM_SETINDEXBUFCHARSET	75
VAPIM_SHOWHITS	76
VAPIM_CONVERT	76
VAPIMWP_CANCONVERT	77
VAPIMWP_DRAW_DRAWPAGE	78
VAPIMWP_DRAW_DRAWTOFILE	79
VAPIMWP_DRAW_GETPAGECOUNT	80
VAPIMWP_DRAW_GETPAGESIZE	81
VAPIMWP_DRAW_GETWORKBOOKPAGECOUNT	82
VAPIMWP_DRAW_INIT	82
VAPIMWP_DRAW_SHUTDOWN	83
VAPIMWP_EDIT_CANCOPY	84
VAPIMWP_EDIT_CANFIND	85
VAPIMWP_EDIT_CANSELECTALL	85

VAPIMWP_EDIT_COPY	86
VAPIMWP_EDIT_FIND	87
VAPIMWP_EDIT_FIND_UNICODE	88
VAPIMWP_EDIT_GETFINDTEXT	88
VAPIMWP_EDIT_SELECTALL	89
VAPIMWP_FILE_CANSAVEAS	90
VAPIMWP_FILE_CANUNZIP	91
VAPIMWP_FILE_CLOSE	91
VAPIMWP_FILE_SAVEAS	92
VAPIMWP_FILE_UNZIP	93
VAPIMWP_INIT_GETCHARSET	93
VAPIMWP_INIT_GETDESCRIP	94
VAPIMWP_INIT_GETDOCCLASS	95
VAPIMWP_INIT_GETDOCFORMAT	96
VAPIMWP_INIT_GETFILENAME	96
VAPIMWP_INIT_GETHWNDVIEWER	97
VAPIMWP_INIT_JUMPTOFIRSTHILIITE	98
VAPIMWP_INIT_OPEN_DOCUMENT	98
VAPIMWP_INIT_SETPASSWORD	100
VAPIMWP_INIT_SETSRCCHARSET	101
VAPIMWP_INIT_SETTRGCHARSET	102
VAPIMWP_MULTIOBJ_CANMULTIOBJ	102
VAPIMWP_MULTIOBJ_CANNEXTOBJ	103
VAPIMWP_MULTIOBJ_CANPREVOBJ	104
VAPIMWP_MULTIOBJ_CANSETCURRENTOBJ	105
VAPIMWP_MULTIOBJ_GETOBJCOUNT	105
VAPIMWP_MULTIOBJ_NEXTOBJ	106
VAPIMWP_MULTIOBJ_OBJNAME	107
VAPIMWP_MULTIOBJ_PREVOBJ	108
VAPIMWP_MULTIOBJ_SETCURRENTOBJ	108
VAPIMWP_OPTIONS_GETOPTIONS_EX	109
VAPIMWP_OPTIONS_SETOPTIONS_EX	110
VAPIMWP_PRINT_ANNOTATIONS	111
VAPIMWP_PRINT_CANPRINT	112
VAPIMWP_PRINT_PAGESETUP	112
VAPIMWP_PRINT_PRINT	113
VAPIMWP_PRINT_PRINTHEADER	114
VAPIMWP_PRINT_PRINTSETUP	115
VAPIMWP_PRINT_PRINTTOPD	115
VAPIMWP_PRINT_PRINTTOPPRINTER	116
VAPIMWP_PRINT_SETPRINTNAME	117
VAPIMWP_VIEW_CANASPECTRATIO	117
VAPIMWP_VIEW_CANDECREASEFONT	118
VAPIMWP_VIEW_CANFITTOWINDOW	119
VAPIMWP_VIEW_CANGOTO	120
VAPIMWP_VIEW_CANGRIDLINES	120

VAPIMWP_VIEW_CANINCREASEFONT	121
VAPIMWP_VIEW_CANINVERT	122
VAPIMWP_VIEW_CANLAYOUT	123
VAPIMWP_VIEW_CANMAGNIFY	124
VAPIMWP_VIEW_CANPAUSE	125
VAPIMWP_VIEW_CANPLAY	125
VAPIMWP_VIEW_CANPREVIEWPANE	126
VAPIMWP_VIEW_CANROTATE	127
VAPIMWP_VIEW_CANSTOP	128
VAPIMWP_VIEW_DECREASEFONT	129
VAPIMWP_VIEW_END	129
VAPIMWP_VIEW_GETASPECTRATIO	130
VAPIMWP_VIEW_GETGRIDLINES	131
VAPIMWP_VIEW_GETINVERT	132
VAPIMWP_VIEW_GETLAYOUT	132
VAPIMWP_VIEW_GETMAGNIFY	133
VAPIMWP_VIEW_GETPLAYMODE	134
VAPIMWP_VIEW_GETPREVIEWPANE	135
VAPIMWP_VIEW_GETROTATE	136
VAPIMWP_VIEW_GOTOPAGE	136
VAPIMWP_VIEW_INCREASEFONT	137
VAPIMWP_VIEW_LOOP	138
VAPIMWP_VIEW_PAUSE	139
VAPIMWP_VIEW_PLAY	139
VAPIMWP_VIEW_SETASPECTRATIO	140
VAPIMWP_VIEW_SETGRIDLINES	141
VAPIMWP_VIEW_SETINVERT	141
VAPIMWP_VIEW_SETLAYOUT	142
VAPIMWP_VIEW_SETMAGNIFY	143
VAPIMWP_VIEW_SETPREVIEWPANE	144
VAPIMWP_VIEW_SETROTATE	144
VAPIMWP_VIEW_STOP	145
Chapter 6: Notification Message Parameters	146
VAPINM_ANNOTATION_HIT	146
VAPINM_EXTENT	147
VAPINM_SELECTION	148
VAPINM_TEXTBUFFER	148
VAPINM_USERCLICK	150
VAPINM_VIEW_FILE	150
VAPINMWP_INIT_DISABLEUI	151
VAPINMWP_INIT_DOCTYPE	152
VAPINMWP_INIT_GETTEMPFILEPATH	152
VAPINMWP_INIT_OPENDOCDONE	153
VAPINMWP_INIT_PAGENUMBER	154
VAPINMWP_MULTIOBJ_OBJNAME	154
VAPINMWP_OPTIONS_GETOPTIONS_EX	155

VAPINMWP_PRINT_PRINTDONE	156
Chapter 7: Structures	157
ADDOCINFO	157
ALL_OPTIONS_EX	158
KPTPIOobj	159
KVSumInfoElemEx	160
KVSummaryInfoEx	160
TPVAPIAnnotation	161
TPVAPIConvert	162
TPVAPICreateParams	163
TPVAPIDrawFileInfo	164
TPVAPIDrawPageInfo	166
TPVAPIExtract	167
TPVAPIFindInfo	167
TPVAPIFirstLast	168
TPVAPIGetText	169
TPVAPIHiLiteColor	169
TPVAPIHiLiteOptions	170
TPVAPIOpenDocumentInfo	170
TPVAPIPageSize	174
TPVAPIPosition	175
TPVAPITextInfo	175
Part III: Viewing ActiveX Control	177
Chapter 8: Use the Viewing ActiveX Control	178
Overview of the Viewing ActiveX Control	178
Open and View a Document	179
Save a Document	179
Convert a Document	180
Print a Document	180
Determine the Document Format	181
Extract Document Metadata	181
Search for Text in a Document	181
Copy a Selected Area of Text	181
Copy all the Text in a Document	182
Create a Thumbnail Image of a Document Page	182
Filter a Document	182
Highlight Text in a Document	182
Annotate Text in a Document	183
Chapter 9: Control Sample Programs	184
Viewing SDK Initialization Information	184
fileview	184
Create a New Visual Basic Project 6.0	184
Draw the Controls	185
Set Objects and Properties	185

Create Event Procedures	186
dotnetview	187
Chapter 10: Control Methods	188
Annotate	189
ChangeObject	190
Close	190
Convert	191
Copy	192
DecreaseFont	193
DrawToFile	193
Find	195
GetNextTextBuffer	195
GetPageFromLogical	196
GetSelectedText	197
GetSummaryInfo	197
GetText	198
GoToPage	199
IncreaseFont	199
Open	200
Play	201
Position	202
PositionHiLite	202
PrintDlg	203
PrintOut	204
PrintOutEx	204
PrintPageSetup	205
SaveAs	206
SelectAll	207
SetCursor	207
SetFocusViewer	208
SetHiLite	208
SetHiLiteOptions	209
SetPassword	210
SetPrintName	210
ShowHits	211
UnZip	212
UnZipEx	212
Chapter 11: Control Properties	214
Introduction	216
Persistent Properties	216
Property Naming Conventions in .NET	216
"OPEN" Properties	216
ASCIICharSet	216
ASCIIFilterNonPrintable	217
ASCIIFontName	217
ASCIIFontSize	217

ASCIIFontStyle	218
ASCIIMarginBottom	218
ASCIIMarginLeft	219
ASCIIMarginRight	219
ASCIIMarginTop	219
ASCIIPrintLandscape	220
AspectRatio	220
CanCopy	220
CanDecreaseFont	221
CanFind	221
CanIncreaseFont	222
CanMultiObj	222
CanNextObj	222
CanPause	223
CanPlay	223
CanPrevObj	224
CanPrint	224
CanSaveAs	225
CanSelectAll	225
CanStop	225
CanUnZip	226
CanViewPane	226
CharSet	226
ContextMenu	227
DocumentClass	227
DocumentFormat	228
DocumentType	228
DrawPageCount	228
DrawPageHeight	229
DrawPageWidth	229
DrawWorkBookPageCount	229
FileName	230
HiLiteBackground	230
HiLiteForeground	230
HotKeys	231
ImageCustomSize	231
ImagePrintHorzAlign	231
ImagePrintMode	232
ImagePrintPercent	232
ImagePrintVertAlign	233
ImageScaling	233
IndexBufCharSet	233
Invert	234
JumpToFirstHiLite	234
MMPlayOption	235
MMScaleMovie	235

ObjName	235
OPENDisableUI	236
OPENHighLight	236
OPENMode	236
OPENWaitOnOpen	237
PrintAnnotations	238
PrintHeaders	238
RegIniMode	238
RegIniName	239
Rotate	239
SrcCharSet	240
SSDisplayGrid	240
SSDisplayHeaders	241
SSViewObjects	241
TrgCharSet	241
ViewPane	242
WPCustomSize	242
WPDisplayPict	242
WPPageLayout	243
WPScaleTable	243
WPViewMode	244
Chapter 12: Control Events	245
Annotation	245
KeyDown	246
MouseUp	246
OpenDocDone	247
PageNumber	247
PrintDone	248
PrintDoneEx	248
Selection	249
TextBuffer	249
UserClick	250
ViewExtent	250
ViewFile	251
Part IV: Appendixes	252
Appendix A: Supported Formats	253
Supported Formats	253
Archive Formats	255
Binary Format	257
Computer-Aided Design Formats	257
Database Formats	259
Desktop Publishing	259
Display Formats	260
Graphic Formats	260

Mail Formats	263
Multimedia Formats	266
Presentation Formats	267
Spreadsheet Formats	269
Text and Markup Formats	271
Word Processing Formats	272
Supported Formats (Detected)	277
Appendix B: Character Sets	284
Multibyte and Bidirectional Support	284
Coded Character Sets	292
Appendix C: File Format Detection	297
Introduction	297
Extract Format Information	297
Determine Format Support	297
Translate Format Information	298
Distinguish Between Formats	299
Determine a Document Reader	299
Category Values in the Initialization File and Registry	299
Appendix D: File Formats and Extensions	318
File Format and Extension Table	318
Appendix E: Extract and Format Lotus Notes Subfiles	343
Overview	343
Customize XML Templates	343
Use Demo Templates	344
Use Old Templates	344
Disable XML Templates	344
Template Elements and Attributes	345
Conditional Elements	345
Control Elements	346
Data Elements	347
Date and Time Formats	349
Lotus Notes Date and Time Formats	349
KeyView Date and Time Formats	350
Appendix F: List of Files Required for Redistribution	356
Core Files	356
Support Files	357
Document Readers and Writers	358
Archive Formats	358
Binary Formats	359
Computer-Aided Design Formats	359
Database Formats	360
Desktop Publishing Formats	360
Display Formats	360
Graphic Formats	360
Mail Formats	362

- Presentation Formats362
 - Spreadsheet Formats363
 - Word Processor Formats364
- Miscellaneous Functionality365
- Viewing ActiveX Control366
- Windows System Files366
- Appendix G: Configuration Options in formats.ini367
 - formats.ini Options367
- Appendix H: Password Protected Files369
 - Supported Password Protected File Types369
 - View Password Protected Files370
- Send documentation feedback371

Part I: Overview of Viewing SDK

This section provides a general overview of Viewing SDK and a description of the sample programs, and includes the following chapters:

- [Introduction to Viewing SDK](#)
- [Getting Started](#)

Chapter 1: Introduction to Viewing SDK

This guide is for developers who incorporate KeyView Viewing SDK components into their own applications. It is intended for readers who are familiar with Windows programming.

• Overview	16
• Features	16
• Viewing SDK and Visual Basic	17
• Platforms, Compilers, and Dependencies	17
• Windows Installation	18
• Package Contents	19
• License Information	19
• Directory Structure	21

Overview

The Viewing 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.

The Viewing SDK enables you to build high-fidelity document viewing capabilities into your own applications. You can incorporate Viewing technology into your document management, web server, Internet or Intranet, groupware, information retrieval, email, or imaging applications. It enables your users to open, view, and print virtually any document, spreadsheet, presentation, graphic, or compression file, without having the native application or plug-in available.

Viewing SDK uses a standard Windows interface which integrates effectively using popular languages such as C++ (including Microsoft Foundation Classes), J#, and Visual Basic.

The SDK includes the following components:

- Viewing API (VAPI) – Windows messaging-based API
- Viewing ActiveX control (OCX) and .NET interface
- Sample programs

Features

With Viewing SDK, you can create an application by using the Viewing API or the Viewing ActiveX control to:

- View and print documents.
- Convert popular word processing and spreadsheet formats to text, Microsoft Rich Text Format (RTF), and HTML.
- Annotate, highlight, and filter documents.

- Generate thumbnail views of documents.
- Automatically recognize document types.

Viewing SDK and Visual Basic

The Viewing ActiveX control is ideally suited for developing Visual Basic® applications with viewing, conversion, and printing capabilities. The control provides most of the functionality of the Viewing Windows messaging API, but in the form of an ActiveX control that can be dropped onto your Visual Basic form.

Platforms, Compilers, and Dependencies

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

Supported Platforms

- Microsoft Windows 8 x86 and x64
- Microsoft Windows 7 x86 and x64
- Microsoft Windows Vista Business Edition x86 and x64
- Microsoft Windows 2003 Server x86 and x64
- Microsoft Windows 2008 Server x86 and x64
- Microsoft Windows XP x86 (Service Pack 1 and 2) and x64

Supported Compilers

Microsoft 32-bit C/C++ Optimizing Compiler Version 12.00.8804 for 80x86

Software Dependencies

Some KeyView components require that you have installed specific third-party software:

- Outlook 2002 or client or later versions—for Microsoft Outlook Personal Folders (PST) file viewing using the MAPI-based PST reader `pstsr.dll`. The native PST reader (`pstnsr`) does not require an Outlook client. See [Use the Native or MAPI-based Reader, on page 42](#).

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: KVErrors_PSTAccessFailed.

- Lotus Notes or Lotus Domino (the minimum requirement is 6.5.1, but version 8.5 is recommended)—for Lotus Notes database (NSF) file viewing.
- Microsoft .NET Framework Version 2.0 Redistributable Package (if programming in .NET environment).
- Microsoft Visual J# .NET Version 2.0 Redistributable Package (if developing J# program in .NET environment).

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:

`KeyViewViewingSDK_11.5_Windows_X86_64.exe`

The installation wizard opens.

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

The License Agreement page opens.

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

The Installation Directory page opens.

4. Select the directory in which to install the SDK. To specify a directory other than the default, click



, and then specify another directory. After choosing where to install the SDK, click **Next**.

The License Key page opens.

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

NOTE:

The installation program validates the company name and license key and generates the file `install\OS\bin\kv.lic` (where *install* is your chosen installation folder and *OS* is the name of the operating system platform). The license information is validated when the KeyView API is used. If you do not enter a license key at this step, or if you enter invalid

information, the KeyView SDK is installed, but the API does not function. When you obtain a valid license key, you can either re-install the KeyView SDK, or manually update the license key file (`kv.lic`) with the new information. For more information, see [License Information, below](#).

The Pre-Installation Summary dialog box opens.

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

The SDK is installed.

7. Click **Finish**.

Package Contents

The Viewing SDK installation contains:

- Dynamic Link Library files and executable files necessary for viewing text from a wide variety of formats.
- Several sample programs that demonstrate Viewing SDK functionality. See [Viewing API Sample Programs, on page 54](#) and [Control Sample Programs, on page 184](#).
- The following files define the functions and structures used by your application to establish an interface with Viewing SDK:

<code>adAPI.h</code>	<code>kwautdef.h</code>
<code>adinfo.h</code>	<code>kwcmfio.h</code>
<code>kv10obj.h</code>	<code>kwcvmgr.h</code>
<code>kvoem.h</code>	<code>kwkpif.h</code>
<code>kvtypes.h</code>	<code>kwoption.h</code>
<code>kvvapi.h</code>	<code>language.h</code>

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  
XXXXXXXX-XXXXXXXX-XXXXXXXX-XXXXXXXX
```

The license key controls whether the following are enabled:

- the full version of the KeyView SDK
- the trial version of the KeyView SDK

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

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

Enable Advanced Document Readers

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

If you are enabling the MBX reader in an existing installation of Viewing SDK, in addition to updating the license key, you must also follow these steps:

If you are using the registry file:

1. Open the `install.reg.txt` in a text editor. The file is installed in the `install\redist` directory, where `install` is the directory in which you installed Viewing SDK.
2. Under the key `[HKEY_LOCAL_MACHINE\Software\Verity\Viewing SDK\KVMAILVE]`, change the parameter `"208=emlsr.dll"` to `"208=mbxsr.dll"`.
3. Save the file as `install.reg` and import the file into your Windows system registry.

If you are using the `kvsdk.ini` file:

1. Open the `kvsdk.ini` file with a text editor. The file is installed in the root of the Windows directory.
2. In the `[KVMAILVE]` section of the `kvsdk.ini` file, change the parameter `208=emlsr.dll` to `208=mbxsr.dll`.

Update License Information

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

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

To update the information, do one of the following:

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

To update the KeyView license information

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

The file contains the following text:

```
COMPANY NAME  
XXXXXXX-XXXXXXX-XXXXXXX-XXXXXXX
```

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

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

5. Save the `kv.lic` file.

Directory Structure

Viewing SDK creates the following directory structure during installation. The variable *install* refers to the installation directory. By default, the installation directory is `C:\Program Files\Autonomy\KeyViewViewingSDK`.

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\KeyViewViewingSDK\WINDOWS\bin`.

Viewing Installed Directory Structure

Directory	Description
<i>install</i> \OS\bin	Libraries, the <code>formats.ini</code> file, the <code>kv.lic</code> file, and a number of other supporting files.
<i>install</i> \OS \bin\system	Shared libraries used by Viewing SDK components.
<i>install</i> \dotnetview	A .NET workspace for Visual Studio. This is a J# sample program demonstrating basic Viewing functionality.
<i>install</i> \drawdemo	The Viewing API thumbnail sample program (draw into supplied DC).
<i>install</i> \filetype	The Viewing API sample program used to determine file type.
<i>install</i> \fileview	A Viewing OCX sample program.
<i>install</i>	Contains the <i>KeyView Viewing SDK Programming Guide</i> in HTML and PDF format.

Viewing Installed Directory Structure, continued

Directory	Description
\guide	
install \helloworldapi	Sample code for a simple program that demonstrates how to use the Viewing API to display documents in a window. HPE recommends that you review this sample first.
install \ihademo	A Viewing API sample program featuring indexing (filtering), highlighting, and annotating.
install \include	The header files required for Viewing SDK.
install \mfckv	A simple MFC (Microsoft Foundation Class) SDI application using Viewing API.
install \prntdemo	A sample program that uses the Viewing API to print documents.
install \redist	Contains the <code>install.reg</code> file, which contains initialization information used by Viewing SDK. See View Initialization Information, on page 23 .
install \rel_notes	Contains the <i>KeyViewViewing SDK Release Notes</i> in HTML and PDF format.
install \rtfdemo	A sample program that demonstrates the use of the Viewing API to convert documents to RTF.
install \uzipdemo	A sample program for unzipping source files to a selected directory.
install \vapidemo	A sample program that demonstrates most of the Viewing API functionality.
Windows system directory	Contains the <code>kvsdk.ini</code> file which contains initialization information used by Viewing SDK. See View Initialization Information, on page 23 .

Chapter 2: Getting Started

This section provides information on developing and deploying Viewing applications. It includes the following topics:

- [Before You Begin](#) 23
- [View Initialization Information](#) 23
- [Deploy Viewing API Applications](#) 26
- [Deploy ActiveX Control Applications](#) 26
- [Develop .NET Applications](#) 27

Before You Begin

Before you use Viewing SDK to build your own programs, review and run the sample programs provided with the product. HPE recommends that you review the `helloworldapi` sample program first. It is a simple program that demonstrates how to use the Viewing API to display documents within your application.

For information on the sample programs, see [Viewing API Sample Programs, on page 54](#) and [Control Sample Programs, on page 184](#).

View Initialization Information

Viewing uses initialization information for its internal operations, for example, to determine which components to load. You can store this information either in an initialization file or in the Windows registry.

The initialization file is called `kvsdk.ini` and is stored in the Windows system directory.

The file used to define registry settings is called `install.reg.txt` and is stored in the `install\redist` directory, where `install` is the directory in which you installed Viewing SDK.

You must customize the information in one of these files and specify in your application where the information is located.

Use an Initialization File

If you are using the initialization file (`kvsdk.ini`) to set initialization information, you must modify the file to reflect your company name and application name. The sample programs demonstrate how to use an initialization file.

NOTE: A copy of the original `kvsdk.ini` file (`install.ini`) is stored in the `install\redist` directory, where `install` is the directory in which you installed Viewing SDK. This file is not required for redistribution and is for reference only.

Viewing API

To specify an initialization file using the Viewing API

1. Create the `TPVAPICreateParams` structure. Set `uProfileType` to `PROFILEDF_USE_INI`, and `lpszIniFileName` to the location of the initialization file.

For example:

```
memset (&CreateParams, 0, sizeof(TPVAPICreateParams));
if (bUseIni)
{
    CreateParams.uProfileType    = PROFILEDF_USE_INI;
    CreateParams.lpszIniFileName = szIniFileName;
}
else
{
    CreateParams.uProfileType    = PROFILEDF_USE_REG;
    CreateParams.lpszRegistryName = REGISTRY_NAME_ASCII;
}
```

2. Create the VAPI window by using the standard Windows API functions `CreateWindow()` or `CreateWindowEx()`.

ActiveX Control

To specify an initialization file by using the ActiveX control

1. Set the `RegIniMode` property to 1.
2. Set the `RegIniName` property to the path and name of the initialization file. For example, `kvsdk.ini` or `c:\myprogram\myini.ini`. By default, Viewing looks for the initialization file in the Windows system directory.

For example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    AxKEYview1.RegIniMode = 1
    AxKEYview1.RegIniName = "c:\windows\kvsdk.ini"
    AxKEYview1.Open("c:\test.doc")
End Sub
```

Use the Windows Registry File

If you are using the Windows registry to set initialization information, you must modify the registry file (`install.reg.txt`) to reflect your company name and application name. The file has a `.txt` extension for easy editing and viewing. After you have finished editing the file, remove the `.txt` extension. When your application is installed, import the `install.reg` file into the Windows Registry.

Viewing API

To specify the Windows registry by using the Viewing API

1. Create the `TPVAPICreateParams` structure. Set `uProfileType` to `PROFILEDF_USE_REG`, and `lpszRegistryName` to the location of the initialization file.

For example:

```
memset (&CreateParams, 0, sizeof(TPVAPICreateParams));
if (bUseRegistry)
{
    CreateParams.uProfileType      = PROFILEDF_USE_REG;
    CreateParams.lpszRegistryName = REGISTRY_NAME_ASCII;
}
else
{
    CreateParams.uProfileType      = PROFILEDF_USE_INI;
    CreateParams.lpszIniFileName = szIniFileName;
}
```

2. Create the VAPI window by using the standard Windows API functions `CreateWindow()` or `CreateWindowEx()`.

ActiveX Control

To specify the Windows registry by using the ActiveX control

1. Set the `RegIniMode` property to 2.
2. Set the `RegIniName` property to the registry key under `HKEY_LOCAL_MACHINE\Software` where the Viewing initialization information resides. For example, *YourCompany\YourProduct*.

For example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    AxKEYview1.RegIniMode = 2
    AxKEYview1.RegIniName = "Autonomy\keyview"
    AxKEYview1.Open("c:\test.doc")
End Sub
```

Remove Functionality from an Application

To remove Viewing functionality from your application

1. Do not redistribute the Dynamic Link Library (DLL) associated with the component you want to remove.

lists the files that you can redistribute with your application. It also shows the Dynamic Link Library (DLL) associated with each component.

2. In the `kvsdk.ini` file or the `install.reg.txt` file, remove references to the component that you want to remove.

For example, to remove support for Windows Animated Cursor, remove the line that references "Windows Animated Cursor" from the registry or initialization file, and do not redistribute the Windows Animated Cursor reader (`kpanirdr.dll`).

NOTE: If you remove Viewing functionality for a graphic format, and you view a document that has an embedded graphic of that format, the graphic is not displayed.

The following is a summary of files required based on functionality:

- **Copy to clipboard**—The following files are required for copy to clipboard functionality:
 - `rtfcnv.dll`, `txtcnv.dll` (for word processor formats)
 - `rtfss.dll` (for spreadsheet formats)
 - `kpifutil.dll` (for picture formats)
- **SaveAs to RTF**—The following files are required for SaveAs to RTF functionality:
 - `kvcnv.dll`
 - `rtfcnv.dll` (for word processor formats)
 - `rtfss.dll` (for spreadsheet formats)
 - `kpifutil.dll` (for picture formats)

Deploy Viewing API Applications

After you have built an application with the Viewing API, you must do the following:

1. Install all required files to the `\bin` directory of your application's installation directory. [List of Files Required for Redistribution, on page 356](#) lists the components that must be redistributed with your application. It also shows the Dynamic Link Library (DLL) associated with each component.
2. Review the `kvsdk.ini` file or the `install.reg.txt` file to make sure that the appropriate files are referenced.
3. Update the `HOME` entry in the initialization file or registry file with the complete path to where you are installing Viewing components.
4. Specify whether you are using an initialization file or registry settings. See [View Initialization Information, on page 23](#).
5. If you are using an initialization file, install the file to the location specified by `lpszIniFileName` when the application is installed. See [TPVAPICreateParams, on page 163](#).
6. If you are using the registry file, import the `install.reg` file into the Windows Registry when the application is installed.

Deploy ActiveX Control Applications

After you have built an application with Viewing ActiveX control, you must do the following:

1. Install all required files to the `\bin` directory of your application's installation directory. [List of Files Required for Redistribution, on page 356](#) lists the components that must be redistributed with your

application. It also shows the Dynamic Link Library (DLL) associated with each component.

2. Review the `kvsdk.ini` file or the `install.reg.txt` file to make sure that the appropriate files are referenced.
3. Update the `HOME` entry in the initialization file or registry file with the complete path to where you are installing Viewing components.
4. Specify whether you are using an initialization file or registry settings. See [View Initialization Information, on page 23](#).
5. If you are using an initialization file, install the file to the location specified by `RegIniName` when the application is installed.
6. If you are using the registry file, import the `install.reg` file into the Windows Registry when the application is installed.
7. Install the Viewing ActiveX control (`kvocx.ocx`) to the `\bin` directory of your application's installation directory and register the control in the system registry by running the following command:

```
regsvr32 C:\MyApp\bin\kvocx.ocx
```

Add `-s` to suppress any dialog boxes when registering the OCX.

To unregister the ActiveX control, run the command:

```
regsvr32 -u C:\MyApp\bin\kvocx.ocx
```

Develop .NET Applications

This section describes how to create and deploy a .NET application by using the KeyView ActiveX Control. Although you can develop .NET applications in many different development environments, the instructions in this section refer to Microsoft Visual Studio 2005.

To create and deploy a .NET application

1. Install the KeyView Viewing SDK.

The installation automatically registers the Viewing ActiveX control, "KeyView OLE Control module (v1.0)" and installs the COM dynamic library (`kvocx.ocx`) to the Viewing `\bin` directory. You can also use the `regsvr32` command to register the ActiveX COM module. For example:

```
regsvr32.exe install\bin\kvocx.ocx
```

2. In Visual Studio 2005, select **Tools** from the main menu, and click **Choose Toolbox Items....**
3. In the **Choose Toolbox Items** dialog box, click the **COM Components** tab.
4. From the list of available COM components, select the **KeyView Control** check box, and then click **OK**.

A Windows control named **KeyView Control** appears in the Toolbox. You can use this KeyView control in the same way as other controls in the Toolbox.

When the .NET application is built, Visual Studio creates the following dynamic libraries:

- `Interop.KEYVIEWLib.dll`
- `AxInterop.KEYVIEWLib.dll`

These libraries are wrappers for the KeyView ActiveX control, and are required to use the control in a .NET environment.

Method and Property Naming Conventions

The .NET control class name for KeyView ActiveX control is `AxKEYVIEWLib.AxKEYview`, where the namespace `AxKEYVIEWLib` is the library name.

In J#, C#, and C++, all ActiveX control method names in the .NET class are the same as their COM counterparts. However, individual properties in .NET are defined using get and set methods of the following format:

`get_property_name`

`set_property name`

For example, `RegIniName` in COM has `get_RegIniName` and `set_RegIniName` methods in the .NET class.

NOTE: Important: In a Visual Basic .NET application, all properties and methods are used in the same way as in a Visual Basic COM application.

Sample Code

The following code demonstrates how to use the .NET class in a J# Windows Form program:

```
private void button1_Click(Object sender, System.EventArgs e)
{
    this.axKEYview1.set_RegIniMode((short)1);
    this.axKEYview1.set_RegIniName("c:\\windows\\kvsdk.ini");
    this.axKEYview1.Open("c:\\test.doc");
}
```

The following code demonstrates how to use the .NET class in a Visual Basic Windows Form program:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    AxKEYview1.RegIniMode = 1
    AxKEYview1.RegIniName = "c:\\windows\\kvsdk.ini"
    AxKEYview1.Open("c:\\test.doc")
End Sub
```

Deploy .NET Applications

After you have built a .NET application using Viewing ActiveX control, follow these steps to deploy the application:

1. Install all required files to the `\bin` directory of your application's installation directory. [List of Files Required for Redistribution, on page 356](#) lists the components that must be redistributed with your application. It also shows the Dynamic Link Library (DLL) associated with each component.

2. Review the `kvsdk.ini` file or the `install.reg.txt` file to make sure that the appropriate files are referenced.
3. Update the `HOME` entry in the initialization file or registry file with the complete path to where you are installing Viewing components.
4. Specify whether you are using an initialization file or registry settings. See [View Initialization Information, on page 23](#).
5. If you are using an initialization file, install the file to the location specified by `RegIniName` when the application is installed. See [RegIniName, on page 239](#).
6. If you are using the registry file, import the `install.reg` file into the Windows Registry when the application is installed.
7. Install the Viewing ActiveX control (`kvocx.ocx`) to the `\bin` directory of your application's installation directory. Register the control in the system registry by running the following command:

```
regsvr32 C:\MyApp\bin\kvocx.ocx
```

Add `-s` to suppress any dialog boxes when registering the OCX.

8. Install the wrapper files `Interop.KEYVIEWLib.dll` and `AxInterop.KEYVIEWLib.dll` to the `\bin` directory of your application's installation directory.
9. Install the Microsoft .NET Framework Version 2.0 Redistributable Package, which is available at <http://msdn.microsoft.com/netframework/downloads/updates/default.aspx>.
10. If the application is developed using J#, install the Microsoft Visual J# .NET Version 2.0 Redistributable Package, which is available at <http://msdn.microsoft.com/netframework/downloads/updates/default.aspx>.

Part II: Viewing API

This section describes the Viewing API and provides detailed reference information and includes the following chapters:

- [Use the Viewing API](#)
- [Viewing API Sample Programs](#)
- [Message Parameters](#)
- [Notification Message Parameters](#)
- [Structures](#)

Chapter 3: Use the Viewing API

This section describes how to use the Viewing API to perform some basic viewing tasks.

• Overview of the Viewing API	31
• Create a Viewing API Window	32
• Open and View a Document	33
• Save a Document	34
• Convert a Document	34
• Print a Document	35
• Change the Print Job Name	35
• Determine the Document Format	36
• Extract Document Metadata	36
• Change Document Options	36
• Annotate, Highlight, or Index a Document	37
• Draw a Page	37
• Edit a Document	38
• Modify the Document View	38
• Change the Current Object in a Document	40
• View Deleted Items and Document Revision Marks	40
• View Container Files	41
• View PDF Documents	48
• View Microsoft Visio Files	51
• Extract Microsoft Excel Formulas	51

Overview of the Viewing API

The Viewing API (VAPI) enables you to build a Windows program that uses Viewing components to manage many types of document, including word processing, spreadsheet, presentation, and graphics. See [Supported Formats, on page 253](#) for more information on supported formats.

You can use the Viewing API to create an application to:

- Open and view a document.
- Draw a page of a word processing document, spreadsheet, or a picture into a supplied Device Context (HDC). This is useful for generating *thumbnail* views of documents.
- Print a document (including the ability to print a document without viewing it) to a specified printer or to the default printer.
- Allow viewed word processing and spreadsheet documents to be saved as RTF, HTML, or text. Also, you can save image formats to other supported image formats.

- Convert word processing and spreadsheet documents to text, RTF, or HTML without viewing them.
- View or extract subfiles from a container file, such as ZIP, TAR, or PST.
- View and manipulate a graphic (including rotate and magnify).
- Annotate documents with a bitmap or selected text. The Viewing API includes annotation event notification for actions such as clicking and double-clicking, allowing for implementation of hyperlink and pop-up text.
- Highlight all occurrences of a word in a document.
- Filter spreadsheets, presentation graphics, and documents to text. A cross-platform C API that provides text filtering is also available. Contact HPE for information on KeyView Filter SDK.
- Determine a document's format based on its contents, not its file extension.
- Obtain document metadata, such as a document's author or title.

Create a Viewing API Window

You must create a new VAPI window for each document that you open; each VAPI window manages only *one* document at a time.

You can create multiple VAPI windows to handle multiple documents simultaneously. After you create a VAPI window, you can use the Viewing API to manage the document by sending messages to the window and receiving notification messages from the window. When you are finished with the document, you destroy its VAPI window.

To create the VAPI window, use the standard Windows API functions `CreateWindow()` or `CreateWindowEx()`, with the following parameter values:

Parameter	Value/Description
LPCTSTR lpClassName	VAPIDF_VAPI_WINDOW_CLASS_NAME (defined in <code>kvvapi.h</code>).
LPCTSTR lpWindowName	NULL.
DWORD dwStyle	WS_CHILD or WS_DISABLED
int x, y	0, 0
int nWidth	The width of the application (parent) window.
int nHeight	The height of the application (parent) window.
HWND hWndParent	The handle of the application window. (See note below.)
HMENU hMenu	NULL.
HINSTANCE hInstance	The handle of the VAPI library.
LPVOID lpParam	A pointer to a TPVAPICreateParams structure that specifies optional parameters. Through this structure, you specify whether you are using an initialization file or registry settings.

For example:


```
hWndVAPI = CreateWindow (VAPIDF_VAPI_WINDOW_CLASS_NAME,  
                        NULL,  
                        WS_CHILD | WS_DISABLED,  
                        rc.left, rc.top, rc.right, rc.bottom,  
                        hWnd,  
                        NULL,  
                        hLibVAPI,  
                        &CreateParams);
```

Get the Viewer Window of the Document

The Viewer window is a document-specific window that the VAPI window creates when you open a document. Because the Viewer window is controlled by the VAPI window, normally you should not need the handle of the Viewer window.

The Viewer window is subclassed by the VAPI window. That is, when the VAPI window creates the Viewer window, it subclasses the Viewer window so that the VAPI window intercepts all messages sent to the Viewer window. This allows the VAPI window to control the Viewer window and to handle the right-mouse context menu and common operations such as **SaveAs**.

Depending on the document type, the Viewer window might also create several child windows in order to handle the document. For example, when you open a spreadsheet document, the VAPI window creates a [WorkBook] Viewer window, which in turn creates a [SpreadSheet] Viewer window for each spreadsheet page when it is accessed. All VAPI child windows are destroyed when the VAPI window is destroyed.

- To get the Viewer window handle of a document, use the [VAPIMWP_INIT_GETHWNDVIEWER](#) message.
- To disable the Viewer user interface for a document (that is, when the Viewer asks if the user interface is disabled before creating a dialog box), respond to the [VAPINMWP_INIT_DISABLEUI](#) notification message.

Open and View a Document

Because a document must be opened before it can be viewed, printed, saved, or can have any other operation performed on it, viewing a document in the Viewing API means to open *and* view a document. It is possible, however, to open a document without viewing it, or in other words, to open a document with view mode disabled. In this mode, you can print or save the document without viewing it.

To open a document

1. Create a [TPVAPIOpenDocumentInfo](#) structure.
To open a document without viewing it (view mode disabled), set the [VAPIDF_FLAGS_OPEN_WITHOUT_VIEW](#) flag in the `nFlags` member of the `TPVAPIOpenDocumentInfo` structure.
2. Send VAPI a [VAPIM_INIT](#) message with the `wParam` set to [VAPIMWP_INIT_OPEN_DOCUMENT](#), and the `lParam` set to the address of the `TPVAPIOpenDocumentInfo` structure. See [VAPIMWP_INIT_OPEN_DOCUMENT](#), on page 98.

For example:

```
memset (&OpenDocInfo, 0, sizeof(TPVAPIOpenDocumentInfo));  
OpenDocInfo.lpszFilePath = szFileName;  
lResult = SendMessage (hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_OPEN_DOCUMENT,  
(LPARAM)OpenDocInfo);
```

Notification Messages

- To receive the status of the open document expressed as the percent done, wait for the [VAPINMWP_INIT_OPENDOCDONE](#) , on [page 153](#) notification message.
- To receive the current page number of the document, wait for the [VAPINMWP_INIT_PAGENUMBER](#) notification message.
- To receive the name of the current object of the document, wait for the [VAPINMWP_MULTIOBJ_OBJNAME](#) notification message.

Save a Document

To save a document

1. Open the document. See [Open and View a Document, on the previous page](#).
To make sure that the entire document is opened before the document is saved, open the document with the `bWait` member in the [TPVAPIOpenDocumentInfo](#) structure set to `TRUE`.
To save a document without viewing it, open the document with view mode disabled.
2. Use the [VAPIMWP_FILE_CANSAVEAS](#) message to determine whether the document is completely processed and can be saved.
3. Use the [VAPIMWP_FILE_SAVEAS](#) message to save the document through a **Save As** dialog box.
To override the file path that VAPI uses to convert an I/O object to a temporary file when saving an I/O object document, respond to the [VAPINMWP_INIT_GETTEMPFILEPATH](#) notification message.

Convert a Document

To convert a document to text, RTF, or HTML

1. Open the document. See [Open and View a Document, on the previous page](#).
To make sure that the entire document is opened before the document is saved, open the document with the `bWait` member in the [TPVAPIOpenDocumentInfo](#) structure set to `TRUE`.
To save a document without viewing it, open the document with view mode disabled.
2. Use the [VAPIMWP_CANCONVERT](#) message to determine whether the document is completely processed and can be converted.
3. Use either the [VAPIMWP_FILE_SAVEAS](#) message to convert the document through a **Save As** dialog box or the [VAPIM_CONVERT](#) message to convert the document without requiring the user

to respond to the **Save As** dialog box.

To override the file path that VAPI uses to convert an I/O object to a temporary file when converting an I/O object document, respond to the `VAPIMWV_INIT_GETTEMPFILEPATH` notification message.

NOTE: Viewing SDK does not convert PDFs, presentations, container files, or graphics files to text, RTF, or HTML.

Print a Document

The `prntdemo` sample program demonstrates how to print by using the Viewing API.

To print a document

1. Open the document. See [Open and View a Document, on page 33](#).
To make sure that the entire document is opened before the document is printed, open the document with the `bwait` member in the `TPVAPIOpenDocumentInfo` structure set to `TRUE`.
To print a document without viewing it, open the document with view mode disabled.
2. Use the `VAPIMWV_PRINT_CANPRINT` message to determine whether a document is completely processed and ready for printing.
3. Optionally, use the `VAPIMWV_PRINT_PRINTHEADER` message to print the file name, page number, and page length at the top of each page of a printed output.
Used in conjunction with `VAPIMWV_PRINT_PRINTHEADER`, the `VAPIMWV_PRINT_SETPRINTNAME` message replaces the default file name field of the header with another string.
4. Optionally, use the `VAPIMWV_PRINT_PRINTTOPD` message to set the standard Windows print options.
5. Optionally, use the `VAPIMWV_PRINT_PAGESETUP` message to set print page scaling for a spreadsheet.
6. Use either the `VAPIMWV_PRINT_PRINT` message to print by using a common **Print** dialog box or the `VAPIMWV_PRINT_PRINTTOPRINTER` message to print to a specific printer without a **Print** dialog box.

Change the Print Job Name

You can change the print job name in the `kvsdk.ini` file. The printer uses the print job name for all documents printed from KeyView Viewing SDK.

To change the print job name

1. Open the `kvsdk.ini` file with a text editor. The file is installed in the root of the Windows directory.
2. In the `[Settings]` section, set the `PrintJobName` parameter to the desired print job name. For example:

```
[Settings]
```

```
PrintJobName=MyPrintJob
```

3. Save the file.

Determine the Document Format

To determine a document format

1. Open the document. See [Open and View a Document](#), on page 33.

To get format information without viewing the document, set the `VAPIDF_FLAGS_OPEN_VAPI_ONLY` flag in the `nFlags` member of the `TPVAPIOpenDocumentInfo` structure.

To quickly determine a document's format, regardless of whether the document is supported for viewing, set the `VAPIDF_FLAGS_OPEN_FORMAT_ONLY` flag in the `nFlags` member of the `TPVAPIOpenDocumentInfo` structure. Only the `VAPIM_INIT` message with the `VAPIMWP_INIT_GETDOCFORMAT` parameter is supported when opening a document with the `VAPIDF_OPEN_FORMAT_ONLY` flag enabled.

2. Use the `VAPIMWP_INIT_GETDOCFORMAT` parameter of the `VAPIM_INIT` message to get the document format of the currently opened document.
3. Use the `VAPIMWP_INIT_GETDOCCLASS` parameter of the `VAPIM_INIT` message to get the general class to which the currently opened document belongs.

Extract Document Metadata

To extract metadata from a document, use the `VAPIM_GETSUMMARYINFO` message.

Change Document Options

Document options control display elements such as window size, zoom settings, margin size, and scaling. Options are defined for each file type category (for example, spreadsheets, multimedia, and word processing). The document options only apply to the current document and document type. In other words, it initializes the in-memory options of the current Viewer. The options are defined in `kwoption.h`.

To set options for a document

1. Create an `ALL_OPTIONS_EX` structure.
2. If you are using the `VAPIMWP_INIT_OPEN_DOCUMENT` message to set options, create a `TPVAPIOpenDocumentInfo` structure.
3. Use either the `VAPIMWP_OPTIONS_SETOPTIONS_EX` or `VAPIMWP_INIT_OPEN_DOCUMENT` message.

To get the options of a document, use the `VAPIMWP_OPTIONS_GETOPTIONS_EX` message.

Annotate, Highlight, or Index a Document

- Use the [VAPIM_ENABLEINDEX](#) message to enable index-only mode. This generates text buffer (VAPINM_TEXTBUFFER) notification messages with document viewing disabled.
- To specify the character set for the returned text buffer, use [VAPIM_SETINDEXBUFCHARSET](#).
- To add and delete annotations, use the [VAPIM_ANNOTATE](#) message. The annotation is placed at a logical address.
- To add a highlight to a document, use the [VAPIM_SETHILITE](#) message.
- See the following messages and notification messages for more functionality related to annotating, highlighting, or indexing documents:
 - [VAPIM_GETNEXTTEXTBUFFER](#), on page 67
 - [VAPIM_GETPAGEFROMLOGICAL](#) , on page 68
 - [VAPIM_GETTEXT](#) , on page 69
 - [VAPIM_GOTO_PAGE](#) , on page 70
 - [VAPIM_POSITION](#) , on page 71
 - [VAPIM_SETCURSOR](#) , on page 73
 - [VAPIM_SHOWHITS](#) , on page 76
 - [VAPINM_ANNOTATION_HIT](#) , on page 146
 - [VAPINM_EXTENT](#) , on page 147
 - [VAPINM_SELECTION](#) , on page 148
 - [VAPINM_USERCLICK](#), on page 150
 - The [VAPIMWP_PRINT_ANNOTATIONS](#), on page 111 parameter of [VAPIM_PRINT](#)

Draw a Page

Draw a Page into a Supplied Device Context

- Use the [VAPIMWP_DRAW_INIT](#) parameter of the [VAPIM_DRAW](#) message to initialize the drawing routine in VAPI. You must send this parameter before you open the document by using the [VAPIMWP_INIT_OPEN_DOCUMENT](#) message.
- To get the number of pages in a document, open the document with the `bWait` parameter in the `TPVAPIOpenDocumentInfo` structure set to `TRUE`, and use the [VAPIMWP_DRAW_GETPAGECOUNT](#) parameter of the [VAPIM_DRAW](#) message. See [VAPIMWP_INIT_OPEN_DOCUMENT](#) for more information. You can set `bWait` to `FALSE` if you do not want to wait for the whole file to be processed and just want to get the size of the first few pages and draw the first few pages, or if you want to draw pages in any order.

For spreadsheets, you must use the [VAPIMWP_DRAW_GETPAGECOUNT](#) parameter to draw the worksheet pages successfully. To change the worksheet, use the [VAPIM_MULTIOBJ](#) message. See [VAPIMWP_MULTIOBJ_CANMULTIOBJ](#), on page 102 for more information.

- To get the size of the specified page, use the [VAPIMWP_DRAW_GETPAGESIZE](#) parameter of the

VAPIM_DRAW message.

- To draw the specified page into the supplied device context, use the [VAPIMWP_DRAW_DRAWPAGE](#) parameter of the VAPIM_DRAW message.
- To create a thumbnail image file of a document page, use the [VAPIMWP_DRAW_DRAWTOFILE](#) parameter of the VAPIM_DRAW message.

Edit a Document

Search for Text

- To determine whether a document can be searched, use the VAPIMWP_EDIT_CANFIND message. See [VAPIMWP_EDIT_CANFIND](#) , on page 85.
- To search a document for the specified text, use the VAPIMWP_EDIT_FIND message. See [VAPIMWP_EDIT_FIND](#) , on page 87.
- To get the currently selected text in a document, use the VAPIMWP_EDIT_GETFINDTEXT message. See [VAPIMWP_EDIT_GETFINDTEXT](#) , on page 88.

Copy Text

- To determine whether the selected text in a document can be copied, use the VAPIMWP_EDIT_CANCOPY message. See [VAPIMWP_EDIT_CANCOPY](#) , on page 84.
- To copy the selected text in a document, use the VAPIMWP_EDIT_COPY message. See [VAPIMWP_EDIT_COPY](#) , on page 86.
- To determine whether all the items in a document can be selected, use the VAPIMWP_EDIT_CANSELECTALL message. See [VAPIMWP_EDIT_CANSELECTALL](#) , on page 85.
- To select all the items in a document, use the VAPIMWP_EDIT_SELECTALL message. See [VAPIMWP_EDIT_SELECTALL](#) , on page 89.

Modify the Document View

Change the Layout of a Document

- To determine whether the layout of a document can be changed, use the [VAPIMWP_VIEW_CANLAYOUT](#) message.
- To get the current layout of a document, use the [VAPIMWP_VIEW_GETLAYOUT](#) message.
- To set the layout of a document, use the [VAPIMWP_VIEW_SETLAYOUT](#) message.

Change the Aspect Ratio of a Document

- To determine whether the aspect ratio of a document can be changed, use the [VAPIMWP_VIEW_CANASPECTRATIO](#) message.

- To get the current aspect ratio of a document, use the `VAPIMWP_VIEW_GETASPECTRATIO` message.
- To set the aspect ratio of a document, use the `VAPIMWP_VIEW_SETASPECTRATIO` message.

Invert, Rotate, or Magnify a Document

- To determine whether a document can be inverted, use the `VAPIMWP_VIEW_CANINVERT` message.
- To get the current invert state of a document, use the `VAPIMWP_VIEW_GETINVERT` message.
- To set the invert state of a document, use the `VAPIMWP_VIEW_SETINVERT` message.
- To determine whether a document can be rotated, use the `VAPIMWP_VIEW_CANROTATE` message.
- To get the current rotation of a document, use the `VAPIMWP_VIEW_GETROTATE` message.
- To set the rotation of a document, use the `VAPIMWP_VIEW_SETROTATE` message.
- To determine whether a document can be magnified, use the `VAPIMWP_VIEW_CANMAGNIFY` message.
- To determine whether a document can be magnified to fit the document selection to the window, use the `VAPIMWP_VIEW_CANFITTOWINDOW` message.
- To get the current magnification of a document, use the `VAPIMWP_VIEW_GETMAGNIFY` message.
- To set the magnification of a document, use the `VAPIMWP_VIEW_SETMAGNIFY` message.

Display or Hide Gridlines in a Document

- To determine whether a document supports gridlines, use the `VAPIMWP_VIEW_CANGRIDLINES` message.
- To get the current gridline state of a document, use the `VAPIMWP_VIEW_GETGRIDLINES` message.
- To set the gridline state of a document, use the `VAPIMWP_VIEW_SETGRIDLINES` message.

Play a Multimedia Document

- To determine whether a multimedia document can be played, use the `VAPIMWP_VIEW_CANPLAY` message.
- To play a multimedia document, use the `VAPIMWP_VIEW_PLAY` message.
- To determine whether the playing of a multimedia document can be paused, use the `VAPIMWP_VIEW_CANPAUSE` message.
- To pause the playing of a multimedia document, use `VAPIMWP_VIEW_PAUSE`.
- To determine whether the playing of a multimedia document can be stopped, use the `VAPIMWP_VIEW_CANSTOP` message.
- To stop the playing of a multimedia document, use the `VAPIMWP_VIEW_STOP` message.
- To get the play mode (that is, to stop or loop at the end of a multimedia document after playing it),

use the `VAPIMWP_VIEW_GETPLAYMODE` message.

- To set the play mode of a multimedia document to stop at the end after playing it, use the `VAPIMWP_VIEW_END` message.
- To set the play mode of a multimedia document to loop at the end after playing it, use the `VAPIMWP_VIEW_LOOP` message.

Change the Current Object in a Document

There are many Viewing parameters that control the objects in a multiple-object document. Examples of a multiple-object document include a Microsoft Excel spreadsheet with multiple worksheets, and a Microsoft PowerPoint presentation with multiple slides.

- To determine whether a document contains multiple objects, use the `VAPIMWP_MULTIOBJ_CANMULTIOBJ` message.
- To determine the number of objects in a multiple-object document, use the `VAPIMWP_MULTIOBJ_GETOBJCOUNT` message.
- To change the current object to the next object in a document, use the `VAPIMWP_MULTIOBJ_NEXTOBJ` message.
- To change the current object to the previous object in a document, use the `VAPIMWP_MULTIOBJ_PREVOBJ` message.
- To get the name of the current object in a document, use the `VAPIMWP_MULTIOBJ_OBJNAME` message.
- To change the current object to a target object in a document, use the `VAPIMWP_MULTIOBJ_SETCURRENTOBJ` message.
- To receive the name of the current object, which VAPI sends when the document is first opened or whenever the object changes, wait for the `VAPIMWP_MULTIOBJ_OBJNAME` notification message.

View Deleted Items and Document Revision Marks

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

If revision tracking was enabled when changes were made to a document, you can configure Viewing to display the deleted content, revision marks, and revision tracking information in the document. Content that was added to the document is underlined. Content that was deleted from the document is displayed with strikethrough formatting. The name of the reviewer who made the change and the date on which the change was made is displayed in a tooltip when you hover the cursor over the revised text.

To display revision tracking information

1. Create a `TPVAPIOpenDocumentInfo` structure.
2. Set the `VAPIDF_FLAGS_INCL_REVISION_MARK` flag in the `nFlags` member of the `TPVAPIOpenDocumentInfo` structure.

3. Send VAPI a `VAPIM_INIT` message with the `wParam` set to `VAPIMWP_INIT_OPEN_DOCUMENT`, and the `lParam` set to the address of the `TPVAPIOpenDocumentInfo` structure. See [VAPIMWP_INIT_OPEN_DOCUMENT](#), on page 98 for more information.

The View API Demo program demonstrates how to implement the revision mark feature.

View Container Files

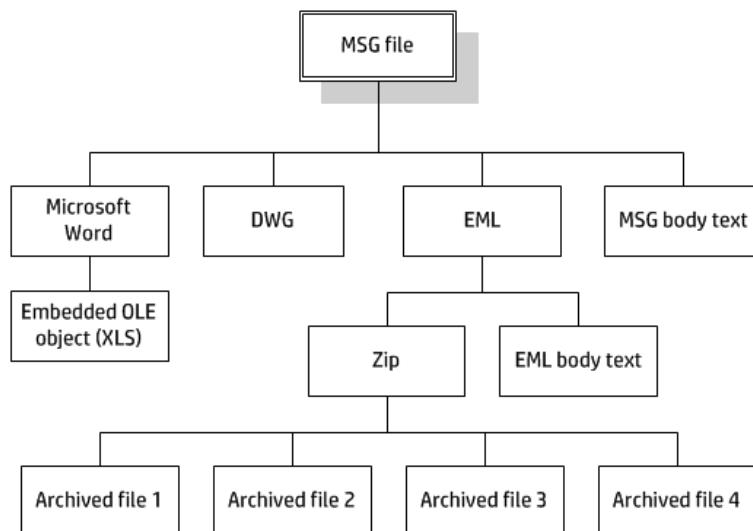
A *container* file has a main file (parent) and subfiles (children) embedded in the main file. The following are examples of container files:

- Compressed 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)

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 containing 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.

Example Container File Tree Structure



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

Microsoft Outlook Personal Folders (PST) Files

The PST reader is an advanced feature and is sold and licensed separately. To enable this reader in a KeyView SDK, you must obtain the appropriate license key from HPE. See [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 by using the Microsoft's Messaging Application Programming Interface (MAPI) reader named `pstsr.dll`.
- directly using the native PST reader named `pstnsr.dll`.

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 only
Outlook client required	No	Yes
MAPI properties supported	Yes All properties are defined in <code>mapitags.h</code> . Object properties are not supported.	Yes All properties are defined in <code>mapitags.h</code> . Object properties are not supported.
Password protection supported	Yes	Yes (using the <code>KVCredential</code> structure)
Compressible encryption supported	Yes	Yes
High encryption supported	No	Yes

To use the native reader for PST files, change the PST entry in either the registry file or the initialization file as follows:

In the `kvsdk.ini` file

1. Open the `kvsdk.ini` file with a text editor. The file is installed in the root of the Windows directory.
2. In the `[KVMAILVE]` section of the `kvsdk.ini` file, change the parameter `297=pstsr.dll` to `297=pstnsr.dll`.

In the registry file

1. Open `install.reg.txt` in a text editor. The file is installed in the `install\redist` directory, where `install` is the directory in which you installed Viewing SDK.
2. Under the `[HKEY_LOCAL_MACHINE\Software\Autonomy\KeyviewViewingSDK\KVMAILVE]` key, change the parameter `"297"="pstsr.dll"` to `"297"="pstnsr.dll"`.
3. Save the file as `install.reg`.
4. Import the file into your Windows system registry.

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-Based Reader (pstsar)

The `pstsar` 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.

System Requirements

Because MAPI relies on functionality in Microsoft Outlook, a Microsoft Outlook client must be installed on the same machine as the application displaying 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: KVErrors_PSTAccessFailed.

Lotus Notes Database (NSF)

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

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 display 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.

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 that displays the NSF files.

KeyView supports Lotus Notes client version 6.5.1, Lotus Domino 6.5.1, and 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)

Installation and Configuration

Before KeyView can display 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 viewing:

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 file `notes.ini` is in the `install\lotus\notes` 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` and KeyView bin directories to the PATH environment variable. HPE recommends that you add the KeyView bin directory because the Lotus Notes installation might contain older KeyView OEM libraries.

Format Notes

The KeyView NSF reader uses XML templates to format Lotus notes. You can customize the templates as required 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 343](#).

View Mail Messages and Mail Stores

You can display mail messages and mail stores in one of two ways:

- The Viewing window displays the file's hierarchy—showing all the children of the parent file—by using the archive format viewing engine, `kvarve.dll`. See [View Archive Files, on page 46](#) for more information.

- The Viewing windows displays the file as it would appear in a Microsoft Outlook Client. This display uses the mail format viewing engine, `kvmailve.dll`.

By default, mail messages and mail stores are displayed with the mail format viewing engine. To use the archive format viewing engine to display the complete file hierarchy, follow these steps:

In the `kvsdk.ini` file

1. Open the `kvsdk.ini` file with a text editor. The file is installed in the root of the Windows directory.
2. Remove the comments from the beginning of the following lines:

```
297=zip 0 kvarcve.dll; PST
295=zip 0 kvarcve.dll; MSG MS Outlook
208=zip 0 kvarcve.dll; EML
299=zip 0 kvarcve.dll; Lotus Notes NSF
```

3. In the `[VAPI]` section of the `kvsdk.ini` file, insert comments at the beginning of the following lines:

```
; Mail formats
;kvmailve.dll=kvMAILVIEW;
;297=mail 0 kvmailve.dll; PST
;295=mail 0 kvmailve.dll; MSG MS Outlook
;208=mail 0 kvmailve.dll; EML
;299=mail 0 kvmailve.dll; Lotus Notes NSF
```

In the registry file

1. Open the `install.reg.txt` in a text editor. The file is installed in the `install\redist` directory, where `install` is the directory in which you installed Viewing SDK.
2. Remove the comments from the beginning of the following lines:

```
297=zip 0 kvarcve.dll; PST
295=zip 0 kvarcve.dll; MSG MS Outlook
208=zip 0 kvarcve.dll; EML
299=zip 0 kvarcve.dll; Lotus Notes NSF
```

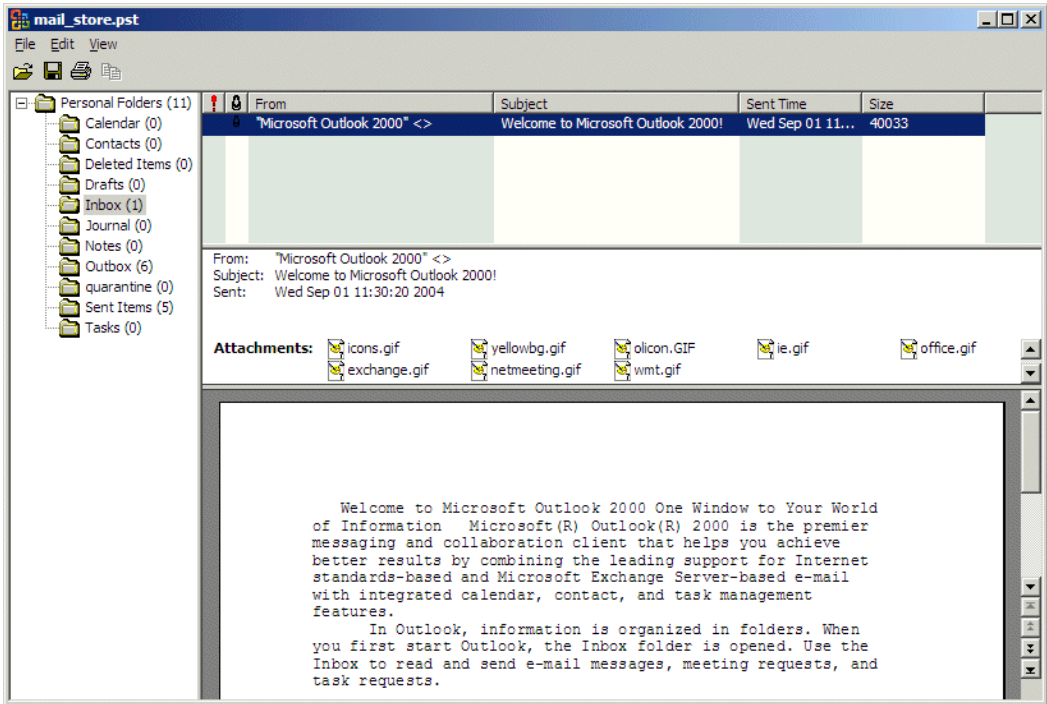
3. Under the `[HKEY_LOCAL_MACHINE\Software\Autonomy\Viewing SDK\VAPI]` key, insert comments at the beginning of the following lines:; Mail formats

```
; Mail formats
;kvmailve.dll=kvMAILVIEW;
;297=mail 0 kvmailve.dll; PST
;295=mail 0 kvmailve.dll; MSG MS Outlook
;208=mail 0 kvmailve.dll; EML
;299=mail 0 kvmailve.dll; Lotus Notes NSF
```

4. Save the file as `install.reg`.
5. Import the file into your Windows system registry.

The following figure shows a PST file displayed in the Viewing API sample program with the mail format viewing engine:

Display mail files with the mail format viewing engine



To extract the main message and its attachments to disk, select the main message and send an Unzip message or method (VAPIMWP_FILE_UNZIP or UnZip). See the implementation of the **Extract** menu in the View API Demo program (vapidemo).

To view an attachment, double-click the file in the **Attachments** field. The file is displayed in a separate window that can be closed.

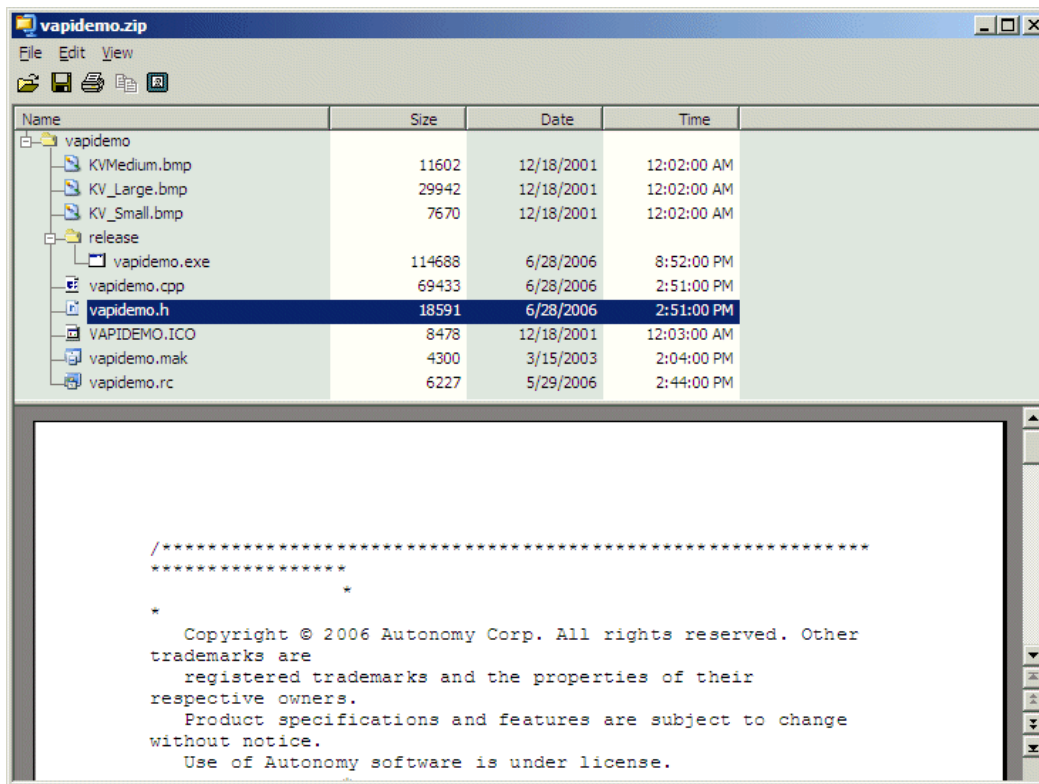
View Archive Files

The Viewing window displays an archive file's hierarchy—showing all the children of the parent file—by using the archive format viewing engine `kvarve.dll`. (You can also display mail files by using the archive format viewing engine. This is optional, and must be configured in the initialization file or registry file. See [View Mail Messages and Mail Stores, on page 44.](#))

When an archive file is opened for viewing, the archive's folders and subfiles are listed in one pane (the file list pane). When a user selects a subfile, the subfile is displayed in another pane (the preview pane). When a user double-clicks a selected subfile, the file's text is displayed in the entire application window. When a subfile is extracted to disk, the user is prompted for a target directory. If the file already exists on disk, a dialog box asks the user whether the file can be overwritten.

The following figure shows a ZIP file displayed in the Viewing API sample program.

Display an archive file with the archive viewing engine



Extract Subfiles to a Viewing Window or Disk

To extract a subfile or files from a container file to a Viewing window or disk

1. Open the container file. See [Open and View a Document, on page 33](#).
2. Use the `VAPIMWP_FILE_CANUNZIP` message to determine whether the selected subfile or files can be extracted.
3. Use the following `VAPIMWP_FILE_UNZIP` message to extract the selected subfile or files to a Viewing window or to disk. If you extract the files to disk, Viewing prompts you for the target directory and overwrite permission.

See the implementation of the **Extract** menu in the View API Demo program (vapidemo).

Display Subfiles in the Preview Pane

NOTE: The preview pane messages apply to the Archive Viewing Engine (kvarcve) only. The Mail Viewing Engine (mailve) does not use these settings.

To display a subfile in the preview pane

1. Use the `VAPIMWP_VIEW_CANPREVIEWPANE` message to determine whether a subfile can be displayed in a preview pane.
2. Use the `VAPIMWP_VIEW_GETPREVIEWPANE` message to determine whether the preview

pane is currently being used.

3. If required, set the size and location of the preview pane using the `ARCHIVE_OPTIONS` structure. This structure is described in `kwoption.h`. See [Change Document Options, on page 36](#).
4. Use the `VAPIMWP_VIEW_SETPREVIEWPANE` message to specify the subfile is displayed in the preview pane.

The View API Demo program (`vapidemo`) demonstrates this functionality.

Set a Password for a Container File

For password-protected ZIP, PST, or NSF files, use the `VAPIMWP_INIT_SETPASSWORD` message to set the password before the file is opened.

View PDF Documents

You can view PDF files with Viewing SDK in one of three ways:

- View the PDF by using the Adobe Acrobat ActiveX control.
- View the PDF by using the Microsoft WebBrowser ActiveX control.
- View an image of each page of the PDF by using a graphic-based PDF reader (`kppdfldr` or `kppdf2ldr`).

By default, Viewing SDK uses the Acrobat ActiveX control to view PDF documents. If you do not want to redistribute the Acrobat Reader with your application, you can use a graphic-based reader instead.

Use the Acrobat ActiveX Control

The Acrobat control is automatically installed with Adobe® Reader® 4.0 or later. To download the Adobe Reader, go to www.adobe.com.

Use the Microsoft WebBrowser ActiveX Control

You can use the Microsoft WebBrowser ActiveX control to view PDF documents. The Microsoft WebBrowser ActiveX control is installed automatically with Microsoft Internet Explorer 3.0 or later. To use the WebBrowser ActiveX control to view PDF documents, follow one of these procedures:

In the `kvsdk.ini` file

1. Open the `kvsdk.ini` file with a text editor. The file is installed in the root of the Windows directory.
2. In the [General] section of the `kvsdk.ini` file, set the `UseHTMLPluginForPDF` parameter to `True`.
3. Pass the highlight or search term in by using `VAPIMWP_INIT_OPENDOCX` (the extended version of `VAPIMWP_INIT_OPEN_DOCUMENT`. Refer to `ihademo.cpp` for details) with the `OpenDocInfo.lpszHighlight` structure. For example:

```
OpenDocInfo.lpszHighLight="search_term";
```

where *search_term* is the highlight or search term.

In the registry file

1. Open `install.reg.txt` in a text editor. The file is installed in the `install\redist` directory, where `install` is the directory in which you installed Viewing SDK.
2. Under the `[HKEY_LOCAL_MACHINE\Software\Autonomy\Viewing SDK\General]` key, set the following parameter:

```
"UseHTMLPluginForPDF"="true"
```

3. Save the file as `install.reg`.
4. Import the file into your Windows system registry.

Use a Graphic-based PDF Reader

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

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

- The `kppdfldr` reader supports highlighting, annotation, and several other features, but also has several graphical limitations.
- The `kppdf2ldr` reader produces high-fidelity raster images but is a viewer only, and does not support highlighting or other features.

Use the `kppdfldr` Reader

The `kppdfldr` graphic-based reader has the following features:

- supports vector images
- supports rotation and scaling
- supports multibyte and bidirectional text

The `kppdfldr` reader has the following limitations:

- Embedded fonts in a PDF file are not translated correctly. They are usually displayed using the question mark (?) replacement character.
- If an unsupported font is encountered during conversion, the default font, Times New Roman, is substituted.
- Supports 180-degree rotation only for raster images.
- Supports the following color spaces: DeviceRGB, DeviceGray, DeviceCMYK, CalGray, and CalRGB color spaces. Indexed color spaces are supported as long as they are used with a supported basic color space.
- Does not support hyperlinks.
- Does not extract summary information (metadata).

Use the kppdf2rdr Reader

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

- Does not support anything beyond viewing, such as highlighting or annotation.
- Does not support PDFs containing XFA forms content.

Specify the Graphic-based Reader

By default, the Acrobat control is used to view PDF documents. To use one of the graphic-based readers to view PDF documents, follow one of these procedures:

In the kvsdk.ini file

1. Open the kvsdk.ini file with a text editor. The file is installed in the root of the Windows directory.
2. In the [VAPI] section of the kvsdk.ini file, change the 200=doc 0 kvaxcc.dll parameter to 200=pic 0 kvpicve.dll.
3. In the [KVPICVE] section, set the following parameter to the graphic-based reader you want to use. Set one of the following values:

- For the kppdfdrdr reader:

200=kppdfdrdr.dll

This is the default setting.

- For the kppdf2rdr reader:

200=kppdf2rdr.dll

In the registry file

1. Open install.reg.txt in a text editor. The file is installed in the *install\redist* directory, where *install* is the directory in which you installed Viewing SDK.
2. Under the [HKEY_LOCAL_MACHINE\Software\Autonomy\Viewing SDK\VAPI] key, change the "200"="doc 0 kvaxcc.dll" parameter to "200"="pic 0 kvpicve.dll".
3. Under the [HKEY_LOCAL_MACHINE\Software\Autonomy\Viewing SDK\KVPICVE] key, set the following parameter to the graphic-based reader you want to use. Set one of the following values:

- For the kppdfdrdr reader:

"200"="kppdfdrdr.dll"

This is the default setting.

- For the kppdf2rdr reader:

"200"="kppdf2rdr.dll"

4. Save the file as install.reg.
5. Import the file into your Windows system registry.

View Microsoft Visio Files

Microsoft Visio files are supported by different readers or components depending on the file version:

- Version 2013 files are supported with ActiveX components included with the free Visio 2013 viewer provided by Microsoft. Image fidelity is supported, but additional features such as highlighting are not. Additional configuration steps are required.
- Version 2003–2010 files are supported with the `kpVSDrdr` reader by default. Image fidelity is supported. If desired, you can use the ActiveX components for these files instead of `kpVSDrdr`; this produces higher quality images but, as with 2013 files, does not support other Viewing features.
- Version 2002 and lower files are supported with the `vsdsr` reader. Image fidelity is not supported.

To enable support for Microsoft Visio 2013 files

1. Download and install the free Microsoft Visio 2013 Viewer from the following website:

<http://www.microsoft.com/en-us/download/details.aspx?id=35811>

2. Update the following entry in the `kvsdk.ini` file:

```
415=doc 0 kvaxcc.dll ; MS Visio 2013
```

To enable the ActiveX solution for Microsoft 2003–2010 files

1. Download and install the free Microsoft Visio 2013 Viewer from the following website:

<http://www.microsoft.com/en-us/download/details.aspx?id=35811>

2. Update the following entries in the `kvsdk.ini` file:

```
;294.6.0.11=prsgfx 0 kvpicve.dll ; MS Visio 2003/2007 (11.0)  
294.6.0.11=doc 0 kvaxcc.dll ; MS Visio 2003/2007 (11.0)
```

To disable the ActiveX functionality for 2003–2010 files and revert to the `kpVSDrdr` reader, update the entries as follows:

```
294.6.0.11=prsgfx 0 kvpicve.dll ; MS Visio 2003/2007 (11.0)  
;294.6.0.11=doc 0 kvaxcc.dll ; MS Visio 2003/2007 (11.0)
```

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 you configure Filter to extract the formula and the formula value, the output might look like this:

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

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

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

To set the extraction option for formulas, add the following lines to the `formats.ini` file:

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

where *option* is one of the following:

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

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

When you enable formula extraction, Filter can extract Microsoft Excel formulas containing the functions listed in [Supported Microsoft Excel functions, below](#):

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()	

Chapter 4: Viewing API Sample Programs

This section describes the sample programs that demonstrate how to use the API.

Overview

The following sample programs are provided for the Viewing API:

helloworldapi, on the next page	vapidemo, on page 62
mfckv, on page 62	rtfdemo, on page 62
pmtdemo, on page 63	filetype, on page 63
ihademo, on page 63	drawdemo, on page 64
uzipdemo, on page 64	

HPE recommends that you review the `helloworldapi` program first to help you get started. It is a simple program that demonstrates the basic functions of the Viewing API.

NOTE: The sample programs are Windows applications, not console applications. In other words, they contain a `WinMain` procedure instead of a `main` procedure.

Compile the Sample Programs

To compile the sample programs, use the makefile provided in each sample program directory. Make sure that the Viewing `include` directory is specified in the include path of the project.

After the executables are compiled and built, you must place them in the `release` subdirectory of each program.

Run the Sample Programs

To run a sample program

1. Install Viewing SDK.
2. Run the sample program from the `release` subdirectory of each sample program.

Viewing SDK Initialization Information

Viewing SDK uses initialization information for its internal operations; for example, to determine which components to load. You can store this information either in the Windows registry or in an initialization file. When you use Viewing SDK you must tell it where to find this information and what form it is in. See [View Initialization Information, on page 23](#) for more information.

`helloworldapi` demonstrates how to use the registry and the `kvsdk.ini` file.

helloworldapi

helloworldapi is a simple program that demonstrates how to use Viewing SDK to display documents in your application. The program creates a Windows application window, and then creates a child window that the Viewing SDK uses to display documents. The Viewing SDK is controlled by sending Windows-style messages to the child window. The set of messages that you send to the child window form the Viewing API (VAPI). The child window is known as a VAPI window. The VAPI messages are described in [Message Parameters, on page 65](#).

Load kvvapi.dll

Before you can create a VAPI window, you must load the VAPI library (kvvapi.dll) by using the LoadLibrary function. In helloworldapi, the library is loaded in the InitializeVAPI function which is called during the processing of the WM_CREATE message.

The library is loaded in the following way:

1. InitializeVAPI calls either the GetPrivateProfileString Windows function or RegQueryValueEx to get the value of HOME in the General section of the kvsdk.ini file or Windows registry.

GetPrivateProfileString gets the value from kvsdk.ini, and RegQueryValueEx gets the value from the registry. The HOME setting specifies the location of the Viewing SDK bin directory. InitializeVAPI demonstrates how to get this setting from both the registry and kvsdk.ini file. By default, InitializeVAPI gets the value from kvsdk.ini.

2. InitializeVAPI then creates the path to the VAPI library and loads it:

```
wsprintf (szDLLPath, TEXT("%s\\%s"), szHome, VAPIDF_VAPI_DLL_NAME);  
hLibVAPI = LoadLibrary (szDLLPath);
```

Create the VAPI Window

After the VAPI library is loaded, the program creates the VAPI window by calling CreateWindow with a class name of VAPIDF_VAPI_WINDOW_CLASS_NAME:

```
hWndVAPI = CreateWindow (VAPIDF_VAPI_WINDOW_CLASS_NAME,  
                        NULL,  
                        WS_CHILD | WS_DISABLED,  
                        rc.left, rc.top, rc.right, rc.bottom,  
                        hWnd,  
                        NULL,  
                        hLibVAPI,  
                        &CreateParams);
```

The last parameter passes in a pointer to VAPI creation information. This creation information is in the structure of type TPVAPICreateParams, and tells VAPI where to locate the initialization settings, and whether they are in the registry or in kvsdk.ini. The helloworldapi program uses the bUseRegistry global variable to specify whether to get the settings from the registry or kvsdk.ini.

By default, `hellovapi` tells VAPI to use the `kvsdk.ini` file that was located with the call to `InitializeVAPI`. The path to the `kvsdk.ini` file is stored in the `szIniFileName` global variable.

To specify the registry

1. Set `uProfileType` to `PROFILEDF_USE_REG`.
2. Set `lpszRegistryName` to the registry name of the Viewing SDK key under the main branch `HKEY_LOCAL_MACHINE\SOFTWARE`. The default is `Autonomy\Viewing SDK`.

To specify an initialization file

1. Set `uProfileType` to `PROFILEDF_USE_INI`.
2. Set `lpszIniFileName` to the location of the initialization file.

NOTE: The strings that you pass in for initialization information must be ASCII strings. If your application is in Unicode, you must convert the strings to ASCII before you pass them in.

```
// Initialize the parameters for the creation of the VAPIwindow.//
memset (&CreateParams, 0, sizeof(TPVAPICreateParams));
if (bUseRegistry)
{
    CreateParams.uProfileType      = PROFILEDF_USE_REG;
    CreateParams.lpszRegistryName = REGISTRY_NAME_ASCII;
}
else
{
    CreateParams.uProfileType      = PROFILEDF_USE_INI;
    CreateParams.lpszIniFileName = szIniFileName;
}
```

Open a Document

To tell VAPI to open a document

1. Create a `TPVAPIOpenDocInfoEx` structure.
2. Send VAPI a `VAPIM_INIT` message with the `wParam` set to `VAPIMWP_INIT_OPENDOCSEX`, and the `lParam` set to the address of the `TPVAPIOpenDocInfoEx` structure.

This is demonstrated in the `OpenDoc` function. It uses a Windows Open dialog box to get the name of a file, and sets the `lpszFilePath` parameter of the `TPVAPIOpenDocInfoEx` structure to this file name. If it is successful, the `VAPIM_INIT` message returns `VAPI_RETURN_SUCCESS`.

NOTE: The file name must be an ASCII string. If your application is in Unicode, you must convert the string to ASCII before passing it in.

```
// Open the document.//
memset (&OpenDocInfo, 0, sizeof(TPVAPIOpenDocInfoEx));

OpenDocInfo.lpszFilePath = szFileName;

lResult = SendMessage (hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_OPENDOCSEX,
```



```
(LPARAM)&OpenDocInfo);
```

helloworldapi.c

```
// helloworldapi.c
This program demonstrates how to use the Viewing API (VAPI) to display documents.//
#ifdef UNICODE
#else
#define _UNICODE
#endif
#include <windows.h>
#include "kvoem.h"
#include "helloworldapi.h"
#define REGISTRY_NAME TEXT("Autonomy\\Viewing SDK")
#define REGISTRY_NAME_ASCII "Autonomy\\Viewing SDK"
HWND      hWndVAPI      = NULL;    //VAPI window
HINSTANCE hLibVAPI      = NULL;    //VAPI library instance
BOOL      bUseRegistry = FALSE;    //Profile type (Set to FALSE for initialization
file)
TCHAR     szIniFileName[MAX_PATH];
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
void OpenDoc (HWND);
BOOL InitializeVAPI (void);
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR szCmdLine,
int nCmdShow)
{
    static TCHAR szAppName[] = TEXT("Hello VAPI demo program");
    HWND        hWnd;
    MSG         msg;
    WNDCLASSEX  wc;
    wc.cbSize    = sizeof (wc);
    wc.style     = 0;
    wc.lpfnWndProc = WndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hInstance = hInstance;
    wc.hIcon     = LoadIcon (NULL, IDI_APPLICATION);
    wc.hCursor   = LoadCursor (NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH) (COLOR_WINDOW + 1);
    wc.lpszMenuName = MAKEINTRESOURCE(IDR_MENU);
    wc.lpszClassName = szAppName;
    wc.hIconSm    = LoadIcon (NULL, IDI_APPLICATION);
    RegisterClassEx (&wc);
    hWnd = CreateWindow (szAppName, TEXT("Hello VAPI"),
        WS_OVERLAPPEDWINDOW,
        CW_USEDEFAULT, CW_USEDEFAULT,
        CW_USEDEFAULT, CW_USEDEFAULT,
```

```
        NULL, NULL, hInstance, NULL);
ShowWindow (hWnd, nCmdShow);
UpdateWindow (hWnd);
while (GetMessage (&msg, NULL, 0, 0))
{
    TranslateMessage (&msg);
    DispatchMessage (&msg);
}
return msg.wParam;
}
LRESULT CALLBACK WndProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg)
    {
        case WM_CREATE:
        {
            TPVAPICreateParams CreateParams;
            RECT rc;
#ifdef UNICODE
                char szName[MAX_PATH];
#endif
                if (!InitializeVAPI())
                {
                    return -1;
                }
                // Initialize parameters for creation of the VAPI window.//
                memset (&CreateParams, 0, sizeof(TPVAPICreateParams));
                if (bUseRegistry)
                {
                    CreateParams.uProfileType      = PROFILEDF_USE_REG;
                    CreateParams.lpszRegistryName = REGISTRY_NAME_ASCII;
                }
                else
                {
                    CreateParams.uProfileType      = PROFILEDF_USE_INI;
#ifdef UNICODE
                        WideCharToMultiByte (CP_ACP, 0, szIniFileName, -1, szName,
                        MAX_PATH, NULL, NULL);
                        CreateParams.lpszIniFileName = szName;
#else
                        CreateParams.lpszIniFileName = szIniFileName;
#endif
                    }
                // Create the VAPI window. //
                GetClientRect (hWnd, &rc);
                hWndVAPI = CreateWindow (VAPIDF_VAPI_WINDOW_CLASS_NAME,
                    NULL,
                    WS_CHILD | WS_DISABLED,
                    rc.left, rc.top, rc.right, rc.bottom,
```

```
        hWnd,
        NULL,
        hLibVAPI,
        &CreateParams);
    return 0;
}
case WM_DESTROY:
    // Destroy the VAPI window. //
    if (hWndVAPI != NULL)
    {
        DestroyWindow (hWndVAPI);
    }
    // Free the VAPI library. //
    if (hLibVAPI != NULL)
    {
        FreeLibrary (hLibVAPI);
    }
    PostQuitMessage (0);
    return 0;
case WM_CLOSE:
    DestroyWindow (hWnd);
    return 0;
case WM_SIZE:
    MoveWindow (hWndVAPI, 0, 0, LOWORD(lParam), HIWORD(lParam),
    TRUE);
    break;
case WM_COMMAND:
    switch (LOWORD(wParam))
    {
    case IDM_OPEN:
        OpenDoc (hWnd);
        return 0;
    case IDM_CLOSE:
        SendMessage (hWnd, WM_CLOSE, 0, 0);
        return 0;
    }
    break;
}
return DefWindowProc (hWnd, uMsg, wParam, lParam);
}
void OpenDoc (HWND hWnd)
{
    LRESULT          lResult;
    OPENFILENAME      ofn;
    TCHAR             szFileName[MAX_PATH];
    TPVAPIOpenDocInfoEx OpenDocInfo;
#ifdef UNICODE
    char szName[MAX_PATH];
#endif
    #endif
```

```
// Get a document name. //
szFileName[0] = TEXT('\0');
memset (&ofn, 0, sizeof(ofn));
ofn.lStructSize = sizeof(OPENFILENAME);
ofn.hwndOwner   = hWnd;
ofn.lpstrFile   = szFileName;
ofn.nMaxFile    = MAX_PATH;
ofn.Flags       = OFN_PATHMUSTEXIST | OFN_HIDEREADONLY |
                  OFN_FILEMUSTEXIST;
if (GetOpenFileName (&ofn) == 0)
{
    return;
}
// Open the document. //
memset (&OpenDocInfo, 0, sizeof(TPVAPIOpenDocInfoEx));
#ifdef UNICODE
    WideCharToMultiByte (CP_ACP, 0, szFileName, -1, szName, MAX_PATH,
        NULL, NULL);
    OpenDocInfo.lpszFilePath = szName;
#else
    OpenDocInfo.lpszFilePath = szFileName;
#endif
lResult = SendMessage (hWndVAPI, VAPIM_INIT,
    VAPIMWP_INIT_OPENDOCCEX, (LPARAM)&OpenDocInfo);
if (lResult != VAPI_RETURN_SUCCESS)
{
    MessageBox (hWnd, TEXT("Unable to view document."),
        TEXT("Hello VAPI"), MB_OK);
    return;
}
return;
}
// Function:InitializeVAPI() //
// Summary: Load and initialize KVVAPI.dll for use with hellovapi. //
BOOL InitializeVAPI (void)
{
    long lResult;
    TCHAR szDLLPath[MAX_PATH];
    TCHAR szHome[MAX_PATH];
    // Get the location of the VAPI DLL.//
    if (bUseRegistry)
    {
        HKEY hKey;
        TCHAR szSubKey[256];
        DWORD dwType;
        DWORD dwcbData;
        // Open the registry key. //
        wsprintf (szSubKey, TEXT("SOFTWARE\\%s\\General"), REGISTRY_NAME);
        lResult = RegOpenKeyEx (HKEY_LOCAL_MACHINE, szSubKey, 0, KEY_READ, &hKey);
    }
```

```
    if (lResult != ERROR_SUCCESS)
    {
        return FALSE;
    }
    dwcbData = sizeof(szHome);
    lResult = RegQueryValueEx (hKey, TEXT("HOME"), NULL, &dwType,
                              (PBYTE)szHome, &dwcbData);
    RegCloseKey (hKey);
    if (lResult != ERROR_SUCCESS)
    {
        return FALSE;
    }
}
else
{
    int    nSize;
    // Get the location of the initialization file. //
    GetWindowsDirectory (szIniFileName, MAX_PATH);
    _tcscat (szIniFileName, TEXT("\\kvsdk.ini"));
    if (GetFileAttributes (szIniFileName) == 0xffffffff)
    {
        return FALSE;
    }
    nSize = GetPrivateProfileString (TEXT("General"), TEXT("HOME"),
    NULL, szHome, MAX_PATH, szIniFileName);
    if (nSize <= 0)
    {
        return FALSE;
    }
}
// Load the VAPI DLL. //
wsprintf (szDLLPath, TEXT("%s\\%s"), szHome, VAPIDF_VAPI_DLL_NAME);
hLibVAPI = LoadLibrary (szDLLPath);
if (hLibVAPI == NULL)
{
    return FALSE;
}
return TRUE;
}
```

hellovapi.h

```
// hellovapi.h
// This file is the main header file for hellovapi.exe
// Resource definitions
#define IDR_MENU            100
#define IDM_OPEN            100
#define IDM_CLOSE          101
```

hellovapi.rc

```
// hellovapi.rc resource script
#include "hellovapi.h"
// Menu
IDR_MENU MENU DISCARDABLE
BEGIN
    POPUP "&File"
        BEGIN
            MENUITEM "&Open",          IDM_OPEN
            MENUITEM "&Exit",          IDM_CLOSE
        END
    END
END
```

vapidemo

The `vapidemo` sample program is a Windows application that demonstrates most of the functionality of the Viewing API. To start the `vapidemo` program, double-click the `vapidemo.exe` file, or type `vapidemo` at a DOS prompt or in the Run dialog box.

NOTE: Menu options in `vapidemo` do not adjust according to the format of the document you view; however, they appear shaded if they do not apply to the format. This is a limitation of the sample program only, not the Viewing SDK.

mfckv

The `mfckv` sample program is a Single Document Interface (SDI) application written with Microsoft Foundation Classes (MFC). To start the `mfckv` program, double-click the `mfckv.exe` file or enter the following command at a DOS prompt or in the Run dialog box:

```
mfckv
```

To open (view) a document, select **Open** from the **File** menu.

rtfdemo

The `rtfdemo` sample program demonstrates how to use the Viewing API to perform conversions of documents to RTF. To start the `rtfdemo` program, enter the following command at a DOS prompt or in the Run dialog box:

```
rtfdemo sourcefile targetfile
```

where *sourcefile* and *targetfile* include the complete path and file name.

prntdemo

The `prntdemo` sample program demonstrates how to use the Viewing API to print documents to a specified printer. To start the `prntdemo` program, enter the following command at a DOS prompt or in the Run dialog box:

```
prntdemo printername,printerdevice,printerport
```

For example:

```
prntdemo \\Calculus\HP Laserjet IIIsi,winspool,NE00:
```

The program displays the **Open** dialog box for you to select the file to print.

filetype

The `filetype` sample program demonstrates how to use the Viewing API to obtain a document's type.

To start the `filetype` program, enter the following command at a DOS prompt or in the Run dialog box:

```
filetype file
```

where *file* includes the complete path and file name.

ihademo

The `ihademo` sample program is a simple Windows application that demonstrates how to index (filter), highlight, and annotate documents.

NOTE: The `ihademo` sample program is intended for word processing documents only. You might encounter limitations if you use it with other formats.

To start the `ihademo` program, double-click the `ihademo.exe` file, or enter the following command at a DOS prompt or in the Run dialog box:

```
ihademo optional_word_to_index
```

To open and view a document, select **Open** from the **File** menu. If you specified a word to index at the command line (the *optional_word_to_index* parameter), all occurrences of the indexed word in the document (index hits) are highlighted when you open a document. To show or hide the index hits, select **Show Hits** from the **View** menu.

To index a document without viewing it, select **Index Only** before you select **Open**. In "Index Only" mode, the document is not displayed; rather, text buffers are returned. The first text buffer returns automatically. To get more text buffers, select **Drive Next Buffer**. The `baseAddress` returned in the text buffer is the starting logical address of the returned text.

To insert an annotation at the cursor position, select **Annotate** from the **View** menu. If no text is selected, the **Annotate** command inserts a bitmap. If any text is selected, the **Annotate** command turns green and the selected text is underlined. You can also annotate by using a double-click. This

method annotates (with a green underline) five characters, starting at the cursor position. If you click the annotation, the annotation text returns in a message box.

To get the current view position (that is, the first and last logical address displayed), select **Get View Position** from the **View** menu.

To get the logical address and page number at the cursor position, select **Current Page Number** from the **View** menu.

To get the text at the cursor position, select **Get Text** from the **View** menu.

drawdemo

The `drawdemo` sample program demonstrates how to draw a page of a word processing, spreadsheet, presentation, or picture file into a supplied Device Context (thumbnail view). The program captures the first few pages (up to `MAX_PAGES_TO_DISPLAY`) of the document in metafiles and later uses these metafiles to draw the pages on the screen.

To start the `drawdemo` program, double-click the `drawdemo.exe` file, or enter the following command at a DOS prompt or in the Run dialog box:

```
drawdemo
```

To draw the first page of a document into a supplied Device Context, select **Open** from the **File** menu.

uzipdemo

The `uzipdemo` sample program demonstrates how to use the Viewing API to extract subfiles from a container file such as a ZIP archive.

To start the `uzipdemo` program, enter the following command at a DOS prompt or in the Run dialog box:

```
uzipdemo sourcefiletargetdirectory
```

where *sourcefile* and *targetdirectory* includes the complete path and file name.

Chapter 5: Message Parameters

This section provides information on the message parameters in the Viewing API.

VAPIM_ANNOTATE

Description

Adds and deletes annotations, and determines whether annotations exist.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI,
            VAPIM_ANNOTATE,
            (WPARAM) wControl,
            (LPARAM) (TPVAPIAnnotation*) lpAnnotation );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
wControl	KV_DEL_ANNOTATION (0) – delete annotation KV_ADD_ANNOTATION (1) – add annotation KV_ANNOTATION_EXISTS (2) – query annotation
lpAnnotation	A pointer to a TPVAPIAnnotation structure that defines the annotation.

Returns

- For KV_DEL_ANNOTATION, SendMessage() returns TRUE if successful; FALSE if the annotation did not exist.
- For KV_ADD_ANNOTATION, SendMessage() returns 0 if successful; 1 if out of memory; 2 if the annotation could not be added because it would cause an overlap with an existing annotation; and 3 if the logical address was invalid.
- For KV_ANNOTATION_EXISTS, SendMessage() returns TRUE if the annotation exists; FALSE otherwise.

Discussion

The size of the bitmap is not relevant.

VAPIM_ENABLEINDEX

Description

Enables index-only mode, also called document filtering. This generates text buffer (VAPINM_TEXTBUFFER) notification messages with document viewing disabled. The first text buffer notification message is generated after a VAPIMWP_INIT_OPEN_DOCUMENT message is sent. To get additional text buffer notification messages in this mode, call the [VAPIM_GETNEXTTEXTBUFFER](#) message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI,
            VAPIM_ENABLEINDEX,
            (WPARAM) n_IndexMode,
            (LPARAM) (TPVAPIHiLiteColor*) lpHiLiteColor );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
n_IndexMode	0 – Combined mode (document view and text buffers) 1 – Text buffers only 2 – Disable index (document view only)
lpHiLiteColor	A pointer to a TPVAPIHiLiteColor structure that defines the highlight color used to signify an index hit.

Returns

SendMessage() returns TRUE if successful; FALSE otherwise.

Discussion

- This message is passed to the VAPI control window to notify the Viewing display engine that a document index is under way. This message *must* be sent before the [VAPIMWP_INIT_OPEN_DOCUMENT](#) message. Check the return value from VAPIMWP_INIT_OPEN_DOCUMENT to make sure

that indexing was really supported.

- Calling this message produces a sequence of VAPINM_TEXTBUFFER notification messages to the calling window—that is, the parent of the VAPI window—as well as enabling the viewing engine to handle highlight and annotation requests. If you are using index-only mode, the VAPI window should be hidden and destroyed when the index is complete. No GDI output is generated and no information is stored to render the document. This results in a faster initial index of the document.

When Viewing is in index-only mode, VAPIM_GETNEXTTEXTBUFFER messages must be used to drive Viewing to obtain VAPINM_TEXTBUFFER notification messages, with the exception of the very first buffer. In other words, after you send a VAPIMWP_INIT_OPEN_DOCUMENT message, you either get back one VAPINM_TEXTBUFFER notification message automatically, or two when there is only one buffer in the file. When you need more, request it.

- When text buffers are no longer necessary, send VAPIM_ENABLEINDEX with n_IndexMode set to 2 and reopen the same document.

VAPIM_GETNEXTTEXTBUFFER

Description

Gets text buffers in index-only mode.

Syntax

```
#include <kvvapi.h>  
SendMessage(hWndVAPI, VAPIM_GETNEXTTEXTBUFFER, 0, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if a text buffer was returned, and FALSE if there are no more text buffers in the document, that is, the end of document is reached.

Discussion

This message is used to generate VAPINM_TEXTBUFFER notification messages when Viewing is in index-only mode (except for the very first text buffer, which comes automatically after a VAPIMWP_INIT_OPEN_DOCUMENT message is sent). The VAPINM_TEXTBUFFER notification messages is received before this message returns.

There might be two notification messages generated, one for the text buffer and one to indicate the end of the document.

VAPIM_GETPAGEFROMLOGICAL

Description

Gets the page number for a logical address.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_GETPAGEFROMLOGICAL, 0L,
            (LPARAM) (long) lLogicalAddress );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lLogicalAddress	A long integer that is the logical address for which to get the page number.

Returns

SendMessage() returns the page number the specified logical address resides on; or -1 on error.

Discussion

For spreadsheets, this message fails if the page containing the specified logical address has not been completely indexed yet.

VAPIM_GETSUMMARYINFO

Description

Gets document metadata, also referred to as summary information.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, STAT wParam,
```

```
(LPARAM) (*KVSummaryInfoEx) pSummaryInfo );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
wParam	0 – get summary information 1 – free summary information
pSummaryInfo	A pointer to a KVSummaryInfoEx structure that contains summary information about the document.

Returns

SendMessage() returns TRUE if successful (pSummaryInfo filled in with valid information, if wParam is 0), FALSE if it fails.

VAPIM_GETTEXT

Description

Gets a text buffer from a specified range.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_GETTEXT, 0L,
            (LPARAM) (TPVAPIGetText*) lpGetText );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpGetText	A pointer to a TPVAPIGetText structure, which defines the text to get.

Returns

SendMessage() returns the number of BYTES it stored in the buffer; or -1 on error.

Discussion

- Send this message to obtain a buffer of text from a specified range. It is assumed that the text buffer is large enough to hold the required number of bytes. The data is not null terminated.
- This message does not wait for a logical address to become valid in the same way as `VAPIM_POSITION` (for non-spreadsheets). This message does not retrieve text across buffer boundaries.
- For spreadsheets, this message also fails if the page containing the entire text buffer—that is, containing the last address in the text buffer—is not completely indexed.

VAPIM_GOTO_PAGE

Description

If indexing is enabled on the document by using the `VAPIM_ENABLEINDEX` message, the document is displayed at the specified page.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_GOTO_PAGE,
            (WPARAM)(int) nPageNumber, 0L );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>nPageNumber</code>	The page to display.

Returns

`SendMessage()` returns `TRUE` if the call succeeds.

Discussion

This message can be used only with word processing files. To use similar functionality with PPT files or the graphic-based PDF reader, see `VAPIMWP_VIEW_GOTOPAGE` , on page 136.

VAPIM_HAVEHILITE

Description

Determines whether there is a Previous or Next highlight relative to the current position.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_HAVEHILITE,
            (WPARAM) (BOOL) bPrevious, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
bPrevious	TRUE checks for a previous highlight, FALSE checks for the next highlight.

Returns

Returns TRUE if there is a Previous or Next highlight (depending on the setting of bPrevious) relative to the current position.

VAPIM_POSITION

Description

Positions the document in the viewing window.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_POSITION, 0L,
            (LPARAM) (TPVAPIPosition*) lpPosition );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpPosition	A pointer to a TPVAPIPosition structure that defines the position.

Returns

SendMessage() returns TRUE on success; or FALSE on error (for example, invalid position).

Discussion

- You can use this message at any time to position the document within the viewing window. If successful, lpPosition->first and lpPosition->last is set on return.
- If lpPosition->position is set to -1, this message fills in only the first and last values, without changing the current view position.
- If the specified position is not processed when this message is called, Viewing takes exclusive control until the position is encountered. In other words, this message does not return until the desired position is set.
- For spreadsheets, if the page containing the specified position has not been indexed at the time this message is called, this message returns FALSE immediately, and does not wait until the page containing the specified position is indexed.

VAPIM_POSITIONHILITE

Description

Changes focus from previous to next highlight.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_POSITIONHILITE,
            (LPARAM) (BOOL) bPrev, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Argument	Description
bPrev	TRUE goes to previous highlight, FALSE goes to next highlight.

Returns

SendMessage() returns TRUE on success; or FALSE on error.

VAPIM_SETCURSOR

Description

Sets the viewing engine cursor.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_SETCURSOR, 0L,
            (LPARAM) (HCURSOR) hCursor );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
hCursor	The handle of the new cursor.

Returns

SendMessage() returns the handle of the active cursor (HCURSOR); or NULL on error.

Discussion

After sending this message to change the cursor, it is assumed you will send another message to change the cursor back to its original shape.

VAPIM_SETHILITE

Description

Highlights a region of text.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_SETHILITE,
            (WPARAM) (int) cbTextToHilite,
            (LPARAM) (long) lLogicalAddress );
```

Arguments

Argument	Description
hWndVAPI	The handle of VAPI window.
cbTextToHilite	An integer that is the number of bytes to highlight.
lLogicalAddress	A long integer that is the logical address from which to start highlighting.

Returns

SendMessage() returns non-zero on success; or zero on error.

Discussion

- This message should be sent to Viewing when a VAPINM_TEXTBUFFER is received, and before processing for that message has returned control to Viewing. When the SendMessage(hWnd, VAPINM_TEXTBUFFER, ..) returns, it is assumed that any highlights have been added to the buffer. There is no limit to the number of highlights that can be added.
- For spreadsheets, this message can also fail if the page containing the entire text region—that is, containing the last address in the text region—is not completely indexed.

VAPIM_SETHILITEOPTIONS

Description

Sets highlight options.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_SETHILITEOPTIONS,
            (LPARAM) (TPVAPIHiLiteOptions) pHiLiteOptions );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
pHiLiteOptions	A pointer to a TPVAPIHiLiteOptions structure that contains information for the highlight options.

Returns

SendMessage() returns TRUE on success; or FALSE on error.

VAPIM_SETINDEXBUFCHARSET

Description

Sets the character set for the returned indexed text buffer.

Syntax

```
#include <kvvapi.h>
SendMessage (g_hWndVAPI, VAPIM_SETINDEXBUFCHARSET, kvcharset, 0 );
```

Arguments

Argument	Description
g_hWndVAPI	The handle of the VAPI window.
kvcharset	A value from the KVCharSet type in kvtypes.h.

Returns

SendMessage() returns TRUE if the call succeeds.

Discussion

If kvcharset is KVCS_UNKNOWN, the character set of the returned buffer is the Windows native character set (for example, KVCS_1252 for an English machine). This is also the default character set on an English machine when VAPIM_SETINDEXBUFCHARSET message is not sent.

VAPIM_SHOWHITS

Description

Shows or hides index hits.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_SHOWHITS,
            (LPARAM) (BOOL) bShowHits, 0 );
```

Arguments

Argument	Description
hWndVAPI	The handle of VAPI window.
bShowHits	TRUE shows hits, FALSE hides hits.

Returns

SendMessage() returns TRUE on success; or FALSE on error.

Discussion

You can send this message at any time to control whether hits are shown.

VAPIM_CONVERT

Description

Converts the currently open document to another format, without requiring the user to respond to the **SaveAs** dialog. To generate the **SaveAs** dialog box, use the VAPIMWP_FILE_SAVEAS message. The TPVAPIConvert structure includes two data members, one a target file and the other a format code, which you can set to RTF, text, or HTML.

Syntax

```
#include <kvvapi.h>
SendMessage( hWndVAPI, VAPIM_CONVERT, 0,
            (LPARAM) (TPVAPIConvert*) lpConvert );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpConvert	A pointer to a TPVAPIConvert structure.

Returns

`SendMessage()` returns TRUE if the conversion succeeds.

Discussion

To make sure that the entire document is opened before the document is converted, open the document with the `bWait` member in the [TPVAPIOpenDocumentInfo](#) structure set to TRUE. Use the [VAPIMWP_CANCONVERT](#) message to determine whether the document has been completely processed and is ready to be converted.

VAPIMWP_CANCONVERT

Description

Determines whether a file can be converted to another format. This is a parameter of the `VAPIM_CONVERT` message.

Syntax

```
#include <kvvapi.h>
BOOL bSupported = FALSE;
SendMessage(hWndVAPI, VAPIM_CONVERT, VAPIMWP_CANCONVERT,
            (LPARAM)&bSupported);
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
bSupported	A pointer to a Boolean variable.

Returns

If `bSupported` is TRUE, the file can be converted. If `bSupported` is FALSE, the file cannot be converted.

Discussion

Some file formats, such as PDF, presentation graphics files, and graphics, cannot be converted to RTF. In these cases, use `VAPIMWP_CANCONVERT` to check whether conversion is possible.

VAPIMWP_DRAW_DRAWPAGE

Description

Draws a page in an area on a device context. This is a parameter of the `VAPIM_DRAW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_DRAW, VAPIMWP_DRAW_DRAWPAGE,
            (LPARAM) (TPVAPIDrawPageInfo*) pDrawInfo );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>pDrawInfo</code>	A pointer to a TPVAPIDrawPageInfo structure that contains information used for drawing. To draw pages in any order, set the <code>bwait</code> parameter in the TPVAPIOpenDocumentInfo structure to <code>FALSE</code> .

Returns

`SendMessage()` returns:

- `VAPI_RETURN_SUCCESS` if the call succeeds.
- `VAPI_RETURN_NOT_INITIALIZED` if the drawing routines have not been initialized.
- `VAPI_RETURN_NO_PAGE` if the requested page does not exist, or is being displayed before all previous pages have been displayed.
- `VAPI_RETURN_NOT_AVAILABLE` if the document does not support this feature (for example, ZIP files, video, audio).
- `VAPI_RETURN_ERROR` if an error has occurred.

Discussion

- Before you send this message, initialize VAPI by sending the [VAPIMWP_DRAW_INIT](#) message, and then open the document by sending the [VAPIMWP_INIT_OPEN_DOCUMENT](#) message.
- Page numbers start at 0. For example, set `uPage` to 0 to draw page 1, and to 1 to draw page 2.
- By default, you must draw pages sequentially: to draw page 3, you must first draw pages 1 and 2. To draw pages in any order, set the `bWait` parameter in the [TPVAPIOpenDocumentInfo](#) structure to `FALSE`.

VAPIMWP_DRAW_DRAWTOFILE

Description

Draws a page of a document to a graphic file (thumbnail). This is a parameter of the `VAPIM_DRAW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_DRAW, VAPIMWP_DRAW_DRAWTOFILE,
            (LPARAM) (TPVAPIDrawFileInfo*) pDrawInfo );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>pDrawInfo</code>	<p>A pointer to a TPVAPIDrawFileInfo structure that contains information used for drawing a page to a file.</p> <p>To draw pages in any order, set the <code>bWait</code> parameter in the TPVAPIOpenDocumentInfo structure to <code>FALSE</code>.</p>

Returns

`SendMessage()` returns:

- `VAPI_RETURN_SUCCESS` if the call succeeds.
- `VAPI_RETURN_NOT_INITIALIZED` if the drawing routines have not been initialized.
- `VAPI_RETURN_NO_PAGE` if the requested page does not exist, or is being displayed before all previous pages have been displayed.
- `VAPI_RETURN_NOT_AVAILABLE` if the document does not support this feature (for example, ZIP files,

video, audio).

- `VAPI_RETURN_ERROR` if an error has occurred.

Discussion

- Before you send this message, initialize VAPI by sending the `VAPIMWP_DRAW_INIT` message, and then open the document by sending the `VAPIMWP_INIT_OPEN_DOCUMENT` message.
- Page numbers start at 0. For example, set `uPageNumbers` to 0 to draw page 1, and to 1 to draw page 2. For word processing documents, pages must be drawn sequentially. For example, to draw page 3, you must first draw pages 0 and 1.

To draw pages in any order, set the `bwait` parameter in the `TPVAPIOpenDocumentInfo` structure to `FALSE`.

VAPIMWP_DRAW_GETPAGECOUNT

Description

Gets the number of pages in a document. This is a parameter of the `VAPIM_DRAW` message.

Syntax

```
#include <kvvapi.h>
SendMessage( hWndVAPI, VAPIM_DRAW, VAPIMWP_DRAW_GETPAGECOUNT,
             (LPARAM) (unsigned int*) pPageCount );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>pPageCount</code>	A pointer to an unsigned <code>int</code> that returns the number of pages in the document.

Returns

`SendMessage()` returns `TRUE` if the call succeeds, in which case `pPageCount` returns the number of pages in the document. `SendMessage()` returns `FALSE` if the call fails.

Discussion

- Before you send this message, initialize VAPI by sending the `VAPIMWP_DRAW_INIT` message, and then open the document by sending the `VAPIMWP_INIT_OPEN_DOCUMENT` message.

- To make sure that the entire document is opened before the page count is retrieved, open the document with the `bWait` member in the [TPVAPIOpenDocumentInfo](#) structure set to `TRUE`. If you do not set `bWait` to `TRUE`, the returned page count might not be accurate.

VAPIMWP_DRAW_GETPAGESIZE

Description

Gets the default size of a page. This is a parameter of the `VAPIM_DRAW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_DRAW, VAPIMWP_DRAW_GETPAGESIZE,
            (LPARAM) (TPVAPIPageSize*) pPageSize );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>pPageSize</code>	A pointer to a TPVAPIPageSize structure which contains information on the page size.

Returns

`SendMessage()` returns:

- `VAPI_RETURN_SUCCESS` if the call succeeds.
- `VAPI_RETURN_NOT_INITIALIZED` if the drawing routines have not been initialized.
- `VAPI_RETURN_NO_PAGE` if the requested page does not exist or is being displayed before all previous pages have been displayed.
- `VAPI_RETURN_NOT_AVAILABLE` if the document does not support this feature (for example, ZIP files, video, audio).
- `VAPI_RETURN_ERROR` if an error has occurred.

Discussion

Before you send this message:

- Initialize VAPI by sending the [VAPIMWP_DRAW_INIT](#) message.
- Open the document by sending the [VAPIMWP_INIT_OPEN_DOCUMENT](#) message.
- Send the [VAPIMWP_DRAW_DRAWPAGE](#) message to draw the page.

VAPIMWP_DRAW_GETWORKBOOKPAGECOUNT

Description

Gets the number of workbook pages in a spreadsheet document. This is a parameter of the VAPIM_DRAW message.

Syntax

```
#include <kvvapi.h>;  
SendMessage( hWndVAPI, VAPIM_DRAW,  
             VAPIMWP_DRAW_GETWORKBOOKPAGECOUNT,  
             (LPARAM) (unsigned int*) pPageCount );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
pPageCount	A pointer to an unsigned int that returns the number of workbook pages in the spreadsheet document.

Returns

SendMessage() returns TRUE if the call succeeds, in which case pPageCount returns the number of workbook pages in the document. SendMessage() returns FALSE if the call fails.

Discussion

Before you send this message, initialize VAPI by sending the [VAPIMWP_DRAW_INIT](#) message and then open the document by sending the [VAPIMWP_INIT_OPEN_DOCUMENT](#) message.

VAPIMWP_DRAW_INIT

Description

Initializes the drawing routines in VAPI. This is a parameter of the VAPIM_DRAW message.

Syntax

```
#include <kvvapi.h>
SendMessage( hWndVAPI, VAPIM_DRAW, VAPIMWP_DRAW_INIT, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds and FALSE if the call fails.

Discussion

You must send this message prior to opening the document, that is, prior to sending the [VAPIMWP_INIT_OPEN_DOCUMENT](#) message.

VAPIMWP_DRAW_SHUTDOWN

Description

Before a new document is opened, this frees up any data from a previously opened document. This is a parameter of the VAPIM_DRAW message.

Syntax

```
SendMessage(g_hWndVAPI, VAPIM_DRAW, DRAW_SHUTDOWN, 0L );
```

Arguments

Argument	Description
g_hWndVapi	The handle of the VAPI window
uMsg	VAPIM_DRAW
wParam	VAPIMWP_DRAW_SHUTDOWN
lParam	Not used, set to 0

Returns

SendMessage() returns:

- VAPI_RETURN_SUCCESS (or TRUE, value 1) if the call succeeds.
- VAPI_RETURN_NOT_INITIALIZED (value 5) if the drawing routines have not been initialized.
- VAPI_RETURN_ERROR (or FALSE, value 0) if the call fails.

Discussion

VAPIM_DRAW messages are used in the sample drawdemo program.

VAPIMWP_EDIT_CANCOPY

Description

Determines whether content is selected in the currently opened document and can be copied to the clipboard. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage( hWndVAPI, VAPIM_EDIT, VAPIMWP_EDIT_CANCOPY,
             (LPARAM) (BOOL*) lpbCanCopy );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanCopy	A pointer to a flag that returns TRUE or FALSE, depending on whether the document contains a selection that you can copy.

Returns

- SendMessage() returns TRUE if the call succeeds, in which case lpbCanCopy returns TRUE or FALSE.
- SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbCanCopy is undefined.

Discussion

Use this message to control the state of a **Copy** menu item or toolbar button.

VAPIMWP_EDIT_CANFIND

Description

Determines whether the document contents can be searched. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_EDIT, VAPIMWP_EDIT_CANFIND,
            (LPARAM) (BOOL*) lpbCanFind );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanFind	A pointer to a flag that returns TRUE or FALSE, depending on whether you can search the document.

Returns

- SendMessage() returns TRUE if the call succeeds, in which case lpbCanFind returns TRUE or FALSE.
- SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbCanFind is undefined.

Discussion

Use this message to control the state of a **Find** menu item or toolbar button.

VAPIMWP_EDIT_CANSELECTALL

Description

Determines whether you can select all items in the document. If lpbCanSelectAll is TRUE, you can select all items in the document by using the [VAPIMWP_EDIT_SELECTALL](#) message. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_EDIT, VAPIMWP_EDIT_CANSELECTALL,
            (LPARAM) (BOOL*) lpbCanSelectAll );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanSelectAll	A pointer to a flag that returns TRUE or FALSE, depending on whether you can select all items in the document.

Returns

SendMessage() returns TRUE if the call succeeds, in which case lpbCanSelectAll returns TRUE or FALSE.

SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbCanSelectAll is undefined.

Discussion

Use this message to control the state of a **Select All** menu item or toolbar button.

VAPIMWP_EDIT_COPY

Description

Copies the current selection in the document to the clipboard. Use the [VAPIMWP_EDIT Cancopy](#) message to determine whether content is selected and can be copied to the clipboard.. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_EDIT, VAPIMWP_EDIT_COPY, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the copy succeeded; FALSE otherwise.

Discussion

Use this message to implement a **Copy** menu item or toolbar button.

VAPIMWP_EDIT_FIND

Description

Searches the document for the specified text. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_EDIT, VAPIMWP_EDIT_FIND,
            (LPARAM) (TPVAPIFindInfo*) lpFindInfo );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpFindInfo	A pointer to a TPVAPIFindInfo structure that contains information about the search text.

Returns

SendMessage() returns TRUE if the find succeeded; FALSE otherwise.

Discussion

Use this message to implement a **Find** menu item or toolbar button.

VAPIMWP_EDIT_FIND_UNICODE

Description

Searches the document for the specified UNICODE text. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_EDIT, VAPIMWP_EDIT_FIND_UNICODE,
            (LPARAM) (TPVAPIFindInfo*) lpFindInfo );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpFindInfo	A pointer to a TPVAPIFindInfo structure that contains information about the text to search for.

Returns

SendMessage() returns TRUE if the find succeeded; FALSE otherwise.

Discussion

Use this message to implement a **Find** UNICODE menu item or toolbar button.

VAPIMWP_EDIT_GETFINDTEXT

Description

Gets the currently selected text in the document. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_EDIT, VAPIMWP_EDIT_GETFINDTEXT,
            (LPARAM) (HGLOBAL*) lphgFindText );
```


Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lphgFindText	A pointer to an HGLOBAL handle, which returns the currently selected text if any text is selected. You must GlobalLock() this handle before using it, and must GlobalFree() it afterwards. However, you should not GlobalAlloc() this handle, because the Viewer does this.

Returns

SendMessage() returns TRUE if the find text was returned successfully; FALSE otherwise.

Discussion

Use this message to set the default text in the **Find** dialog box for a **Find** menu item or toolbar button.

VAPIMWP_EDIT_SELECTALL

Description

Selects all content in the currently opened document. Use the [VAPIMWP_EDIT_CANSELECTALL](#) message to determine whether you can select all content in the document. This is a parameter of the VAPIM_EDIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(g_hWndFocus, VAPIM_EDIT, VAPIMWP_EDIT_SELECTALL, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
uMsg	VAPIM_EDIT
wParam	VAPIMWP_EDIT_SELECTALL
lParam	Not used, set to 0

Returns

`SendMessage()` returns `TRUE` if the selection succeeded; `FALSE` otherwise.

Discussion

Use this message to implement a **Select All** menu item or toolbar button.

VAPIMWP_FILE_CANSAVEAS

Description

Determines whether the document can be saved and converted. This is a parameter of the `VAPIM_FILE` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_FILE, VAPIMWP_FILE_CANSAVEAS,
            (LPARAM) (BOOL*) lpbCanSaveAs );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpbCanSaveAs</code>	A pointer to a flag that returns <code>TRUE</code> or <code>FALSE</code> , depending on whether you can save and convert the document.

Returns

`SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbCanSaveAs` returns `TRUE` or `FALSE`.

`SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanSaveAs` is undefined.

Discussion

Use this message to control the state of a **Save As** menu item or toolbar button.

VAPIMWP_FILE_CANUNZIP

Description

If a container file is open and there are subfiles selected in the file, this parameter determines whether the subfile or files can be extracted. This is a parameter of the VAPIM_FILE message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_FILE, VAPIMWP_FILE_CANUNZIP,
            (LPARAM) (BOOL*) lpbCanUnZip );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanUnZip	A pointer to a flag that returns TRUE or FALSE depending on whether you can extract the subfiles.

Returns

SendMessage() returns TRUE if the call succeeds, in which case lpbCanUnZip returns TRUE or FALSE.
SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbCanUnZip is undefined.

Discussion

Use this message to control the state of an **Extract** menu item or toolbar button.

VAPIMWP_FILE_CLOSE

Description

Closes the currently opened document. It is not necessary to use this message, because issuing a second VAPIMWP_INIT_OPEN_DOCUMENT message automatically closes the currently opened document. This is a parameter of the VAPIM_FILE message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_FILE, VAPIMWP_FILE_CLOSE, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE on success; FALSE otherwise.

VAPIMWP_FILE_SAVEAS

Description

Saves the current document in another format through a **Save As** dialog box. This is a parameter of the VAPIM_FILE message. See [Save a Document, on page 34](#) and [Convert a Document, on page 34](#).

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_FILE, VAPIMWP_FILE_SAVEAS, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE on success; FALSE otherwise.

Discussion

Use this message to implement a **Save As** menu item or toolbar button. You can save the document in its current format, or use Viewing conversions to convert it.

VAPIMWP_FILE_UNZIP

Description

Use this message to extract selected subfiles in a container file either to disk or to a Viewing window. Container file types include ZIP, TAR, or PST. This prompts the user to specify a target directory and password (if required).

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_FILE, VAPIMWP_FILE_UNZIP,
            (LPARAM) (BOOL) bUnzipToDisk );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
bUnZipToDisk	A flag that is TRUE to extract to disk, FALSE (the default) to extract and view.

Returns

SendMessage() returns TRUE on success; FALSE otherwise.

Discussion

Use this message to implement an **Extract** menu or toolbar button. You can extract the selected file to disk, or extract and view it.

VAPIMWP_INIT_GETCHARSET

Description

Gets the character set of the VAPI window. This is a parameter of the VAPIM_INIT message.

Syntax

```
SendMessage(g_hWndFocus, VAPIM_INIT, VAPIMWP_INIT_GETCHARSET, 0)
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
uMsg	VAPIM_INIT
wParam	VAPIMWP_INIT_GETCHARSET
lParam	Not used, set to 0

Returns

This message returns a value from the `KVCharSet` type (see the `kvtypes.h` file for a description) corresponding to the character set that VAPI is using for viewing the document. You can use this message to control the state of a menu item or toolbar button that allows a user to select a character encoding for viewing.

Discussion

- `vapidemo.cpp` demonstrates how to use the message.
- A relevant message is `VAPIMWP_INIT_GETAUTOSELECT`.

VAPIMWP_INIT_GETDESCRIP

Description

Gets a description of the format of the currently opened document. You must send this *after* the `VAPIM_INIT_OPEN_DOCUMENT` message returns. This is a parameter of the `VAPIM_INIT` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_GETDESCRIP,
            (LPARAM) (char*) lpszDescription );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpszDescription	A pointer to a Pascal string that returns a description of the format of the currently opened document.

Returns

SendMessage() returns TRUE on success; FALSE otherwise.

VAPIMWP_INIT_GETDOCCLASS

Description

Indicates the general class to which the currently opened document belongs. This is a parameter of the VAPIM_INIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_GETDOCCLASS,
            (LPARAM) (int*) lpnClass );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpnClass	A pointer to an integer that returns the document class: <ul style="list-style-type: none">• 1 - Text document (ASCII)• 2 - Word processor document (WP)• 3 - Spreadsheet document (SS)• 4 - Image (Image)• 5 - Multimedia document (MM)• 6 - Fax document (FAX)• 7 - Presentation (PG)• 8 - Archive• 9 - Other

Returns

SendMessage() returns TRUE if successful (that is, if lpnClass contains valid information); otherwise it returns FALSE.

VAPIMWP_INIT_GETDOCFORMAT

Description

Gets document format information. This is a parameter of the VAPIM_INIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_GETDOCFORMAT,
            (LPARAM) (*ADDOCINFO) pDocInfo );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
pDocInfo	A pointer to the ADDOCINFO structure that contains information about the document format.

Returns

SendMessage() returns TRUE if successful (pDocInfo is filled in with valid information); otherwise it returns FALSE.

Discussion

If you want to get format information without viewing the document, set the VAPIDF_FLAGS_OPEN_VAPI_ONLY flag in the nFlags member of the [TPVAPIOpenDocumentInfo](#) structure. This structure is supplied when you open the document by using the VAPIMWP_INIT_OPEN_DOCUMENT message parameter.

VAPIMWP_INIT_GETFILENAME

Description

Gets the file name of the current document. This is a parameter of the VAPIM_INIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_GETFILENAME,
            (LPARAM) (char*) lpstzFileName );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpstzFileName	A pointer to a Pascal string that returns the file name string.

Returns

SendMessage() returns TRUE on success; otherwise it returns FALSE.

Discussion

This message returns the file name (not the full path), so the file name can be used in the window title. This message must be sent after the document is opened, otherwise the file name does not exist yet. If the file is a container file, such as a ZIP, TAR, or PST file, this file name is the name of an extracted subfile.

VAPIMWP_INIT_GETHWNDVIEWER

Description

Gets the handle of the Viewer window. This is a parameter of the VAPIM_INIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_GETHWNDVIEWER,
            (LPARAM) (HWND*) lphWndViewer );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lphWndViewer	A pointer to a handle that returns the Viewer window handle.

Returns

`SendMessage()` returns `TRUE` if the Viewer window exists; otherwise it returns `FALSE`.

Discussion

You must send this message *after* you send the `VAPIMWP_INIT_OPEN_DOCUMENT` message to open the document, otherwise the Viewer window does not exist yet.

VAPIMWP_INIT_JUMPTOFIRSTHILITE

Description

Jumps to the first highlight. You must send this message before you send the `VAPIM_INIT_OPEN_DOCUMENT` message. This applies only when you use XML documents created with the Verity Developer's Kit (VDK) to specify highlights. This is a parameter of the `VAPIM_INIT` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_JUMPTOFIRSTHILITE,
           0L );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.

Returns

`SendMessage()` returns `TRUE` on success, or `FALSE` on error.

VAPIMWP_INIT_OPEN_DOCUMENT

Description

Opens a document for one of the following operations:

- viewing
- determining a document format without viewing

- printing/converting/saving without viewing

This is a parameter of the VAPIM_INIT message. See [Open and View a Document](#), on page 33.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_OPEN_DOCUMENT,
            (LPARAM) (TPVAPIOpenDocumentInfo*) pOpenDocumentInfo );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
pOpenDocumentInfo	A pointer to a TPVAPIOpenDocumentInfo structure that contains information about the document to be opened.

Returns

- `SendMessage()` returns `VAPI_RETURN_SUCCESS` if the document opening is successfully initiated.
- If there is no viewer for the specified document, `SendMessage()` returns `VAPI_RETURN_NO_VIEWER`.
- If the format of the document could not be determined, `SendMessage()` returns `VAPI_RETURN_UNKNOWN_FORMAT`.
- If the document is password protected, `SendMessage()` returns `VAPI_RETURN_PASSWORD_PROTECTED`.
- If the drawing routines have not been initialized, `SendMessage()` returns `VAPI_RETURN_NOT_INITIALIZED`.
- If the requested page does not exist, or is being displayed before all previous pages have been displayed, `SendMessage()` returns `VAPI_RETURN_NO_PAGE`.
- If the document does not support this feature (for example, ZIP files, video, audio), `SendMessage()` returns `VAPI_RETURN_NOT_AVAILABLE`.
- If the KeyView license is invalid, `SendMessage()` returns `VAPI_RETURN_INVALID_LICENSE_KEY`.
- If the KeyView license is expired, `SendMessage()` returns `VAPI_RETURN_EXPIRED_LICENSE_KEY`.
- If the input file or stream is invalid or corrupt, `SendMessage()` returns `VAPI_RETURN_BAD_INPUT`.
- Any other error condition causes `VAPI_RETURN_ERROR` to be returned.

Discussion

- This message initiates the document opening and returns before the document opening is complete unless you set the `bWait` parameter in the `TPVAPIOpenDocumentInfo` structure to `TRUE`.
- This message produces several notification messages:

- One or more VAPINMWP_INIT_OPENDOCDONE notification messages are received to report the status of the document opening.
- In addition, if the document to be opened contains pages, a VAPINMWP_INIT_PAGENUMBER notification message is received to report the current page number.
- Finally, if the document to be opened contains objects (for example, a spreadsheet document containing pages), a VAPINMWP_MULTIOBJ_OBJNAME notification message is received to report the current object name (for example, the name of the current spreadsheet page).
- To open a document to view as text, set the `bViewAsText` parameter in the `TPVAPIOpenDocumentInfo` structure to `TRUE`.
- To open and process a document (print, convert, and so on) without viewing, set the `nFlags` parameter in the `TPVAPIOpenDocumentInfo` structure to `VAPIDF_FLAGS_OPEN_WITHOUT_VIEW`. This flag tells VAPI to create a hidden Viewer window. You must also set the `bWait` parameter in the `TPVAPIOpenDocumentInfo` structure to `TRUE`, except when you use APIs to draw documents without viewing. See [Draw a Page, on page 37](#) for more information.

For example, to print a document without viewing, open the document with the `bWait` parameter set to `TRUE` and the `nFlags` parameter set to `VAPIDF_FLAGS_OPEN_WITHOUT_VIEW`, and then send the `VAPIMWP_PRINT_PRINT` message.
- To open a document and return format information without viewing, set the `nFlags` parameter in the `TPVAPIOpenDocumentInfo` structure to `VAPIDF_FLAGS_OPEN_VAPI_ONLY`. This flag does not create a Viewer window.
- To make sure that a document is fully processed before an operation (such as printing, converting, or searching) is performed, set the `bWait` parameter in the `TPVAPIOpenDocumentInfo` structure to `TRUE`. This is useful when you want to use an operation immediately after opening the document.

VAPIMWP_INIT_SETPASSWORD

Description

Sets the password to use to open a password-protected file before the file is opened. Currently, you can use this to set a password for ZIP and PST files. This is a parameter of the `VAPIM_INIT` message.

Syntax

```
SendMessage (hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_SETPASSWORD,  
            (LPARAM) (LPCTSTR) pPasswordInfo );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>pPasswordInfo</code>	A pointer to a password string.

Returns

`SendMessage()` returns TRUE if successful, or FALSE if it fails.

Discussion

- For password-protected PST files, you must call this message before the `VAPIMWP_INIT_OPEN_DOCUMENT` message.
- For password-protected ZIP files, you can call this message after the `VAPIMWP_INIT_OPEN_DOCUMENT`, but you must call it before the protected subfile is extracted or viewed.
- Unicode passwords are not supported.

VAPIMWP_INIT_SETSRCCHARSET

Description

Sets the source character set of a document. This is a parameter of the `VAPIM_INIT` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_SETSRCCHARSET,
            (LPARAM) eCharset );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>eCharset</code>	A value from the enumerated type <code>KVCharSet</code> . See the <code>kvtypes.h</code> file for a description.

Returns

`SendMessage()` returns TRUE if successful, or FALSE if it fails.

Discussion

This message is used to specify the character set for documents when the character set cannot be determined by Viewing, such as in the case of plain text documents.

VAPIMWP_INIT_SETTRGCHARSET

Description

Sets the target character set of a document. This is a parameter of the VAPIM_INIT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_INIT, VAPIMWP_INIT_SETTRGCHARSET,
            (LPARAM) eCharset );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
eCharset	A value from the enumerated type KVCharSet. See the kvtypes.h file for a description.

Returns

SendMessage() returns TRUE if successful, or FALSE if it fails.

Discussion

This message forces the character set Viewing uses to display a document. For example, this allows Japanese documents to be accurately displayed on an English Windows machine if the Japanese fonts are available.

VAPIMWP_MULTIOBJ_CANMULTIOBJ

Description

Determines whether the document contains multiple objects. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the VAPIM_MULTIOBJ message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_MULTIOBJ,
```

```
VAPIMWP_MULTIOBJ_CANMULTIOBJ,  
(LPARAM) (BOOL*) lpbCanMultiObj );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanMultiObj	A pointer to a flag that returns TRUE or FALSE, depending on whether the document contains multiple objects.

Returns

- `SendMessage()` returns TRUE if the call succeeds, in which case `lpbCanMultiObj` returns TRUE or FALSE.
- `SendMessage()` returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanMultiObj` is undefined.

Discussion

Use this message to control the state of a **Next Object** or **Previous Object** menu item or toolbar button.

VAPIMWP_MULTIOBJ_CANNEXTOBJ

Description

Determines whether the next object can be selected in a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the `VAPIM_MULTIOBJ` message.

Syntax

```
#include <kvvapi.h>  
SendMessage(hWndVAPI, VAPIM_MULTIOBJ, VAPIMWP_MULTIOBJ_CANNEXTOBJ,  
            (LPARAM) (BOOL*) &lpbCanNextObj );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanNextObj	A pointer to a flag that returns TRUE or FALSE, depending on whether the document can select the next objects.

Returns

`SendMessage()` returns `TRUE` if the current object can be changed to the next object; otherwise it returns `FALSE`.

Discussion

Use this message to control the state of a **Next Object** menu item or toolbar button.

VAPIMWP_MULTIOBJ_CANPREVOBJ

Description

Determines whether the previous object can be selected in a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the `VAPIM_MULTIOBJ` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_MULTIOBJ, VAPIMWP_MULTIOBJ_CANPREVOBJ,
            (LPARAM) (BOOL*) &lpbCanPrevObj );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpbCanPrevObj</code>	A pointer to a flag that returns <code>TRUE</code> or <code>FALSE</code> , depending on whether the document can select the previous object.

Returns

`SendMessage()` returns `TRUE` if the current object can be changed to the previous object; otherwise it returns `FALSE`.

Discussion

Use this message to control the state of a **Previous Object** menu item or toolbar button.

VAPIMWP_MULTIOBJ_CANSETCURRENTOBJ

Description

Determines whether the target object can be selected in a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the VAPIM_MULTIOBJ message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_MULTIOBJ,
             VAPIMWP_MULTIOBJ_CANSETCURRENTOBJ, (LPARAM) (int)
             nTargetObj);
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
nTargetObj	A zero-based integer that is the target object.

Returns

SendMessage() returns TRUE if the current object can be set to the target object; otherwise it returns FALSE.

Discussion

Use this message to control the state of a **Set Current Object** menu item or toolbar button.

VAPIMWP_MULTIOBJ_GETOBJCOUNT

Description

Get the total number of objects in a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the VAPIM_MULTIOBJ message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_MULTIOBJ,
             VAPIMWP_MULTIOBJ_GETOBJCOUNT, (LPARAM) (int)
             lpbTotalObj);
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbTotalObj	A pointer to the total number of objects.

Returns

SendMessage() returns TRUE.

Discussion

You can use this message to implement an object count item.

VAPIMWP_MULTIOBJ_NEXTOBJ

Description

Changes the current object to the next object in a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the VAPIM_MULTIOBJ message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_MULTIOBJ, VAPIMWP_MULTIOBJ_NEXTOBJ, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

`SendMessage()` returns `TRUE` if the current object was changed to the next object; otherwise it returns `FALSE`.

Discussion

You can use this message to implement a **Next Object** menu item or toolbar button. This message generates a `VAPIMWP_MULTIOBJ_OBJNAME` notification message that reports the new object name.

When viewing spreadsheets or presentations in index mode, this message fails if the next page is not completely indexed yet, or if the current page is the last page. In other words, the program does not allow the last page to wrap around to the first page, as it does in non-index mode.

VAPIMWP_MULTIOBJ_OBJNAME

Description

Gets the current object name for a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the `VAPIM_MULTIOBJ` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_MULTIOBJ, VAPIMWP_MULTIOBJ_OBJNAME,
            (LPARAM) (char*) lpstzObjectName );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpstzObjectName</code>	A pointer to a Pascal string that returns the object name string.

Returns

`SendMessage()` returns `TRUE` on success; otherwise it returns `FALSE`.

Discussion

You can use this message only with multiple-object documents.

VAPIMWP_MULTIOBJ_PREVOBJ

Description

Changes the current object to the previous object in a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the VAPIM_MULTIOBJ message.

Syntax

```
#include <kvvapi.h>  
SendMessage(hWndVAPI, VAPIM_MULTIOBJ, VAPIMWP_MULTIOBJ_PREVOBJ, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the current object was changed to the previous object; otherwise it returns FALSE.

Discussion

- You can use this message to implement a **Previous Object** menu item or toolbar button. This message generates a VAPIMWP_MULTIOBJ_OBJNAME notification message that reports the new object name.
- When viewing spreadsheets or presentations in index mode, this message fails if the next page has not been completely indexed yet, or if the current page is the first page. It does not allow the first page to wrap around to the last page, as it does in non-index mode.

VAPIMWP_MULTIOBJ_SETCURRENTOBJ

Description

Changes the current object to the target object in a multiple-object document. See [Change the Current Object in a Document, on page 40](#). This is a parameter of the VAPIM_MULTIOBJ message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_MULTIOBJ,
              VAPIMWP_MULTIOBJ_SETCURRENTOBJ, (LPARAM) (int)
              nTargetObj;)
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
nTargetObj	A zero-based integer that is the target object.

Returns

SendMessage() returns TRUE if the current object was changed to the target object; otherwise it returns FALSE.

Discussion

You can use this message to implement a **Set Current Object** menu item or toolbar button. This message generates a VAPIMWP_MULTIOBJ_OBJNAME notification message that reports the new object name.

VAPIMWP_OPTIONS_GETOPTIONS_EX

Description

Gets the document options. Document options control display elements such as window size, zoom settings, margin size, scaling, and revision tracking information. Options are defined for each file type category (for example, spreadsheets, multimedia, and word processing). This is a parameter of the VAPIM_OPTIONS message. See [Change Document Options, on page 36](#).

Syntax

```
#include <kvvapi.h>
#include <kwoption.h>
SendMessage(hWndVAPI, VAPIM_OPTIONS,
              VAPIMWP_OPTIONS_GETOPTIONS_EX,
              (LPARAM) (ALL_OPTIONS_EX*) lpAllOptions );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpAllOptions	A pointer to an ALL_OPTIONS_EX structure that returns the document options.

Returns

`SendMessage()` returns `TRUE` if the call succeeds, in which case `lpAllOptions` returns the options.

`SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpAllOptions` is undefined.

Discussion

Use this message to get the options for the document. Only the options for the current document type are returned, not those for all document types.

VAPIMWP_OPTIONS_SETOPTIONS_EX

Description

Sets the current document options. Document options control display elements such as window size, zoom settings, margin size, and scaling. Options are defined for each file type category (for example, spreadsheets, multimedia, and word processing). This is a parameter of the `VAPIM_OPTIONS` message. See [Change Document Options, on page 36](#).

Syntax

```
#include <kvvapi.h>
#include <kwoption.h>
SendMessage(hWndVAPI, VAPIM_OPTIONS,
             VAPIMWP_OPTIONS_SETOPTIONS_EX,
             (LPARAM) (ALL_OPTIONS_EX*) lpAllOptions );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Argument	Description
lpAllOptions	A pointer to an ALL_OPTIONS_EX structure that shows the document options.

Returns

`SendMessage()` returns `TRUE` if the call succeeds; otherwise it returns `FALSE`.

Discussion

Use this message to set the options for the document. This message does not save the document options in the registry. In addition, this message sets the document options for the current document and document type only. In other words, it initializes the in-memory options of the current Viewer.

VAPIMWP_PRINT_ANNOTATIONS

Description

Specifies whether annotations are included in the printed output. To print a document, use either the [VAPIMWP_PRINT_PRINT](#), or [VAPIMWP_PRINT_PRINTTOPRINTER](#) message. This is a parameter of the [VAPIM_PRINT](#) message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_ANNNOTATIONS,
            (LPARAM) (BOOL) bPrintAnnotations );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
bPrintAnnotations	If <code>TRUE</code> , the document is printed with annotations. If <code>FALSE</code> , the document is printed without annotations.

Returns

`SendMessage()` returns `TRUE` if the call succeeds.

VAPIMWP_PRINT_CANPRINT

Description

Determines whether the document can be printed. This is a parameter of the VAPIM_PRINT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_CANPRINT,
            (LPARAM) (BOOL*) lpbCanPrint );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanPrint	A pointer to a flag that returns TRUE or FALSE, depending on whether the document can be printed.

Returns

- SendMessage() returns TRUE if the call succeeds, in which case lpbCanPrint returns TRUE or FALSE.
- SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbCanPrint is undefined.

Discussion

Use this message to control the state of a **Print** menu item or toolbar button.

VAPIMWP_PRINT_PAGESETUP

Description

Sets up the print page scaling for a spreadsheet. This is a parameter of the VAPIM_PRINT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_PAGESETUP, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Print Page Setup** menu item or toolbar button.

VAPIMWP_PRINT_PRINT

Description

Prints the current document by calling the common **Print** dialog box. This is a parameter of the VAPIM_PRINT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_PRINT, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

- This message prints the current document by calling the common **Print** dialog box to set the printer parameters. You can use this message to implement a **Print** menu item or toolbar button.
- When printing in an application that is a Windows service, a default printer must be installed for the user account using the application.
- To make sure that the entire document is opened before the document is printed, open the document with the `bWait` member in the [TPVAPIOpenDocumentInfo](#) structure set to `TRUE`. Use the [VAPIMWP_PRINT_CANPRINT](#) message to determine whether the document has been completely processed and is ready to be printed.

VAPIMWP_PRINT_PRINTHEADER

Description

Specifies whether a print header appears at the top of the printed output. The print header consists of a left-justified file name and a right-justified page number followed by the page-length on the next line.

You can change the file name value by using the [VAPIMWP_PRINT_SETPRINTNAME](#) message. This is a parameter of the `VAPIM_PRINT` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_PRINTHEADER,
            (LPARAM) (BOOL) bPrintHeaders );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>bPrintHeaders</code>	If <code>TRUE</code> , the document is printed with a header. If <code>FALSE</code> , the document is printed without a header.

Returns

`SendMessage()` returns `TRUE` if the call succeeds.

VAPIMWP_PRINT_PRINTSETUP

Description

Opens a standard **Print Setup** dialog box. This is a parameter of the VAPIM_PRINT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_PRINTSETUP, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Allows the user to select general printing options, including the printer, page size, and page orientation.

VAPIMWP_PRINT_PRINTTOPD

Description

Sets the standard Windows print options for printing files. This is a parameter of the VAPIM_PRINT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_PRINTTOPD,
            (LPARAM) (PRINTDLG*) lpPD );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpPD	A pointer to a Windows PRINTDLG structure.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

VAPIMWP_PRINT_PRINTTOPRINTER

Description

Prints the document to the specified printer. This is a parameter of the VAPIM_PRINT message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_PRINTTOPRINTER,
            (LPARAM) (LPCSTR*) lpzPrinterDriver );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpzPrinterDriver	A string that is the name of the printer driver, or NULL for the default printer. This string must be of the form printername, printerdevice, printerport. For example: \\Calculus\HP LaserJet IIISi,winspool,NE00:.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

- This message prints to the specified printer without calling the common **Print** dialog box. You can use this message to implement a **Print** menu item or toolbar button.

- When printing in an application that is a Windows service, a default printer must be installed for the user account using the application.
- To make sure that the entire document is opened before the document is printed, open the document with the `bWait` member in the `TPVAPIOpenDocumentInfo` structure set to `TRUE`. Use the `VAPIMWP_PRINT_CANPRINT` message to determine whether the document has been completely processed and is ready to be printed.

VAPIMWP_PRINT_SETPRINTNAME

Description

Used in conjunction with `VAPIMWP_PRINT_PRINTHEADER`, this message replaces the default file name field of the print header with a specified string. This is a parameter of the `VAPIM_PRINT` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_PRINT, VAPIMWP_PRINT_SETPRINTNAME,
            (LPARAM) (char*) szPrintName );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>szPrintName</code>	A string used to replace the file name field of the print header.

Returns

`SendMessage()` returns `TRUE` if the call succeeds; otherwise it returns `FALSE`.

VAPIMWP_VIEW_CANASPECTRATIO

Description

Determines whether the document supports an aspect ratio. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANASPECTRATIO,
```

```
(LPARAM) (BOOL*) lpbCanAspectRatio );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanAspectRatio	A pointer to a flag that returns TRUE or FALSE, depending on whether the document supports an aspect ratio.

Returns

- `SendMessage()` returns TRUE if the call succeeds, in which case `lpbCanAspectRatio` returns TRUE or FALSE.
- `SendMessage()` returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanAspectRatio` is undefined.

Discussion

Use this message to control the state of an **Aspect Ratio** menu item or toolbar button.

VAPIMWP_VIEW_CANDECREASEFONT

Description

Determines whether the document font can be decreased. If `lpbCanDecreaseFon` is TRUE, the font size can be decreased by using [VAPIMWP_VIEW_DECREASEFONT](#). This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANDECREASEFONT,
            (LPARAM) (BOOL*) lpbCanDecreaseFont );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanDecreaseFont	A pointer to a flag that returns TRUE or FALSE, depending on whether the document font size can be decreased.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbCanDecreaseFont` returns `TRUE` or `FALSE`.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanDecreaseFont` is undefined.

Discussion

Use this message to control the state of a **Decrease Font** menu item or toolbar button.

VAPIMWP_VIEW_CANFITTOWINDOW

Description

Determines whether the document can be magnified to fit the document selection to the window. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANFITTOWINDOW,
            (LPARAM) (BOOL*) lpbCanFitToWindow );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpbCanFitToWindow</code>	A pointer to a flag that returns <code>TRUE</code> or <code>FALSE</code> , depending on whether the document has a selection, and whether the document can be magnified to fit the selection to the window.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbCanFitToWindow` returns `TRUE` or `FALSE`.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanFitToWindow` is undefined.

Discussion

Use this message to control the state of a **Magnify** menu item or toolbar button.

VAPIMWP_VIEW_CANGOTO

Description

Determines whether the document can go to a specified page or slide. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANGOTO,
            (LPARAM) (BOOL*) lpbCanGoTo );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanGoTo	A pointer to a flag that returns TRUE or FALSE, depending on whether the document can go to a specified page or slide.

Returns

SendMessage() returns TRUE if the call succeeds, in which case lpbCanGoTo returns TRUE or FALSE.

Discussion

- Use this message to control the state of a **Go To Page** menu item or toolbar button.
- To implement a **Go To Page** menu item or toolbar button, use the [VAPIMWP_VIEW_GOTOPAGE](#) message.
- This message can be used only with word processing and presentation files.

VAPIMWP_VIEW_CANGRIDLINES

Description

Determines whether the document supports gridlines. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANGRIDLINES,
            (LPARAM) (BOOL*) lpbCanGridlines );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanGridlines	A pointer to a flag that returns TRUE or FALSE, depending on whether the document supports gridlines.

Returns

- `SendMessage()` returns TRUE if the call succeeds, in which case `lpbCanGridlines` returns TRUE or FALSE.
- `SendMessage()` returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanGridlines` is undefined.

Discussion

Use this message to control the state of a **Toggle Gridlines** menu item or toolbar button.

VAPIMWP_VIEW_CANINCREASEFONT

Description

Determines whether the document font can be increased. If `lpbCanIncreaseFont` is TRUE, the font size can be increased by using the `VAPIMWP_VIEW_INCREASEFONT` message. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANINCREASEFONT,
            (LPARAM) (BOOL*) lpbCanIncreaseFont );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanIncreaseFont	A pointer to a flag that returns TRUE or FALSE, depending on whether the document font size can be increased.

Returns

- `SendMessage()` returns TRUE if the call succeeds, in which case `lpbCanIncreaseFont` returns TRUE or FALSE.
- `SendMessage()` returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanIncreaseFont` is undefined.

Discussion

Use this message to control the state of an **Increase Font** menu item or toolbar button.

VAPIMWP_VIEW_CANINVERT

Description

Determines whether the document colors can be inverted. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANINVERT,
            (LPARAM) (BOOL*) lpbCanInvert );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanInvert	A pointer to a flag that returns TRUE or FALSE, depending on whether the document colors can be inverted (for example, from black to white and white to black).

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbCanInvert` returns `TRUE` or `FALSE`.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanInvert` is undefined.

Discussion

Use this message to control the state of an **Invert** menu item or toolbar button.

VAPIMWP_VIEW_CANLAYOUT

Description

Determines whether the document layout can be changed. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANLAYOUT,
            (LPARAM) (BOOL*) lpbCanLayout );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpbCanLayout</code>	A pointer to a flag that returns <code>TRUE</code> or <code>FALSE</code> , depending on whether the document layout can be changed.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbCanLayout` returns `TRUE` or `FALSE`.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanLayout` is undefined.

Discussion

Use this message to control the state of a **Wrap to Window**, **Page Layout**, or **Window Width** menu item or toolbar button.

VAPIMWP_VIEW_CANMAGNIFY

Description

Determines whether the document can be magnified. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANMAGNIFY,
            (LPARAM) (BOOL*) lpbCanMagnify );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanMagnify	A pointer to a flag that returns TRUE or FALSE, depending on whether the document can be magnified.

Returns

- SendMessage() returns TRUE if the call succeeds, in which case lpbCanMagnify returns TRUE or FALSE.
- SendMessage() returns FALSE if the call (for example, if there are invalid arguments or if no document is open), in which case lpbCanMagnify is undefined.

Discussion

Use this message to control the state of a **Magnify** menu item or toolbar button.

VAPIMWP_VIEW_CANPAUSE

Description

Determines whether the multimedia document can be paused. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANPAUSE,
            (LPARAM) (BOOL*) lpbCanPause );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanPause	A pointer to a flag that returns TRUE or FALSE, depending on whether the multimedia document can be paused.

Returns

- SendMessage() returns TRUE if the call succeeds, in which case lpbCanPause returns TRUE or FALSE.
- SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbCanPause is undefined.

Discussion

Use this message to control the state of a **Pause** menu item or toolbar button.

VAPIMWP_VIEW_CANPLAY

Description

Determines whether the multimedia document can be played. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANPLAY,
            (LPARAM) (BOOL*) lpbCanPlay );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanPlay	A pointer to a flag that returns TRUE or FALSE, depending on whether the multimedia document can be played.

Returns

- SendMessage() returns TRUE if the call succeeds, in which case lpbCanPlay returns TRUE or FALSE.
- SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbCanPlay is undefined.

Discussion

Use this message to control the state of a **Play** menu item or toolbar button.

VAPIMWP_VIEW_CANPREVIEWPANE

Description

Determines whether a file can be viewed in a preview pane. The message indicates TRUE only when container formats such as ZIP, TAR, or PST files are viewed.

To determine whether the preview pane is being used, use the VAPIMWP_VIEW_GETPREVIEWPANE message.

To specify if the preview pane should be used, use the VAPIMWP_VIEW_SETPREVIEWPANE message.

This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANPREVIEWPANE,
            (LPARAM) (BOOL*) lpbCanPreviewPane );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanPreviewPane	A pointer to a flag that returns TRUE or FALSE, depending on whether the document can be viewed in a preview pane. Only container files use the preview pane.

Returns

- `SendMessage()` returns TRUE if the call succeeds, in which case `lpbCanPreviewPane` returns TRUE or FALSE.
- `SendMessage()` returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanPreviewPane` is undefined.

VAPIMWP_VIEW_CANROTATE

Description

Determines whether the document can be rotated. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANROTATE,
            (LPARAM) (BOOL*) lpbCanRotate );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbCanRotate	A pointer to a flag that returns TRUE or FALSE, depending on whether the document can be rotated.

Returns

- `SendMessage()` returns TRUE if the call succeeds, in which case `lpbCanRotate` returns TRUE or FALSE.

- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanRotate` is undefined.

Discussion

Use this message to control the state of a **Rotate** menu item or toolbar button.

VAPIMWP_VIEW_CANSTOP

Description

Determines whether the playing of the multimedia document can be stopped. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_CANSTOP,
            (LPARAM) (BOOL*) lpbCanStop );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpbCanStop</code>	A pointer to a flag that returns <code>TRUE</code> or <code>FALSE</code> , depending on whether the multimedia document can be stopped.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbCanStop` returns `TRUE` or `FALSE`.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbCanStop` is undefined.

Discussion

Use this message to control the state of a **Stop** menu item or toolbar button.

VAPIMWP_VIEW_DECREASEFONT

Description

Decreases the document font size. Use the [VAPIMWP_VIEW_CANDECREASEFONT](#) message to determine whether the font size can be decreased. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_DECREASEFONT, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Decrease Font** menu item or toolbar button.

VAPIMWP_VIEW_END

Description

Sets the play mode of a multimedia document to stop at the end after playing. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_END, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Stop At End** menu item or toolbar button.

VAPIMWP_VIEW_GETASPECTRATIO

Description

Gets the aspect ratio of a document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETASPECTRATIO,
            (LPARAM) (int*) lpnAspectRatio );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpnAspectRatio	A pointer to an integer that returns the aspect ratio: 0 - None 1 - Based on document 2 - Normal (use scanlines) 3 - Letter (times 2)

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpnAspectRatio` returns the aspect ratio.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpnAspectRatio` is undefined.

Discussion

Use this message to set the state of an **Aspect Ratio** menu item or toolbar button.

VAPIMWP_VIEW_GETGRIDLINES

Description

Gets the gridlines state of the document. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETGRIDLINES,
            (LPARAM) (BOOL*) lpbGridlines );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpbGridlines</code>	A pointer to a flag that returns <code>TRUE</code> or <code>FALSE</code> , depending on whether the document has gridlines set.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbGridlines` returns `TRUE` or `FALSE`.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbGridlines` is undefined.

Discussion

Use this message to set the state of a **Toggle Gridlines** menu item or toolbar button.

VAPIMWP_VIEW_GETINVERT

Description

Gets the invert state of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETINVERT,
            (LPARAM) (BOOL*) lpbInverted );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpbInverted	A pointer to a flag that returns TRUE or FALSE, depending on whether the document colors are inverted (for example, from black to white and white to black).

Returns

SendMessage() returns TRUE if the call succeeds, in which case lpbInverted returns TRUE or FALSE.

SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case lpbInverted is undefined.

Discussion

Use this message to set the state of an **Invert** menu item or toolbar button.

VAPIMWP_VIEW_GETLAYOUT

Description

Gets the layout of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETLAYOUT,
            (LPARAM) (long*) lpLayout );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpLayout	A pointer to a long integer that returns the document layout: LOWORD(*lpLayout) 0 – Wrap to window. LOWORD(*lpLayout) 1 – Page layout. HIWORD(*lpLayout) 0 – Scale page to window width. HIWORD(*lpLayout) n – Scale page to custom percentage.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to set the state of a **Wrap to Window**, **Page Layout**, or **Window Width** menu item or toolbar button.

VAPIMWP_VIEW_GETMAGNIFY

Description

Gets the magnification of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETMAGNIFY,
            (LPARAM) (int*) lpnMagnify );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpnMagnify	A pointer to an integer that returns the document magnification: 0 – Custom -1 – Page width -2 – Page size -3 – Fit selection to window

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpnMagnify` returns the magnification.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpnMagnify` is undefined.

Discussion

Use this message to set the state of a **Magnify** menu item or toolbar button.

VAPIMWP_VIEW_GETPLAYMODE

Description

Gets the play mode of a multimedia document. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETPLAYMODE,
            (LPARAM) (int*) lpnPlayMode );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lpnPlayMode	A pointer to an integer that returns the play mode of the multimedia

Argument	Description
	document: 0 for stop at end; 1 for loop at end.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpnPlayMode` returns the play mode.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpnPlayMode` is undefined.

Discussion

Use this message to check or press a **Stop At End** or **Loop At End** menu item or toolbar button.

VAPIMWP_VIEW_GETPREVIEWPANE

Description

Determines whether the preview pane is being used. The preview pane is only used to display a subfile in a container file. When it is enabled, the viewing area is divided into two panes: one pane displays the contents of the container file, the other displays the contents of the selected subfile. For more information, see [VAPIMWP_VIEW_SETPREVIEWPANE](#) , on page 144. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETPREVIEWPANE,
            (LPARAM) (BOOL*) lpbPreviewPane );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpbPreviewPane</code>	A pointer to a flag that returns <code>TRUE</code> or <code>FALSE</code> , depending on whether the preview pane was returned. Only container files use the preview pane.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpbPreviewPane` returns `TRUE` or `FALSE`.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpbPreviewPane` is undefined.

VAPIMWP_VIEW_GETROTATE

Description

Gets the rotation of the document. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GETROTATE,
            (LPARAM) (int*) lpnRotate );
```

Arguments

Argument	Description
<code>hWndVAPI</code>	The handle of the VAPI window.
<code>lpnRotate</code>	A pointer to an integer that returns the rotation in degrees.

Returns

- `SendMessage()` returns `TRUE` if the call succeeds, in which case `lpnRotate` returns the rotation.
- `SendMessage()` returns `FALSE` if the call fails (for example, if there are invalid arguments or if no document is open), in which case `lpnRotate` is undefined.

Discussion

Use this message to set the state of a **Rotate** menu item or toolbar button.

VAPIMWP_VIEW_GOTOPAGE

Description

Goes to a specified page or slide in a document. This is a parameter of the `VAPIM_VIEW` message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_GOTOPAGE,
            (LPARAM) (int) nPage );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
nPage	A zero-based integer that is the page or slide number you want to go to.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

- Use this message to implement a **Go To Page** menu item or toolbar button.
- The [VAPIMWP_VIEW_CANGOTO](#) message determines whether you can go to specific pages or slides in a document.
- You can use this message only with PPT files or the graphic-based PDF reader (see [Use the kppdfrdr Reader, on page 49](#)). To use similar functionality with word processing files, see [VAPIM_GOTO_PAGE](#) , on page 70.

VAPIMWP_VIEW_INCREASEFONT

Description

Increases the document font size. Use [VAPIMWP_VIEW_CANINCREASEFONT](#) to determine whether the font size can be increased. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_INCREASEFONT, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement an **Increase Font** menu item or toolbar button.

VAPIMWP_VIEW_LOOP

Description

Sets the play mode of a multimedia document to loop at the end after playing. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>  
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_LOOP, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Loop At End** menu item or toolbar button.

VAPIMWP_VIEW_PAUSE

Description

Pauses the playing of a multimedia document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>  
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_PAUSE, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Pause** menu item or toolbar button.

VAPIMWP_VIEW_PLAY

Description

Plays a multimedia document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>  
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_PLAY, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

You can use this message to implement a **Play** menu item or toolbar button.

VAPIMWP_VIEW_SETASPECTRATIO

Description

Sets the aspect ratio of a document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_SETASPECTRATIO,
            (LPARAM) (int) nAspectRatio );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
nAspectRatio	An integer that is the aspect ratio: 0 - None 1 - Based on document 2 - Normal (use scanlines) 3 - Letter (times 2)

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement an **Aspect Ratio** menu item or toolbar button.

VAPIMWP_VIEW_SETGRIDLINES

Description

Sets the gridlines state of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_SETGRIDLINES,
            (LPARAM) (BOOL) bGridlines );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
bGridlines	A flag that is TRUE or FALSE to enable or disable the document gridlines.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Toggle Gridlines** menu item or toolbar button.

VAPIMWP_VIEW_SETINVERT

Description

Sets the invert state of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_SETINVERT, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

The message toggles the current invert state of the document. You can use this message to implement an **Invert** menu item or toolbar button.

VAPIMWP_VIEW_SETLAYOUT

Description

Sets the layout of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_SETLAYOUT,
            (LPARAM) (long) lLayout );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
lLayout	A long integer that is the document layout: LOWORD(lLayout) 0 Wrap to window. LOWORD(lLayout) 1 Page layout. HIWORD(lLayout) 0 Scale page to window width. HIWORD(lLayout) n Scale page to custom percentage.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Wrap to Window**, **Page Layout**, or **Window Width** menu item or toolbar button.

VAPIMWP_VIEW_SETMAGNIFY

Description

Sets the magnification of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_SETMAGNIFY,
            (LPARAM) (int) nMagnify );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
nMagnify	An integer that is the document magnification: <i>n</i> – Custom Percentage Value -1 – Page width -2 – Page size -3 – Fit selection to window

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Magnify** menu item or toolbar button.

The range of acceptable values is 10 to 400 percent.

VAPIMWP_VIEW_SETPREVIEWPANE

Description

Specifies whether the preview pane is used to display a subfile in a container file. When the preview pane is enabled, the viewing area is divided into two panes: one pane displays the contents of the container file, the other displays the contents of the selected subfile. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_SETPREVIEWPANE,
            (LPARAM) (BOOL) bNewValue );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
bNewValue	A flag that returns TRUE or FALSE, depending on whether the preview pane was set. Only container files use the preview pane.

Returns

- SendMessage() returns TRUE if the call succeeds, in which case bNewValue returns TRUE or FALSE.
- SendMessage() returns FALSE if the call fails (for example, if there are invalid arguments or if no document is open), in which case bNewValue is undefined.

VAPIMWP_VIEW_SETROTATE

Description

Sets the rotation of the document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_SETROTATE,
            (LPARAM) (int) nRotate );
```


Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.
nRotate	An integer that is the rotation in degrees.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Rotate** menu item or toolbar button.

VAPIMWP_VIEW_STOP

Description

Stops the playing of a multimedia document. This is a parameter of the VAPIM_VIEW message.

Syntax

```
#include <kvvapi.h>
SendMessage(hWndVAPI, VAPIM_VIEW, VAPIMWP_VIEW_STOP, 0L );
```

Arguments

Argument	Description
hWndVAPI	The handle of the VAPI window.

Returns

SendMessage() returns TRUE if the call succeeds; otherwise it returns FALSE.

Discussion

Use this message to implement a **Stop** menu item or toolbar button.

Chapter 6: Notification Message Parameters

This section provides information on the notification message parameters in the Viewing API. It includes the following topics:

• VAPINM_ANNOTATION_HIT	146
• VAPINM_EXTENT	147
• VAPINM_SELECTION	148
• VAPINM_TEXTBUFFER	148
• VAPINM_USERCLICK	150
• VAPINM_VIEW_FILE	150
• VAPINMWP_INIT_DISABLEUI	151
• VAPINMWP_INIT_DOCTYPE	152
• VAPINMWP_INIT_GETTEMPFILEPATH	152
• VAPINMWP_INIT_OPENDOCDONE	153
• VAPINMWP_INIT_PAGENUMBER	154
• VAPINMWP_MULTIOBJ_OBJNAME	154
• VAPINMWP_OPTIONS_GETOPTIONS_EX	155
• VAPINMWP_PRINT_PRINTDONE	156

VAPINM_ANNOTATION_HIT

Description

Reports an annotation hit when the user clicks on an annotation.

Syntax

```
#include <kvvapi.h>
VAPINM_ANNOTATION_HIT = uMsg;
BOOL bDoubleClick = (BOOL) wParam;
long lLogicalAddress = (long) lParam;
```

Arguments

Argument	Description
bDoubleClick	A flag that is TRUE if the user double-clicked; FALSE if the user single-clicked.
lLogicalAddress	A long integer that is the logical address of the annotation.

Returns

None

Discussion

The logical address of the annotation is the same as that specified in the `VAPIM_ANNOTATE` message.

VAPINM_EXTENT

Description

Reports that the user changed the view extent.

Syntax

```
#include <kvvapi.h>
VAPINM_EXTENT = uMsg;
(wParam is not used)
TPVAPIFirstLast lpFirstLast = (TPVAPIFirstLast*) lParam;
```

Arguments

Argument	Description
lpFirstLast	A pointer to a TPVAPIFirstLast structure that defines the view extent.

Returns

None

Discussion

This message is received to report the first and last logical addresses that are currently displayed. This message is generated when the user scrolls the document or resizes the client area. This message is not generated until a `SB_ENDSCROLL` or `WM_KEYUP` message is received in the case of a scroll.

VAPINM_SELECTION

Description

Reports that the user changed the selection state.

Syntax

```
#include <kvvapi.h>
VAPINM_SELECTION = uMsg;
BOOL bHaveSelection = (BOOL) wParam;
TPVAPIFirstLast lpFirstLast = (TPVAPIFirstLast*) lParam;
```

Arguments

Argument	Description
bHaveSelection	A flag that is TRUE if a selection exists; FALSE if a selection does not exist.
lpFirstLast	A pointer to a TPVAPIFirstLast structure that defines the selection, if a selection exists. It is undefined if a selection does not exist.

Returns

None

Discussion

The lpFirstLast parameter is not valid if the bHaveSelection parameter is FALSE.

VAPINM_TEXTBUFFER

Description

Returns a text buffer.

Syntax

```
#include <kvvapi.h>
VAPINM_TEXTBUFFER = uMsg;
```

```
(wParam is unused).  
TPVAPITextInfo lpTextInfo = (TPVAPITextInfo*) lParam;
```

Arguments

Argument	Description
lpTextInfo	Pointer to a TPVAPITextInfo structure that defines the text buffer.

Returns

None

Discussion

- The `lpTextInfo->cbText` parameter is the number of bytes of text in the buffer. Typically, the buffer is approximately 4 KB, but might be larger. It cannot exceed 10 KB.
Text buffers are usually created at an even boundary, such as the end of a paragraph, table row, or page column. However, if a table row or page column contains a large amount of text, it might be split across text buffers to make sure that `lpTextInfo->cbText` does not exceed 10 K. Individual words are never split across buffers. If `lpTextInfo` is `NULL`, the end of the document is reached.
- The `lpTextInfo->lpText` parameter is a pointer to the buffer of characters. Typically, the buffer is in the Windows ANSI character set; however, the Viewing API allows the user to select either the OEM or ANSI character set for text files. Depending on your integration of Viewing, this might or might not be an issue.

The buffer is zero-terminated. The terminator is not counted in the byte count. The text is an allocated buffer returned to the system upon return of this message. Therefore, you can write within this buffer if it is convenient.

Embedded control codes exist as follows:

KV_EOP	0x01	End of paragraph.
KV_EOC	0x02	End of cell.
KV_PIC	0x03	Picture exists at this logical address.

- To form a logical address from a TEXTBUFFER message, take the base address and add to it the number of BYTES from the start of the text buffer (`lpText`) that the base address references.

For example, the following TEXTBUFFER messages might occur in a document:

```
{ 0, 1000, xxxx } { 1000, 2300, yyyy } { 3300, 1000, zzzz }
```

In this case, addresses 0 through 999 exist in the first TEXTBUFFER, addresses 1000 through 3299 in the second buffer, and 3300 through 4299 in the third buffer.

VAPINM_USERCLICK

Description

Reports that the user clicked the mouse on the document.

Syntax

```
#include <kvvapi.h>
VAPINM_USERCLICK = uMsg;
BOOL bDoubleClick = (BOOL) wParam;
long lLogicalAddress = (long) lParam;
```

Arguments

Argument	Description
bDoubleClick	A flag that is TRUE if the user double-clicked; FALSE if the user single-clicked.
lLogicalAddress	A long integer that is the logical address of the mouse click.

Returns

None

Discussion

You can use the positional information to insert an annotation. This message is generated by a WM_LBUTTONDOWN or WM_LBUTTONDBLCLK.

NOTE: This message is not sent when the user has the Shift key depressed, because this indicates the selection is to be extended. In fact, this causes a VAPINM_SELECTION message to be sent.

VAPINM_VIEW_FILE

Description

Specifies a file that should be viewed. For example, this notification message is generated when the user double-clicks a subfile in a container file displayed in VAPI. This message is also generated when the user clicks a link to a local file from within an HTML file displayed in VAPI.

Syntax

```
#include <kvvapi.h>
VAPINM_VIEW_FILE = uMsg;
VAPINMWP_VIEW_KEEPPFILE or VAPINMWP_VIEW_DELETEFILE = wParam;
char* lpzFileName = (char*) lParam;
```

Arguments

Argument	Description
lpzFileName	A pointer to the complete file specification, including the path, of a file that should be viewed. If the wParam is VAPINMWP_VIEW_DELETEFILE, the file should be deleted after it is viewed (the file is a temporary file).

Returns

Returns TRUE if the message is processed.

VAPINMWP_INIT_DISABLEUI

Description

Determines whether the user interface should be disabled. This is a parameter of the VAPINM_INIT notification message.

Syntax

```
#include <kvvapi.h>
VAPINM_INIT = uMsg;
VAPINMWP_INIT_DISABLEUI = wParam;
(lParam is not used)
```

Returns

Returns TRUE to disable the user interface; otherwise it returns FALSE. The default is FALSE.

Discussion

This message is received when a Viewer asks VAPI if the user interface is disabled. The Viewer does this before a user interface action, such as creating a dialog box.

VAPINMWP_INIT_DOCTYPE

Description

This notification message is received during the opening of a file that cannot be opened. It indicates the document's format. If the document is successfully opened, use VAPIMWP_INIT_GETDESCRIP to obtain a description of the document's format. This is a parameter of the VAPINM_INIT notification message.

Syntax

```
#include <kvvapi.h>
VAPINM_INIT = uMsg;
VAPINMWP_INIT_DOCTYPE = wParam;
char *lpSzDescription = (char *) lParam;
```

Returns

The return value is ignored. The pointer to the lpSzDescription becomes invalid after returning from this message.

VAPINMWP_INIT_GETTEMPFILEPATH

Description

Asks for a temporary file path for VAPI to use. This is a parameter of the VAPINM_INIT notification message.

Syntax

```
#include <kvvapi.h>
VAPINM_INIT = uMsg;
VAPINMWP_INIT_GETTEMPFILEPATH = wParam;
char* lpstzTempFilePath = (char*) lParam;
```

Arguments

Argument	Description
lpstzTempFilePath	A pointer to a Pascal string that returns the temporary file path string as a C string.

Returns

Returns TRUE if the lpstzTempFilePath string was set; otherwise it returns FALSE.

Discussion

This message is received when VAPI converts an I/O object to a file during a **Save As** operation. If the temporary file path is not set, VAPI creates one.

VAPINMWP_INIT_OPENDOCDONE

Description

Reports the status of the document open process. This is a parameter of the VAPINM_INIT notification message.

Syntax

```
#include <kvvapi.h>
VAPINM_INIT = uMsg;
VAPINMWP_INIT_OPENDOCDONE = wParam;
int nPercentDone = (int) lParam;
```

Arguments

Argument	Description
nPercentDone	The percentage done of the document open process.

Returns

None

Discussion

- This message is received during and after the processing of the VAPIMWP_INIT_OPEN_DOCUMENT message. Multiple messages might be received, with increasing values of percentage done. The document is open when the percentage done is $\geq 100\%$.
- A negative value of nPercentDone indicates that an error occurred during the processing of the document.

VAPINMWP_INIT_PAGENUMBER

Description

Reports the current page number of the document. This is a parameter of the VAPINM_INIT notification message.

Syntax

```
#include <kvvapi.h>
VAPINM_INIT = uMsg;
VAPINMWP_INIT_PAGENUMBER = wParam;
int nCurrentPage = (int) LOWORD(lParam);
int cTotalPages = (int) HIWORD(lParam);
```

Arguments

Argument	Description
nCurrentPage	The current page number of the document.
cTotalPages	The total number of pages in the document.

Returns

None

Discussion

This message is received after the document is opened and whenever the document page is changed.

VAPINMWP_MULTIOBJ_OBJNAME

Description

Reports the current object name of the document. This is a parameter of the VAPINM_MULTIOBJ notification message.

Syntax

```
#include <kvvapi.h>
VAPINM_MULTIOBJ = uMsg;
```

```
VAPINMWP_MULTIOBJ_OBJNAME = wParam;  
LPCSTR lpzObjectName = (LPCSTR) lParam;
```

Arguments

Argument	Description
lpzObjectName	The current object name of the document.

Returns

None

Discussion

This message is received after the document is opened and whenever the document object is changed.

VAPINMWP_OPTIONS_GETOPTIONS_EX

Description

Asks for the options for the current document. This is a parameter of the VAPINM_OPTIONS notification message.

Syntax

```
#include <kvvapi.h>  
VAPINM_OPTIONS = uMsg;  
VAPINMWP_OPTIONS_GETOPTIONS_EX = wParam;  
ALL_OPTIONS_EX* lpAllOptions = (ALL_OPTIONS_EX*) lParam;
```

Arguments

Argument	Description
lpAllOptions	A pointer to an ALL_OPTIONS_EX structure to get the document options.

Returns

Returns TRUE if the lpAllOptions structure was processed and initialized; otherwise it returns FALSE.

Discussion

This message is received during the document open process. VAPI initializes the Viewer before the Viewer opens the document. Therefore, this message is received before the VAPIMWP_INIT_OPEN_DOCUMENT or VAPIMWP_INIT_OPENDOCWAIT message returns.

VAPINMWP_PRINT_PRINTDONE

Description

Reports the status of a document that is being printed. This is a parameter of the VAPINM_PRINT notification message.

Syntax

```
#include <kvvapi.h>
VAPINM_PRINT = uMsg;
VAPINMWP_PRINT_PRINTDONE = wParam;
long lStatus = (long) lParam;
```

Arguments

Argument	Description
lStatus	1 - print successful
	2 - user cancelled operation

Returns

None

Chapter 7: Structures

This section describes the structures of the Viewing API. It includes the following topics:

• ADDOCINFO	157
• ALL_OPTIONS_EX	158
• KPTPIObj	159
• KVSumInfoElemEx	160
• KVSummaryInfoEx	160
• TPVAPIAnnotation	161
• TPVAPIConvert	162
• TPVAPICreateParams	163
• TPVAPIDrawFileInfo	164
• TPVAPIDrawPageInfo	166
• TPVAPIExtract	167
• TPVAPIFindInfo	167
• TPVAPIFirstLast	168
• TPVAPIGetText	169
• TPVAPIHiLiteColor	169
• TPVAPIHiLiteOptions	170
• TPVAPIOpenDocumentInfo	170
• TPVAPIPageSize	174
• TPVAPIPosition	175
• TPVAPITextInfo	175

ADDOCINFO

Description

This structure defines the parameters used by the [VAPIMWP_INIT_GETDOCFORMAT](#) message. It provides the format, file class, and version number of the source document. It is defined in `adinfo.h`.

Syntax

```
#include <adinfo.h>
typedef struct
{
    ENdocClass      eClass;
    ENdocFmt        eFormat;
```

```
        long            lVersion;  
        unsigned long    ulAttributes;  
    }  
    ADDOCINFO, *ADDOCINFOPTR;
```

Members

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

ALL_OPTIONS_EX

Description

This structure defines the document options. Document options control display elements such as window size, zoom settings, margin size, scaling, and revision tracking information. Options are defined for each file type category (for example, spreadsheets, multimedia, and word processing). See [Change Document Options, on page 36](#).

Syntax

```
#include <kwoption.h>  
typedef struct ALL_OPTIONS_EX_TAG  
{  
    int            size;  
    MMD_OPTIONS    MMDOptions;  
    WPD_OPTIONS    WPDOptions;  
    SSD_OPTIONS    SSDOptions;  
    ASCII_OPTIONS  ASCIIOptions;  
    IMAGE_OPTIONS  IMGOptions;  
    GX_OPTIONS     GFXOptions;  
    FX_OPTIONS     FAXOptions;  
    GL_OPTIONS     GeneralOptions;  
    ARCHIVE_OPTIONS ArchiveOptions;  
    BOOL           SaveOptions;  
    char           szSectionName[ 16 ];
```

```
    char                szSectionTitle[ 16 ];  
    HTML_OPTIONS        HTMLOptions;  
    PG_OPTIONS          PGOptions;  
}  
ALL_OPTIONS_EX;
```

See the `kwoption.h` file for a description of this structure.

Members

<code>size</code>	The size of the structure.
<code>MMOptions</code>	A pointer to the document options for multimedia files.
<code>WPOptions</code>	A pointer to the document options for word processing files.
<code>SSOptions</code>	A pointer to the document options for spreadsheet files.
<code>ASCIIOptions</code>	A pointer to the document options for ASCII files.
<code>IMGOptions</code>	A pointer to the document options for graphic files.
<code>GFXOptions</code>	A pointer to the document options for GFX files.
<code>FAXOptions</code>	A pointer to the document options for FAX files.
<code>GeneralOptions</code>	A pointer to general options.
<code>ArchiveOptions</code>	A pointer to options that affect archive files.
<code>SaveOptions</code>	Currently not used.
<code>szSectionName[16]</code>	Currently not used.
<code>szSectionTitle[16]</code>	Currently not used.
<code>HTMLOptions</code>	A pointer to options that affect HTML files.
<code>PGOptions</code>	A pointer to options affecting presentation files.

KPTPIOobj

Description

This structure defines the I/O object.

Syntax

```
#include <kwkpfif.h>
```

See the `kvioobj.h` file for a description of this structure.

KVSumInfoElemEx

Description

This structure defines the individual metadata elements, and is defined in `kvtypes.h`.

Syntax

```
typedef struct tag_KVSumInfoElemEx
{
    int                isValid;
    KVSumInfoType      type;
    void               *data;
    char               *pcType;
}
KVSumInfoElemEx;
```

Members

- | | |
|----------------------|--|
| <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 <code>KVSumInfoType</code> in <code>kvtypes.h</code> . |
| <code>data</code> | <p>The content of the metadata field.</p> <p>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.</p> <p><code>KV_DateTime</code> and <code>KV_IEEE8</code> point to an 8-byte value.</p> <p><code>KV_String</code> and <code>KV_Unicode</code> point to the beginning of the string containing the text. <code>KV_Unicode</code> is replaced with <code>KV_String</code> when the UNICODE value has been character mapped to the desired output character set.</p> |
| <code>pcType</code> | A pointer to the name (text description) of the metadata field. |

KVSummaryInfoEx

Description

This structure defines the parameters used by the [VAPIM_GETSUMMARYINFO](#) message. It provides a count of the number of metadata elements, and a pointer to the first element of the array of individual elements. (Metadata is also referred to as document summary information.)

Syntax

```
#include <kvtypes.h>
typedef struct tag_KVSummaryInfoEx
{
    int                nElem;
    KVSuMInfoElemEx    *pElem;
}
KVSummaryInfoEx;
```

Members

nElem The number of metadata elements contained in the array. This value is derived from the enumerated type **KVSuMType**.

pElem Points to the first element of the array of document metadata elements defined by the **KVSuMInfoElemEx** structure.

Discussion

- **nElem** might be zero. This indicates that the document did not contain metadata, such as an ASCII text document. If **nElem** is not zero, **nElem** is at least 42, and possibly more. This value is derived from the **KVSuMType** enumerated type in **kvtypes.h**. The first 42 members of **pElem** are ordered according to the sort order of **KVSuMType**. For example, **pElem[0]** is the code page of the document, and **pElem[10]** is the date the document was last printed.
- If **nElem** is equal or greater than 42, the returned value is a non-standard metadata field.

TPVAPIAnnotation

Description

This structure defines the parameters used by the **VAPIM_ANNOTATE** message.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIAnnotation
{
    long                position;
    ENVAPIAnnotationType type;
    HBITMAP             hBitmap;
    Int                 cbText;
    COLORREF            color;
```

```

        HCURSOR                hCursor;
    }
    TPVAPIAnnotation;

```

Members

Member	Description
position	A long integer. The position where the annotation applies. This is required.
type	<p>The annotation type as defined in <code>ENVAPIAnnotationType</code> in <code>kvvapi.h</code>. This is required. The following options are available:</p> <p><code>kvBitMap</code> – use a bitmap for the annotation.</p> <p><code>kvUnderline</code> – use an underline for the annotation.</p> <p><code>kvDottedUnderline</code> – use a dotted underline for the annotation. Currently not implemented.</p> <p><code>kvStrikeout</code> – use strikethrough as the annotation. Currently not implemented.</p>
hBitmap	If the annotation type is bitmap, this is the handle of the bitmap.
cbText	If the annotation type is underline, this is the length of the underlined text.
color	If the annotation type is underlined, this is the <code>COLORREF</code> value of the underlined text.
hCursor	The handle of the cursor when the mouse hovers over the annotation.

TPVAPIConvert

Description

This structure defines the parameters used by the `VAPIM_CONVERT` message.

Syntax

```

#include <kvvapi.h>
typedef struct tag_TPVAPIConvert
{
    LPSTR        lpszCode;
    LPSTR        lpszTarget;
}
TPVAPIConvert;

```

Members

lpszCode	The format code of the format to which to convert the document. The following options are available: txt — conversion to text format rtf — conversion to RTF format htm — conversion to HTML format
lpszTarget	The target path and file name for the converted file.

TPVAPICreateParams

Description

This structure defines the parameters used to create the VAPI window. A VAPI window is created using the standard Windows API functions `CreateWindow()` or `CreateWindowEx()`. See [Create a Viewing API Window, on page 32](#).

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPICreateParams
{
    UINT            uProfileType;
    LPSTR           lpszRegistryName;
    LPSTR           lpszIniFileName;
    LPVOID          lpvViewerCreateParams;
    BOOL            bSendErrorNM;
    BOOL            bToolBar;
    LPVOID*         lpIpvKeyView;
}
TPVAPICreateParams;
```

Members

uProfileType	Specifies whether initialization information is located in an initialization file or registry. This is optional. The following options are available: PROFILEDF_USE_UNDEFINED 0 Use default (Registry) PROFILEDF_USE_INI 1 Use initialization file PROFILEDF_USE_REG 2 Use Registry
--------------	--

The default is PROFILEDF_USE_REG.

See [View Initialization Information, on page 23](#).

lpszRegistryName	If you are using the registry file to specify initialization information, this is the registry name of key under HKEY_LOCAL_MACHINE\SOFTWARE. This is optional. The default is VAPIDF_REGISTRY_NAME = [Autonomy\Keyview]).
lpszIniFileName	If you are using an initialization file to specify initialization information, this is the file name (not the full path) of the initialization file. This is optional. The default is VAPIDF_INI_FILE_NAME = [KeyView.ini].
lpvViewerCreateParams	Viewer window create. This is optional. Use for custom Viewer.
bSendErrorNM	A flag to tell VAPI to send VAPINM_ERROR notification messages. This is optional. The default is TRUE.
bToolBar	Reserved. This must be FALSE.
lpvpKeyView	Reserved. This must be NULL.

TPVAPIDrawFileInfo

Description

This structure defines the parameters used by the [VAPIMWP_DRAW_DRAWTOFILE](#) message to draw a page to a graphic file.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIDrawFileInfo
{
    LPSTR    lpszTarget;
    LPSTR    lpszGfxOutput;
    UINT     uPageNumber;
    int      iWidth;
    int      iHeight;
    int      iPicXRes;
    int      iPicYRes;
    int      iCompressionQuality;
}
TPVAPIDrawFileInfo;
```

Members

<code>lpszTarget</code>	<p>The target path and file name to which the page is written. The file should use one of the following four extensions:</p> <ul style="list-style-type: none"><code>.bmp</code><code>.jpg</code><code>.png</code><code>.tif</code>
<code>lpszGfxOutput</code>	<p>The output graphics format. The following options are available:</p> <ul style="list-style-type: none"><code>KVGFX_OUTPUT_BMP</code><code>KVGFX_OUTPUT_JPEG</code><code>KVGFX_OUTPUT_PNG</code><code>KVGFX_OUTPUT_TIFF</code> (uncompressed TIFFs)
<code>uPageNumber</code>	<p>The number of the page to be rasterized into a thumbnail.</p> <p>Page numbers start at 0. For example, set <code>uPageNumber</code> to 0 to draw page 1, and to 1 to draw page 2. For word processing documents, pages must be drawn sequentially. For example, to draw page 3, you must first draw pages 0 and 1.</p> <p>If the <code>bWait</code> member of <code>TPVAPIOpenDocumentInfo</code> is set to <code>FALSE</code>, you can draw any page. See TPVAPIOpenDocumentInfo, on page 170. If the page you request is beyond the last page, a <code>VAPI_RETURN_NO_PAGE</code> error is returned.</p>
<code>iWidth</code>	<p>The maximum picture width (in TWIPS).</p>
<code>iHeight</code>	<p>The maximum picture height (in TWIPS).</p>
<code>iPicXRes</code>	<p>The desired horizontal resolution (0 for default).</p>
<code>iPicYRes</code>	<p>The desired vertical resolution (0 for default).</p>
<code>iCompressionQuality</code>	<p>This parameter controls the output quality of graphics that support compression quality (for example, JPEG). The valid range is 0 to 100. 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>The default is 0.</p>

TPVAPIDrawPageInfo

Description

This structure defines the parameters used by the [VAPIMWP_DRAW_DRAWPAGE](#) message to draw a page into a device context.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIDrawPageInfo
{
    HDC          hdc;
    unsigned int nPage;
    RECT         Rect;
    RECT         RectUsed;
    BOOL         bCentre;
    HBRUSH       hBrush;
}
TPVAPIDrawPageInfo;
```

Members

hdc	The device context in which to draw the page.
nPage	<p>The number of the page to draw. Page numbers start at 0. For example, set <code>nPage</code> to 0 to draw page 1, and to 1 to draw page 2. For word processing documents, pages must be drawn sequentially. For example, to draw page 3, you must first draw pages 0 and 1.</p> <p>If the <code>bWait</code> member of TPVAPIOpenDocumentInfo is set to FALSE, you can draw any page. See TPVAPIOpenDocumentInfo, on page 170. If the page you request is beyond the last page, a <code>VAPI_RETURN_NO_PAGE</code> error is returned.</p>
Rect	A rectangle in the device coordinates specifying where to draw the page.
RectUsed	Returns the rectangle that the page was actually drawn in. <code>RectUsed</code> might be different than <code>Rect</code> because Viewing maintains the aspect ratio of the document.
bCentre	Because <code>RectUsed</code> might be different than <code>Rect</code> , this flag specifies whether the page should be centered in <code>Rect</code> .
hBrush	The handle of a brush used to paint the background when <code>RectUsed</code> is different than <code>Rect</code> . If this is <code>NULL</code> , the background is not filled.

TPVAPIExtract

Description

This structure defines the parameters used by the `VAPIMWP_FILE_EXTRACT` message to extract subfiles from a container file.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIExtract
{
    LPSTR    szTargetDirectory;
    LPSTR    szPassWord;
    BOOL     bPreserveDirStructure;
    BOOL     bFailIfExists;
}
TPVAPIExtract;
```

Members

<code>szTargetDirectory</code>	The target directory to which to extract the subfiles.
<code>szPassWord</code>	Password required to open a password-protected subfile.
NOTE: This member is obsolete.	
<code>bPreserveDirStructure</code>	Specifies whether to preserve the directory structure.
<code>bFailIfExists</code>	Specifies whether to fail if the file already exists.

TPVAPIFindInfo

Description

This structure defines the parameters used by `VAPIMWP_EDIT_FIND` for a text search request.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIFindInfo
{
    LPSTR    lpszFindText;
    BOOL     bFindDown;
```

```
        BOOL    bMatchCase;  
        BOOL    bWholeWordOnly;  
    }  
    TPVAPIFindInfo;
```

Members

lpszFindText	A string containing the text to find.
bFindDown	Specifies whether to search from the selected point in the document to the beginning or to the end of the document.
bMatchCase	Specifies whether to match the case of the search term.
bWholeWordOnly	Specifies whether to search for whole words or partial words. Currently not implemented.

TPVAPIFirstLast

Description

This structure defines a text extent or selection.

Syntax

```
#include <kvvapi.h>  
typedef struct tag_TPVAPIFirstLast  
{  
    long    first;  
    long    last;  
}  
TPVAPIFirstLast;
```

Members

first	A long integer that is the first logical address of the text.
last	A long integer that is the last logical address of the text.

TPVAPIGetText

Description

This structure defines the parameters used by the [VAPIM_GETTEXT](#) message to get a text buffer from a specified range.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIGetText
{
    long        start;
    int         cbText;
    BYTE*       lpText;
}
TPVAPIGetText;
```

Members

start	A long integer that is the starting logical address of the text.
cbText	An integer that is the number of bytes of text to copy.
lpText	A pointer to the byte buffer in which to return the text.

TPVAPIHiLiteColor

Description

This structure defines the highlight color used by the [VAPIM_ENABLEINDEX](#) message to mark an index hit.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIHiLiteColor
{
    COLORREF    foreground;
    COLORREF    background;
}
TPVAPIHiLiteColor;
```

Members

foreground	The COLORREF value that is the highlight foreground color.
background	The COLORREF value that is the highlight background color.

TPVAPIHiLiteOptions

Description

This structure defines the highlight color options used by the [VAPIM_SETHILITEOPTIONS](#) message.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIHiLiteOptions
{
    TPVAPIHiLiteColor    hlColorRec;
    int                  TextFontSize;
    char                  TextFontName[LF_FACESIZE];
}
TPVAPIHiLiteOptions;
```

Members

HlColorRec	The handle of a highlight color as described in the TPVAPIHiLiteColor structure.
TextFontSize	The font size in points.
TextFontName	The font name.

TPVAPIOpenDocumentInfo

Description

This structure defines the parameters used by the [VAPIMWP_INIT_OPEN_DOCUMENT](#) message to open a document.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIOpenDocumentInfo
```

```
{
    int                size;
    LPSTR              lpszFilePath;
    KPTPIOobj*         lpIOobj;
    adDocDesc*         lpadDocDesc;
    BOOL               bADInstallViewer;
    int                nViewAsMode;
    LPSTR              lpszOrigFilePath;
    Int                nFlags;
    LPSTR              lpszResID;
    LPSTR              lpszVMLFilePath;
    Void*              lpOptions;
    HGLOBAL             hGlobalMem;
    DWORD              dwcbGlobalMem;
    BOOL               bWait;
    int                nGeneralTab;
    BOOL               bAutoViewAsText;
    LPSTR              lpszHighLight;
    BOOL               bMatchHighLightCase;
}
TPVAPIOpenDocumentInfo;
```

Members

size	This parameter must be initialized to sizeof(TPVAPIOpenDocumentInfo) before calling VAPIMWP_INIT_OPEN_DOCUMENT.
lpszFilePath	A string containing the full file path of the document to open. This is not required if you are using lpIOobj or hGlobalMem.
lpIOobj	A pointer to a KPTPIOobj structure that contains the I/O object of the document to open. This is not required if you are using lpszFilePath or hGlobalMem.
lpadDocDesc	A pointer to an adDocDesc structure that contains the document format information. Set this to NULL.
bADInstallViewer	A flag to install Viewer if the document is not supported (optional).
nViewAsMode	A flag to display the document as formatted text, text, or hexadecimal (optional). The options are described in Options for nViewAsMode, on page 173 .
lpszOrigFilePath	If lpszFilePath is not the original file path, this optional string contains the full path to the original file.
nFlags	A bit field that contains additional options for opening a document (optional). The options are described in Options for nFlags, on page 174 .
lpszResID	A string that contains the resource ID to use (optional).

lpzVMLFilePath	Reserved. Set this to NULL.
lpOptions	A pointer to an ALL_OPTIONS_EX structure to change the default options for the document (optional). See Change Document Options, on page 36 for more information.
hGlobalMem	A block of memory that contains the input file data (optional). If you use this member, lpIOobj and lpzFilePath must be NULL.
dwcbGlobalMem	The size in bytes of the memory specified by hGlobalMem. This is ignored if hGlobalMem is not used.
bwait	<p>Set to TRUE to make SendMessage() on open <i>not</i> return until the document is fully processed.</p> <p>This ensures that the document is fully processed before an operation (such as printing, converting, or searching) is performed, and is useful when you want to use an operation immediately after opening the document.</p> <p>If you are opening a document for viewing only, set this to FALSE so that the first page of the document can be viewed as soon as it is processed.</p> <p>If you are drawing a word processing document and only want to process it up to the specified page, set bwait to FALSE. When the specified page is reached, processing is paused. This setting minimizes delays.</p>
nGeneralTab	This parameter must be set to 0.
bAutoViewAsText	<p>If you set this flag to TRUE, the document is automatically displayed as unformatted text when the document format cannot be determined or is not supported (optional).</p> <p>This member will be deprecated in a future release. To display an unknown or unsupported format as text or hexadecimal, set nViewAsMode, on the previous page to VIEW_MODE_AUTO_TEXT or VIEW_MODE_AUTO_HEX.</p>
lpzHighLight	A string that contains text to be highlighted if it is found in the document (optional). You can either search for an intact string or for individual words that might or might not be adjacent. To search for individual words, separate each word with \t, which indicates a tab.
bMatchHighLightCase	A flag to indicate if the text in lpzHighLight is to be matched case-sensitively (TRUE if it is case sensitive).

Discussion

- If the bwait parameter is set to TRUE, you can determine whether the document has been completely processed and is ready for an operation by using the appropriate "Can" messages, such as VAPIMWP_CANCONVERT, VAPIMWP_FILE_CANSAVEAS, and VAPIMWP_PRINT_CANPRINT.
- The nViewAsMode member can be one of the following options:

Options for nViewAsMode

Option	Description
VIEW_MODE_NORMAL	Displays the document as formatted text when the format can be determined and is supported. This is the default.
VIEW_MODE_TEXT	Displays each byte as ASCII when the format can be determined.
VIEW_MODE_HEX	Displays each byte as hexadecimal when the format can be determined.
VIEW_MODE_AUTO_TEXT	Automatically displays each byte as ASCII when the format cannot be determined or is not supported. This option overrides the setting in bAutoViewAsText , on the previous page.
VIEW_MODE_AUTO_HEX	Automatically displays each byte as hexadecimal when the format cannot be determined or is not supported. This option overrides the setting in bAutoViewAsText , on the previous page.

- For the nViewAsMode member, the VIEW_MODE_NORMAL, VIEW_MODE_TEXT, and VIEW_MODE_HEX options are mutually exclusive; and the VIEW_MODE_AUTO_TEXT and VIEW_MODE_AUTO_HEX options are mutually exclusive. This means you can set a maximum of two options at one time. For example, you can set nViewAsMode as:

```
nViewAsMode= VIEW_MODE_TEXT | VIEW_MODE_AUTO_HEX
```

This configuration results in the following behavior:

File characteristic	Behavior
The file format cannot be determined	The VAPI_RETURN_UNKNOWN_FORMAT message is returned, and the file is displayed as hexadecimal.
The file format can be determined, but is not supported	The VAPI_RETURN_NO_VIEWER message is returned, and the file is displayed as ASCII text.
The file format can be determined and is supported	The VAPI_RETURN_SUCCESS message is returned, and the file is displayed as ASCII text.

- The nFlags member can be one of the following options:

Options for nFlags

Option	Description
VAPIDF_FLAGS_OPEN_FORMAT_ONLY	Opens a document to determine the document format, regardless of whether the document is supported for viewing. After the document is opened, you can then call the VAPIMWP_INIT_GETDOCFORMAT message to get the format information. This flag does not create a Viewer window.
VAPIDF_FLAGS_OPEN_WITHOUT_VIEW	Opens a document in a hidden viewer window. Use this flag to process a document (print, convert, and so on) without viewing. (You must also set the bWait member to TRUE.) For example, to print a document, set the VAPIDF_FLAGS_OPEN_WITHOUT_VIEW flag, and then send the VAPIMWP_PRINT_PRINT message.
VAPIDF_FLAGS_OPEN_VAPI_ONLY	Opens a document without viewing and returns format information with the notification message VAPINMWP_INIT_DOCTYPE. This flag does not create a Viewer window.
VAPIDF_FLAGS_NO_UI	Suppresses GUI elements that are not called explicitly. For example, if you set this flag and the document format is not supported, the "unsupported format" dialog box does not display. However, if you set this flag and send the VAPI message to request the SaveAs dialog box, the SaveAs dialog box displays.
VAPIDF_FLAGS_INCL_REVISION_MARK	Displays the deleted content, revision marks, and revision tracking information in a document. See View Deleted Items and Document Revision Marks , on page 40.

TPVAPIPageSize

Description

This structure defines the parameters used by the [VAPIMWP_DRAW_GETPAGESIZE](#) message to get the size of a page.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPIPageSize
{
    unsigned int    nPage;
```

```
        unsigned int    nWidth;  
        unsigned int    nHeight;  
    }  
    TPVAPIPageSize;
```

Members

nPage	The page number.
nWidth	The default width of the page.
nHeight	The default height of the page.

TPVAPIPosition

Description

This structure defines the position of the viewing window, and is used by the [VAPIM_POSITION](#) message to position the document within the viewing window.

Syntax

```
#include <kvvapi.h>  
typedef struct tag_TPVAPIPosition  
{  
    long    first;  
    long    last;  
    long    position;  
}  
TPVAPIPosition;
```

Members

first	A long integer that returns the first visible logical address.
last	A long integer that returns the last visible logical address.
position	A long integer that is the position (logical address) to make visible.

TPVAPITextInfo

Description

This structure defines a text buffer. See [VAPINM_TEXTBUFFER](#), on page 148.

Syntax

```
#include <kvvapi.h>
typedef struct tag_TPVAPITextInfo
{
    long        lBaseAddress;
    int         cbText;
    BYTE*       lpText;
}
TPVAPITextInfo;
```

Members

lBaseAddress	A long integer that returns the base logical address at the start of the buffer.		
cbText	An integer that returns the number of bytes in the buffer.		
lpText	A pointer to the byte buffer, which returns the pointer to the text buffer. The text buffer can contain text or commands. Commands are ANSI strings with the following extensions (embedded control codes):		
	KV_EOP	0x01	End of paragraph.
	KV_EOC	0x02	End of cell.
	KV_PIC	0x03	Picture exists at this logical address.

Part III: Viewing ActiveX Control

This section provides procedural and reference information for the Viewing ActiveX control and includes the following chapters:

- [Use the Viewing ActiveX Control](#)
- [Control Sample Programs](#)
- [Control Methods](#)
- [Control Properties](#)
- [Control Events](#)

Chapter 8: Use the Viewing ActiveX Control

This section describes how to use the Viewing ActiveX control. It includes the following topics:

• Overview of the Viewing ActiveX Control	178
• Open and View a Document	179
• Save a Document	179
• Convert a Document	180
• Print a Document	180
• Determine the Document Format	181
• Extract Document Metadata	181
• Search for Text in a Document	181
• Copy a Selected Area of Text	181
• Copy all the Text in a Document	182
• Create a Thumbnail Image of a Document Page	182
• Filter a Document	182
• Highlight Text in a Document	182
• Annotate Text in a Document	183

Overview of the Viewing ActiveX Control

Viewing includes an ActiveX control that provides the same functionality of the Viewing API. This control is ideally suited to Visual Basic developers, although it can be used in other development environments that support controls.

NOTE: For information on using the Viewing ActiveX control in a .NET application, see [Develop .NET Applications](#), on page 27.

You can use the Viewing controls to create an application (or HTML page) to:

- Open and view a document.
- Draw a page of a word processing document, spreadsheet, or a picture into a supplied Device Context (HDC). This is useful for generating *thumbnail* views of documents.
- Print a document (or print a document without viewing it) to a specified printer or to the default printer.
- Allow viewed word processing and spreadsheet documents to be saved as RTF, HTML, or text. Also, you can save image formats to other supported image formats.
- Convert word processing and spreadsheet documents to text, RTF, or HTML without viewing them.
- View or extract subfiles from a container file, such as ZIP, TAR, or PST.
- View and manipulate a graphic (including rotate and magnify).
- Annotate documents with a bitmap or selected text. The View API includes annotation event

notification for actions such as clicking and double-clicking, allowing for implementation of hyperlinks and pop-up text.

- Highlight all occurrences of a word in a document.
- Filter spreadsheets, presentation graphics, and documents to text. A cross-platform C API that provides text filtering is also available. Contact HPE for information on KeyView Filter SDK.
- Determine the format of a document based on its contents rather than its file extension.

Open and View a Document

Because you must open a document before you can view, print, or save it, or perform any other operation on it, viewing a document means to open *and* view a document. It is possible, however, to open a document without viewing it, or in other words, to open a document with view mode disabled. In this mode, you can print or save the document without viewing it.

To open and view a document, call the `Open` method. See [Open , on page 200](#) for a description of the possible return codes.

For example:

```
nRet = KeyView1.Open("c:\docs\bigtree.jpg")
```

Using the default settings, this example results in a view of the specified document. However, a number of properties are available to change the default behavior of the `Open` method. See [Control Properties, on page 214](#) for a list of relevant properties. The properties that begin with "OPEN" (for example, the `OPENMode` property and `OPENWaitOnOpen` property) apply to the `Open` method.

When a document is opened successfully, the `OpenDocDone` and `PageNumber` events are generated. These events indicate the progress of the document processing.

Save a Document

To save a document

1. Open the document. See [Open , on page 200](#).
Set the `OPENWaitOnOpen` property to `TRUE` to prevent the `Open` method from returning before the document is completely processed.
To save a document without viewing it, set the `OPENMode` property to 2 (open without generating a view).
2. Use the `CanSaveAs` property to determine whether the document is completely processed and can be saved.
3. Call the `SaveAs` method. To display the **Save As** dialog box and allow the user to save to a target file name, set the `FileName` parameter to an empty string.

Convert a Document

To convert a document to text, RTF, or HTML

1. Open the document. See [Open , on page 200](#) for more information.
Set the [OPENWaitOnOpen](#) property to TRUE to prevent the Open method from returning before the document is completely processed.
To convert a document without viewing it, set the [OPENMode](#) property to 2 (open without generating a view).
2. Use the [CanSaveAs](#) property to determine whether the document is completely processed and can be saved.
3. Call the [Convert , on page 191](#) method.

NOTE: Viewing SDK does not convert PDF, presentations, container files, or graphics files to text, RTF, or HTML.

Print a Document

To print a document

1. Open the document. See [Open , on page 200](#) for more information.
Set the [OPENWaitOnOpen](#) property to TRUE to make sure that the entire document is opened before the document is printed.
To print a document without viewing it, set the [OPENMode](#) property to 2 (open without generating a view). You can also set the [Visible](#) property of the Viewing control object to FALSE.
2. Use the [CanPrint](#) property to determine whether the document is completely processed and ready for printing.
3. Optionally, use the [PrintHeaders](#) property to specify whether the file name and page number header are printed at the top of each page of the printed output.
Used in conjunction with the [PrintHeaders](#) property, the [SetPrintName](#) method replaces the default file name field of the print header with another string.
4. Use either the [PrintDlg](#) method to print using a common **Print** dialog box, or the [PrintOut](#) method to print to a specific printer without a **Print** dialog box.

NOTE: When printing in an application that is an NT service, a default printer must be installed for the user account using the application.

Determine the Document Format

To determine the document format

1. Open the document whose format you want to determine. See [Open](#) for more information.
2. Use the [DocumentType](#) or [DocumentFormat](#) property to determine the format.

The [DocumentType](#) property returns the format as a text description, such as "Microsoft Word for Windows". The [DocumentFormat](#) property returns the format as a numeric value. These properties are set to the document format regardless of whether the document can be viewed.

3. To get the general class to which the currently opened document belongs, use the [DocumentClass](#) property.

Extract Document Metadata

To extract document metadata

1. Open the document whose metadata you want to extract. See [Open](#) for more information.
2. Set `nItem` to 0 (zero).
3. Use the [GetSummaryInfo](#) method to get the total number of summary information items available in the document (parameter `nTotalItem`).
4. Set `nItem` to a number between zero and `nTotalItem` ($0 \leq nItem < nTotalItem$).
5. Use the [GetSummaryInfo](#) method again to get the specific information for the `nItem` item. After the call, the other parameters (such as `nValid`, `nType`, `lVal`, `szVal`, and `szUserVal`) hold the values for the summary information item.

Search for Text in a Document

To search for specified text in a document

1. Open the document that you want to search. See [Open](#) for more information.
2. Use the [CanFind](#) property to determine whether the document is completely processed and can be searched.
3. Call the [Find](#) method without a search string specified to open a **Find** dialog box.
4. To perform a **Find Next**, call the [Find](#) method repeatedly with the original search term.

Copy a Selected Area of Text

To select and copy text

1. Open the document from which you want to copy the text. See [Open](#) for more information.
2. Highlight a region of text to copy.

3. Use the [CanCopy](#) property to determine whether the selected text can be copied.
4. Use the [GetSelectedText](#) method to get the selected text.
5. Use the [Copy](#) method.
6. Paste the text in another document, such as a text file.

Copy all the Text in a Document

To copy all the text in a document

1. Open the document from which you want to copy all the text. See [Open](#) for more information.
2. Use the [CanSelectAll](#) method to determine whether all contents in the document can be selected.
3. Use the [SelectAll](#) method.
4. Use the [CanCopy](#) property to determine whether the text can be copied.
5. Use the [Copy](#) method.
6. Paste the text in another document, such as a text file.

Create a Thumbnail Image of a Document Page

To create thumbnails of a document page

1. Set the [OPENMode](#) property to 7 (drawing mode enabled).
2. Open the document for which you want to create a thumbnail image. See [Open](#) for more information.
3. Use the [DrawToFile](#) method.

Filter a Document

To filter a document

1. Set the [OPENMode](#) property to 6 (index-only mode).
2. Open the document that you want to filter. See [Open](#) for more information.
3. Use the [GetNextTextBuffer](#) method to get the text buffers.

Highlight Text in a Document

To highlight text in a document

1. Set the [OPENMode](#) property to 5 (enable index mode).
2. Open the document in which you want to place highlights. See [Open](#) for more information.

3. Use the [SetHiLiteOptions](#) method to set the highlight color and font.
4. Use the [SetHiLite](#) method to specify the text to be highlighted.

Annotate Text in a Document

To add annotations to or remove annotations from a document

1. Set the [OPENMode](#) property to 5(enable index mode).
2. Open the document you want to annotate. See [Open](#) for more information.
3. Use the [Annotate](#) method to add and delete annotations.

Chapter 9: Control Sample Programs

This section describes the sample programs that demonstrate how to use the ActiveX control.

• Viewing SDK Initialization Information	184
• fileview	184
• dotnetview	187

Viewing SDK Initialization Information

Viewing SDK uses initialization information for its internal operations; for example, to determine which components to load. You can store this information either in the Windows registry or in an initialization file. When you use Viewing SDK you must tell it where to find this information and what form it is in. See [View Initialization Information, on page 23](#) for more information.

fileview

FILEVIEW is a sample program that demonstrates how to insert the Viewing ActiveX control into a Visual Basic application and use it to display documents.

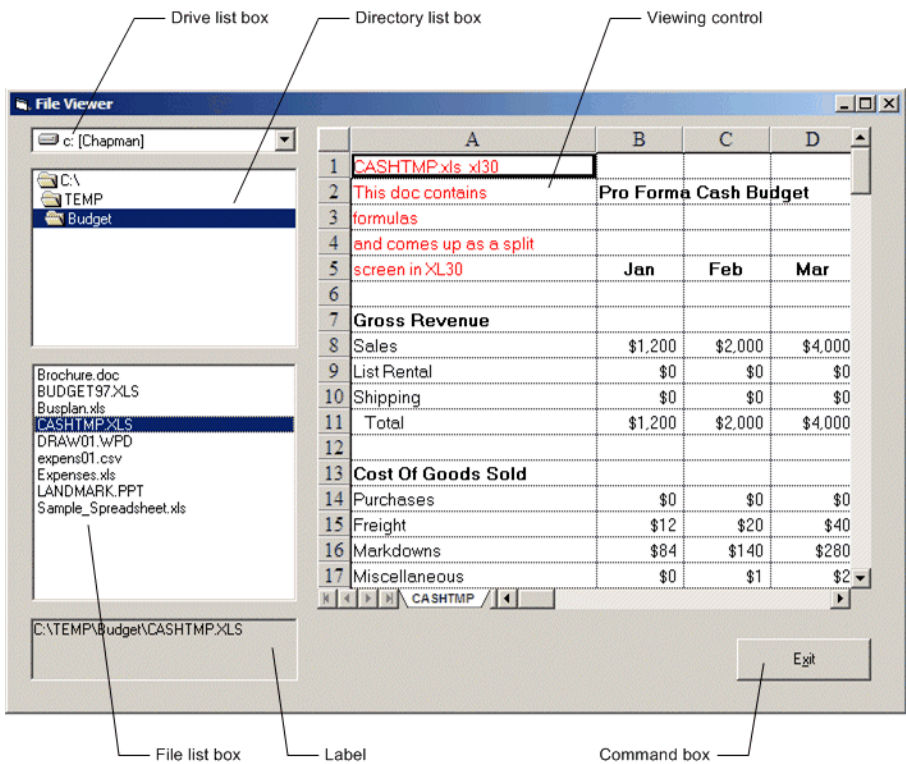
Create a New Visual Basic Project 6.0

1. Start a new project by choosing **New Project** from the **File** menu.
2. Add the Viewing Control to the project. From the **Project** menu, select **Components**.
3. In the **Components** dialog box, select **KeyView OLE Control module** from the **Controls** list box. Click **OK**.







If the KeyView OLE Control module is not listed in the **Controls** list box, click **Browse** to locate and register the control.

The Viewing icon  appears in the Toolbox.

Draw the Controls



Draw the controls on the form according to the diagram above. Use the following controls:

Button	Control
	Command button
	Label
	Drive list box
	Directory list box
	File list box
	Viewing control

Set Objects and Properties

After you design the form, you need to set the following properties:

Object	Property	Setting
Form	Caption	File Viewer
Label	BorderStyle	1-Fixed Single
	Caption	(Empty)
Command button	Caption	Exit

NOTE: Use the default settings for all other properties and objects.

Create Event Procedures

In the File Viewer application, create the following event procedures for five different controls. After you have completed these event procedures, you can compile and run the application.

Form Load event: The `Form_Load` event sets the drive and path to the drive and directory where the sample application is loaded, and specifies whether initialization information is stored in the registry or the `kvsdk.ini` file. Add the following code to the `Form_Load` event procedure:

```
Private Sub Form_Load()  
    Drive1.Drive = App.Path  
    Dir1.Path = App.Path  
    KEYview1.RegIniMode = 1  
    KEYview1.RegIniName = "kvsdk.ini"  
End Sub
```

Command button Click event: The command button's `Click` event ends the application. Add the following code to the `Command1_Click` event procedure:

```
Private Sub Command1_Click ()  
    Unload Me  
    End ' Ends the application.  
End Sub
```

Drive list and directory list boxes Change events: To make the drive, directory, and file list boxes work together, add the following code to the `Drive1_Change` and `Dir1_Change` event procedures:

```
Private Sub Drive1_Change ()  
    Dir1.Path = Drive1.Drive ' Update directory path.  
End Sub  
Private Sub Dir1_Change ()  
    File1.Path = Dir1.Path ' Update files.  
End Sub
```

File list box Double-click event: The `File1_DblClick` event procedure for the file list box displays the full path name of the selected file in the label control, and displays the picture itself in the image control. You need the following code:

```
Private Sub File1_DblClick ()  
    If Right(File1.Path, 1) <> "\" Then  
        Label1.Caption = File1.Path & "\" & File1.FileName
```

```
Else          ' If root directory
    Label1.Caption = File1.Path & File1.FileName
End If
nRet = KEYview1.Open(Label1.Caption)
End Sub
```

Notice how the full path name is constructed:

- `File1.Path` returns the drive and directory path, "\" adds a backslash separator, and `File1.FileName` returns the file name.
- The `Right` function checks to see whether the path name is the root directory (\). If it is not, the full path name is assigned to the label's caption. If the path name is the root directory, the backslash is omitted.
- The `Open` method loads the file specified by the path name in the label caption.

dotnetview

Viewing SDK includes a .NET workspace for Visual Studio called `dotnetview`. This is a J# application that demonstrates basic Viewing functionality. To use the .NET sample, open the workspace file `dotnetview.sn1` in Visual Studio 2005 and follow the instructions in [Develop .NET Applications, on page 27](#). The source file is `form1.js1`.

Chapter 10: Control Methods

This section describes the Viewing ActiveX control methods.

• Annotate	189
• ChangeObject	190
• Close	190
• Convert	191
• Copy	192
• DecreaseFont	193
• DrawToFile	193
• Find	195
• GetNextTextBuffer	195
• GetPageFromLogical	196
• GetSelectedText	197
• GetSummaryInfo	197
• GetText	198
• GoToPage	199
• IncreaseFont	199
• Open	200
• Play	201
• Position	202
• PositionHiLite	202
• PrintDlg	203
• PrintOut	204
• PrintOutEx	204
• PrintPageSetup	205
• SaveAs	206
• SelectAll	207
• SetCursor	207
• SetFocusViewer	208
• SetHiLite	208
• SetHiLiteOptions	209
• SetPassword	210
• SetPrintName	210
• ShowHits	211
• UnZip	212

- [UnZipEx](#)212

NOTE: The Viewing control's methods in this chapter show the syntax in Visual Basic.

Annotate

Description

Adds and deletes annotations, and determines whether annotations exist. Use this method with the [OPENMode](#) property set to 5 (indexing enabled).

Syntax

```
bRet = KeyView.Annotate(action, position, type, bitmapFile, cbText, colorref, cursorFile)
```

Parameters

Parameter	Type	Description
action	Integer	0 – delete annotation.
		1 – add annotation.
		2 – query annotation.
position	Long	Logical address to apply the annotation.
type	Integer	0 – use a bitmap for the annotation.
		1 – use an underline as the annotation.
		2 – use a dotted underline as the annotation. Currently not implemented.
		3 – use a strikethrough as the annotation. Currently not implemented.
bitmapFile	String	If the annotation type is bitmap, this is the bitmap file name (*.bmp).
cbText	Integer	If the annotation type is underline, this is the length of the underlined text.
colorref	Long	If the annotation type is underline, this is the color of the underlined text.
cursorFile	String	The cursor file (*.cur) to use when mouse hovers over the annotation.

Returns

Type	Description
Boolean	TRUE if the method succeeds.

Type	Description
	FALSE if the method fails.

ChangeObject

Description

Changes the current object in a multiple-object document. Examples of a multiple-object document include:

- A Microsoft Excel spreadsheet with multiple worksheets where each worksheet is considered an object.
- A Microsoft PowerPoint presentation with multiple slides where each slide is considered an object.

Syntax

```
bRet = KeyView.ChangeObject(direction)
```

Parameters

Parameter	Type	Description
direction	Integer	0 – select the next object. 1 – select the previous object.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

Close

Description

Closes the currently opened document. The Close method is not required because calling the Open method automatically closes the currently opened document.

Syntax

```
bRet = KeyView.Close
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

Convert

Description

Converts a document to text, RTF, or HTML format.

Before you convert a document, you must open it by using the [Open](#) method. To make sure that the entire document is opened before the document is converted, set the [OPENWaitOnOpen](#) property to TRUE. Use the [CanSaveAs](#) property to check whether the document is completely processed and is ready for conversion.

To convert a document without viewing it, set the [OPENMode](#) property to 2 (open without generating a view).

Syntax

```
bRet = KeyView.Convert(FormatCode, TargetFile)
```

Parameters

Parameter	Type	Description
FormatCode	String	The format code of the format to which to convert the document. The following options are available: txt — conversion to text format rtf — conversion to RTF format htm — conversion to HTML format

Parameter	Type	Description
-----------	------	-------------

TargetFile	String	The target path and file name for the converted file.
------------	--------	---

Returns

Type	Description
------	-------------

Boolean	TRUE if the method succeeds. FALSE if the method fails.
---------	--

Example

The following code converts a file to RTF:

```
Function ConvertToRTF(FileName As String, TargetFile As String) As Boolean
    KeyView.OPENMode = 2
    nRet = KeyView.Open(FileName)
    If (nRet = 1) Then
        bRet = KeyView.Convert("rtf",TargetFile)
    End If
    bRet2 = KeyView.Close
    ConvertToRTF = bRet
End Function
```

In this example, OPENMode is set to 2 to prevent a view of the document.

Copy

Description

Copies the current selection to the clipboard. Use the [CanCopy](#) property to determine whether the content is selected and can be copied by using the Copy method.

Syntax

```
bRet = KeyView.Copy
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

DecreaseFont

Description

Decreases the document font size. Use the [CanDecreaseFont](#) property to determine whether the font size can be decreased, and therefore, whether the `DecreaseFont` method succeeds.

Syntax

```
bRet = KeyView.DecreaseFont
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

DrawToFile

Description

Draws a page of a document to a graphic file (thumbnail). Use this method with the [OPENMode](#) property set to 7 (drawing mode enabled).

Syntax

```
bRet = KeyView.DrawToFile(szTargetFile, szGfxOutput, uPageNumber, nWidth, nHeight,  
nPicXRes, nPicYRes, nQuality)
```

Parameters

Parameter	Type	Description
szTargetFile	String	The target path and file name to which the page is written. The file should use one of the following four extensions: <ul style="list-style-type: none">• .bmp• .jpg• .png• .tif
SzGfxOutput	String	The output graphics format. The following options are available: <ul style="list-style-type: none">• bmp• jpg• png• tif (uncompressed TIFFs) You must specify the parameter in lower case.
uPageNumber	UINT	The number of the page to be rasterized into a thumbnail. Page numbers start at 0. For example, set uPageNumber to 0 to draw page 1, and to 1 to draw page 2. For word processing documents, pages must be drawn sequentially. For example, to draw page 3, you must first draw pages 0 and 1.
nWidth	Integer	The maximum picture width (in TWIPS).
nHeight	Integer	The maximum picture height (in TWIPS).
nPicXRes	Integer	The desired horizontal resolution (0 for default).
nPicYRes	Integer	The desired vertical resolution (0 for default).
nQuality	Integer	This parameter controls the output quality of graphics that support compression quality (for example, JPEG). The valid range is 0 to 100. 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). The default is 0.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

Find

Description

Searches the currently opened document for the specified text. To perform a **Find Next**, call this method repeatedly with the original search term. If you specify an empty string for the `FindText`, a **Find** dialog box appears.

Syntax

```
bRet = KeyView.Find(FindText, FindDown, MatchCase)
```

Parameters

Parameter	Type	Description
<code>FindText</code>	String	A string that contains the text to find.
<code>FindDown</code>	Boolean	Specifies whether to search from the selected point in the document to the beginning (FALSE) or to the end (TRUE) of the document.
<code>MatchCase</code>	Boolean	Specifies whether to match the case of the search term.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

GetNextTextBuffer

Description

Gets text buffers. Use this method with the `OPENMode` property set to 6 (index only mode).

Syntax

```
bRet = KeyView.GetNextTextBuffer
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

GetPageFromLogical

Description

Gets the page number for a logical address.

Syntax

```
lRet = KeyView.GetPageFromLogical(lLogicalAddress)
```

Parameters

Parameter	Type	Description
lLogicalAddress	Long	The logical address for which to get the page number.

Returns

Type	Description
Long	<i>n</i> – Page number the specified logical address resides on -1 – Error

GetSelectedText

Description

Returns any currently selected text. The [CanCopy](#) property returns TRUE if any text is currently selected.

Syntax

```
SelectedText = KeyView.GetSelectedText
```

Parameters

None

Returns

Type	Description
String	A description of the specified format.

GetSummaryInfo

Description

Gets the document metadata, also referred to as summary information. See [Extract Document Metadata, on page 181](#).

Syntax

```
bRet = KeyView.GetSummaryInfo(nItem, nTotalItem, nValid, nType, lVal, szVal, szUserVal)
```

Parameters

Parameter	Type	Description
nItem	Integer	The summary information item number.
nTotalItem	Integer	The total number of summary information items available in the source document.

Parameter	Type	Description
nValid	Integer	Specifies whether the data value is present in the document. 0 – The summary information field is not available. 1 – The summary information field is available.
nType	Integer	The data type of the metadata element. 1 – szVal contains the field contents. 2 – lVal contains the field contents. The types are defined in KVSumInfoType in kvtypes.h.
lVal	Integer	Contains the contents of the summary information field if the contents are numeric.
szVal	String	Contains the contents of the summary information field if the contents are non-numeric.
szUserVal	String	The summary information field name.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

GetText

Description

Gets a text buffer from a specified range. Use this method with the [OPENMode](#) property set to 5 (indexing enabled).

Syntax

```
bRet = KeyView.GetText(start, cbText, pszText)
```

Parameters

Parameter	Type	Description
start	Long	The starting logical address of the text.
cbText	Integer	The number of bytes of text to copy.

Parameter	Type	Description
pszText	String	A pointer to the logical address where the text will be stored.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

GoToPage

Description

Displays the document at the specified page. Use this method with the [OPENMode](#) property set to 5 (indexing enabled).

Syntax

```
nRet = KeyView.GoToPage(nPageNumber)
```

Parameters

Parameter	Type	Description
nPageNumber	Integer	The page to display.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

IncreaseFont

Description

Increases the document font size. Use the [CanIncreaseFont](#) property to determine whether the font size can be increased, and therefore, whether the IncreaseFont method succeeds.

Syntax

```
bRet = KeyView.IncreaseFont
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

Open

Description

Opens a document.

The properties that begin with "OPEN" (for example, the `OPENMode` property and `OPENWaitOnOpen` property) apply to the `Open` method. However, they do not apply to the initial call to the `Open` method; that is, they do not affect the currently opened document and apply only to documents opened with subsequent calls to the `Open` method. See [Control Properties, on page 214](#).

Syntax

```
nRet = KeyView.Open(FileName)
```

Parameters

Parameter	Type	Description
Filename	String	A string containing the file name of the document to open.

Returns

Type	Description
Short	0 – if an error occurs during open. 1 – if the document opened successfully.

Type	Description
------	-------------

- | | |
|----|--|
| 2 | – if the document is of an unknown format. |
| 3 | – if no viewer is available for the document format. |
| 4 | – if the document is password-protected. |
| 5 | – if the drawing routines have not been initialized. |
| 6 | – if the requested page does not exist, or is being displayed before all previous pages have been displayed. |
| 7 | – if the document does not support this feature (for example, ZIP files, video, or audio). |
| 8 | – if the KeyView license is invalid. |
| 9 | – if the KeyView license is expired. |
| 10 | – if the input file or stream is invalid or corrupt. |

Play

Description

Plays, pauses, or stops a multimedia (digital video or sound) document.

Syntax

```
bRet = KeyView.Play(Mode)
```

Parameters

Parameter	Type	Description
Mode	Integer	0 – Play
		1 – Pause
		2 – Stop

Returns

Type	Description
Boolean	TRUE if the method succeeds.
	FALSE if the method fails.

Position

Description

Positions the document in the viewing window.

Syntax

```
bRet = KeyView.Position(first, last, position)
```

Parameters

Parameter	Type	Description
first	Long	The first visible logical address.
last	Long	The last visible logical address.
position	Long	The position (logical address) to make visible.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

PositionHiLite

Description

Changes focus from the previous to next highlight. This only applies when using XML files with the Verity Developer's Kit (VDK) to specify highlights.

Syntax

```
bRet = KeyView.PositionHiLite(bPrev)
```

Parameters

Parameter	Type	Description
bPrev	Boolean	TRUE – go to previous highlight FALSE – go to next highlight.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

PrintDlg

Description

Prints a document by calling the common **Print** dialog box. [Print a Document, on page 180.](#)

Syntax

```
bRet = KeyView.PrintDlg
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

PrintOut

Description

Prints a document to a specified printer without calling the common **Print** dialog box, or to the default printer if none is specified. See [Print a Document, on page 180](#).

Syntax

```
bRet = KeyView.PrintOut(Printer)
```

Parameters

Parameter	Type	Description
Printer	String	The name of the printer to use. If the string is empty, the default printer is used. An example of a valid string value is \\Calculus\HP LaserJet IIISi

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

PrintOutEx

Description

Prints a document to a specified printer without calling the common **Print** dialog box, or to the default printer if none is specified. The method can specify the page range and number of copies to print. See [Print a Document, on page 180](#).

Syntax

```
bRet = KeyView.PrintOutEx(Printer, FromPage, ToPage, Copies, Flags)
```

Parameters

Parameter	Type	Description
Printer	String	The name of the printer to use. If the string is empty, the default printer is used. An example of a valid string value is <code>\\Calculus\HP LaserJet IIISi</code>
FromPage	Integer	The number of the first page to print. For spreadsheet documents, this is the first sheet number.
ToPage	Integer	The number of the last page to print. For spreadsheet documents, this is the last sheet number.
Copies	Integer	The number of copies to print.
Flags	Integer	Reserved. This parameter should be set to 0.

NOTE:

If FromPage, ToPage, and Copies are all set to 0, this method behaves the same way as [PrintOut](#)

If FromPage and ToPage are both set to 0, all pages are printed.

If Copies is set to 0, one copy is printed.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

PrintPageSetup

Description

Displays a dialog box that allows the user to set up print page scaling for a spreadsheet.

Syntax

```
bRet = KeyView.PrintPageSetup
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SaveAs

Description

Saves the current document in another format.

If the `FileName` parameter is an empty string, this method displays the **Save As** dialog box, which allows the user to specify conversion, compression, and encodings, as well as the target file name. Otherwise, the `FileName` parameter should be the complete path and file name of the desired target file. A copy operation from the currently opened file to the specified target file occurs. Use the [CanSaveAs](#) property to check whether the document has been completely processed and can be saved.

Syntax

```
bRet = KeyView.SaveAs(FileName)
```

Parameters

Parameter	Type	Description
<code>FileName</code>	String	The complete path and file name of the target file. This parameter can be an empty string that displays the Save As dialog box.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SelectAll

Description

Selects all items in the currently opened document. Use the [CanSelectAll](#) method to determine whether all contents in the document can be selected.

Syntax

```
bRet = KeyView.SelectAll
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SetCursor

Description

Sets the viewing engine cursor.

Syntax

```
bRet = KeyView.SetCursor(CursorFile, bRestore)
```

Parameters

Parameter	Type	Description
CursorFile	String	The cursor file (*.cur) to use.
bRestore	Boolean	TRUE restores the default cursor. FALSE uses the cursor file.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SetFocusViewer

Description

Sets the current focus to the Viewer window.

Syntax

```
bRet = KeyView.SetFocusViewer
```

Parameters

None

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SetHiLite

Description

Highlights a region of text. Use this method with the [OPENMode](#) property set to 5 (indexing enabled).

Syntax

```
bRet = KeyView.SetHiLite(cbTextToHiLite, lLogicalAddress)
```


Parameters

Parameter	Type	Description
<code>cbTextToHiLite</code>	Integer	The number of bytes to highlight.
<code>lLogicalAddress</code>	Long	The logical address from which to start highlighting.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SetHiLiteOptions

Description

Set the highlight options. Use this method with the [OPENMode](#) property set to 5 (indexing enabled).

Syntax

```
bRet = KeyView.SetHiLiteOptions(BackColor, ForeColor, FontSize, FontName)
```

Parameters

Parameter	Type	Description
<code>BackColor</code>	Integer	The COLORREF value that is the highlight background color.
<code>ForeColor</code>	Integer	The COLORREF value that is the highlight foreground color.
<code>FontSize</code>	Integer	The font size in points.
<code>FontName</code>	String	The font name.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SetPassword

Description

Sets a password to use to open a password-protected file before the file is opened. Currently, you can use this to set a password for ZIP, PST, and NSF files.

NOTE: Unicode passwords are not supported.

Syntax

```
bRet = KeyView.SetPassword (szPassword);
```

Parameters

Parameter	Type	Description
szPassword	String	The password string.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

SetPrintName

Description

Used in conjunction with the `PrintHeaders` property, this method replaces the default file name field of the print header with a specified string.

Syntax

```
bRet = KeyView.SetPrintName(szPrintName)
```

Parameters

Parameter	Type	Description
szPrintName	String	A string used to replace the file name field of the <code>PrintHeader</code> property. For example, you could replace the file name with "Copyright 2003".

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

ShowHits

Description

Shows or hides index hits. Use this method with the [OPENMode](#) property set to 5 (indexing enabled).

Syntax

```
bRet = KeyView.ShowHits(bShow)
```

Parameters

Parameter	Type	Description
bShow	Boolean	TRUE shows hits. FALSE hides hits.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

UnZip

Description

Extracts selected subfiles in the currently opened container file to disk or to a Viewer window.

Syntax

```
bRet = KeyView.UnZip(UnZipToDisk)
```

Parameters

Parameter	Type	Description
UnZipToDisk	Boolean	If TRUE, a dialog box prompts the user to specify a path where the subfiles are extracted. If FALSE, the subfiles are extracted and displayed in the preview pane. This is the default.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

UnZipEx

Description

Extracts the subfiles in the currently opened container file to disk without requiring the user to respond to dialog boxes.

Syntax

```
bRet = KeyView.UnZipEx(TargetDirectory, Password, bFailIfExists,  
bPreserveDirectory)
```

Parameters

Parameter	Type	Description
TargetDirectory	String	The directory to which to extract the subfiles.
PassWord	String	The password to use to open a password-protected container file.
bFailIfExists	Boolean	If TRUE and a subfile is extracted and a file of the same name already exists in the target directory, the existing file is not overwritten. If FALSE, the existing subfile is overwritten.
bPreserveDirectory	Boolean	If TRUE, the directory structure is preserved. If FALSE, the directory structure is not preserved.

Returns

Type	Description
Boolean	TRUE if the method succeeds. FALSE if the method fails.

Chapter 11: Control Properties

This section describes the Viewing ActiveX control properties.

• Introduction	216
• ASCIICharSet	216
• ASCIIFilterNonPrintable	217
• ASCIIFontName	217
• ASCIIFontSize	217
• ASCIIFontStyle	218
• ASCIIMarginBottom	218
• ASCIIMarginLeft	219
• ASCIIMarginRight	219
• ASCIIMarginTop	219
• ASCIIPrintLandscape	220
• AspectRatio	220
• CanCopy	220
• CanDecreaseFont	221
• CanFind	221
• CanIncreaseFont	222
• CanMultiObj	222
• CanNextObj	222
• CanPause	223
• CanPlay	223
• CanPrevObj	224
• CanPrint	224
• CanSaveAs	225
• CanSelectAll	225
• CanStop	225
• CanUnZip	226
• CanViewPane	226
• CharSet	226
• ContextMenu	227
• DocumentClass	227
• DocumentFormat	228
• DocumentType	228
• DrawPageCount	228

• DrawPageHeight	229
• DrawPageWidth	229
• DrawWorkBookPageCount	229
• FileName	230
• HiLiteBackground	230
• HiLiteForeground	230
• HotKeys	231
• ImageCustomSize	231
• ImagePrintHorzAlign	231
• ImagePrintMode	232
• ImagePrintPercent	232
• ImagePrintVertAlign	233
• ImageScaling	233
• IndexBufCharSet	233
• Invert	234
• JumpToFirstHiLite	234
• MMPlayOption	235
• MMScaleMovie	235
• ObjName	235
• OPENDisableUI	236
• OPENHighLight	236
• OPENMode	236
• OPENWaitOnOpen	237
• PrintAnnotations	238
• PrintHeaders	238
• RegIniMode	238
• RegIniName	239
• Rotate	239
• SrcCharSet	240
• SSDisplayGrid	240
• SSDisplayHeaders	241
• SSViewObjects	241
• TrgCharSet	241
• ViewPane	242
• WPCustomSize	242
• WPDDisplayPict	242
• WPPageLayout	243
• WPScaleTable	243

- [WPViewMode](#) 244

Introduction

Persistent Properties

Persistent properties are those that can be set in the **Properties** page at design time. The selected value is stored as a part of your project and used at runtime.

Property Naming Conventions in .NET

In J#, C#, and C++, all ActiveX control method names in the .NET class are the same as their COM counterparts. However, individual properties in .NET are defined using *get* and *set* methods of the following format:

get_property_name

set_property name

For example:

```
private void button1_Click(Object sender, System.EventArgs e)
{
    this.axKEYview1.set_RegIniMode((short)1);
    this.axKEYview1.set_RegIniName("c:\\windows\\kvsdk.ini");
    this.axKEYview1.Open("c:\\test.doc");
}
```

NOTE: Important: In a Visual Basic .NET application, all properties and methods are used in the same way as in a Visual Basic COM application.

See [Develop .NET Applications](#), on page 27 for more information.

"OPEN" Properties

The "Open" properties (such as `OPENWaitUponOpen` and `OPENMode`) apply to the `Open` method. However, they do not apply to the initial call to the `Open` method; that is, they do not affect the currently opened document, and apply only to documents opened with subsequent calls to the `Open` method. These properties are persistent and are available through the **Properties** page.

ASCIISet

Description

Specifies the character set used to display an ASCII text file. The ANSI character set is the Microsoft Windows default. The DOS code page is useful if the text file was created with a DOS editor. This is a

persistent property.

Returns

Type	Description
Integer	0 – Displays the document using the ANSI character set. 1 – Displays the document using the DOS code page.

ASCIIFilterNonPrintable

Description

Specifies whether non-printable characters can be viewed in an ASCII text file. This is a persistent property.

Returns

Type	Description
Boolean	TRUE – non-printable characters can be viewed. FALSE – non-printable characters cannot be viewed.

ASCIIFontName

Description

Specifies the font name to be used in an ASCII text file. This is a persistent property.

Returns

Type	Description
String	A description of the desired font name.

ASCIIFontSize

Description

Specifies the font size to be used in an ASCII text file. This is a persistent property.

Returns

Type	Description
Integer	n – The font size in points.

ASCIIFontStyle

Description

Specifies the font style used to display an ASCII text file.

Returns

Type	Description
Integer	0 – Displays text in normal font. 1 – Displays text in bold font. 2 – Displays text in italic font. 3 – Displays text in bold and italic font.

ASCIIMarginBottom

Description

Specifies the bottom page margin of an ASCII text file (between 0 - 4 inches or 100 mm). This is a persistent property.

Returns

Type	Description
Integer	n – The bottom page margin in TWIPS (the default is 1440).

ASCIIMarginLeft

Description

Specifies the left page margin of an ASCII text file. This is between zero and 4 inches, or zero and 100 mm. This is a persistent property.

Returns

Type	Description
Integer	<i>n</i> – The left page margin in TWIPS (the default is 1440).

ASCIIMarginRight

Description

Specifies the right page margin of an ASCII text file. This is between zero and 4 inches, or zero and 100 mm. This is a persistent property.

Returns

Type	Description
Integer	<i>n</i> – The right page margin in TWIPS (the default is 1440).

ASCIIMarginTop

Description

Specifies the top page margin of an ASCII text file (between 0 and 4 inches or 100 mm). This is a persistent property.

Returns

Type	Description
Integer	<i>n</i> – The top page margin in TWIPS (the default is 1440).

ASCIIPrintLandscape

Description

Specifies whether the ASCII text file is printed in portrait or landscape mode. This is a persistent property.

Returns

Type	Description
Integer	0 – Prints the document in portrait mode. 1 – Prints the document in landscape mode.

AspectRatio

Description

Specifies the current aspect ratio of the document (the pictures). **This property is read-only.**

Returns

Type	Description
Integer	-1 – Not applicable. 0 – None. 1 – Based on the document. 2 – Normal (use scanlines). 3 – Letter (times 2).

CanCopy

Description

Specifies whether content is selected in the currently opened document. If this returns TRUE, the selection can be copied to the clipboard by using the [Copy](#) method. Any selected text can be obtained by using the [GetSelectedText](#) method. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE if the selected text can be copied. FALSE if the selected text cannot be copied.

CanDecreaseFont

Description

Specifies whether the document font size can be decreased. If this returns TRUE, the font size can be decreased by using the [DecreaseFont](#) method. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE if the font size can be decreased. FALSE if the font size cannot be decreased.

CanFind

Description

Specifies whether the currently opened document can be searched. If this returns TRUE, the document can be searched by using the [Find](#) method. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE if the document can be searched. FALSE if the document cannot be searched.

CanIncreaseFont

Description

Specifies whether the document font size can be increased. If this returns `TRUE`, the font size can be increased by using the `IncreaseFont` method. **This property is read-only.**

Returns

Type	Description
Boolean	<code>TRUE</code> if the font size can be increased. <code>FALSE</code> if the font size cannot be increased.

CanMultiObj

Description

Specifies whether a document contains multiple objects. If this property is `TRUE`, the `ObjName` property and `ChangeObject` method are applicable to the opened document. **This property is read-only.**

Returns

Type	Description
Boolean	<code>TRUE</code> if the document contains multiple objects. <code>FALSE</code> if the document does not contain multiple objects.

CanNextObj

Description

Specifies whether the next object is available in a multiple-object document. **This property is read-only.**

Examples of a multiple-object document include:

- A Microsoft Excel spreadsheet with multiple worksheets, where each worksheet is considered an object.
- A Microsoft PowerPoint presentation with multiple slides, where each slide is considered an object.

Returns

Type	Description
Boolean	TRUE – the next object is available. FALSE – the next object is not available.

CanPause

Description

Specifies whether the playing of a multimedia document can be paused. This returns TRUE only if you have a multimedia document opened and playing. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE if the document can be paused. FALSE if the document cannot be paused.

CanPlay

Description

Specifies whether a multimedia document can be played. This property is TRUE if the document is a multimedia (sound or digital video) document, and if the document is not currently playing. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE if the document can be played. FALSE if the document cannot be played.

CanPrevObj

Description

Specifies whether the previous object is available in a multiple-object document. **This property is read-only.**

Examples of a multiple-object document include:

- A Microsoft Excel spreadsheet with multiple worksheets, where each worksheet is considered an object.
- A Microsoft PowerPoint presentation with multiple slides, where each slide is considered an object.

Returns

Type	Description
Boolean	TRUE—the previous object is available. FALSE—the previous object is not available.

CanPrint

Description

Specifies whether the currently opened document is completely open and ready for printing. A document must be fully processed before it can be printed. Use the [OPENWaitOnOpen](#) property to make sure that the Open method does not return until the entire document is open. **This property is read-only.**

If this property is TRUE, the document can be printed by using the [PrintDlg](#) or [PrintOut](#) method.

Returns

Type	Description
Boolean	TRUE if the document can be printed. FALSE if the document cannot be printed.

CanSaveAs

Description

Specifies whether the currently opened document can be saved or converted to a different file. If this is `TRUE`, the document can be saved or converted by using the `SaveAs` and `Convert` methods. **This property is read-only.**

Returns

Type	Description
Boolean	<code>TRUE</code> if the document can be saved.

CanSelectAll

Description

Specifies whether all items in the currently opened document can be selected. If this property is `TRUE`, all items in the document can be selected by using the `SelectAll` method. **This property is read-only.**

Returns

Type	Description
Boolean	<code>TRUE</code> if the document can be selected.

CanStop

Description

Specifies whether the playing of a multimedia document can be stopped. This property returns `TRUE` only if you have a multimedia document opened and playing. **This property is read-only.**

Returns

Type	Description
Boolean	<code>TRUE</code> if the document can be stopped.

CanUnZip

Description

Returns TRUE if the currently opened document is a container file and there are subfiles selected in the file. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE if the document can be unzipped.

CanViewPane

Description

Specifies whether a preview pane can be viewed. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE – can show preview pane. This applies only to container files. FALSE – cannot show a preview pane.

CharSet

Description

Enables you to get and set the character set of a document that is open in the Viewing window.

Returns

Type	Description
Integer	A value from the enumerated type KVCharSet. See the kvtypes.h file for a description.

ContextMenu

Description

Turns the context menu on or off. The context menu is the menu that appears when you right mouse click in the client area. This is a persistent property.

Returns

Type	Description
Boolean	TRUE if the context menu is enabled.

DocumentClass

Description

Indicates which general class the currently opened document belongs to. **This property is read-only.**

The classes are as follows:

1 – Text document (ASCII)	6 – Fax document (FAX)
2 – Word processor document (WP)	7 – Presentation (PG)
3 – Spreadsheet document (SS)	8 – Archive document
4 – Image (Image)	9 – Other
5 – Multimedia document (MM)	

The document class is useful to determine whether a specific set of properties is applicable to the currently opened document. The codes specified in the brackets above are prefixes used for all persistent properties specific to that document type.

For example, a multimedia document has a `DocumentClass` of 5, and the properties `MMPlayOption` and `MMScaleMovie` are relevant only to multimedia documents. The value of these properties is ignored for any other kind of document.

The one exception to the above rule is the `WPPageLayout` property, which is applicable to both text and word processor documents.

Returns

Type	Description
Short	The document class (a value of 1 through 9, as described above).

DocumentFormat

Description

Specifies the document format value of a document. This value corresponds to the numerical equivalent of the `DocumentType` property value. For example, the "Microsoft Word for Windows" `DocumentType` corresponds to "44" for the `DocumentFormat` property. **This property is read-only.**

The document formats are defined in the `adinfo.h` header file.

Returns

Type	Description
Integer	n – The document format value.

DocumentType

Description

Specifies the type of the currently opened document, such as "Microsoft Word for Windows". If an `Open` fails because the document type is not supported for viewing by Viewing SDK, this property still contains the document type until another call to the `Open` method is made, even though no document is currently opened. **This property is read-only.**

The document formats are defined in the `adinfo.h` header file.

Returns

Type	Description
String	A description of the type of the currently opened document.

DrawPageCount

Description

Specifies the total number of pages in the document. Use this property with the `OPENMode` property set to 7 (drawing mode enabled). **This property is read-only.**

To make sure that the entire document is opened before the page count is retrieved, set the `OPENWaitOnOpen` property to `TRUE`. If `OPENWaitOnOpen` is not set to `TRUE`, the returned page count might not be accurate.

Returns

Type	Description
Long	n – The number of pages.

DrawPageHeight

Description

Specifies the page height. Use this property with the [OPENMode](#) property set to 7 (drawing mode enabled). **This property is read-only.**

Returns

Type	Description
Long	n – The page height.

DrawPageWidth

Description

Specifies the page width. Use this property with the [OPENMode](#) property set to 7 (drawing mode enabled). **This property is read-only.**

Returns

Type	Description
Long	n – The page width.

DrawWorkbookPageCount

Description

Specifies the number of workbook pages in spreadsheet documents. Use this property with the [OPENMode](#) property set to 7 (drawing mode enabled). **This property is read-only.**

Returns

Type	Description
Long	n – Number of pages.

FileName

Description

Specifies the file name of the currently opened document, for example `file1.doc`. **This property is read-only.**

Returns

Type	Description
String	The file name of the currently opened document.

HiLiteBackground

Description

Specifies the background highlight color. Use this property with the [OPENMode](#) property set to 5 (indexing enabled).

Returns

Type	Description
Long	n – The color value.

HiLiteForeground

Description

Specifies the foreground highlight color. Use this property with the [OPENMode](#) property set to 5 (indexing enabled).

Returns

Type	Description
Long	n – The color value.

HotKeys

Description

Specifies whether or not hotkeys are enabled for KeyView and the parent application. The default is TRUE.

Returns

Type	Description
Boolean	TRUE – Hotkeys are enabled in KeyView and disabled in parent applications. FALSE – Hotkeys are disabled in KeyView and enabled in parent applications.

ImageCustomSize

Description

Specifies the image document view in a custom size, but only if the ImageScaling property is set to 10. The range of acceptable values is 10 to 400 percent. This is a persistent property.

Returns

Type	Description
Integer	n – Specify a custom view size.

ImagePrintHorzAlign

Description

Specifies the horizontal print alignment of the image document. This is a persistent property.

Returns

Type	Description
Integer	0 – Print the image aligned to the left. 1 – Print the image aligned to horizontal center. 2 – Print the image aligned to the right.

ImagePrintMode

Description

Specifies the image document print mode. This is a persistent property.

Returns

Type	Description
Integer	0 – Print the image in its original size. 1 – Print the image at full-page size. 2 – Print the image at a customized scaling factor based on the value of the <code>ImagePrintPercent</code> property (see ImagePrintPercent , below).

ImagePrintPercent

Description

Specifies the image print mode in a custom size, but only if the [ImagePrintMode](#) property is set to 2 (customized scaling). The range of acceptable values is 10 to 400 percent. This is a persistent property.

Returns

Type	Description
Integer	<i>n</i> – Specify a custom print size.

ImagePrintVertAlign

Description

Specifies the vertical print alignment of the image document. This is a persistent property.

Returns

Type	Description
Integer	0 – Print the image aligned to the top of the page. 2 – Print the image aligned to the bottom of the page.

ImageScaling

Description

Specifies the image document view. This is a persistent property.

Returns

Type	Description
Integer	1 – Display the image to fit the selected portion of the image to the window, while maintaining the image's aspect ratio. 2 – Display the image to fit to the window, while maintaining the image's aspect ratio. 3 – Display the image in its original size. 10 – Display the image at a customized scaling factor based on the value of the ImageCustomSize property.

IndexBufCharSet

Description

Sets the character set for the returned indexed text buffer of an open document. See [GetNextTextBuffer](#) , on page 195 for more information.

Returns

Type	Description
Integer	A value from the KVCharSet enumerated type. See the kvtypes.h file for a description.

Invert

Description

Indicates and allows the inversion status of a document to be specified, for example, turning black to white and white to black.

Returns

Type	Description
Short	-1 – Not applicable to the opened document. 0 – The opened document is not inverted. 1 – The opened document is inverted.

JumpToFirstHiLite

Description

Specifies whether to jump to the first highlight. This applies only when using XML files with the Verity Developer's Kit (VDK) to specify highlights.

Returns

Type	Description
Boolean	TRUE – jump to the first highlight. FALSE – do not jump to the first highlight.

MMPlayOption

Description

Specifies whether a multimedia file plays continuously (loops). This is a persistent property.

Returns

Type	Description
Integer	0 – Play the multimedia file once and stop. 1 – The multimedia file loops.

MMScaleMovie

Description

Specifies whether the movie should be played at original size or if it should fit to window.

Returns

Type	Description
Boolean	TRUE – fit the movie to the window size. FALSE – play the movie at the original size.

ObjName

Description

Specifies the name of the currently active object. This applies only to multiple-object documents, such as spreadsheets with multiple worksheets. **This property is read-only.**

Returns

Type	Description
Boolean	TRUE if the context menu is enabled.

OPENDisableUI

Description

This property disables the generation of dialog boxes or message boxes by the KeyView control. Dialog boxes such as **Save As** still appear if you specifically call the `SaveAs` method.

Returns

Type	Description
Boolean	TRUE – do not generate message boxes or dialog boxes.

OPENHighLight

Description

Specify the text to appear highlighted in subsequently opened documents. For example, `OPENHighLight = 1 dog and mouse` highlights all occurrences of `dog` and `mouse` (case sensitive) in documents subsequently opened by using the `Open` method.

You can also search for non-adjacent words by separating them with a tab character, indicated by `(tab)`. For example, `OPENHighLight = 1 dog(tab)mouse` highlights all occurrences of `dog` or `mouse` (case sensitive).

Returns

Type	Description
String	The first character should be a 1 (case sensitive) or a 0 (case insensitive), followed by the text to be highlighted.

OPENMode

Description

The `OPENMode` property specifies the behavior of the `Open` method.

Returns

Type	Description
------	-------------

- | | |
|-------|---|
| Short | <ul style="list-style-type: none">0 – Default open, generates a formatted view of the document if the document format is supported.1 – Open with a text view of the document.2 – Open without generating a view. This is useful when, for example, you want to print a document without viewing it.3 – Open with a text view of the document automatically when the document is an unsupported format.5 – Open the file with indexing enabled. This generates text buffer events.6 – Open the file for indexing only. This generates text buffer events with document viewing disabled. To get text buffer events in this mode, you need to call the <code>GetNextTextBuffer</code> method—except for the first text buffer).7 – Open the file with drawing methods and properties enabled.8 – Open the files with revision marks enabled. |
|-------|---|

OPENWaitOnOpen

Description

If this property is set to `TRUE`, the `Open` method does not return until the entire document is open. This ensures that the document is fully processed before an operation (such as printing, converting, or searching) is performed, and is useful when you want to use an operation immediately after opening the document. Use the various "Can" properties, such as `CanPrint`, `CanSaveAs`, and `CanFind`, to determine whether the document has been completely processed and is ready for the operation.

If you are opening the document for viewing only, set this property to `FALSE` so that the first page of the document can be viewed as soon as it is processed.

Returns

Type	Description
------	-------------

- | | |
|---------|---|
| Boolean | If <code>TRUE</code> , the <code>Open</code> method does not return until the document is completely processed. |
|---------|---|

PrintAnnotations

Description

Specifies whether annotations should be printed in the printed output of a document.

Returns

Type	Description
Boolean	TRUE – print annotations. FALSE – do not print annotations.

PrintHeaders

Description

Specifies whether a file name and page number header should be printed at the top of each page of the printed output of a document.

Returns

Type	Description
Boolean	TRUE – print the header. FALSE – do not print the header.

RegIniMode

Description

The `RegIniMode` property specifies whether Viewing gets initialization information from an initialization file (.ini) or the registry. The `RegIniName` property must also be set. See [View Initialization Information, on page 23](#) for more information. This is a persistent property.

NOTE: By default, Viewing SDK looks for the initialization file in the Windows system directory; however, you can specify a file in another location by using the `RegIniName` property.

Returns

Type	Description
Short	1 – use the initialization file specified in the <code>RegIniName</code> property. 2 – use the registry entry specified in the <code>RegIniName</code> property.

RegIniName

Description

The initialization file (if `RegIniMode` is 1) or the registry key (if `RegIniMode` is 2) where the Viewing initialization information resides. The `RegIniMode` property must also be set. See [View Initialization Information, on page 23](#) for more information. This is a persistent property.

Returns

Type	Description
String	When <code>RegIniMode</code> is 1, this is the path and name of the initialization file. For example, <code>kvsdk.ini</code> or <code>C:\myprogram\myini.ini</code> . By default, Viewing looks for the initialization file in the Windows system directory. When <code>RegIniMode</code> is 2, this is the registry key under <code>HKEY_LOCAL_MACHINE\Software</code> . For example, <code>YourCompany\YourProduct</code> . Do not specify the complete registry name, only the <code>Company\Product</code> portion.

Rotate

Description

Rotates the currently opened document (fax or picture) and returns the current rotation angle.

Returns

Type	Description
Short	If the value is -1, the file cannot be rotated. To rotate the file, specify one of the following angles: 0, 90, 180, 270, or 360.

Sample Code

```
If KEYview1.Rotate > -1 Then ' Can the file be rotated?
    KEYview1.Rotate = 180    ' Rotate 180 degrees.
End If
```

SrcCharSet

Description

Sets the source character set of a document to be opened. This property is used to specify the character set for documents when the character set cannot be determined by Viewing, such as in the case of plain text documents.

Returns

Type	Description
Integer	A value from the KVCharSet enumerated type. See the kvtypes.h file for a description.

SSDisplayGrid

Description

Specifies whether the spreadsheet document displays gridlines. This is a persistent property.

Returns

Type	Description
Integer	0 – Do not display gridlines. 1 – Display gridlines. 2 – Display gridlines based on the setting in the original document.

SSDisplayHeaders

Description

Specifies whether the spreadsheet document displays and prints headings, such as row numbers and column letters. This is a persistent property.

Returns

Type	Description
Integer	0 – Do not display headers.
	1 – Display headers.
	2 – Display headers based on the setting in the original document.

SSViewObjects

Description

Specifies whether the spreadsheet document displays embedded graphic objects. This is a persistent property.

Returns

Type	Description
Integer	0 – Do not display graphic objects.
	1 – Display graphic objects.

TrgCharSet

Description

Sets the target character set of a document to be opened. This property forces the character set Viewing uses to display a document. For example, this allows Japanese documents to be accurately displayed on an English Windows machine if the Japanese fonts are available.

Returns

Type	Description
Integer	A value from the <code>KVCharSet</code> enumerated type. See the <code>kvtypes.h</code> file for a description.

ViewPane

Description

Specifies whether the preview pane of a container file is enabled. When the preview pane is enabled, the viewing area is divided into two panes: one pane displays the contents of the container file, the other displays the contents of the selected subfile.

Returns

Type	Description
Boolean	TRUE – the preview pane is enabled. FALSE – the preview pane is not enabled.

WPCustomSize

Description

Specifies the word processing document view in a custom size, but only if `WPViewMode` is set to 1 (page layout mode) and the `WPPageLayout` property is set to 10 (customized scaling factor). The range of acceptable values is 10 to 400 percent. This is a persistent property.

Returns

Type	Description
Integer	<i>n</i> – Specify a custom view size.

WPDisplayPict

Description

Specifies whether pictures in a word processing document are displayed. This is a persistent property.

Returns

Type	Description
Integer	0 – Do not display pictures. 1 – Display pictures.

WPPageLayout

Description

Specifies the page layout mode of the word processing document. This property applies only if [WPViewMode](#) is set to 1 (page layout mode). This is a persistent property.

Returns

Type	Description
Integer	0 – Display the document at full size (100%). 1 – Display the document at the current page width. 10 – Display the document at a customized scaling factor based on the value of the WPCustomSize property.

WPScaleTable

Description

Specifies whether tables in the word processing document are scaled. This property applies only if [WPViewMode](#) is set to 0 (fit to window mode). In this mode, setting the property to TRUE scales tables within documents. This is a persistent property.

Returns

Type	Description
Boolean	TRUE – scale tables to fit window. FALSE – tables retain their original size.

WPViewMode

Description

Specifies whether the word processing document fits to the window size or appears in page layout mode. This is a persistent property.

Returns

Type	Description
Integer	0 – Display the document in fit to window mode.
	1 – Display the document in page layout mode.

Chapter 12: Control Events

This section describes the Viewing ActiveX control events. It includes the following topics:

- [Annotation](#) 245
- [KeyDown](#) 246
- [MouseUp](#) 246
- [OpenDocDone](#) 247
- [PageNumber](#) 247
- [PrintDone](#) 248
- [PrintDoneEx](#) 248
- [Selection](#) 249
- [TextBuffer](#) 249
- [UserClick](#) 250
- [ViewExtent](#) 250
- [ViewFile](#) 251

NOTE: The Viewing control's events in this chapter show the syntax in Visual Basic.

Annotation

Description

These events are generated to report that the user clicked an annotation.

Syntax

`Annotation(bDoubleClick, lLogicalAddress)`

Parameters

Parameter	Type	Description
bDoubleClick	Boolean	This flag is TRUE if the user double-clicked, and FALSE if the user single-clicked.
lLogicalAddress	Long	The logical address of the mouse click.

KeyDown

Description

These events are generated to indicate that a key is pressed. The `HotKeys` property must be set to `FALSE` before the file is opened for these events to be generated.

For more information about `HotKeys`, see [HotKeys](#), on page 231.

Syntax

```
KeyDown(KeyCode, Shift)
```

Parameters

Parameter	Type	Description
<code>KeyCode</code>	Integer	The virtual key code of the key that was pressed.
<code>Shift</code>	Integer	This parameter is always 0.

MouseUp

Description

These events are generated to indicate that the right mouse button is released. The `ContextMenu` property must be set to `FALSE` before the file is opened for these events to be generated. Users can combine `MouseUp` and `ContextMenu` to disable `KeyView`'s context menu and implement their own.

For more information about `ContextMenu`, see [ContextMenu](#) , on page 227.

Syntax

```
MouseUp(Button, Shift, X, Y)
```

Parameters

Parameter	Type	Description
<code>Button</code>	Short	This parameter should always be set to 2.
<code>Shift</code>	Short	This parameter should always be set to 0.
<code>X</code>	Long	The x-coordinate of the cursor when a mouse button was released. The

Parameter	Type	Description
-----------	------	-------------

		coordinate is relative to the upper-left corner of the control window.
Y	Long	The y-coordinate of a cursor when a mouse button was released. The coordinate is relative to the upper-left corner of the control window.

OpenDocDone

Description

These events are generated to indicate the percentage (0-100) of the document processed. These events are generated in response to the `Open` method call.

Syntax

```
OpenDocDone(PercentageDone)
```

Parameters

Parameter	Type	Description
PercentageDone	Short	The percentage of the document processed.

PageNumber

Description

This event is generated in response to an `Open` method call. It indicates the total number of pages in the document and the page number at which the document is currently being viewed.

Syntax

```
PageNumber(CurrentPage, TotalPages)
```

Parameters

Parameter	Type	Description
CurrentPage	Short	The current page number of the document.
TotalPages	Short	The total number of pages in the document. This number reflects the total number of processed pages, until the <code>OpenDocDone</code> event reaches 100.

PrintDone

Description

This event indicates that the printing of the document is complete.

NOTE: This event does not return any status reports. If you want to receive a status report about your print job, use [PrintDoneEx](#).

Syntax

PrintDone()

Parameters

None

PrintDoneEx

Description

This event indicates that the printing of the document is complete, and reports the status of the print job.

NOTE: If you do not want to receive status reports, use [PrintDone](#) .

Syntax

PrintDoneEx(printStatus)

Parameters

Parameter	Type	Description
printStatus	Short	Returns one of four status reports: 0: General error 1: Success 2: Printing aborted by user 3: Printing aborted due to Windows GDI call failure

Selection

Description

These events are generated to report that the user changed the selection state.

Syntax

```
Selection(bHaveSelection, lStart, lEnd)
```

Parameters

Parameter	Type	Description
bHaveSelection	Boolean	The flag is TRUE if a selection exists, or FALSE if a selection does not exist.
lStart	Long	The logical address of the start of the view.
lEnd	Long	The logical address of the end of the view.

TextBuffer

Description

This event returns a text buffer.

Syntax

```
TextBuffer(lBaseLogicalAddress, szBuffer)
```

Parameters

Parameter	Type	Description
lBaseLogicalAddress	Long	The logical address of the text buffer.
szBuffer	String	A pointer to the text buffer.

UserClick

Description

These events are generated to report that the user clicked the mouse on the document. To generate the events, set the [OPENMode](#) property to 5 before the document is opened. This indicates that the file should be opened with indexing enabled.

Syntax

```
UserClick(bDoubleClick, lLogicalAddress)
```

Parameters

Parameter	Type	Description
bDoubleClick	Boolean	The flag is TRUE if the user double-clicked, and FALSE if the user single-clicked.
lLogicalAddress	Long	The logical address of the mouse click.

ViewExtent

Description

These events are generated to report that the user changed the view extent.

Syntax

```
ViewExtent(lStart, lEnd)
```

Parameters

Parameter	Type	Description
lStart	Long	The logical address of the start of the view.
lEnd	Long	The logical address of the end of the view.

ViewFile

Description

This event is generated in response to a user double-clicking on the contents of a displayed container file or using the UnZip method. You can then respond to this event to Open the file.

Syntax

```
ViewFile(FileName, DeleteFile)
```

Parameters

Parameter	Type	Description
FileName	String	The path and file name of the file that the user double-clicked within the container file.
DeleteFile	Boolean	If this is TRUE, delete the file when you are finished with it (it is a temporary file).

Part IV: Appendixes

This section provides information on files required for redistribution, character sets, and supported and detected formats, and includes the following appendixes:

- [Supported Formats, on page 253](#)
- [Character Sets, on page 284](#)
- [File Formats and Extensions, on page 318](#)
- [Extract and Format Lotus Notes Subfiles, on page 343](#)
- [List of Files Required for Redistribution, on page 356](#)
- [File Format Detection, on page 297](#)
- [Password Protected Files, on page 369](#)

Appendix A: Supported Formats

This section lists information about the file formats that can be detected and processed (either filtered, converted, or displayed) by the KeyView suite of products. The KeyView suite includes KeyView Filter SDK, KeyView Export SDK, and KeyView Viewing SDK.

- [Supported Formats](#) 253
- [Supported Formats \(Detected\)](#) 277

Supported Formats

The tables in this section provide the following information:

- The file formats supported by the Filter API, Export API, Viewing API, and File Extraction API. The supported versions and the format's extension are also listed.

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

- The file formats for which KeyView can detect and extract the character set and metadata information (properties such as title, author, and subject).

Even though a file format might be able to provide character set information, some documents might not contain character set information. Therefore, the document reader would not be able to determine the character set of the document. In this case, either the operating system code page or the character set specified in the API is used.

- The document reader used to filter each format.

Key to Support Tables

Symbol	Description
Y	The format is supported. You can extract metadata for this format. You can determine the character set for this format.
N	The format is not supported. You cannot extract metadata for this format. You cannot determine the character set for this format.
P	Partial metadata is extracted from this format. Some non-standard fields are not extracted.
T	Only text is extracted from this format. Formatting information is not extracted.
M	Only metadata (title, subject, author, and so on) is extracted from this format. Text and

Key to Support Tables, continued

Symbol	Description
	formatting information are not extracted.

Archive Formats

Supported Archive Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
7-Zip	4.57	z7zsr, multiarcsr ¹	7Z	N	N	Y	Y	N	n/a	N
AD1	n/a	ad1sr	AD1	N	N	Y	Y	N	n/a	N
ARJ	n/a	multiarcsr	ARJ	N	N	N	Y	N	n/a	N
B1	n/a	b1sr	B1	N	N	Y	Y	N	n/a	N
BinHex	n/a	kvhqxsr	HQX	N	N	Y	Y	N	n/a	N
Bzip2	n/a	bzip2sr	BZ2	N	N	Y	Y	N	n/a	N
Expert Witness Compression Format (EnCase)	6	encasesr	E01, L01	N	N	Y	Y	N	n/a	N
	7	encase2sr	Lx01	N	N	Y	Y	N	n/a	N
GZIP	2	kvgzsr	GZ	N	N	N	Y	N	n/a	N
		kvgz	GZ	N	N	Y	N	N	n/a	N
ISO	n/a	isosr	ISO	N	N	Y	Y	N	n/a	N
Java Archive	n/a	unzip	JAR	N	N	Y	Y	N	n/a	N
Legato EMailXtender	n/a	emxsr	EMX	N	N	Y	Y	N	n/a	N

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
Archive										
MacBinary	n/a	macbinsr	BIN	N	N	Y	Y	N	n/a	N
Mac Disk Copy Disk Image	n/a	dmgsr	DMG	N	N	Y	Y	N	n/a	N
Microsoft Backup File	n/a	bkfsr	BKF	N	N	Y	Y	N	n/a	N
Microsoft Cabinet format	1.3	cabsr	CAB	N	N	Y	Y	N	n/a	N
Microsoft Compiled HTML Help	3	chmsr	CHM	N	N	Y	Y	N	n/a	N
Microsoft Compressed Folder	n/a	lzhsr	LZH LHA	N	N	N	Y	N	n/a	N
PKZIP	through 9.0	unzip	ZIP	N	N	Y	Y	N	n/a	N
RAR archive	2.0 through 3.5	rarsr	RAR	N	N	N	Y	N	n/a	N
RAR5 archive	5	multiarcsr	RAR5	N	N	N	Y	N	n/a	N
Tape Archive	n/a	tarsr	TAR	N	N	Y	Y	N	n/a	N
UNIX Compress	n/a	kvzeesr	Z	N	N	N	Y	N	n/a	N
		kvzee	Z	N	N	Y	N	N	n/a	N
UUEncoding	all versions	uudsr	UUE	N	N	Y	Y	N	n/a	N
XZ	n/a	multiarcsr	XZ	N	N	N	Y	N	n/a	N

Supported Archive Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Windows Scrap File	n/a	olesr	SHS	N	N	N	Y	N	n/a	N
WinZip	through 10	unzip	ZIP	N	N	Y	Y	N	n/a	N

Binary Format**Supported Binary Formats**

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

Computer-Aided Design Formats**Supported CAD Formats**

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
AutoCAD Drawing	R13, R14, R15/2000, 2004, 2007, 2010, 2013	kpODArdr kpDWGrdr ¹	DWG	Y	Y ²	Y ³	N	Y	Y	N

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

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

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

Supported CAD Formats, continued

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

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

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

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

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

⁵Viewing 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.

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

⁷Visio 2013 is supported in Viewing only, with the support of ActiveX components from the Microsoft Visio 2013 Viewer. Image fidelity is supported but other features, such as highlighting, are not.

Supported CAD Formats, continued

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

Database Formats**Supported Database Formats**

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

Desktop Publishing**Supported Desktop Publishing Formats**

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

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

Display Formats

Supported Display Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe PDF	1.1 to 1.7	pdfsr	PDF	Y	Y	N	Y ¹	Y	Y	N
		pdf2sr	PDF	N	Y	N	N	N	N	N
		kppdfldr	PDF	N	Y	Y	N	N	N	N
		kppdf2ldr ²	PDF	N	N	Y	N	N	N	N

Graphic Formats

Supported Graphic Formats

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Computer Graphics Metafile	n/a	kpcgmrdr ³	CGM	Y	Y	Y	N	N	N	N
CorelDRAW ⁴	through 9.0 10, 11, 12, X3	kpcdrdr	CDR	N	Y	Y	N	N	N	N

¹Includes support for extraction of subfiles from PDF Portfolio documents.

²kppdf2ldr is an alternate graphic-based reader that produces high-fidelity output but does not support other features such as highlighting or text searching.

³Files with non-partitioned data are supported.

⁴CDR/CDR with TIFF header.

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
DCX Fax System	n/a	kpdcxrdr	DCX	N	Y	Y	N	N	N	N
Digital Imaging & Communications in Medicine (DICOM)	n/a	dcmsr	DCM	M	N	N	N	Y	N	N
Encapsulated PostScript (raster)	TIFF header	kpepsrdr	EPS	N	Y	Y	N	N	N	N
Enhanced Metafile	n/a	kpemfrdr	EMF	Y	Y	Y	N	Y	N	N
GIF	87, 89	kpgifrdr	GIF	N	Y	Y	N	N	N	N
		gifsr		M	M	N	N	Y	N	N
JBIG2	n/a	kpJBIG2rdr	JBIG2	N	Y	Y	N	N	N	N
JPEG	n/a	kpjpgdr	JPEG	N	Y	Y	N	N	N	N
		jpgsr		M	M	N	N	Y	N	N
JPEG 2000	n/a	kpjp2000rdr	JP2, JPF, J2K, JPWL, JPX, PGX	N	Y	Y	N	N	N	N
		jp2000sr		M	M	N	N	Y	N	N
Lotus AMIDraw Graphics	n/a	kpsdwrdr	SDW	N	Y	Y	N	N	N	N
Lotus Pic	n/a	kppicrdr	PIC	Y	Y	Y	N	N	N	N
Macintosh Raster	2	kppctrdr	PIC PCT	N	Y	Y	N	N	N	N
MacPaint	n/a	kpmacrdr	PNTG	N	Y	Y	N	N	N	N

Supported Graphic Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Office Drawing	n/a	kpmrdr	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
SGI RGB Image	n/a	kpsgirdr	RGB	N	Y	Y	N	N	N	N
Sun Raster Image	n/a	kpsunrdr	RS	N	Y	Y	N	N	N	N
Tagged Image File	through 6.0 ¹	tifsr	TIFF	M	M	N	N	Y	N	N
		kptifdr	TIFF	N	Y	Y	N	N	N	N
Truevision Targa	2	kptrdr	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	kpicondr	ICO	N	Y	Y	N	N	N	N
Windows Metafile	3	kpwmfrdr	WMF	Y	Y	Y	N	N	N	N
WordPerfect Graphics 1	1	kpwpgrdr	WPG	N	Y	Y	N	N	N	N

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.

Supported Graphic Formats, continued

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

Mail Formats**Supported Mail Formats**

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

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
	Eudora 6.2									
Microsoft Entourage Database	2004	entsr	various	N	N	Y	Y	Y	Y	N
Microsoft Outlook	97, 2000, 2002, 2003, 2007, 2010, 2013, 2016	msgsr ¹	MSG, OFT	Y	T	T	Y	Y	Y ²	N
Microsoft Outlook DBX	5.0, 6.0	dbxsr	DBX	N	N	Y	Y	Y	Y	N
Microsoft Outlook Express	Windows 6 MacIntosh 5	emlsr ³	EML	Y	T	T	Y	Y	Y	N
		mbxsr ⁴	EML	N	N	T	Y	Y	Y	N
Microsoft Outlook iCalendar	1.0, 2.0	icssr	ICS, VCS	N	N	Y	Y	Y	Y	N
Microsoft Outlook for Macintosh	2011	olmsr	OLM	N	N	Y	Y	N	Y	N
Microsoft Outlook Offline Storage File	97, 2000, 2002, 2003, 2007, 2010,	pffsr ⁵	OST	N	N	Y	Y	Y	Y	N

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
	2013									
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). Note that the native PST reader (*pstsr*) works only on Windows, and requires that you have Microsoft Outlook installed. As an alternative, the MAPI reader (*pstnsr*) runs on all platforms, and does not require Microsoft Outlook. For more information on using the native PST reader or the MAPI reader, see the sections 'Use the Native PST Reader (*pstnsr*)' and 'Use the MAPI Reader (*pstsr*)' in Chapter 3.

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
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
		aiffr	AIFF	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
MPEG-1 Audio layer 3	ID3 v1 and v2	MCI	MP3	N	N	Y	N	N	N	N
		mp3sr	MP3	M	M	Y	N	Y	N	N
MPEG-1 Video	2, 3	MCI	MPG	N	N	Y	N	N	N	N
MPEG-2 Audio	n/a	MCI	MPEGA	N	N	Y	N	N	N	N
MPEG-4 Audio	n/a	mpeg4sr	MP4 3GP	M	N	N	N	Y	N	N
NeXT/Sun Audio	n/a	MCI	AU	N	N	Y	N	N	N	N
QuickTime Movie	2, 3, 4	MCI	QT MOV	N	N	Y	N	N	N	N

Supported Multimedia Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Windows Video	2.1	MCI	AVI	N	N	Y	N	N	N	N

NOTE:

Depending on the default multimedia player installed on your computer, the View API might not be able to play some supported multimedia formats. To play multimedia files, the View API uses the Windows Media Control Interface (MCI) to communicate with the multimedia player installed on your computer. If the player does not play a multimedia file that is supported by the Viewing SDK, the View API cannot play the file.

If you cannot play a supported multimedia file by using the View API, install a different multimedia player or compressor/decompressor (codec) component.

Presentation Formats**Supported Presentation Formats**

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Keynote	2, 3, '08, '09	kpIWPGdrdr	GZ	Y	Y	Y	N	Y	Y	N
Applix Presents	4.0, 4.2, 4.3, 4.4	kpagrdr	AG	Y	Y	Y	N	N	N	N
Corel Presentations	6, 7, 8, 9, 10, 11, 12, X3	kpshwrdr	SHW	Y	Y	Y	N	N	N	N
Extensible Forms Description Language	n/a	kpXFDLrdr	XFD XFDL	Y	Y	Y	N	Y	Y	N
Lotus Freelance Graphics	96, 97, 98, R9, 9.8	kpprzrdr	PRZ	Y	Y	Y	N	N	N	N
Lotus Freelance Graphics 2	2	kpprerdr	PRE	Y	Y	Y	N	N	N	N

Supported Presentation Formats, continued

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

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

Supported Presentation Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
OASIS Open Document Format	1, 2 ¹	kpodfrdr	SXD SXI ODG ODP	Y	Y	Y	Y ²	Y	Y	N
OpenOffice Impress, LibreOffice Impress	1 to 5	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N
StarOffice Impress	6, 7, 8, 9	sosr	SXI SXP ODP	Y	T	T	N	Y	Y	N

Spreadsheet Formats**Supported Spreadsheet Formats**

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Numbers	'08, '09	iwsssr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16	iwss13sr	NUMBERS	Y	T	T	N	N	Y	N
Applix Spreadsheets	4.2, 4.3, 4.4	assr	AS	Y	Y	Y	N	N	Y	N
Comma Separated Values	n/a	csvsr	CSV	Y	Y	Y	N	N	N	N

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

²Supported using the olesr embedded objects reader.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Corel Quattro Pro	5, 6, 7, 8	qpssr	WB2 WB3	Y	Y	Y	N	P	Y	N
	X4	qpwsr	QPW	Y	N	Y	N	P	Y	N
Data Interchange Format	n/a	difsr		Y	Y	Y	N	N	N	N
Lotus 1-2-3	96, 97, R9, 9.8	l123sr	123	Y	Y	Y	N	P	Y	N
Lotus 1-2-3	2, 3, 4, 5	wkssr	WK4	Y	Y	Y	N	N	Y	N
Lotus 1-2-3 Charts	2, 3, 4, 5	kpchtrdr	123	N	Y	Y	N	N	N	N
Microsoft Excel Charts	2, 3, 4, 5, 6, 7	kpchtrdr	XLS	N	Y	Y	N	N	N	N
Microsoft Excel Macintosh	98, 2001, v.X, 2004	xlssr	XLS	Y	Y	Y	Y ¹	Y	Y	N
Microsoft Excel Windows	2.2 through 2003	xlssr	XLS XLW XLT XLA	Y	Y	Y	Y ²	Y	Y	Y
Microsoft Excel Windows XML	2007, 2010, 2013, 2016	xlxsxr	XLSX XLTX XLSM XLTM XLAM	Y	Y	Y	Y	Y	Y	Y
Microsoft Excel Binary	2007, 2010,	xlsbsr	XLSB	Y	Y	Y	N	N	N	N

¹Supported using the embedded objects reader olesr.²Supported for versions 97 and higher using the embedded objects reader olesr.

Supported Spreadsheet Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Format	2013, 2016									
Microsoft Works Spreadsheet	2, 3, 4	mwssr	S30 S40	Y	Y	Y	N	N	Y	N
OASIS Open Document Format	1, 2 ¹	odfssr	ODS SXC STC	Y	Y	Y	Y ²	Y	Y	N
OpenOffice Calc, LibreOffice Calc	1 to 5	sosr	SXC ODS OTS	Y	T	T	N	Y	Y	N
StarOffice Calc	6, 7, 8, 9	sosr	SXC ODS	Y	T	T	N	Y	Y	N

Text and Markup Formats**Supported Text and Markup Formats**

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
ANSI	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
ASCII	n/a	afsr	TXT	Y	Y	Y	N	N	N	N
HTML	3, 4	htmsr	HTM	Y	Y	Y	N	P	Y	N
Microsoft Excel Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N

¹Generated by OpenOffice Calc 2.0, StarOffice 8 Calc, and IBM Lotus Symphony Spreadsheet 3.0.²Supported using the embedded objects reader olesr.

Supported Text and Markup Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Microsoft Word Windows XML	2003	xmlsr	XML	Y	T	T	N	Y	Y	N
Microsoft Visio XML	2003	xmlsr	VDX VTX	Y	T	T	N	Y	Y	N
MIME HTML	n/a	mhtsr	MHT	Y	Y	Y	N	Y	Y	N
Rich Text Format	1 through 1.7	rtfsr	RTF	Y	Y	Y	N	P	Y	Y
Unicode HTML	n/a	unihtmsr	HTM	Y	Y	Y	N	Y	Y	N
Unicode Text	3, 4	unisr	TXT	Y	Y	Y	N	N	Y	N
XHTML	1.0	htmsr	HTM	Y	Y	Y	N	Y	Y	N
XML (generic)	1.0	xmlsr	XML	Y	T	T	N	Y	Y	N

Word Processing Formats**Supported Word Processing Formats**

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Adobe FrameMaker Interchange Format	5, 5.5, 6, 7	mifsr	MIF	Y	Y	Y	N	N	Y	N
Apple iChat Log	1, AV 2 AV 2.1, AV 3	ichatsr	ICHAT	Y	Y	Y	N	N	N	N

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Apple iWork Pages	'08, '09	iwwpsr	GZ	Y	Y	Y	N	Y	Y	N
	'13, '16	iwwp13sr	PAGES	Y	T	T	N	N	N	N
Applix Words	3.11, 4, 4.1, 4.2, 4.3, 4.4	awsr	AW	Y	Y	Y	N	N	Y	Y
Corel WordPerfect Linux	6.0, 8.1	wp6sr	WPS	Y	Y	Y	N	P	Y	N
Corel WordPerfect Macintosh	1.02, 2, 2.1, 2.2, 3, 3.1	wpmsr	WPM	Y	Y	Y	N	N	Y	N
Corel WordPerfect Windows	5, 5.1	wosr	WO	Y	Y	Y	N	P	Y	Y
Corel WordPerfect Windows	6, 7, 8, 9, 10, 11, 12, X3	wp6sr	WPD	Y	Y	Y	N	P	Y	Y
DisplayWrite	4	dw4sr	IP	Y	Y	Y	N	N	Y	N
Folio Flat File	3.1	foliosr	FFF	Y	Y	Y	N	Y	Y	Y
Founder Chinese E-paper Basic	3.2.1	cebsr ¹	CEB	Y	N	N	N	N	N	N
Fujitsu Oasys	7	oa2sr	OA2	Y	Y	Y	N	P	N	N

¹This reader is only supported on Windows 32-bit platforms.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Haansoft Hangul	97	hwpsr	HWP	Y	N	N	N	N	Y	N
	2002, 2005, 2007, 2010	hwposr	HWP	Y	T	T	Y	Y	Y	N
Health level7	2.0	hl7sr	HL7	Y	Y	Y	N	Y	Y	N
IBM DCA/RFT (Revisable Form Text)	SC23-0758-1	dcasr	DC	Y	Y	Y	N	N	Y	N
JustSystems Ichitaro	8 through 2013	jtdsr	JTD	Y	Y	Y	N	P	N	Y
Lotus AMI Pro	2, 3	lasr	SAM	Y	Y	Y	N	P	Y	Y
Lotus AMI Professional Write Plus	2.1	lasr	AMI	Y	Y	Y	N	N	N	Y
Lotus Word Pro	96, 97, R9	lwpsr	LWP	Y	Y	Y	N	P	N	Y
Lotus SmartMaster	96, 97	lwpsr	MWP	Y	Y	Y	N	N	N	N
Microsoft Word Macintosh	4, 5, 6, 98	mbsr	DOC	Y	Y	Y	N	Y	N	Y
	2001, v.X, 2004	mw8sr	DOC DOT	Y	Y	Y	Y ¹	Y	Y	N
Microsoft Word PC	4, 5, 5.5, 6	mwsr	DOC	Y	Y	Y	N	N	N	Y
Microsoft Word Windows	1.0 and 2.0	misr	DOC	Y	Y	Y	N	N	N	Y

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 Windows	6, 7, 8, 95	mw6sr	DOC	Y	Y	Y	N	Y	Y	Y
Microsoft Word Windows	97, 2000, 2002, 2003	mw8sr	DOC DOT	Y	Y	Y	Y ¹	Y	Y	Y
Microsoft Word Windows XML	2007, 2010, 2013, 2016	mwxsr	DOCM DOCX DOTX DOTM	Y	Y	Y	Y	Y	Y	Y
Microsoft Works	1, 2, 3, 4	mswsr	WPS	Y	Y	Y	N	N	N	Y
Microsoft Works	6, 2000	msw6sr	WPS	Y	Y	Y	N	N	N	Y
Microsoft Windows Write	1, 2, 3	mwsr	WRI	Y	Y	Y	N	N	Y	N
OASIS Open Document Format	1, 2 ²	odfwpsr	ODT SXW STW	Y	Y	Y	Y ³	Y	Y	Y
Omni Outliner	v3, OPML, OOutline	oo3sr	OO3 OPML OOUTLINE	Y	Y	Y	N	N	Y	N
OpenOffice Writer, LibreOffice Writer	1 to 5	sosr	SXW ODT	Y	T	T	N	Y	Y	N

¹Supported using the embedded objects reader olesr.²Generated by OpenOffice Writer 2.0, StarOffice 8 Writer, and IBM Lotus Symphony Documents 3.0.³Supported using the embedded objects reader olesr.

Supported Word Processing Formats, continued

Format	Version	Reader	Extension	Filter	Export	View	Extract	Metadata	Charset	Header/Footer
Open Publication Structure eBook	2.0, 3.0	epubsr	EPUB	Y	Y	Y	N	Y	Y	N
StarOffice Writer	6, 7, 8, 9	sosr	SXW ODT	Y	T	T	N	Y	Y	N
Skype Log	3	skypesr	DBB	Y	Y	Y	N	N	N	N
WordPad	through 2003	rtfsr	RTF	Y	Y	Y	N	P	Y	N
XML Paper Specification	n/a	xpssr	XPS	Y	T	T	N	N	N	N
XyWrite	4.12	xywsr	XY4	Y	Y	Y	N	N	N	N
Yahoo! Instant Messenger	n/a	yimsr ¹	DAT	Y	Y	Y	N	N	N	N

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.

Supported Formats (Detected)

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

The formats listed in [Supported Formats, on page 253](#) can be detected as well as filtered, exported, and viewed.

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

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

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

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

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

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

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

Appendix B: Character Sets

This section provides information on the handling of character sets in the KeyView suite of products, which includes KeyView Filter SDK, KeyView Export SDK, and KeyView Viewing SDK.

- [Multibyte and Bidirectional Support](#) 284
- [Coded Character Sets](#) 292

Multibyte and Bidirectional Support

The KeyView SDKs can process files that contain multibyte characters. A multibyte character encoding represents a single character with consecutive bytes. KeyView can also process text from files that contain bidirectional text. Bidirectional text contains both Latin-based text which is read from left to right, and text that is read from right to left (Hebrew and Arabic).

[Multibyte and bidirectional support](#), below indicates which character encodings are supported by KeyView for each format.

Multibyte and bidirectional support

Format	Single-byte	Multibyte	Bidirectional
Archive			
7-Zip (7Z)	n/a	n/a	n/a
AD1 Evidence file	n/a	n/a	n/a
ADJ	n/a	n/a	n/a
B1	n/a	n/a	n/a
BinHex (Hqx)	n/a	n/a	n/a
Bzip2 (BZ2)	n/a	n/a	n/a
EnCase – Expert Witness Compression Format (E01)	n/a	n/a	n/a
GZIP (GZ)	n/a	n/a	n/a
ISO (ISO)	n/a	n/a	n/a
Java Archive (JAR)	n/a	n/a	n/a
Legato EMailXtender Archive (EMX)	n/a	n/a	n/a
MacBinary (BIN)	n/a	n/a	n/a
Mac Disk Copy Disk Image (DMG)	n/a	n/a	n/a
Microsoft Backup File (BKF)	n/a	n/a	n/a

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
Display			
Adobe Portable Document Format (PDF) (basic reader)	Y	Y ¹	Y

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Adobe Portable Document Format (PDF) (graphic-based reader)	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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
WordPerfect Graphics 1 (WPG)	Y	N	N
WordPerfect Graphics 2 (WPG)	Y	N	N
Mail			
Documentum EMC MF Format	Y	Y	Y
Domino XML Language (DXL)	Y	Y	N
GroupWise FileSurf	Y	N	N
Legato Extender (ONM)	Y	Y	N
Lotus Notes database (NSF)	Y	Y	Y
Mailbox (MBX)	Y	Y	Y
Microsoft Entourage Database	Y	Y	Y
Microsoft Outlook (MSG)	Y	Y	Y
Microsoft Outlook Express (EML)	Y	Y	Y
Microsoft Outlook iCalendar	Y	Y	Y
Microsoft Outlook for Macintosh	Y	Y	Y
Microsoft Outlook Offline Storage File	Y	Y	Y
Microsoft Outlook Personal File Folders (PST)	Y	Y	Y
Microsoft Outlook vCard Contact	?	?	?
Text Mail (MIME)	Y	Y	Y
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
Comma Separated Values (CSV)	character set 1252 only	N	N
Corel Quattro Pro (QPW/WB3)	Y	N	N
Data Interchange Format (DIF)	Y	Y	Y ²
Lotus 1-2-3 (123)	Y	Y	Y
Lotus 1-2-3 (WK4)	Y	Y	N
Lotus 123 Charts (123)	Y	Y	N
Microsoft Excel Charts (XLS)	Y	Y	N
Microsoft Excel Macintosh (XLS)	Y	N	N
Microsoft Excel Windows (XLS)	Y	Y	Y ²
Microsoft Excel Windows XML 2007 (XLSX)	Y	Y	N
Microsoft Office Excel Binary Format (XLSB)	Y	Y	N
Microsoft Works Spreadsheet (S30/S40)	Y	N	N
OASIS Open Document (ODS)	Y	Y	N
OpenOffice Calc (ODS)	Y	Y	N
StarOffice Calc (ODS)	Y	Y	N
Text and Markup			
ANSI (TXT)	Y	Y	Y ²
ASCII (TXT)	Y	Y	Y ²
HTML (HTM)	Y	Y	Y ^{2, 3}
Microsoft Excel Windows XML 2003	Y	Y	Y
Microsoft Word for Windows XML 2003	Y	Y	Y
Microsoft Visio XML 2003	Y	Y	Y
Rich Text Format (RTF)	Y	Y	Y ³
Unicode HTML	Y	Y	Y ^{2, 3}
Unicode Text (TXT)	Y	Y	Y ²
XHTML	Y	Y	Y ³

Multibyte and bidirectional support, continued

Format	Single-byte	Multibyte	Bidirectional
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
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

Multibyte and bidirectional support, continued

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

1

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

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

2

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

3

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

Coded Character Sets

This section lists which character set you can use to specify the target character set. The coded character sets are enumerated in `kvtypes.h` and defined in the class.

Code Character Sets

Coded Character Set	Description	Can be set as target charset?
KVCS_UNKNOWN	Unknown character set	N
KVCS_SJIS	Japanese (uses multibyte encoding), cp932	Y
KVCS_GB	Simplified Chinese (China, Singapore, Malaysia) cp936	Y
KVCS_BIG5	Traditional Chinese (Taiwan, Hong Kong, Macaw) cp950	Y
KVCS_KSC	Korean, cp949	Y
KVCS_1250	Windows Latin 2 (Central Europe)	Y
KVCS_1251	Windows Cyrillic (Slavic)	Y
KVCS_1252	Windows Latin 1 (ANSI)	Y
KVCS_1253	Windows Greek	Y
KVCS_1254	Windows Latin 5 (Turkish)	Y
KVCS_1255	Windows Hebrew	Y
KVCS_1256	Windows Arabic	Y
KVCS_1257	Windows Baltic Rim	Y
KVCS_1258	Windows Vietnamese	Y
KVCS_8859_1	ISO 8859-1 Latin 1 (Western Europe, Latin America)	Y
KVCS_8859_2	ISO 8859-2 Latin 2 (Central Eastern Europe)	Y
KVCS_8859_3	ISO 8859-3 Latin 3 (S.E. Europe)	Y
KVCS_8859_4	ISO 8859-4 Latin 4 (Scandinavia/Baltic)	Y
KVCS_8859_5	ISO 8859-5 Latin/Cyrillic	Y
KVCS_8859_6	ISO 8859-6 Latin/Arabic	Y

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
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
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

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_037	EBCDIC code page 037	Y
KVCS_1026	EBCDIC code page 1026	Y
KVCS_500	EBCDIC code page 500	Y
KVCS_875	EBCDIC code page 875	Y
KVCS_LMBCS	Lotus multibyte character set Group 1 and Group 2	N
KVCS_UNICODE	Unicode, UCS-2	Y
KVCS_UTF16	16-bit Unicode transformation format	Y
KVCS_UTF8	8-bit Unicode transformation format	Y
KVCS_UTF7	7-bit Unicode transformation format	Y
KVCS_2022_JP	ISO 2022-JP, Japanese mail and news safe encoding (JIS-7)	N
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

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
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_CNS1	Adobe-CNS1-2 character collection	N
KVCS_2022_JP_8	ISO 2022-JP, Japanese mail and news safe encoding (JIS8)	N
KVCS_720	Arabic DOS-720	Y
KVCS_VISCII	Vietnamese VISCII	Y
KVCS_8859_10	ISO 8859-10 (Latin 6 Nordic)	Y ¹
KVCS_8859_13	ISO 8859-13 (Latin 7 Baltic)	Y ¹
KVCS_57002	ISCII Devanagari (x-iscii-de)	Y ¹
KVCS_57003	ISCII Bengali (x-iscii-be)	Y ¹
KVCS_57004	ISCII Tamil (x-iscii-ta)	Y ¹
KVCS_57005	ISCII Telugu (x-iscii-te)	Y ¹
KVCS_57006	ISCII Assamese (x-iscii-as)	Y ¹
KVCS_57007	ISCII Oriya (x-iscii-or)	Y ¹
KVCS_57008	ISCII Kannada (x-iscii-ka)	Y ¹
KVCS_57009	ISCII Malayalam (x-iscii-ma)	Y ¹
KVCS_57010	ISCII Gujarathi (x-iscii-gu)	Y ¹
KVCS_57011	ISCII Panjabi (x-iscii-pa)	Y ¹
KVCS_GB18030b2	Reserved for internal use	n/a
KVCS_GB18030	GB18030 (Chinese 4-byte character set)	Y
KVCS_8859_11	ISO 8859-11 (Thai)	Y
KVCS_8859_16	ISO 8859-16 (Latin-10 South-Eastern Europe)	Y
KVCS_ARABICMAC	Arabic Mac (x-mac-arabic)	Y

Code Character Sets, continued

Coded Character Set	Description	Can be set as target charset?
KVCS_KOI8U	Cyrillic (KOI8U Ukrainian)	Y
KVCS_HZGB2312	The 7-bit representation of GB 2312 / RFC 1842	n/a

1

The character set cannot be forced as output in Export SDK and Viewing SDK because the character set is not supported by the major browsers.

Appendix C: File Format Detection

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

• Introduction	297
• Extract Format Information	297
• Determine Format Support	297
• Translate Format Information	298
• Determine a Document Reader	299
• Category Values in the Initialization File and Registry	299

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 format reader for further processing.

For a list of supported formats, see [Supported Formats, on page 253](#).

Extract Format Information

To extract format information from a document, use the [VAPIMWP_INIT_GETDOCFORMAT](#) parameter of the [VAPIM_INIT](#) message. This parameter gets format information (such as the file class, format, and version), and populates the [ADDOCINFO](#) structure. This structure, which is defined in the header file [adinfo.h](#), specifies the formats that KeyView can detect. If required, this format information can then be reported to the developer's application.

For information on how to translate the extracted format information, see [Translate Format Information, on the next page](#).

Determine Format Support

After the file format is extracted, the detection module then uses an initialization file ([kvsdk.ini](#)) or the Windows registry to determine whether the format is supported by KeyView, and the appropriate structured access layer and reader to load. See [View Initialization Information, on page 23](#) for more information.

The initialization file and Windows registry contain the following information:

- Coded format information. To translate this information, see [Translate Format Information, on the next page](#).
- The reader associated with each format. See [Determine a Document Reader, on page 299](#).
- Initialization information. See [View Initialization Information, on page 23](#).

Below are some entries from the initialization file:

```
153=afsr.dll
207=afsr.dll
210=htmsr.dll
251=htmsr.dll
296=htmsr.dll
282=foliosr.dll
```

NOTE: The information in the initialization file and Windows registry applies to all formats except graphics. Detection of graphics formats is handled by an internal module named KeyView Picture Interchange Format (KPIF).

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 include only 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 the initialization file and the Windows registry. When extracting format information by using the VAPIMWP_INIT_GETDOCFORMAT parameter, the same format information is represented as 294.2.0.9.

NOTE: The format values returned by VAPIMWP_INIT_GETDOCFORMAT differ from those in the initialization file and Windows registry 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 `ADDONINFO` structure provides a unique ID for each major format. For example, `VAPIMWP_INIT_GETDOCFORMAT` returns `351.1.0` for a Microsoft Word 2003 XML format. The major format 351 is unique to this format.

Unlike `ADDONINFO`, the initialization file and the Windows registry use the major version number to distinguish between formats. For example, in the initialization file, a Microsoft Word 2003 XML format is defined as `285.1.0.100.0`. The major format 285 and file class 1 are the same values for generic XML. The major version 100 distinguishes the format as Microsoft Word 2003 XML.

The major version is used in the `kvsdk.ini` file or the Windows registry 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 entries in the initialization file or the Windows registry list each format's coded value, and the reader used to parse that format. For example, the entry below specifies that a Lotus 1-2-3 Spreadsheet file version 9.0 is parsed by the Lotus 1-2-3 reader, `l123sr.dll`:

`81.2.0.9.0=l123sr.dll`

[List of Files Required for Redistribution, on page 356](#) lists the document readers provided with KeyView.

Category Values in the Initialization File and Registry

This section lists the possible category values for format information in the initialization file and the Windows registry. The corresponding values for the format information extracted by using the `VAPIMWP_INIT_GETDOCFORMAT` parameter are listed in `adinfo.h`.

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

Major Formats

Number	Format	File Class
1	AES Multiplus Comm Format	Word processor
2	ASCII File word processor/MS DOS Batch File format	Word processor
3	Applix Asterix	Word processor
4	Microsoft Windows Bitmap image (BMP)	Raster image
5	Convergent Tech DEF Comm. format	Word processor
6	Corel Draw (CDR)	Vector graphic
7	Keyword COM.FILE (KSIF)	
8	Computer Graphics Metafile (CGM)	Vector graphic
9	Word Connection	Word processor
10	COMET TOP Word	Word processor
11	DG CEOwrite	Word processor
12	Honey Bull DSA101	Word processor
13	IBM DCA-RFT	Word processor
14	DDIF	Word processor
15	Dummy File (Internal)	
16	DG Common Data Stream (CDS)	Word processor
17	Dummy Print File (Internal)	
18	Windows Micrografx Draw (DRW)	Vector graphic
19	Data Point VISTAWORD	Word processor
20	DECdx	Word processor
21	Enable	Word processor
22	Encapsulated PostScript (EPS)	Raster image
23	DOS/Windows Executable (EXE, DLL)	Executable
24	CCITT Group 3 1-Dimensional (G31D)	Raster image
25	Graphics Interchange format (GIF)	Raster image
26	Hewlett Packard	Word processor
27	IBM 1403 Line Printer	Word processor

Major Formats, continued

Number	Format	File Class
28	IBM DCF Script	Word processor
29	IBM DCA-FFT	Word processor
30	Interleaf	Word processor
31	GEM Bit Image	Raster image
32	IBM Display Write 4	Word processor
33	Raster Graphics	Raster image
34	Keywords PICL	
35	Lotus AMI Pro	Word processor
36	MORE Database Outliner (Mac)	Outline/planning
37	Lyrinx	Word processor
38	MASS-11	Word processor
39	MacPaint	Raster image
40	Microsoft Word Mac	Word processor
41	Informix SmartWare II Communication File	Communications
42	Microsoft Word for Windows	Word processor
43	MultiMate 4.0	Word processor
44	Multiplan Spreadsheet	Spreadsheet
45	Microsoft Rich Text Format (RTF)	Word processor
46	Microsoft Word 5.0 (PC)	Word processor
47	NBI Async Archive Format	Word processor
48	Navy DIF	Word processor
49	NBI Net Archive Format	Word processor
50	NIOS TOP	Word processor
51	FileMaker (Mac)	Database
52	ODA/ODIF	Word processor
53	OLIDIF	Word processor
54	Keyword OSM	

Major Formats, continued

Number	Format	File Class
55	Office Writer	Word processor
56	PC Paint Brush Graphics (PCX)	Raster image
57	CPT Communication Format	Word processor
58	Lotus PIC	Vector graphic
59	Macintosh Quick Draw Picture Format (PICT)	Raster image
60	Philips Script	Word processor
61	PostScript File	Vector graphic
62	PRIMEWORD	Word processor
63	Quadratron Q-One (V1.93J)	Word processor
64	Quadratron Q-One (V2.0)	Word processor
65	SAMNA Word IV	Word processor
66	Lotus AMI Pro Draw (SDW)	Raster image
67	SYLK Spreadsheet	Spreadsheet
68	Informix SmartWare II	Word processor
69	Symphony Spreadsheet	Spreadsheet
70	Truevision Targa	Raster image
71	Tagged Image File (TIFF)	Raster image
72	Targon Word (V 2.0)	Word processor
73	Uniplex Ucalc Spreadsheet	Spreadsheet
74	Uniplex (V6.01)	Word processor
75	Microsoft Word (UNIX)	Word processor
76	WANG PC	Word processor
77	WordERA (V 1.0)	Word processor
78	WANG WPS Comm. format	Word processor
79	WordPerfect Mac	Word processor
80	WordPerfect 5.2	Word processor
81	Lotus 1-2-3 Spreadsheet	Spreadsheet

Major Formats, continued

Number	Format	File Class
82	WordMARC word processor	Word processor
83	Microsoft Windows Metafile (WMF) Graphics	Raster image
84	Informix SmartWare II Database	Database
85	WordPerfect Graphics V1.0 (WPG)	Raster image
86	WordPerfect	Word processor
87	WordStar	Word processor
88	Wang WITA	Word processor
89	Xerox 860 Comm. format	Word processor
90	Microsoft Excel Spreadsheet	Spreadsheet
91	Xerox Writer word processor	Word processor
92	DIF Spreadsheet	Spreadsheet
93	ENABLE Spreadsheet	Spreadsheet
94	Supercalc Spreadsheet	Spreadsheet
95	Ultracalc Spreadsheet	Spreadsheet
96	Informix SmartWare Spreadsheet	Spreadsheet
97	Serialized Object Format (SOF) Encapsulation format	Encapsulation
98	Microsoft PowerPoint (PC)	Presentation
99	Microsoft PowerPoint (Mac)	Presentation
100	Aldus PageMaker (Mac)	Desktop Publishing
101	Aldus PageMaker (DOS)	Desktop Publishing
103	Microsoft Works (Mac)	Word processor
104	Microsoft Works Database (Mac)	Database
105	Microsoft Works Spreadsheet (Mac)	Spreadsheet
106	Microsoft Works Communication (Mac)	Communication
107	Microsoft Works (PC)	Word processor
108	Microsoft Works Database (PC)	Database
109	Microsoft Works Spreadsheet (PC)	Spreadsheet

Major Formats, continued

Number	Format	File Class
111	PC Library Module	Library module
112	MacWrite	Word processor
113	MacWrite II	Word processor
114	Aldus Freehand Mac	Vector graphic
115	Disk Doubler Compression format	Encapsulation
116	HP Graphics Language (HP-GL)	Vector graphic
117	Adobe Maker Interchange Format (MIF)	Desktop Publishing
118	JPEG File Interchange Format (JFIF)	Raster image
119	Reflex Database	Database
120	Framework II	Mixed format
121	Paradox (PC) Database	Database
123	Microsoft Windows Write	Word processor
124	Quattro Pro Spreadsheet (DOS)	Spreadsheet
126	Persuasion Presentation	Presentation
127	Corel Presentation	Presentation
128	Microsoft Windows Icon Format (ICO) Graphics	Raster image
129	Microsoft Project	Time scheduling
131	Harvard Graphics	Desktop publishing
132	Zip Archive Format	Encapsulation
133	Microsoft Windows Cursor (CUR) Graphics	Raster image
134	Quark Express (Mac)	Desktop publishing
135	ARC/PAK Archive format	Encapsulation
136	Adobe FrameMaker	Desktop publishing
137	Microsoft Publisher	Desktop publishing
138	Plan Perfect	Time scheduling
139	WordPerfect General File Format	Miscellaneous
140	Lotus Freelance	Presentation

Major Formats, continued

Number	Format	File Class
141	Microsoft Wave Sound File	Sound
142	MIDI Sound File	Sound
143	AutoCAD DXF Graphics	Vector graphic
144	dBase Database	Database
145	OS/2 PM Metafile Graphics	Vector graphic
146	Lasergraphics Language	Vector graphic
147	AutoShade Rendering File Format	Vector graphic
148	Graphics Environment Manager (GEM VDI)	Vector graphic
149	Microsoft Windows Help File	Miscellaneous
150	Volkswriter	Word processor
151	Ability Office (SS, DB, GR, WP, COM)	
152	XyWrite/Nota Bene	Word processor
153	Comma Separated Values (CSV)	Spreadsheet
154	Writing Assistant word processor	Word processor
155	WordStar 2000	Word processor
156	WordStar 6.0	Word processor
157	HP Printer Control Language (PCL)	Vector graphic
158	(UNIX/VAX/SUN) Executable	Executable
159	(UNIX/VAX/SUN) Object Module	Object module
160	(UNIX/VAX/SUN) Link Library	Library module
161	NeXT SUN Audio Data	Sound
162	NeWS font file (SUN)	Font
163	cpio Archive Format (UNIX/VAX/SUN)	Encapsulation
164	PEX Binary Archive (SUN)	Encapsulation
165	SUN vfont definition	Font
166	Curses Screen Image (UNIX/VAX/SUN)	Raster image
167	UU Encoded Encryption File	Encapsulation

Major Formats, continued

Number	Format	File Class
168	WriteNow	Word processor
169	PC Object Module	Object module
170	Microsoft Windows Group File	Miscellaneous
171	PC True Type Font	Font
172	Program Information File	Miscellaneous
173	PC COM executable file	Executable
174	Adobe FrameMaker Markup Language	Desktop publishing
175	Stuff It Archive (Mac)	Encapsulation
176	PeachCalc Spreadsheet	Spreadsheet
177	Wang Office GDL Header Encapsulation	Encapsulation
178	WordPerfect 6.0	Word processor
179	Q & A for DOS	Word processor
180	Q & A for Windows	Word processor
181	DEC WPS PLUS	Word processor
182	DCX Fax format	Fax
183	Microsoft Windows OLE 2 Encapsulation	Encapsulation
184	Quattro Pro for Windows	Spreadsheet
185	Keyword Viewer Markup Format	
186	EBCDIC Text	Word processor
187	DCS	Word processor
188	Microsoft Excel Spreadsheet 95, 2000	Spreadsheet
189	Microsoft Word for Windows 95	Word processor
190	UNIX SHAR Encapsulation	Encapsulation
191	Lotus Notes Bitmap	Raster image
192	UNIX Compress Encapsulation	Encapsulation
193	Lotus Notes CDF	Word processor
194	UNIX TAR Encapsulation	Encapsulation

Major Formats, continued

Number	Format	File Class
195	WordPerfect Graphics V2.0 (WPG2)	Raster image Vector graphic
196	ODA/ODIF (FOD 26)	Word processor
197	ALIS	Word processor
198	GZ Compress Encapsulation	Encapsulation
199	Envoy (EVY)	Word processor
200	Adobe Portable Document Format (PDF)	Word processor
201	KW ODA Internal Raw Bitmap (RBM)	Raster image
202	KW ODA G4 (G4)	Raster image
203	KW ODA G31D (G31)	Raster image
204	KW ODA Internal G32D (G32)	Raster image
205	Microsoft Word for Mac V 4.x/5.x	Word processor
206	BinHex 4.0 encoded file	Encapsulation
207	SMTP document	Encapsulation
208	MIME format - Microsoft Outlook Express (EML)/Mailbox (MBX)	Encapsulation
209	SGML document	Word processor
210	HTML document XHTML ¹	Word processor
211	ACT Format	Word processor
212	Microsoft PowerPoint 95	Presentation
213	Portable Network Graphics (PNG)	Raster image
214	Video for Windows	Movie
215	Windows Animated Cursor	Raster image
216	Windows C++ Object Storage	Mixed format
217	Windows Palette	Raster image
218	RIFF Device Independent Bitmap	Raster image
219	RIFF MIDI	Sound

Major Formats, continued

Number	Format	File Class
220	RIFF Multimedia Movie	Movie
221	MPEG Movie	Movie
222	QuickTime Movie	Movie
223	Audio Interchange File Format (AIFF) Sound	Sound
224	Amiga MOD Sound	Sound
225	Amiga IFF (8SVX) Sound	Sound
226	Creative Voice (VOC) Sound	Sound
227	Microsoft Works (Windows)	Word processor
228	Microsoft Works Spreadsheet (Windows)	Spreadsheet
229	AutoDesk Animator FLIC Animation	Animation
230	AutoDesk Animator Pro FLIC Animation	Animation
231	Microsoft Works Database (Windows)	Database
232	Microsoft Works Communication (Windows)	Communications
233	Compactor / Compact Pro Archive	Encapsulation
234	VRML	Vector graphic
235	QuickDraw 3D Metafile (3DMF)	Vector graphic
236	PGP Secret Keyring	Encapsulation
237	PGP Public Keyring	Encapsulation
238	PGP Encrypted Data	Encapsulation
239	PGP Signed Data	Encapsulation
240	PGP Signed and Encrypted Data	Encapsulation
241	PGP Signature Certificate	Encapsulation
242	ASCII-armored PGP Public Keyring	Encapsulation
243	ASCII-armored PGP encoded	Encapsulation
244	ASCII-armored PGP signed	Encapsulation
245	OLE DIB object	Raster image
246	PGP Compressed Data	Encapsulation

Major Formats, continued

Number	Format	File Class
247	SGL Image	Raster image
248	Lotus Screen Cam	Animation
249	MPEG Audio	Sound
250	FTP Session Data	Communications
251	Netscape Bookmark file	Word processor
252	Corel Draw CMX	Vector image
253	AutoCAD Drawing (DWG)	Vector graphic
254	AutoDesk WHIP	Vector graphic
255	Macromedia Director	Animation
256	Real Audio	Sound
257	MS DOS Device Driver	Executable
258	Micrografx Designer	Vector graphic
259	Simple Vector format (SVF)	Vector graphic
260	WordPerfect Office document (WPD)	
261	Applix Words	Word processor
262	Applix Graphics	Presentation
263	Microsoft Access	Database
264	Usenet format	Word processor
265	MacBinary	Encapsulation
266	Apple Single	Encapsulation
267	Apple Double	Encapsulation
268	Lotus Word Pro	Word processor
269	Microsoft Word 97, 2000	Word processor
270	Enhanced Window Metafile	Vector graphic
271	Microsoft Office Drawing	Vector graphic
272	Microsoft PowerPoint 97, 2000	Presentation
273	Extended or Custom XML	Word processor

Major Formats, continued

Number	Format	File Class
274	Device Independent file (DVI)	Vector graphic
275	Unicode	Word processor
276	Framework	Mixed
277	KPIF Chart Stream	
278	Applix Spreadsheet	Spreadsheet
279	Microsoft Device Independent Bitmap	Raster image
280	KeyView GPF Filter	
281	Microsoft Project 98, 2000, 2002	Time scheduling
282	Folio Flat file	Word processor
283	HWP (Arae-Ah Hangul)	Word processor
284	JustSystems Ichitaro	Word processor
285	Generic XML format Microsoft Office 2003 XML format ²	Word processor
286	Fujitsu Oasys	Word processor
287	Portable Bitmap Utilities (PBM)	Raster image
288	Portable Greymap Utilities (PGM)	Raster image
289	Portable Pixmap Utilities (PPM)	Raster image
290	X Bitmap (XBM)	Raster image
291	X Pixmap (XPM)	Raster image
292	X Image	Raster image
293	PCD Image	Raster image
294	Microsoft Visio	Presentation
295	Microsoft Outlook (MSG)	Encapsulation
296	XHTML document	Word processor
297	Microsoft Outlook Personal Folders file (PST)	Encapsulation
298	WinRAR Compressed Archive format (RAR)	Encapsulation
299	Lotus Notes Database (NSF) Legato Extender ONM	Encapsulation

Major Formats, continued

Number	Format	File Class
300	Macromedia Flash	Word processor
301	Microsoft Word 2007 (XML format)	Word processor
302	Microsoft Excel 2007 (XML format)	Spreadsheet
303	Microsoft PowerPoint 2007 (XML format)	Presentation
304	Open PGP (new format packets only)	Encapsulation
305	Intergraph version 7 DGN	Vector graphic
306	Microstation version 8 DGN	Vector graphic
307	Microsoft Word 2007 Macro	Word processor
308	Microsoft Excel 2007 Macro	Spreadsheet
309	Microsoft PowerPoint Macro	Presentation
310	Microsoft Compression folder (LZH)	Encapsulation
311	Office 2007 Document	Miscellaneous
312	XML Paper Specification	Word processor
313	Lotus Domino Extensible Language	Encapsulation
314	OASIS Open Document (ODT)	Word processor
315	OASIS Open Document (ODS)	Spreadsheet
316	OASIS Open Document (ODP)	Presentation
317	Legato EMailXtender Native Message	Word Processor
319	Transfer Neutral Encapsulation Format (TNEF)	Encapsulation
320	CADAM Drawing	Vector graphic
321	CADAM Drawing Overlay	Vector graphic
322	NURSTOR Drawing	Vector graphic
323	HP Graphics Language (Plotter)	Vector graphic
324	Advanced Systems Format	Miscellaneous
325	Windows Media Audio Format	Sound
326	Windows Media Video Format	Movie
327	Legato EMailXtender Archive	Encapsulation

Major Formats, continued

Number	Format	File Class
328	7-Zip	Encapsulation
329	Microsoft Office 2007 Excel Binary Format	Spreadsheet
330	Microsoft Cabinet File	Encapsulation
331	CATIA formats	Vector graphic
332	Yahoo! Instant Messenger	Word processor
333	Founder Chinese E-paper Basic	Word processor
334	Corel Quattro Pro X4	Spreadsheet
335	MIME HTML	Word processor
336	Microsoft Document Imaging Format	Raster image
337	Microsoft Office Groove File Format	Word processor
338	Apple iWorks Pages	Word processor
339	Apple iWorks Numbers	Spreadsheet
340	Apple iWorks Keynote	Presentation
341	Microsoft Backup File	Encapsulation
342	Microsoft Access 2007	Database
343	Microsoft Entourage Database	Encapsulation
344	Mac Disk Copy Disk Image File	Encapsularion
345	Appleworks File	Word processor
346	Omni Outliner (OO3) File	Word processor
347	Omni Outliner (OPML) File	Word processor
348	Omni Graffle XML File	Vector graphic
349	Apple Photoshop Document	Raster image
350	Apple Binary Property List	Miscellaneous
351	Apple iChat Format	Word processor
352	Omni Outliner (OOUTLINE) File	Word processor
353	Bzip 2 Compressed File	Encapsulation
354	ISO-9660 CD Disc Image Format	Encapsulation

Major Formats, continued

Number	Format	File Class
355	Xerox DocuWorks	Word processor
356	RealMedia Streaming Media	Movie
357	AC3 Audio File Format	Sound
358	Nero Encrypted File	Encapsulation
359	SolidWorks	Vector graphic
362	UniGraphics NX	Vector graphic
366	Extensible Forms Description Language	Presentation
367	Apple XML Property List	Miscellaneous
368	OneNote Note Format	Presentation
370	Digital Imaging and Communications in Medicine (DICOM)	Raster image
371	Expert Witness Compression Format	Encapsulation
372	Shell Scrap Object File	Encapsulation
373	Microsoft Project 2007	Time scheduling
374	Microsoft Publisher 98–	Desktop publishing
375	Skype Log File	Word processor
376	Lotus Notes Bitmap Format (DXL embedded images)	Raster image
377	Health level7 message	Word processor
378	Microsoft Outlook Offline Storage File	Encapsulation
379	Open Publication Structure eBook	Word processor
380	Microsoft Outlook Express DBX	Encapsulation
381	BlackBerry Activation File	Word processor
382	Disk Image	Encapsulation
383	Milestone	Raster Image
384	RealLegal E-Transcript File	Word processor
385	PostScript Type 1 Font	Font
386	Ghost Disk Image File	Encapsulation
387	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	Raster Image

Major Formats, continued

Number	Format	File Class
388	Unicode HTML	Word processor
389	Microsoft Compiled HTML Help	Encapsulation
390	Documentum EMCMP	Encapsulation
393	JBIG2 File	Raster image
395	AD1 Evidence file	Encapsulation
397	Group Wise File Surf email	Encapsulation
402	ARJ	Encapsulation
409	Microsoft Outlook for Macintosh	Encapsulation
412	Microsoft Outlook vCard Contact	Word processor
414	Microsoft Outlook iCalendar	Encapsulation
418	Apple iWork 2013 Pages	Word processor
419	Apple iWork 2013 Numbers	Spreadsheet
420	Apple iWork 2013 Keynote	Presentation
421	XZ	Encapsulation
427	B1	Encapsulation
428	MP4	Movie
429	Rar5	Encapsulation
430	PTC Creo	Vector graphic
431	Keyhole Markup Language	
432	Zipped Keyhole Markup Language	
433	Wireless Markup Language	
435	Star Office Writer Text	
436	Star Office Calc Spreadsheet	
437	Star Office Impress Presentation	
438	Star Office Math	

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

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

Excel: 101

Visio: 110

File Classes

Attribute Number	File Class
0	No file class
01	Word processor
02	Spreadsheet
03	Database
04	Raster image
05	Vector graphic
06	Presentation
07	Executable
08	Encapsulation
09	Sound
10	Desktop publishing
11	Outline/planning
12	Miscellaneous
13	Mixed format
14	Font
15	Time scheduling
16	Communications
17	Object module
18	Library module
19	Fax
20	Movie
21	Animation

Minor Formats

Attribute Number	Minor Format
00	Minor format not defined
01	Standard
02	Book
03	Chart
04	Macro
05	Text
06	Binary
07	PC
08	Windows
09	DOS
10	Macintosh
11	RGB
12	TIFF
13	IFF
14	Experimental
15	Format Information
16	RLE
17	Symbol
18	Old
19	Footnote
20	Style
21	Palette
22	Configuration
23	Activity
24	Resource
25	Calculation
26	Glossary

Minor Formats, continued

Attribute Number	Minor Format
27	Spelling
28	Thesaurus
29	Hyphenation
30	Miscellaneous
31	UNIX
32	VAX
33	Driver
34	Archive

Appendix D: File Formats and Extensions

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

- [File Format and Extension Table](#)318

File Format and Extension Table

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

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

KeyView file formats and extensions

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
		packet format)	
Intergraph_V7_DGN_Fmt	364	Intergraph Standard File Format (ISFF) V7 DGN (non-OLE)	DGN ¹
MicroStation_V8_DGN_Fmt	365	MicroStation V8 DGN (OLE)	DGN ¹
MS_Word_Macro_2007_Fmt	366	Microsoft Word Macro 2007 XML	DOCM, DOTM
MS_Excel_Macro_2007_Fmt	367	Microsoft Excel Macro 2007 XML	XLSM, XLTM, XLAM
MS_PPT_Macro_2007_Fmt	368	Microsoft PPT Macro 2007 XML	PPTM, POTM, PPSM, PPAM
LZH_Fmt	369	LHA Archive	LZH, LHA
Office_2007_Fmt	370	Office 2007 document	XLSB
MS_XPS_Fmt	371	Microsoft XML Paper Specification (XPS)	XPS
Lotus_Domino_DXL_Fmt	372	IBM Lotus representation of Domino design elements in XML format	DXL
ODF_Text_Fmt	373	ODF Text	ODT ¹ , SXW ¹ , STW
ODF_Spreadsheet_Fmt	374	ODF Spreadsheet	ODS ¹ , SXC ¹ , STC
ODF_Presentation_Fmt	375	ODF Presentation	SXD ¹ , SXI ¹ , ODG ¹ , , ODP ¹
Legato_Extender_ONM_Fmt	376	Legato Extender Native Message ONM	ONM
bin_Unknown_Fmt	377	n/a	
TNEF_Fmt	378	Transport Neutral Encapsulation Format (TNEF)	various
CADAM_Drawing_Fmt	379	CADAM Drawing	CDD
CADAM_Drawing_Overlay_Fmt	380	CADAM Drawing Overlay	CDO
NURSTOR_	381	NURSTOR Drawing	NUR

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

Format Name	Format Number	Format Description	Associated File Extension
MS_Publisher_98_Fmt	428	Microsoft Publisher 98/2000/2002/2003/2007/	PUB ¹
Skype_Fmt	429	Skype Log File	DBB
HL7_Fmt	430	Health level7 message	HL7
MS_OutlookOST_Fmt	431	Microsoft Outlook OST	OST
Epub_Fmt	432	Electronic Publication	EPUB
MS_OEDBX_Fmt	433	Microsoft Outlook Express DBX	DBX
BB_Activ_Fmt	434	BlackBerry Activation File	DAT ¹
DiskImage_Fmt	435	Disk Image	
Milestone_Fmt	436	Milestone Document	MLS, ML3, ML4, ML5, ML6, ML7, ML8, ML9
E_Transcript_Fmt	437	RealLegal E-Transcript File	PTX
PostScript_Font_Fmt	438	PostScript Type 1 Font	PFB
Ghost_DiskImage_Fmt	439	Ghost Disk Image File	GHO, GHS
JPEG_2000_JP2_File_Fmt	440	JPEG-2000 JP2 File Format Syntax (ISO/IEC 15444-1)	JP2, JPF, J2K, JPWL, JPX, PGX
Unicode_HTML_Fmt	441	Unicode HTML	HTM ¹ , HTML ¹
CHM_Fmt	442	Microsoft Compiled HTML Help	CHM
EMCMF_Fmt	443	Documentum EMCMF format	EMCMF
MS_Access_2007_Tmpl_Fmt	444	Microsoft Access 2007 Template	ACCDT
Jungum_Fmt	445	Samsung Electronics Jungum Global document	GUL
JBIG2_Fmt	446	JBIG2 File Format	JB2, JBIG2
EFax_Fmt	447	eFax file	EFX
AD1_Fmt	448	AD1 Evidence file	AD1
SketchUp_Fmt	449	Google SketchUp	SKP

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

KeyView file formats and extensions, continued

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

1

This file extension can return more than one format number.

2

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

3

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

Appendix E: 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	343
• Customize XML Templates	343
• Template Elements and Attributes	345
• Date and Time Formats	349

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 345](#).
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>, <ift dne>, <ift dlt>, <ift dle>, <ift dgt>, <ift dge>	<p>Respectively, if time/date ==, !=, <, >, <=, >, >=.</p> <p>Time/date comparison converts dates to text in local time using the Notes default, TZFMT_NEVER, because Notes also sometimes converts fields to text internally. For example:</p> <pre>text="06/30/2005 02:52:04 PM"</pre>
<iftzeq>, <iftzne>	<p>Respectively, if the time zone equals or does not equal the comparison text, for example CDT, EST, and so on.</p>

Conditional elements, continued

Element	Description
<ifini>	If the value of the INI option specified in name equals the text value.
<else>	If the condition from the last <if> or <switch> was false, process XML.
<switch>	<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> <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.

Control Elements, continued

Element	Description
	<ul style="list-style-type: none"> 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>, on the previous page.
<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> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.
<body>	<p>Output the message body in rich text (MHTML). As with <rich>, above, images are output in the next part or parts of the MHTML.</p>
<form>	<p>Output the message form (usually \$Body field) in rich text (MHTML).</p> <p>Attributes</p> <ul style="list-style-type: none"> name. (Required if there is no parent) The name of the item to output.

Data elements, continued

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

Data elements, continued

Element	Description
	<ul style="list-style-type: none"> ◦ 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>	Output a time zone as UTC, for example (UTC-06:00).
<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 347 and <body>, on page 347.</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> <ul style="list-style-type: none"> • link. (Required if there is no file) The image link, such as a form or title name. For example: • link="StdNotesLtr0" • file. (Required if there is no link) The name of the image file. The file must exist in the ../../templates/images directory. For example: • file="boxcheck.gif"

Date and Time Formats

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

Lotus Notes Date and Time Formats

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

Lotus Notes date and time formats

Format	Integer Value	Description
TDFMT_FULL	0	(The Notes default) Year, month, and day
TDFMT_CPARTIAL	1	Month and day, year if not this year
TDFMT_PARTIAL	2	Month and day
TDFMT_DPARTIAL	3	Year and month
TDFMT_FULL4	4	Four-digit year, month, and day
TDFMT_CPARTIAL4	5	Month and day, four-digit year if not this year
TDFMT_DPARTIAL4	6	Four-digit year and month
TTFMT_FULL	0	(Notes default) Hour, minute, and second
TTFMT_PARTIAL	1	Hour and minute
TTFMT_HOUR	2	Hour
TZGMT_NEVER	0	(Notes default) All time zones are converted to the current time zone
TZGMT_SOMETIMES	1	Show only when outside the current time zone
TZGMT_ALWAYS	2	Show for all time zones
TSFMT_DATE	0	Date
TSFMT_TIME	1	Time
TSFMT_DATETIME	2	(The Notes default) Date and time
TSFMT_CDATETIME	4	Date and time, or time today or time yesterday

KeyView Date and Time Formats

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

Month Month = full month name
 Mon = abbreviated month name
 m = month (number)
 mm = two-digit month (leading 0)

Weekday	weekday = full weekday name wday = abbreviated weekday name
Year	yy = two-digit year yyyy = four-digit year
Day	d = day (number) dd = two-digit day (leading 0)
Time	h = 12-hour H = 24-hour m = minutes s = seconds P = AM/PM p = am/pm
Separators	_ = space c = comma s = slash a = dash o = dot

KeyView date and time formats

Format	Output	Integer Value
12-Hour and 24-Hour Time Formats		
KVDTF_P	P	1
KVDTF_P_hmm	P h:mm	2
KVDTF_hmm_P	h:mm P	3
KVDTF_P_hhmm	P hh:mm	4
KVDTF_hhmm_P	hh:mm P	5
KVDTF_P_hmmss	P h:mm:ss	6
KVDTF_hmmss_P	h:mm:ss P	7
KVDTF_P_hhmmss	P hh:mm:ss	8
KVDTF_hhmmss_P	hh:mm:ss P	9
KVDTF_Hmm	H:mm	10
KVDTF_HHmm	HH:mm	11

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_mmss	mm:ss	12
KVDTF_Hmmss	H:mm:ss	13
KVDTF_HH:mmss	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_mmsddsyysy	mm/dd/yyyy	18
KVDTF_ddsmm	dd/mm	19
KVDTF_ddsmmsyy	dd/mm/yy	20
KVDTF_ddsmmsyy_Hmm	dd/mm/yy H:mm	21
KVDTF_ddsmm_P_hmm	dd/mm P h:mm	22
KVDTF_ddsmm_hmm_P	dd/mm h:mm P	23
KVDTF_ddsmm_P_hhmm	dd/mm P hh:mm	24
KVDTF_ddsmm_hhmm_P	dd/mm hh:mm P	25
KVDTF_ddsmmsyy_P_hmm	dd/mm/yy P h:mm	26
KVDTF_ddsmmsyy_hmm_P	dd/mm/yy h:mm P	27
KVDTF_ddsmmsyy_P_hmmss	dd/mm/yy P h:mm:ss	28
KVDTF_ddsmmsyy_hmmss_P	dd/mm/yy h:mm:ss P	29
KVDTF_ddsmmsyy_P_hhmmss	dd/mm/yy P hh:mm:ss	30
KVDTF_ddsmmsyy_hhmmss_P	dd/mm/yy hh:mm:ss P	31
KVDTF_yysmmsdd_P_hhmmss	yy/mm/dd P hh:mm:ss	32
KVDTF_yysmmsdd_hhmmss_P	yy/mm/dd hh:mm:ss P	33
KVDTF_msdsyy_Hmm	m/d/yy H:mm	34
KVDTF_mmsddsyy_Hmm	mm/dd/yy H:mm	35
KVDTF_msdsyy_P_hmm	m/d/yy P h:mm	36
KVDTF_msdsyy_hmm_P	m/d/yy h:mm P	37

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_mmsddsy_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_mmsddsy_P_hhmmss	mm/dd/yy P hh:mm:ss	41
KVDTF_mmsddsy_hhmmss_P	mm/dd/yy hh:mm:ss P	42
KVDTF_ms_d	m/d	43
KVDTF_yysm	yy/m	44
KVDTF_yysmm	yy/mm	45
KVDTF_yysmsd	yy/m/d	46
KVDTF_yysmmsdd	yy/mm/dd	47
KVDTF_yyyysmmsdd	yyyy/mm/dd	48
Numerical Date Formats with Dashes		
KVDTF_ddammayy	dd-mm-yy	49
KVDTF_mmadd	mm-dd	50
KVDTF_mmayy	mm-yy	51
KVDTF_yyammadd	yy-mm-dd	52
KVDTF_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

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_ddaMonayyyy	dd-Mon-yyyy	62
KVDTF_Mon	Mon	63
KVDTF_Monayy	Mon-yy	64
KVDTF_Monayyyy	Mon-yyyy	65
KVDTF_Monaddayy	Mon-dd-yy	66
KVDTF_yyammadd_P_hhmmss	yy-mm-dd P hh:mm:ss	67
KVDTF_mmadd_P_hhmm	mm-dd P hh:mm	68
KVDTF_Mon_yy	Mon yy	69
KVDTF_Monc_yy	Mon, yy	70
KVDTF_Month	Month	71
KVDTF_Monthayy	Month-yy	72
KVDTF_Month_yy	Month yy	73
KVDTF_Monthc_yy	Month, yy	74
KVDTF_Monthayyyy	Month-yyyy	75
KVDTF_Month_yyyy	Month yyyy	76
KVDTF_Monthc_yyyy	Month, yyyy	77
KVDTF_Mon_dc_yyyy	Mon d, yyyy	78
KVDTF_d_Monc_yyyy	d Mon, yyyy	79
KVDTF_yyyy_Mon_d	yyyy Mon d	80
KVDTF_Month_dc_yyyy	Month d, yyyy	81
KVDTF_d_Monthc_yyyy	d Month, yyyy	82
KVDTF_yyyy_Month_d	yyyy Month d	83
Weekday Date Formats		
KVDTF_wday	wday	84
KVDTF_Weekday	Weekday	85
KVDTF_wdayc_Mon_dc_yyyy	wday, Mon d, yyyy	86
KVDTF_Weekdayc_Month_dc_yyyy	Weekday, Month d, yyyy	87

KeyView date and time formats, continued

Format	Output	Integer Value
KVDTF_Weekdayc_d_Monthc_yyyy	Weekday, d Month, yyyy	88

Appendix F: List of Files Required for Redistribution

This section lists the files required for redistributing applications based on Viewing SDK.

- [Core Files](#) 356
- [Support Files](#) 357
- [Document Readers and Writers](#) 358
- [Miscellaneous Functionality](#) 365
- [Viewing ActiveX Control](#) 366
- [Windows System Files](#) 366

The following files should be installed to the \bin directory of your application's installation directory.

Core Files

The following core files can be redistributed with your application.

File	Description
chartb1s.ux	Character mapping tables.
htmcnv.dll	HTML converter for the document token stream.
kvarcve.dll	Archive format viewing engine.
kvdocve.dll	Word processing format viewing engine.
kvmailve.dll	Mail format viewing engine.
kvmve.dll	Multimedia format viewing engine.
kvpicve.dll	Picture format viewing engine.
kvolefio.dll	Embedded OLE object reader.
kvsdk.ini	Initialization file.
kvutil.dll	Utility.
kvvapi.dll	Viewing API.
kvwkbve.dll	Spreadsheet format viewing engine.
kvxssa.dll	Interface between spreadsheet readers and Viewing API.
kvxtract.dll	File Extraction interface for container file support.
kvxwpsa.dll	Interface between word processing document readers and Viewing API.

File	Description
kwad.dll	Format detection module.
kwcm.dll	Conversion Manager.
kwlm.dll	URL Launch Manager.
kwres.dll	Resources.

Support Files

The following support files can be redistributed with your application.

File	Description
bentofio.dll	Required by l123sr.dll and kpprzrdr.dll.
cbmap.map	Character mappings for Adobe Portable Document Format (PDF).
chmdll.dll	Required by chmsr.dll.
kp3dwrld.dll	Required for 3D charts.
kpifcnvt.dll	Picture conversion routines.
kpifutil.dll	Picture utility routines.
kpjpeg.dll	JPEG file interchange format shared routines.
kppng.dll	Portable Network Graphics (PNG) utilities.
kv.lic	Contains license information for KeyView products. This file is opened and validated when a KeyView API is used.
kvaxcc.dll	Required for viewing HTML using Internet Explorer within View API window
kvgraph.dll	Required for all spreadsheets (chart support).
kvpageve.dll	An alternate viewing engine for Word processing formats.
kvpie.dll	Required for all spreadsheets (chart support).
kvplug.dll	<p>Required for PDF support through the Acrobat plug-in if you use a version of Acrobat Reader earlier than 4.0.</p> <p>You might have to manually install the Acrobat plug-in nppdf32.dll. This is determined by the browser you use. If you use Netscape, nppdf32.dll installs automatically with Acrobat Reader. However, if you use other browsers, such as Internet Explorer, you must manually install nppdf32.dll into a subdirectory of the Viewing Home directory called plugins. You must then set up the registry or initialization file according to the description for PDF in kvsdk.ini or install.reg.</p>
kvradar.dll	Required for all spreadsheet formats (chart support).

File	Description
kvreg.dll	Sheet Registry processing.
kvssvwr.dll	Required for all spreadsheet formats.
kvxmlve.dll	XML format viewing engine.
kwbase64.dll	Required for MIME
wpmap.dll	Extended character mapping for WordPerfect and Corel Presentation.
xmlsh.dll	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.

Archive Formats

File	Description
ad1sr.dll	AD1 Evidence file reader
b11sr.dll	B1 archive reader
bkfsr.dll	Microsoft Backup File reader
bzip2sr.dll	Bzip2 reader
cabsr.dll	Microsoft Cabinet format reader
chmsr.dll	Microsoft Compiled HTML Help reader
dmgsr.dll	Mac Disk Copy Disk Image File reader
dunzip32.dll	used by ZIP reader
emxsr.dll	Legato EMailXtender archive (EMX) reader
encasesr.dll	Expert Witness Compression Format (EnCase) v6 reader
encase2sr.dll	Expert Witness Compression Format (EnCase) v7 reader
isosr.dll	ISO-9660 CD Disc Image Format reader
kvgz.dll	GZIP reader
kvhqx.dll	BinHex reader
kvzee.dll	Unix Compress reader

File	Description
kw2hqx.dll	BinHex writer
kw2tar.dll	Tape Archive writer
kw2uue.dll	UUEncoding writer
kw2zee.dll	Unix Compressed writer
kw2zip.dll	ZIP writer
lzhsr.dll	Microsoft Compression Folder reader.
macbinsr.dll	MacBinary reader
multiarcsr	ARJ reader
rarsr.dll	RAR Archive reader
tarsr.dll	Tape Archive (TAR) reader
unzip.dll	ZIP reader
uudsr.dll	UUEncoding reader
z7zsr.dll	7-Zip reader

Binary Formats

File	Description
exesr.dll	DOS/Windows Executables/DLLs

Computer-Aided Design Formats

File	Description
kpDWGrdr.dll	AutoCAD Drawing reader
kpDXFrdr.dll	AutoCAD Drawing Exchange reader
kpODAndr.*	AutoCAD reader (Windows only)
kpVSDrdr.dll ¹	Microsoft Visio reader
kpVSDXrdr.dll	Microsoft Visio 2013 reader
vsdsr.dll	Microsoft Visio reader

¹kpVSDrdr is only distributed with Windows 32-bit installations. All other platforms use vsdsr.

Database Formats

File	Description
dbfsr.dll	dBase Database reader
mdbsr.dll	Microsoft Access reader
mppsr.dll	Microsoft Project reader

Desktop Publishing Formats

File	Description
mspubsr.dll	Microsoft Publisher reader

Display Formats

File	Description
kppdfldr.dll	Adobe Portable Document File (PDF) graphic-based reader
kppdf2ldr.dll	High-fidelity Adobe Portable Document File (PDF) graphic-based reader

Graphic Formats

File	Description
jp2000sr.dll	JPEG 2000 metadata reader
kpanldr.dll	Windows Animated cursor reader
kpbmprdr.dll	Windows Bitmap reader
kpbmpwrt.dll	Windows Bitmap writer
kpcdrdr.dll	Corel Draw reader
kpcgmrdr.dll	Computer Graphics Metafile (CGM)
kpcxrdr.dll	DCX (fax) reader
kpem2ldr.dll	Enhanced Windows Metafile (EMF) reader
kpemfrdr.dll	Enhanced Windows Metafile (EMF) reader
kpepsrdr.dll	Encapsulated PostScript (EPS) reader

File	Description
kpgifdr.dll	Graphic Interchange Format (GIF) reader
kpicondr.dll	Windows Icon reader
kpjbig2rdr.dll	JBIG2 reader
kpjp2000rdr.dll	JPEG 2000 reader
kpjpgdr.dll	JPEG file interchange format reader
kpjpgwrt.dll	JPEG file interchange format writer
kpmacrdr.dll	MacPaint reader
kpmsondr.dll	Microsoft Office Drawing Objects reader
kpnbmprdr.dll	Lotus Notes Bitmap reader (for embedded images in DXL files)
kppctrdr.dll	Macintosh Quick Draw Picture (PICT) reader
kppcxrdr.dll	PC Paintbrush (PCX) reader
kppicrdr.dll	Pictor PC Paint format (PIC) reader
kppngrdr.dll	Portable Network Graphics (PNG) reader
kppngwrt.dll	Portable Network Graphics (PNG) writer
kpsdwrdr.dll	Lotus Ami Pro Graphics reader
kpsgirdr.dll	SGI RGB reader
kpsunrdr.dll	Sun Raster reader
kptgdr.dll	Truevision Targa reader
kptifdr.dll	Tagged Image File Format reader
kptifwrt.dll	Tagged Image File Format writer
kpwg2rdr.dll	WordPerfect Graphics 2.0 reader
kpwm2rdr.dll	Windows Metafile (WMF) reader
kpwmfrdr.dll	Windows Metafile (WMF) reader
kpwmfwrt.dll	Windows Metafile writer
kpwpgrdr.dll	WordPerfect Graphics 1.0 reader

Mail Formats

File	Description
dbxsr.dll	Microsoft Outlook Express DBX reader
dxlsr.dll	Domino XML Language reader
emlsr.dll	Microsoft Outlook Express (EML) reader
entsr.dll	Microsoft Entourage Database Format reader
gwfssr.dll	GroupWise FileSurf reader
icssr.dll	Microsoft Outlook iCalendar reader
msgsr.dll	Microsoft Outlook (MSG) reader
mbxsr.dll	Mailbox (MBX) and Microsoft Outlook Express (EML) reader ¹
nsfsr.dll	Lotus Notes Database readerThis reader is considered an advanced feature and is sold and licensed separately from the Viewing SDK. See License Information, on page 19., above
olmsr.dll	Microsoft Outlook for Macintosh reader
pffsr.dll	Microsoft Outlook Offline Storage File reader
pstsr.dll	Microsoft Outlook Personal Folders file MAPI-based reader (supported on Windows platform only)This reader is considered an advanced feature and is sold and licensed separately from the Viewing SDK. See License Information, on page 19., above
pstnsr.dll	Microsoft Outlook Personal Folders file native readerThis reader is considered an advanced feature and is sold and licensed separately from the Viewing SDK. See License Information, on page 19., above
tnfsr.dll	Transfer Neutral Encapsulation Format reader
vcfsr.dll	Microsoft Outlook vCard Contact reader

Presentation Formats

File	Description
kpagrdr.dll	Applix Presentations reader
kpiwpgdr.dll	Apple iWork Keynote reader

¹This reader is considered an advanced feature and is sold and licensed separately from the Viewing SDK. See [License Information, on page 19.](#)

File	Description
kpodfrdr.dll	Oasis Open Document Format presentation (ODP) reader
kpONErdr.dll	Microsoft OneNote reader
kpp40rdr.dll	Microsoft PowerPoint 4.0 reader
kpp95rdr.dll	Microsoft PowerPoint 95 reader
kpp97rdr.dll	Microsoft PowerPoint 97, 2000, and 2002 reader
kpppxrdr.dll	Microsoft PowerPoint XML reader 2007
kpprerdr.dll	Lotus Freelance 96/97 reader
kpprzrdr.dll	Lotus Freelance 2.x reader
kpshwrdr.dll	Corel Presentation Graphics reader
kpXFDLrdr.dll	Extensible Forms Description Language reader
swfsr.dll	Macromedia Flash reader
vsdsr.dll	Microsoft Visio reader

Spreadsheet Formats

File	Description
assr.dll	Applix spreadsheet reader
csvsr.dll	Comma Separated Values reader
difsr.dll	Data Interchange Format reader
htmss.dll	Required to save spreadsheets as HTML.
iwsssr.dll	Apple iWork Numbers reader
kpchtrdr.dll	Required for all spreadsheets (chart support)
l123sr.dll	Lotus 123 V96/97 reader
mwssr.dll	Microsoft Works Spreadsheet reader
odfsssr.dll	Oasis Open Document Format spreadsheets (ODS) reader
qpssr.dll	Quattro Pro Spreadsheet reader
qpwsr.dll	Corel Quattro Pro version X4 spreadsheet reader
wkssr.dll	Lotus 123 V2 to 5 reader
xlsbsr.dll	Microsoft Office 2007 Excel Binary Format reader

File	Description
xlssr.dll	Microsoft Excel reader
xlxsxr.dll	Microsoft Excel 2007 XML reader

Word Processor Formats

File	Description
afsr.dll	ASCII reader
awsr.dll	Applix Words V4.x reader
dcasr.dll	Document Content Architecture/Revisable Form Text (DCA/RFT) reader
dw4sr.dll	DisplayWrite 4 reader
epubsr.dll	Open Publication Structure eBook reader
foliosr.dll	Folio Flat File 3.1 reader
hexsr.dll	Hexadecimal reader
hl7sr.dll	Health level7 reader
htmsr.dll	Hypertext Markup Language (HTML) reader
hwposr.dll	Hangul reader
ichatsr.dll	Apple iChat Log reader
iwwpsr.dll	Apple iWork Pages reader
jtdsr.dll	JustSystems Ichitaro reader
lasr.dll	Lotus AMI Pro reader
ltbenn30.dll	Lotus Word Pro support
ltscsn10.dll	Lotus Word Pro support
lwpapin.dll	Lotus Word Pro support
lwppann.dll	Lotus Word Pro support
lwpsr.dll	Lotus Word Pro reader.
mbsr.dll	Microsoft Word Mac reader
mhtsr.dll	MIME HTML reader
mifsr.dll	Adobe Maker Interchange Format (.mif) reader

File	Description
misr.dll	Microsoft Word 2 reader
msw6sr.dll	Microsoft Works 6, 2000 reader
mswsr.dll	Microsoft Works 1, 2, 3, 4 reader
mw6sr.dll	Microsoft Word 95 reader
mw8sr.dll	Microsoft Word 97, 2000, XP reader
mwsr.dll	Microsoft Word for DOS and Microsoft Write reader
mwxsr.dll	Microsoft Word 2007 XML reader
oa2sr.dll	Fujitsu Oasys reader
odfwpsr.dll	Oasis Open Document Format word processing (ODS) reader
oo3sr.dll	Omni Outliner reader
rtfsr.dll	Microsoft Rich Text Format reader
skypesr.*	Skype log file reader
sosr.dll	StarOffice/OpenOffice reader
unihtmsr.dll	Unicode HTML reader
unisr.dll	Unicode reader
wosr.dll	WordPerfect 5.x reader
wp6sr.dll	WordPerfect 6.0 through 10.0 reader
wpmsr.dll	WordPerfect for Macintosh reader
xmlsr.dll	XML reader
xpssr.dll	XML Paper Specification reader
xywsr.dll	XyWrite reader
yimsr.dll	Yahoo! Instant Messenger reader

Miscellaneous Functionality

File	Description
htmcnv.dll	SaveAs HTML (through SaveAs dialog box)
kvcnv.dll	SaveAs

File	Description
kvtlbar.dll	Toolbar with MFC library dynamically loaded (need to redistribute mfc42.dll).
kvtlbst.dll	Toolbar with MFC library statically linked
rtfcnv.dll	SaveAs RTF (through SaveAs dialog box or VAPIM_CONVERT), Copy to Clipboard
rtfss.dll	SaveAs RTF, Copy to Clipboard
txtcnv.dll	SaveAs Text (through SaveAs dialog box) Copy to Clipboard

Viewing ActiveX Control

File	Description
kvocx.ocx	Viewing ActiveX control
kvocx.oca	Viewing ActiveX control

Windows System Files

The following files should be installed in the Windows System directory.

File	Description
mfc42.dll	Microsoft Foundation Class V4.2.
msvbvm60.dll	Microsoft Visual Basic Runtime library V6.0.
msvcpr50.dll	Microsoft Visual C++ Runtime Library V5.0.
msvcpr60.dll	Microsoft Visual C++ Runtime Library V6.0.
msvcrt.dll	Microsoft Visual C Runtime library.
oleaut32.dll	Microsoft OLE Automation Controls.
regsvr32.exe	A Microsoft Windows program used to register in-process COM objects.

Appendix G:

Configuration Options in formats.ini

This appendix lists and explains configuration parameters available in the `formats.ini` file.

- [formats.ini Options, below](#)

formats.ini Options

The following table lists configuration parameters available in the `formats.ini` file.

formats.ini configuration options

Configuration option	Description	For details, see
[DiskCache] section		
DiskCacheSize type = integer default = 64 range =	Specify the amount of memory in KB that KeyView will use for caching. Generally, when you increase the memory, performance improves. To determine a reasonable value, divide the maximum amount of memory you want KeyView to use by the number of threads. For example, if you want KeyView to use 50MB of memory and have 10 threads, set the value to 5MB, or 5120.	
[nsfsr] section		
ExtractAllNotes type = Boolean default = 0	Set to 1 to extract all classes of notes and all subfiles regardless of whether they contain mail headers	
ExtractAllFields type = Boolean default = 0	Set to 1 to extract all fields to a subfile. Applies to non-mail subfiles only.	
TempDir type = file path	(Windows only) Specify a new temp directory.	

formats.ini configuration options, continued

Configuration option	Description	For details, see
default = current temp directory		
ExportDXL type = Boolean default = 0	Export as DXL instead of MHT	
[pdf_flags] section		
remove_invisible_text type = Boolean default = 0	Set to 1 to	

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, below](#)
- [View Password Protected Files, on the next page](#)

Supported Password Protected File Types

The following table lists the password-protected file types that KeyView supports.

Key to support table

Symbol	Description
Y	Format is supported.
N	Format is not supported.
S	Support for viewing subfiles.
V	Support for viewing content.
P	Password required.
C	Password and certificate or User ID file required.

Supported password-protected file types

File Type	Version	Filter	Export	Extract	View	Credentials
PST (Windows)	n/a	N	N	Y	S	P
PST (non-Windows) ¹	n/a	N	N	Y	S	N
ZIP	n/a	N	N	Y	S	P
7-Zip	n/a	N	N	Y	S	P
RAR	n/a	N	N	Y	S	P
SMIME in MSG, EML, MBX	n/a	N	N	Y	N	C
Lotus Notes NSF	n/a	N	N	Y	N	C

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
Adobe PDF	n/a	Y	Y	Y	V	P
Microsoft Office	97-2003 2007 2010	Y	Y	Y	V	P

View Password Protected Files

This section describes how to view password-protected files by using the Viewing API.

To view password-protected files

- Set the password with the `VAPIMWP_INIT_SETPASSWORD` message parameter.
 - For password-protected PST files, this message must be called before the `VAPIMWP_INIT_OPEN_DOCUMENT` message.
 - For password-protected Microsoft Office 2007 and 2010 files, this message must be called before the `VAPIMWP_INIT_OPEN_DOCUMENT` message.
 - For password-protected ZIP files, this message can be called after the `VAPIMWP_INIT_OPEN_DOCUMENT`, but must be called before the protected subfile is extracted or viewed.

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 Viewing SDK Programming Guide (KeyView 11.5)

Add your feedback to the email and click **Send**.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to AutonomyTPFeedback@hpe.com.

We appreciate your feedback!