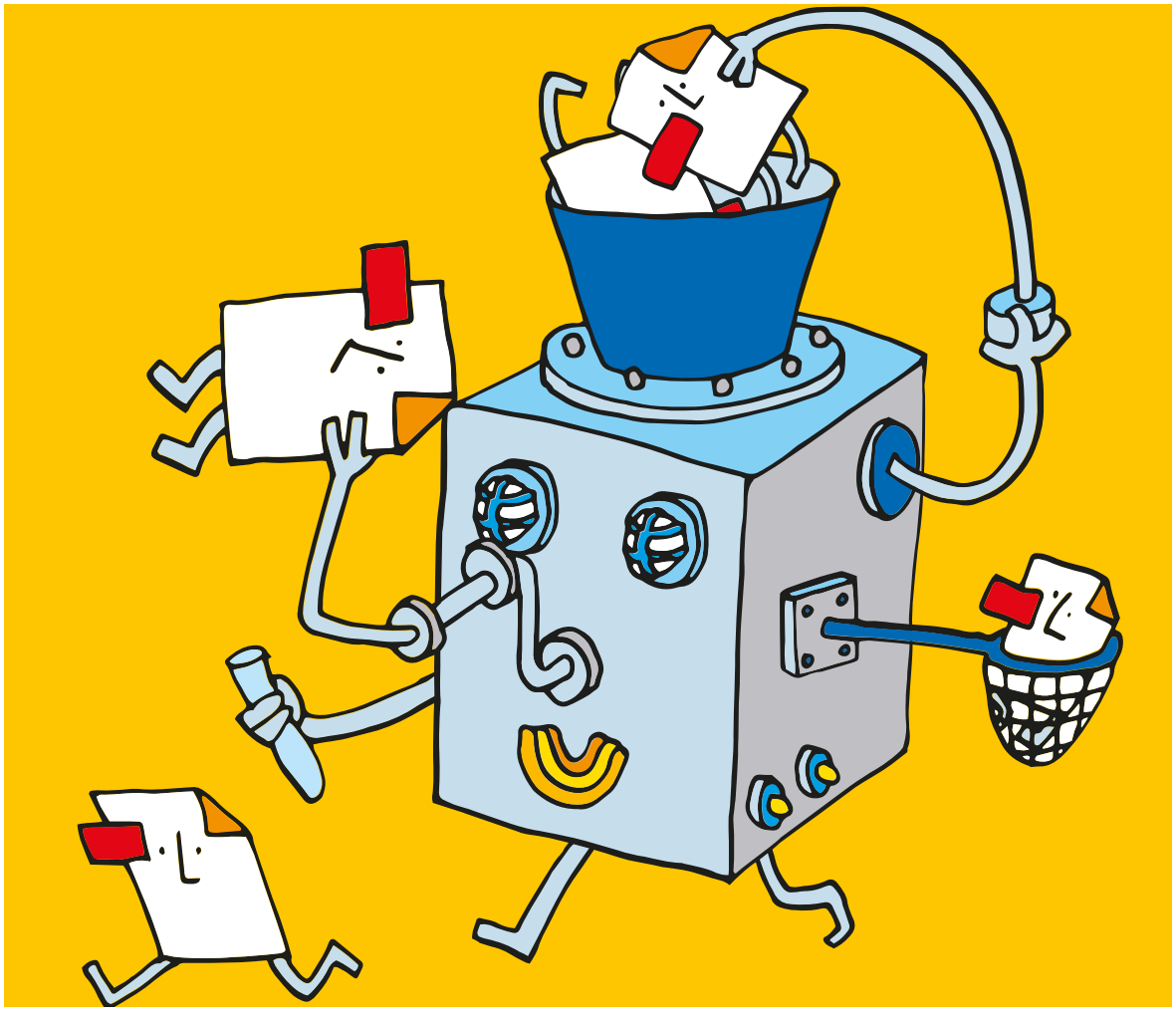
PDFlib GmbH

TET PDF IFilter

Version 5.3

Enterprise PDF Search for Windows



PDFlib GmbH
Franziska-Bilek-Weg 9, 80339 München, Germany
www.pdflib.com
phone +49 • 89 • 452 33 84-0

sales@pdflib.com
support@pdflib.com (please include your license number)

Contents

0 Installing TET PDF IFilter

TET PDF IFilter is delivered as an installer for Windows systems. All TET PDF IFilter packages contain a signed IFilter DLL plus support files, documentation, and samples. Running the installer requires administrator privileges. The installer installs and registers TET PDF IFilter. Additional steps for specific search environments (e.g. Windows Search, SharePoint) as well as custom configuration are discussed in this manual.

32-bit and 64-bit versions. TET PDF IFilter is available for 32-bit and 64-bit platforms. Both versions are available in separate installers, and can be installed on the same system in parallel if required. The 64-bit version is a native 64-bit implementation which works only with 64-bit executables. While the 64-bit installer refuses to install on 32-bit systems, the 32-bit version works on both 32-bit and 64-bit systems.

It is crucial to install the appropriate 32- or 64-bit version which matches the intended IFilter client software. For current server software this will usually be the 64-bit version.

Updating to a newer version of TET PDF IFilter. If an older version of TET PDF IFilter is already installed on the machine, you should uninstall the older version before installing the new version. Installation packages always contain the full product, and never rely on an existing installation.

Applying the TET PDF IFilter license key. Using TET PDF IFilter for production purposes on a server system requires a valid license key. Once you purchased a TET PDF IFilter license you must apply your license key in order to allow processing of arbitrarily large documents. Generally you enter the license key when installing TET PDF IFilter with the installer. However, you can also manually apply the license key in the registry after installation (see »Manual installation«, page 6). 32-bit and 64-bit versions accept the same license keys.

Use the license key 0 (zero) to install the evaluation version on a server system, or to install the free desktop version for non-commercial use.

Restrictions of the evaluation version. The TET PDF IFilter can be used as fully functional evaluation version even without a commercial license. Unlicensed versions support all features, but process only PDF documents with up to 10 pages and 1 MB size. Evaluation versions of TET PDF IFilter must not be used for production purposes. Using TET PDF IFilter for production purposes requires a commercial license.

Free desktop version for non-commercial use. TET PDF IFilter for desktop systems, i.e. Windows 8/10 may freely be used for personal use, i.e. non-commercial purposes. Deploying the desktop version in any situation which can be considered commercial use requires a commercial license, though. TET PDF IFilter for Windows Server always requires a commercial license.

Supported IFilter clients. TET PDF IFilter implements Microsoft's IFilter interface. A variety of indexing products support the IFilter interface. In this documentation such products are called IFilter clients. TET PDF IFilter has been tested with the following products, but may also work with other Microsoft and third-party products which support the IFilter interface:

- SharePoint 2013 and above
- SQL Server 2012 and above
- Exchange Server 2010 and above
- Windows Search is integrated in Windows 8/10

TET PDF IFilter is available in 32-bit and 64-bit versions. The 64-bit version of the IFilter works only with 64-bit versions of the products above.

Registration of XML configuration files and post-installation steps. The installer offers an option for registering suitable XML configuration files for use with various IFilter clients (Windows Search, SharePoint, etc.). If an IFilter client is selected, the corresponding XML configuration file for TET PDF IFilter is added to the registry. In the case of Windows Search the predefined properties of TET PDF IFilter are registered (see Section 3.1.2, »Custom Properties«, page 35).

Some IFilter clients may require additional steps after TET PDF IFilter has been installed. These steps are discussed in the respective sections in Chapter 1, »Getting Started«, page 7.

Manual installation. While the installer applies all steps required to use TET PDF IFilter you may need to apply certain steps manually in some situations. Refer to the following information in this case.

To add the license key manually enter it at the following registry value:

```
HKEY_LOCAL_MACHINE\SOFTWARE\PDFlib\TET PDF IFilter5\license
```

To register the TET PDF IFilter DLL in the system (to make sure it will be found by IFilter clients) run the following command (possibly modified for a different installation directory) in a console window:

```
regsvr32 "C:\Program Files\PDFlib\TET PDF IFilter 5.3 64-bit\bin\TETPDFIFilter.dll"
```

Make sure to run this command from a command prompt with administrator privilege (see below).

If there are IFilter clients which are already using TET PDF IFilter, make sure to stop all related services which access TET PDF IFilter before registering the DLL again. Also, make sure that the Windows event log is closed.

Running privileged commands. Write access to the registry (e.g. by the *regsvr32* and *proptool* programs) requires administrator privileges. You can launch a command prompt with administrator privileges as follows: click on Start, type *cmd.exe* into the search box, right-click the resulting *cmd.exe* entry, and select *Run as administrator*. This triggers the UAC prompt, and after confirmation a command prompt with administrator privileges opens.

1 Getting Started

This chapter describes the initial steps required to configure and use some of the search and retrieval products (IFilter clients) supported by TET PDF IFilter. This description is intended to get you up and running with TET PDF IFilter. Advanced configuration aspects are discussed in Chapter 2, »Indexing Metadata Properties«, page 19.

In this chapter we assume that the appropriate 32- or 64-bit of TET PDF IFilter has been installed on the system.

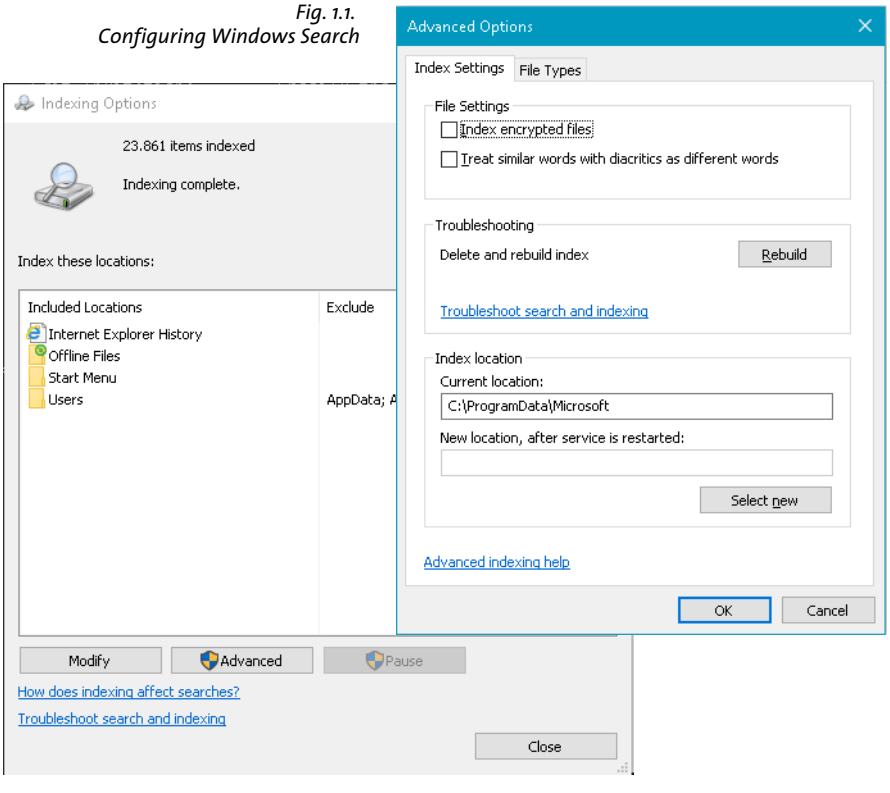
1.1 Windows Search

1.1.1 Configuration

System requirements. Windows Search implements a property system for metadata which is supported by TET PDF IFilter. Windows Search is available for Windows 8/10 and Windows Server 2012 and above. Note that Windows Search is disabled by default on Windows Server 2016 and above.

Setup and configuration. By default, Windows Search indexes documents in libraries (e.g. *Documents)* and offline files. You can instruct Windows Search to index documents in other locations (including network drives) as follows:

Fig. 1.1. Configuring Windows Search



- Click *Start*, *Control Panel*, *Indexing Options*.
  On Windows 10 you can alternatively click into the search field of an Explorer Window (this activates the *Search* item in the menu bar) and then click *Advanced options, Change indexed locations*.
- Click *Modify*. In the *Change selected locations* section, choose the directories that you want to index and click OK.
- Click *Advanced* and *Rebuild* to force the documents to be indexed immediately.

Starting and stopping the Windows Search service. You can start and stop the search service (more precisely: the indexing process which in turn calls TET PDF IFilter) manually with the following methods:

- In a console window with administrator privileges type the following to start the service:

  ```
  net start wsearch
  ```

  Type the following to stop the service:

  ```
  net stop wsearch
  ```

- To control the service in the control panel:
  Click on *Start, Control Panel, Administrative Tools, Services* and locate *Windows Search* in the list of available services. Double-click on the service name to bring up a dialog which offers *Start/Stop* and other controls.
- To rebuild the catalog:
  Click *Start*, *Control Panel*, *Indexing Options, Advanced* and *Rebuild*
  This will re-index all documents.

Note that Windows Search automatically starts the indexing service in certain situations.

1.1.1 Interactive Search

Content search. You can enter search terms in the search box near the top right corner of a Windows Explorer window

In addition to typing simple search terms you can use additional operators to narrow your search, using AQS *(Advanced Query Syntax)*. A comprehensive description of AQS is available at

msdn.microsoft.com/en-us/library/windows/desktop/bb266512%28v=vs.85%29.aspx

The Boolean keywords *AND*, *OR* and *NOT* must be spelled in uppercase. They are not subject to localization, i.e. the English forms must be used regardless of the Windows UI language. Table 1.1 contains examples of content searches.

Table 1.1 Searching content with Windows Search

query	***explanation***
`Hol`	*document contents or any property contains words starting with* `Hol`
`"Sherlock Holmes"`	*document contents or any property contains the exact phrase* `Sherlock Holmes`

Table 1.1 Searching content with Windows Search

query	*explanation*

`Holmes AND Watson`
`Holmes + Watson`
`Holmes Watson`
`(Holmes Watson)`
	document contents or any property contains both `Holmes` *and* `Watson`

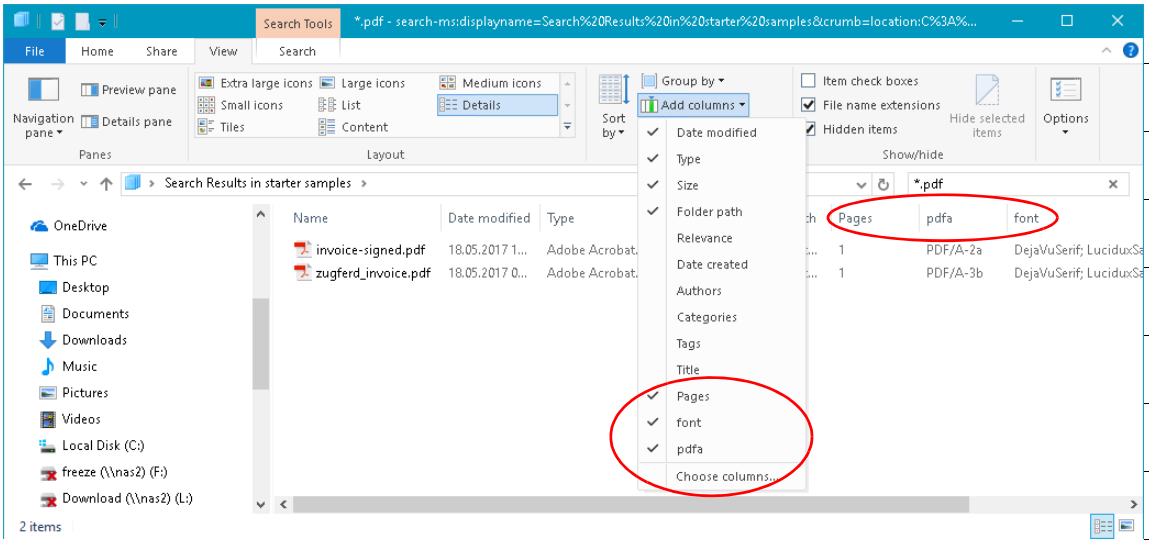
`Holmes OR Watson`	*document contents or any property contains either* `Holmes` *or* `Watson`
`Holmes NOT Mowgli` `Holmes -Mowgli`	*document contents or any property contains the term* `Holmes`*, but not the term* `Mowgli`
`System.Search.Contents:nutshell`	*text contents of the document contains* `nutshell`*; properties are ignored*

Properties in Windows Explorer. In addition to text searches you can search for metadata properties. You can display properties in an Explorer window via *View, Add columns, Choose columns* and selecting one or more additional properties. Now click *View, Details* to display the detailed file information. If a query is entered in the search box (even * or **.pdf* will do) and a property value is available for a document the value is displayed in the new column (see Figure 1.2). New properties are not used in existing Explorer Windows, but only in new Windows.

Canonical property name and localized display name. In addition to canonical names (e.g. *System.Author)* the localized display names or labels (e.g. *Authors)* can be used in queries depending on the Windows UI language. Recognized property names in a query are displayed in blue color in the search window.

In addition to the canonical property name (e.g. *System.Author)* Windows uses localized display names (e.g. *Author)*. If the display name contains space characters (e.g. *Date*

Fig. 1.2
Search results in Windows Explorer and additional columns with predefined properties



modified) these must be removed when querying with the property *(datemodified)*. The display names are used as column headers in Explorer windows.

The canonical names and localized English display names of all shell properties supported in TET PDF IFilter are listed in Appendix A.1, »Shell Property Set Collection«. To determine other localized names of shell properties (e.g. German) you can use the *proptool* utility with the *--list* option:

```
proptool --list System.Author
```

The resulting output on a German system looks as follows:

```
System.Author       Autoren             F29F85E0-4FF9-1068-AB91-08002B27B3D9/4
```

Property queries. Table 1.2 contains examples of property queries. See Section 3.1, »Metadata Properties in Windows Search«, page 35, for advanced topics related to metadata queries and custom properties. Properties come from the following groups:

- Shell properties are defined by the Windows operating system. The canonical (language-independent) Windows property names are listed at

  msdn.microsoft.com/de-de/library/windows/desktop/dd561977%28v=vs.85%29.aspx

  Some Windows shell properties are populated by TET PDF IFilter based on information found in the PDF document, e.g. *System.Document.PageCount* and *System.Keywords*. Table A.1 lists all shell properties which are emitted by TET PDF IFilter. This table lists canonical names and localized English display names (labels).
- Additional predefined properties are supported by TET PDF IFilter. They are grouped in property sets and listed in Table A.2 to Table A.5. Section 2.3, »Predefined Properties«, page 22, discusses configuration of predefined properties.
- Custom properties are user-defined metadata properties which can be configured in TET PDF IFilter; see Section 2.5, »Custom Properties«, page 24.

Table 1.2 Searching properties with Windows Search; long form with canonical property name and short form with English label are shown

query	***explanation***
search for Windows shell properties	
`System.Author:Doyle` `Authors:Doyle`	*author contains* `Doyle`
`System.Author:"Conan Doyle"` `Authors:"Conan Doyle"`	*author contains the words* `Conan Doyle`
`System.Author:Doy` `Authors:Doy`	*author starts with* `Doy`
`System.DateModified:=2017-03-27` `datemodified:=2017-03-27`	*modification date is March 03, 2017; the canonical date format works regardless of the Windows UI language*
`System.Document.DateCreated: = 2017-03-27` `contentcreated: = 2017-03-27`	*creation date is March 03, 2017*
`System.Document.PageCount: >= 100` `Pages: >= 100`	*document contains 100 or more pages*
search for properties predefined in TET PDF IFilter	

Table 1.2 Searching properties with Windows Search; long form with canonical property name and short form with English label are shown

query	*explanation*
`PDFlib.TET.pdfa:=PDF/A-2b` `pdfa:=PDF/A-2b`	*document conforms to the PDF/A-2b flavor*
`PDFlib.TET.pdfa:~<PDF/A` `pdfa:~<PDF/A`	*document conforms to any of PDF/A-1, PDF/A-2 or PDF/A-3*
`PDFlib.TET.producer:~=Microsoft` `producer:~=Microsoft`	*document contains a* `Producer` *entry with the term* `Microsoft`
`PDFlib.TET.fullpdfversion: <170` `fullpdfversion: <170`	*document conforms to a PDF version older than PDF 1.7*
`PDFlib.TET.font:=Calibri` `font:=Calibri`	*document contains text with a font whose name starts with* `Calibri`

1.1.2 Programmatic Search

In addition to interactive queries you can use the Advanced Query Syntax programmatically. You can use SQL syntax extensions which present the search index through a database-like programming interface.

SQL queries for metadata properties. SQL queries can search for predefined as well as custom properties. Some examples are provided below; they assume that indexing of all predefined properties has been enabled in the XML configuration file, and that the *predefined_properties.propdesc* property description has been registered. We use ADO *(ActiveX Data Objects)* and PowerShell scripts to submit SQL-based queries. However, you can use the SQL statements in any other ADO or ADO.NET environment as well. A description of the SQL syntax extensions for Windows search is available at

msdn.microsoft.com/en-us/library/bb231256(VS.85).aspx

Property searches can be applied in two directions:

- Query a specific property value, e.g. which documents have *Doyle* as author?
- Query the value of a specific property in one or more files, e.g. who is the author of this document?

PowerShell for submitting SQL queries. Sample PowerShell query scripts are installed with TET PDF IFilter. Some hints if you are not familiar with PowerShell:

- In order to run unsigned PowerShell scripts you must apply the following command once as Administrator:

```
set-executionpolicy remotesigned
```

- Run a script, e.g. q*uery_text_in_pdf.ps1* as follows in a PowerShell window:

```
& query_text_in_pdf.ps1
```

The following PowerShell script lists PDF/A conformance properties for all documents:

```
$objConnection = New-Object -comobject ADODB.Connection
$objRecordset = New-Object -comobject ADODB.Recordset
$objConnection.Open("Provider=Search.CollatorDSO;Extended
Properties='Application=Windows';")
```

```
$objRecordSet.Open(
"SELECT System.ItemPathDisplay, `"PDFlib.TET.pdfa`" FROM SYSTEMINDEX ", $objConnection)

While ($objRecordset.EOF -ne $True) {
    $private:item = $objRecordset.Fields.Item("System.ItemPathDisplay")
    Write-Output $item.Value
    $item = $objRecordset.Fields.Item("PDFlib.TET.pdfa")
    Write-Output $item.Value
    $objRecordset.MoveNext()
}
```

The following PowerShell script lists all documents where the PDF/A conformance contains *PDF/A-2:*

```
$objConnection = New-Object -comobject ADODB.Connection
$objRecordset = New-Object -comobject ADODB.Recordset
$objConnection.Open("Provider=Search.CollatorDSO;Extended
Properties='Application=Windows';")

$objRecordSet.Open("SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE " +
        "`"PDFlib.TET.pdfa`" = SOME ARRAY ['PDF/A-2']", $objConnection)

While ($objRecordset.EOF -ne $True) {
    $private:item = $objRecordset.Fields.Item("System.ItemPathDisplay")
    Write-Output $item.Value
    $objRecordset.MoveNext()
}
```

VBScript for submitting SQL queries. The following VBScript code lists all documents along with the name of the application which created the document. Unfortunately, only some shell properties can be used with VBScript queries; other properties cannot be queried:

```
On Error Resume Next

Set objConnection = CreateObject("ADODB.Connection")
Set objRecordSet = CreateObject("ADODB.Recordset")

objConnection.Open "Provider=Search.CollatorDSO;Extended
Properties='Application=Windows';"

objRecordSet.Open "SELECT System.ItemPathDisplay, System.ApplicationName FROM
SYSTEMINDEX", objConnection

objRecordSet.MoveFirst

Do Until objRecordset.EOF
    Wscript.Echo objRecordset.Fields.Item("System.ItemPathDisplay")
    Wscript.Echo objRecordset.Fields.Item("System.ApplicationName")
    Wscript.Echo ""
    objRecordset.MoveNext
Loop
```

Complex property queries with SQL. The following samples contain only the relevant SQL statement and can be used in any SQL environment. For using these statements in PowerShell scripts you must apply proper quoting, e.g. *`"PDFlib.TET.pdfa`"* instead of

"PDFlib.TET.pdfa". Many examples below use array queries for vector properties (see Section , »Multivalued properties«, page 26). Details on the syntax for array queries can be found at

msdn.microsoft.com/en-us/library/bb231264(VS.85).aspx

- List all documents where the author contains *Doyle:*

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE CONTAINS("System.Author",
'Doyle')
```

- List all documents where the author starts with *Rudy:*

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE CONTAINS("System.Author",
'"Rudy*"')
```

- List all documents which conform to PDF/A-1a:

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE "PDFlib.TET.pdfa" = 'PDF/A-
1:2005'
```

- List all documents containing at least one of the fonts *Bembo* and *TimesNewRoman:*

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE "PDFlib.TET.font" = SOME ARRAY
['Bembo', 'TimesNewRoman']
```

- List all documents containing both the *Bembo* and *Bembo-Bold* fonts:

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE "PDFlib.TET.font" = SOME ARRAY
['Bembo'] AND "PDFlib.TET.font" = SOME ARRAY ['Bembo-Bold']
```

- List all documents with at least one tennis image (Photoshop category *TEN=tennis*):

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE
"PDFlib.TET.images.photoshop.SupplementalCategories" = SOME ARRAY ['TEN']
```

- List all documents with a PDF version higher than 1.6 (the PDF version is returned by TET PDF IFilter as a string with the version number times ten, e.g. 16 for PDF 1.6):

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE "PDFlib.TET.pdfversion" > '16'
```

1.2 SharePoint

System Requirements. TET PDF IFilter works with the following SharePoint configurations:

- SharePoint Server 2016 and 2019
- SharePoint Server 2013 and SharePoint Foundation 2013; hotfix KB2883000 or July 8, 2014 Cumulative Update for SharePoint Server 2013 is required.

Installation. Proceed as follows to configure TET PDF IFilter for use with SharePoint-Server:

- Install TET PDF IFilter with the installer.
- Open a *SharePoint Management Shell* window and enter the following commands to enable TET PDF IFilter for PDF indexing:

```
$ssa = Get-SPEnterpriseSearchServiceApplication
Set-SPEnterpriseSearchFileFormatState -SearchApplication $ssa -Identity pdf ↵
     -UseIFilter $true -Enable $true
```

- You can use the following command to check whether the previous command was successful:

```
Get-SPEnterpriseSearchFileFormat -SearchApplication $ssa -Identity pdf
```

  The output of this command should look similar to the following (the entry *UseIFilter* should have the value *True):*

```
Identity   : pdf
Name       : PDF
MimeType   : application/pdf
Extension  : .pdf
BuiltIn    : True
Enabled    : True
UseIFilter : True
```

- Now restart the SharePoint search service:

```
net stop OSearch16                  % for SharePoint 2013: net stop OSearch15
net stop SPSearchHostController
net start SPSearchHostController
```

  Starting the *SPSearchHostController* service implicitly also starts the *OSearch16* service.

Simple and advanced Text Search. Several methods are available for building search queries in SharePoint:

- Query keywords
- SQL queries
- Queries encoded in URLs

See Section 3.2, »Metadata Properties in SharePoint«, page 39, for metadata queries.

1.3 Exchange Server

System requirements. TET PDF IFilter works with Microsoft Exchange Server 2010 and above.

Setup and configuration. Proceed as follows to configure TET PDF IFilter for use with Exchange Server:

- Install TET PDF IFilter with the installer.
- Apply the post-installation steps described below.

Post-installation steps. TET PDF IFilter must be registered for use with Microsoft Exchange Server by running the following PowerShell script with Administrator privileges:

```
register_in_exchange_2010.ps1
```

This script is installed in the subdirectory *IFilter clients\Exchange* of the TET PDF IFilter installation directory. After the script was executed successfully the *Microsoft Exchange Search Indexer* service must be restarted. Either perform this task from the Services control panel or from PowerShell on the command line:

```
stop-service MSExchangeSearch -Force
start-service MSExchangeSearch
```

The registration script must be executed again when a newer version of TET PDF IFilter is installed, as the default installation directory of TET PDF IFilter changes with an update.

After TET PDF IFilter has been registered the PDF attachments of all new mails are indexed by Exchange. In order to index PDF attachments of existing messages all mailboxes must be indexed again. The following article on the MSDN website describes the possible procedures for rebuilding the Exchange full-text index:
technet.microsoft.com/en-us/library/aa995966(v=EXCHG.80).aspx

1.4 SQL Server

System requirements. TET PDF IFilter works with the following editions of SQL Server:

- SQL Server 2012 and above

More information about full-text search in SQL Server can be found at

msdn.microsoft.com/en-us/library/mt590198(v=sql.1).aspx

Setup and configuration. In order to give you full control over the use of filters in SQL Server, the installer does not automatically register TET PDF IFilter in any instance of SQL Server. Instead, you must manually register TET PDF IFilter separately for all instances of SQL Server.

The following steps instruct SQL Server to access IFilters which are installed system-wide:

- Install TET PDF IFilter with the installer.
- Run *SQL Server Management Studio* and execute the following statements to make the system-wide document filters available to this instance of SQL Server (see *msdn.microsoft.com/en-us/library/dd207002%28v=sql.120%29.aspx* for details):

```
exec sp_fulltext_service 'load_os_resources', 1;
GO
exec sp_fulltext_service 'update_languages'
GO
exec sp_fulltext_service 'restart_all_fdhosts'
GO
```

Testing the configuration. You can check the configuration results to make sure that TET PDF IFilter is available for an instance of SQL Server. Use the following statements:

```
SELECT document_type, path FROM sys.fulltext_document_types WHERE document_type = '.pdf'
```

A resulting output line similar to the following indicates that TET PDF IFilter has successfully been configured for the instance (the exact path depends on your installation path):

```
.pdf    C:\Program Files\PDFlib\TET PDF IFilter 5.3 64-bit\bin\TETPDFIFilter.dll
```

Preparing a database table for full-text PDF indexing. TET PDF IFilter will be used by SQL Server for creating the full-text index for PDF documents stored in a column of type *varbinary(max)*. Since the document type is not available in this situation, the file extension must be stored in a separate column in the table, the so-called *type column*. The type column can be of any character-based data type. We use *VARCHAR(4)* and store the file extension *pdf*.

The following statements create a *DocumentTable* containing a sample PDF document in the *data* column, the file name in the *name* column, and the associated type in the *extension* column:

```
CREATE DATABASE TestDatabase
GO
USE TestDatabase
GO
CREATE TABLE DocumentTable
```

```
(pk INT NOT NULL IDENTITY CONSTRAINT DocumentTablePK PRIMARY KEY,
data VARBINARY(MAX), name VARCHAR(100), extension VARCHAR(4))
GO
INSERT INTO DocumentTable(data, name, extension) SELECT *,'The_Hound_of_the_
Baskervilles.pdf','pdf' FROM OPENROWSET(BULK 'C:\PDF\The_Hound_of_the_Baskervilles.pdf',
SINGLE_BLOB) AS Document
GO
```

Now you can create the full-text index:

```
sp_fulltext_database 'enable'
GO
CREATE FULLTEXT CATALOG TestCatalog AS DEFAULT
GO
CREATE FULLTEXT INDEX ON DocumentTable (data TYPE COLUMN extension)
KEY INDEX DocumentTablePK
GO
```

Dropping and recreating the full-text index. You can use the following statements to drop the full-text index:

```
USE TestDatabase
GO
DROP FULLTEXT INDEX ON DocumentTable
GO
```

Recreate the full-text index:

```
USE TestDatabase
CREATE FULLTEXT INDEX ON DocumentTable (data TYPE COLUMN extension)
KEY INDEX DocumentTablePK
GO
```

Simple and advanced text search. You can query for individual words in the full-text index:

```
SELECT name FROM DocumentTable WHERE CONTAINS(*, 'Watson')
GO
```

In order to search for a phrase consisting of multiple words enclose the phrase in double quotes:

```
SELECT name FROM DocumentTable WHERE CONTAINS(*,'"Arthur Conan Doyle"')
GO
```

A sample script for performing these steps with the supplied PDF samples is installed with TET PDF IFilter. More information about the *CONTAINS* predicate in Transact-SQL can be found at

msdn.microsoft.com/en-us/library/ms187787(SQL.100).aspx

See Section 3.3, »Metadata Properties in SQL Server«, page 44, for metadata queries.

2 Indexing Metadata Properties

In addition to the main text TET PDF IFilter can also feed a variety of metadata properties to the indexer. This allows powerful searches even if you are not looking for a particular text on the page.

2.1 Sources of Metadata in PDF

Most PDF documents contain document information entries, such as the *Author* and *Title* fields. In addition to document information entries PDF documents may contain XMP metadata. TET PDF IFilter supports indexing of several kinds of metadata in PDF documents.

Predefined and custom document info entries. Document information entries are considered the old and simple kind of PDF metadata. They can be displayed in Acrobat via *File, Properties...* . The PDF standard documents the following document info entries:

```
Title Author Subject Keywords Creator Producer CreationDate ModDate Trapped
```

In addition to these standard entries custom entries can be added to the set of document info entries. They can be displayed and edited in Acrobat (but not Adobe Reader) via *File, Properties..., Custom*.

Both predefined and custom document info entries can be addressed with pCOS paths in TET PDF IFilter (see below).

XMP properties at the document and image level. XMP *(Extensible Metadata Platform*[1]*)* is a framework for metadata. XMP is required, for example, for PDF/A conformance, and is supported by many applications. XMP metadata is organized in schemas which contain a number of properties. Properties are addressed using a namespace prefix and the property name.

XMP metadata is usually associated with the whole document. However, in PDF it is also possible to associate XMP with individual pages, images, or other objects. In practice this feature is mainly used for raster images. For example, a digital image may carry the name of the photographer, copyright notes, scene details and other information. An important source of image-related XMP metadata is Adobe Photoshop. If you create PDF from Photoshop or use Photoshop-created images in Adobe workflows, the XMP metadata is usually included in the PDF document and can be indexed with TET PDF IFilter.

The XMP specification includes a description of all predefined XMP schemas and properties.

XMP properties on the document or image level (as well as XMP associated with other objects) can be addressed in two steps: a pCOS path identifies the relevant PDF object, and a two-part XMP property name addresses the target XMP schema and property.

Extended pCOS paths. The pCOS *(PDFlib Comprehensive Object Syntax)* interface provides a simple and elegant facility for retrieving arbitrary information from all sections of a PDF document which do not describe page contents, such as page dimensions,

1. See www.adobe.com/products/xmp

metadata, interactive elements, etc. Examples for using the pCOS interface and a description of the pCOS path syntax are contained in the pCOS Path Reference which is available as a separate document. Additional examples can be found in the pCOS Cookbook at *www.pdflib.com/pcos-cookbook/*.

pCOS can be used in TET PDF IFilter to address information about a document. pCOS paths represent some aspect of the PDF document such as bookmarks, font name, or page size, and can also address document info entries or XMP metadata streams (but not properties within an XMP stream). While the pCOS API functions are not available in TET PDF IFilter, you can supply pCOS paths as expressions in the XML configuration file in order to index information about a document

Table 2.1 lists the pCOS paths for some commonly used PDF objects. Many pCOS objects are represented by arrays which require an array index in angle brackets, e.g. *pages[0]* denotes the first page (pages are counted starting at 0). In addition to all pCOS paths supported by the TET product (which forms the basis of TET PDF IFilter), the IFilter supports the following syntax extension for pCOS paths: You can use a »*« (asterisk) wildcard character instead of an array or dictionary index. This means that TET PDF IFilter will iterate over all possible values of the array index, and include all corresponding object values in the indexing process. An arbitrary number of wildcards may be used within a single pCOS path.

Table 2.1 pCOS paths for various PDF objects

pCOS path	***type***	***explanation***
`length:pages`	*number*	*number of pages in the document*
`/Info/Title`	*string*	*standard document info field Title*
`/Info/ArticleNumber`	*string*	*custom document info field* `ArticleNumber` *(document info entries can use arbitrary names)*
`/Root/Metadata`	*stream*	*XMP stream with the document's metadata*
`images[*]/Metadata`	*stream*	*XMP metadata streams for all images in the document*
`fonts[*]/name`	*string*	*name of a font*
`length:fonts`	*string*	*number of fonts in a document*
`length:images`	*string*	*number of images in a document*
`fonts[*]/embedded`	*boolean*	*embedding status of a font*
`pages[*]/width`	*number*	*width of the visible area of the page*
`pages[*]/annots[*]/A/URI`	*string*	*target URL of the Web links on all pages*
`tagged`	*Boolean*	*true if the document contains tags with structure information*

XML configuration for metadata sources. Sources of metadata properties can be configured in the *Source* element (child of the *Property* element) of the XML configuration file. While the *pdfObject* attribute contains an extended pCOS path for a PDF object, the *xmpName* attribute contains the schema prefix and name of an XMP property:

```
<Source pdfObject="/Info/ArticleNumber"/>
<Source xmpName="acme:number"/>
```

One or more sources can be configured for a property.

2.2 Metadata Organization

Metadata is organized in the following hierarchical way:

- *Properties* are the fundamental building blocks for metadata. Properties in the Windows operating system and the IFilter interface are organized by a unique numeric identifier (see below).
- *Property sets* comprise a group of properties which usually have some logical relationship. All properties in a set share the same GUID (see below). Property sets can be specified in the XML configuration file.
- *Property set collections* comprise a group of property sets. TET PDF IFilter implements several predefined property set collections. They can be used to collectively enable or disable multiple property sets together. It is not required to configure additional property set collections.

Property identification and GUIDs. Properties are identified in the IFilter interface by an identifier which consists of two parts:

- The first part is the GUID *(Globally Unique IDentifier)*, also known as UUID *(Universally Unique IDentifier)*, a unique 128-bit identifier in case-insensitive hexadecimal notation according to RFC 4122. The parts must be separated by dash characters »-«. There are various tools available for creating GUIDs, e.g. the online service at

  www.uuidgenerator.net/

  A sample GUID looks as follows: *7a737220-0cd0-11dd-bd75-0002a5d5c51b*.
- The second part uniquely identifies the property within its property set. It can consist of a positive integer called the *identifier*, or *ID* for short. Property identifiers in a set must start with the value 2, but are otherwise arbitrary. Property identifiers are supported in all IFilter clients.
  Alternatively, the second part may consist of a cleartext name. The use of names instead of IDs is deprecated, and is not supported by some IFilter clients, e.g. Windows Search. However, it can make configuration more convenient for those IFilter clients which support it, e.g. SharePoint. See »XML configuration for GUID+name treatment of properties«, page 25, for information on enabling the GUID+name method.

The GUID+ID or GUID+name combination is required to configure metadata property queries in search products. Other aspects of metadata properties are detailed in Section 2.5, »Custom Properties«, page 24.

2.3 Predefined Properties

The following predefined property set collections are built into TET PDF IFilter:

- *Shell properties* are known to Windows and have user-friendly names. TET PDF IFilter populates all shell properties which have equivalents in PDF documents. The values of shell properties are retrieved from the document info fields and document XMP metadata. Common examples are *System.Author*, *System.Title*, and *System.Document.DateCreated*. For a list see Appendix A.1, »Shell Property Set Collection«. More information can be found at

  msdn.microsoft.com/en-us/library/windows/desktop/dd561977(v=vs.85).aspx

- *PDF* properties are specific to PDF documents. They are populated from pCOS paths and are enabled by default. Examples are the PDF version number, PDF/A standard to which the document conforms, bookmark contents, or page size. For a full list see Appendix A.2, »PDF Property Set Collection«.
- *Document XMP* properties cover many document properties in the XMP specification; they are disabled by default. For a full list see Appendix A.3, »Document XMP Metadata Property Set Collection«.
- *Image XMP* properties cover XMP properties which are attached to images in the document; they are disabled by default. For a full list see Appendix A.4, »XMP Image Metadata Property Set Collection«.
- *Internal properties* are auxiliary properties which are not intended for production use, but as development and debugging aids. They are enabled by default and include software version numbers and the time of indexing. For a full list see Appendix A.5, »Internal Property Set Collection«.

XML configuration for enabling property set collections. Predefined property set collections can be enabled with the corresponding attributes in the *Metadata/PropertySetCollection* element:

```
<PropertySetCollection
        shell="true"
        pdf="true"
        documentXmp="true"
        imageXmp="true"
        internal="true"/>
```

By default, the *Shell*, *PDF* and *Internal* property set collections are enabled, while the *documentXMP* and *imageXMP* property set collections are disabled. Custom properties are enabled as soon as they are specified.

Handling of all predefined and custom metadata properties can be disabled completely with the *metdataHandling* attribute of the *Filtering* element:

```
<Filtering metadataHandling="ignore">
```

Note *If you enable or disable predefined properties you must recreate the index.*

2.4 Examples with Predefined Properties

Section 1.1.1, »Interactive Search«, page 8, lists several uses of shell and predefined properties in Windows Search. Below we demonstrate more advanced queries based on predefined properties.

Unique XMP document identifiers. XMP defines unique identifiers in the following properties within the XMP Media Management property set:

```
xmpMM:DocumentID
xmpMM:InstanceID
xmpMM:VersionID
xmpMM:OriginalDocumentID
```

These XMP properties are emitted by Adobe InDesign and other applications. TET PDF IFilter supports them with the predefined properties *PDFlib.TET. xmpMM.DocumentID* etc. These document identifiers can be used as follows:

- *xmpMM:InstanceID* is created as a unique entry for each modified version of a document. Regardless of the file name it can be used to identify duplicates.
- *xmpMM:OriginalDocumentID* remains constant across modifications of a document. This can be used to identify document variants and derived documents.

In order to use these properties you must enable the XMP metadata property set collection (see Section A.4, »XMP Image Metadata Property Set Collection«, page 79, for the complete list of properties) with the following XML configuration snippet:

```
<n:PropertySetCollection documentXmp="true" imageXmp="true" internal="true" pdf="true"
        shell="true"/>
```

Photoshop XMP image metadata. Professional images may contain metadata regarding the photographer's or agency's name, city or country where the image was taken, or the content category. Adobe Photoshop and other applications record such properties in XMP metadata attached to the image, where the XMP may also be stored along with the image data in PDF.

In order to use these properties you must enable the XMP image metadata property set collection (see Section A.4, »XMP Image Metadata Property Set Collection«, page 79, for the complete list of properties) with the following XML configuration snippet:

```
<n:PropertySetCollection documentXmp="true" imageXmp="true" internal="true" pdf="true"
        shell="true"/>
```

This instructs TET PDF IFilter to emit image-related XMP metadata. After recreating the index you can use image-related XMP properties in searches. The following query in Windows Search retrieves a list of documents containing images with the supplemental category TEN (=tennis), assuming the images carry appropriate XMP metadata:

```
PDFlib.TET.images.photoshop.SupplementalCategories:=TEN
```

As an alternative to the full canonical name the shorter display name can also be used in the query:

```
images.photoshop.SupplementalCategories:=TEN
```

2.5 Custom Properties

Custom metadata properties are additional properties beyond the predefined properties which meet specific requirements within an enterprise, organization, industry etc. TET PDF IFilter gives you full control over custom properties: they can be specified in the configuration file so that they will be generated by TET PDF IFilter and indexed by the search engine. For example, if you assign a product number as metadata property in all documents you can search for documents by product number instead of by content.

Planning custom metadata properties. In order to specify custom properties you must consider the following aspects (see »Property identification and GUIDs«, page 21, for details on GUIDs, identifiers, and friendly names):

- You can group one or more properties in a property set. Each property set needs a unique 128-bit identifier called GUID.
- The property *identifier* is a unique integer which identifies the property within its property set. Property identifiers in a set start with the value 2. With some IFilter clients the identifier can be replaced with a friendly name. You can override predefined properties by assigning the corresponding GUID+ID combination.
- The *friendly name* for a property is optional if an identifier is available, and required otherwise. It can be an arbitrary name which must be unique within the configuration file. While for some IFilter clients such as SharePoint it can be used instead of the identifier, friendly names do not work in other IFilter clients (e.g. Windows Search).
- Property *source:* properties can be populated from document metadata or general PDF information according to Section 2.1, »Sources of Metadata in PDF«, page 19.
- The *data type* of the property: *Int32* (32-bit integer), *Double* (floating point number with double precision), *Boolean* (true/false), *DateTime* (see below), and *String*.
- The *precedence* rule: if there is more than one data source for the property you can specify whether the first available non-empty data source has precedence (i.e. subsequent sources are ignored), or whether data from all non-empty sources are collected.
- Specify whether the property is emitted as a *vector*, i.e. multiple values are passed to the IFilter interface in an array structure instead of a flat value (see Section , »Multivalued properties«, page 26).
- A *prefix* which is prepended to the property name if properties are indexed as part of the full text (see Section 2.7, »Index Properties as Text«, page 32).

The DateTime property type. In order to specify a point in time you can use the *DateTime* data type for properties. While the output created for *DateTime* properties is always in the format required by the IFilter interface, the input supports different formats depending on the source of the property:

- If the data source is a pCOS path, e.g. for a standard or custom document info field such as */Info/CustomDate,* the value is expected in standard PDF date format as specified in ISO 32000-1, section 7.9.7. The general format of a PDF date string is *D:YYYYMMDDHHmmSSOHH'mm*. Note that the PDF date format (unlike the XMP format discussed below) does not support fractions of a second. Some examples:

```
D:202009231858                  GMT
D:202009231058-08'00'           same instant expressed in U.S. Pacific Standard Time
D:202009231958+01'00'           same instant expressed in Central European Time
```

- If the data source is an XMP property (e.g. *xmp:ModifyDate)*, the value is expected as specified for the basic value type *Date* in the XMP Reference, identical to the format specified in ISO 8601[1]. The general format of an XMP date string is *YYYY-MM-DDThh:mm:ss.sTZD*, where *TZD* is the time zone designator *(Z* or *+hh:mm* or *-hh:mm)*. Note that some parts are optional: XMP dates support six levels of granularity with increasing accuracy, while the PDF date format supports only a single level of granularity. Some examples:

```
2020-09-23T18:58:30Z            GMT
2020-09-23T13:58:30-05:00       same instant expressed in US Eastern Standard Time
2020-09-23T19:58:30+01:00       same instant expressed in Central European Time
```

TET PDF IFilter normalizes *DateTime* properties to UTC as specified in the IFilter interface specification. As a result, searches for *DateTime* properties can always be performed with respect to the local time zone, regardless of the time zone in which the PDF document has been created.

XML configuration for custom properties. One or more custom properties can be specified in the *PropertySet* element, where each *Property* element describes a property in the set:

```
<PropertySet guid="E544AFE6-13E2-40F1-A702-DCEBE8FB7B03">
        <Property friendlyName="MailTo" identifier="2" type="String">
            <Source xmpName="acme:mailto"/>
        </Property>
</PropertySet>
```

Multiple PDF sources can be mapped to the same Windows property. The presence of a *Property* element will automatically enable processing for the specified property. However, handling of all predefined and custom metadata properties can be completely disabled with the *metadataHandling* attribute of the *Filtering* element:

```
<Filtering metadataHandling="ignore">
```

XML configuration for GUID+name treatment of properties. TET PDF IFilter will use GUID+ID to identify properties in the IFilter interface if the *identifier* is available. Custom properties which do not have the *identifier* attribute but only the *friendlyName* attribute, will be identified by GUID+name instead. In order to enable GUID+name treatment for predefined properties as well (and to globally force GUID+name treatment) you can use the *useIdentifier* attribute of the *Filtering* element:

```
<Filtering useIdentifier="false">
```

The names used for the predefined properties are similar to the property prefixes shown in Table 2.2, except that the leading *TET_* and the trailing underscore '_' are dropped (e.g. *System_Author, pdfversion, dc_contributor, photoshop_DateCreated)*.

Using GUID+name treatment for properties can be more convenient than GUID+ID in some environments, especially SharePoint.

Suppressing a particular property value. In some situations it may be desirable to suppress a specific property value. For example, the value which represents an uninteresting default is dropped, while only the interesting values are passed on to the IFilter cli-

1. See www.w3.org/TR/NOTE-datetime

ent. This can be achieved with the *suppressValue* attribute. If it is supplied for the *Source* element a specific value returned by the source will be suppressed. If it is supplied for the *Property* element all configured sources for the property are processed, the result is compared against the *suppressValue* attribute, and dropped if it matches.

Multivalued properties. Metadata properties can contain one or more values. *Single-valued properties* consist of a flat value which describes the document as a whole. Examples for single-valued properties are the creation date (property sources: *xmp:CreateDate* and */Info/CreationDate)* and the unique document identifier *(dc:identifier)*.

Multivalued properties may occur more than once per document. Examples for multivalued properties are the list of document authors and the list of document keywords. Multivalued properties can be created with TET PDF IFilter in different ways:

- The source of the property is an XMP container type and can therefore hold multiple entries at once, e.g. *dc:creator* has type *Seq* in XMP.
- The property is populated from a multivalued pCOS path with wildcards, where the wildcards may be expanded to any number of individual entries, e.g. *bookmarks[*]/Title*.
- The property is defined with more than one *Source* elements, and the *precedence* attribute of the property has the value *try-all*, e.g. *pdf:Keywords* and */Info/Keywords*. In contrast, the default *precedence=first-wins* processes only the first non-empty property source.

Vector treatment of properties. By default, TET PDF IFilter processes all relevant sources in the property definition (subject to the *precedence* attribute), and emits as many non-empty property values as available. In other words, each value is returned as a single entity to the IFilter client. Property values are returned in unspecified order.

Alternatively, multivalued properties can be provided as vectors to the IFilter client. This means that a single array entity is emitted which contains one or more values. There are several relevant aspects of vector processing for properties:

- SharePoint supports multivalued properties only if they are processed as vector entities.
- Some IFilter clients, e.g. Windows Search, support vector queries where you can search for one or more values in a multivalued vector property in a single query.

Keep in mind that there are two separate concepts: *multivalued* refers to an aspect of the property's source, while *vector processing* refers to the way in which property values are transferred to the IFilter client. Vector processing can be applied to multivalued properties even if they contain only a single value.

Some of the predefined properties are multi-valued (see Appendix A, »Predefined Metadata Properties«).

XML configuration for vector properties. Vector processing for custom properties can be enabled with the *emitAsVector* attribute of the *Property* element:

```
<Property friendlyName="MailTo" type="String" precedence="try-all" emitAsVector="true">
        <Source xmpName="acme:mailto"/>
        <Source xmpName="gov:mailto"/>
</Property>
```

2.6 Examples with Custom Properties

In the examples below we demonstrate several custom metadata properties. In addition to TET PDF IFilter XML configuration snippets we also present the corresponding *.propdesc* snippets for Windows Search. A detailed discussion of *.propdesc* files and the corresponding *proptool* utility can be found in Section 3.1.2, »Custom Properties«, page 35.

Common steps for custom metadata. In all samples the following steps are required:

- Add new property definitions to the XML configuration file for TET PDF IFilter. In some cases it is also required to enable predefined property set collections in the XML configuration file. The XML configuration file *starter_samples.xml* contains entries for all examples below so that you can use it as a starting point. It must be configured in the registry as detailed in Section 5.1, »Working with Configuration Files«, page 59. The *starter_samples.xml* configuration file is prepared for use with Windows Search. In order to use it with SharePoint or SQL Server you must apply minor adjustments (see Table 3.2, page 39, and Table 3.5, page 44, respectively).
- In order to use the new properties with Windows Search you must configure them with a suitable XML property description file and register it in Windows as follows (see »Register property descriptions in Windows«, page 36):

  ```
  proptool --register starter_samples.propdesc
  ```

  Now you can configure Windows Explorer to display the new properties (see »Property queries«, page 10). The property description file *starter_samples.propdesc* contains entries for all examples below so that you can use it as a starting point.
- Recreate the index to collect the new properties. You must open a new Search or Explorer Window to enable additional property columns in the Details dialog.

pCOS properties for number of fonts or images. The total number of fonts in a PDF document can be queried with the pCOS path *length:fonts;* similarly for images. In order to emit these properties use the following XML configuration snippet within the *PropertySet* element:

```
<!-- PDF properties: number of fonts and images in the document -->
<n:Property friendlyName="fontcount" type="Int32" identifier="2">
        <n:Source pdfObject="length:fonts" />
</n:Property>

<n:Property friendlyName="imagecount" type="Int32" identifier="3">
        <n:Source pdfObject="length:images" />
</n:Property>
```

In order to use the new *fontcount* and *imagecount* properties with Windows Search you must configure and register it in a Windows property description file with the following *.propdesc* snippet:

```
<!-- Number of fonts in the document -->
<propertyDescription name="fontcount" formatID="{E544AFE6-13E2-40F1-A702-DCEBE8FB7B03}"
        propID="2">
   <typeInfo type="Int32" isInnate="true" isViewable="true"/>
   <labelInfo label="fontcount"/>
   <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>
```

```
<!-- Number of images in the document -->
propertyDescription name="imagecount" formatID="{E544AFE6-13E2-40F1-A702-DCEBE8FB7B03}"
        propID="3">
   <typeInfo type="Int32" isInnate="true" isViewable="true"/>
   <labelInfo label="imagecount"/>
   <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>
```

Now you can display the numbers of fonts and images in a new column in Explorer windows and search for documents with a particular number of fonts or images with the following query:

```
imagecount: >10
```

pCOS property for PDF standards. The predefined properties *PDFlib.TET.pdfa, PDFlib.TET.pdfx* etc. describe the standard conformance status of a document. These properties are based on the corresponding pCOS paths. If a PDF document doesn't conform to any PDF/A standard flavor, the pCOS pseudo object *pdfa* returns *none*. However, the property *PDFlib.TET.pdfa* suppresses the value *none*. In the following example we define a multi-valued custom property *Standards* which works similar to the individual predefined properties *PDFlib.TET.pdfa, PDFlib.TET.pdfx* etc., but emits all standard conformance entries in a single vector property. It uses the corresponding pCOS paths *pdfa* etc. as sources, but drops the uninteresting value *none*. The following XML configuration snippet within the *PropertySet* element defines the *Standards* property:

```
<n:Property friendlyName="Standards" emitAsVector="true" suppressValue="none"
        identifier="4" precedence="try-all">
        <n:Source pdfObject="pdfa"/>
        <n:Source pdfObject="pdfe"/>
        <n:Source pdfObject="pdfx"/>
        <n:Source pdfObject="pdfua"/>
        <n:Source pdfObject="pdfvt"/>
</n:Property>
```

In order to use the new *Standards* property with Windows Search you must configure and register it in a Windows property description file with the following *.propdesc* snippet:

```
<propertyDescription name="Standards" formatID="{E544AFE6-13E2-40F1-A702-DCEBE8FB7B03}"
        propID="4">
           <typeInfo type="String" multipleValues="true" isInnate="true"
                isViewable="true"/>
           <labelInfo label="Standards"/>
           <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>
```

Now you can display the *Standards* column in an Explorer window where all applicable standards are displayed, separated by semicolons, e.g.

```
PDF-A-2a; PDF/UA-1
```

Undesired entries *none* are suppressed: if a document doesn't conform to any of the recognized standards the *Standards* column remains empty. You can search for PDF/A documents with the following query which checks whether the property value contains PDF/A:

```
Standards:~<PDF/A
```

The following more specific searches for the particular standard flavor PDF/A-3b:

```
Standards:=PDF/A-3b
```

pCOS property for PDF signature types. The pCOS path *signaturefields[0]/sigtype* checks the type of the first signature field. It returns one of the strings *none, approval, certification* or *doctimestamp*. In the corresponding property definition we will suppress the value *none* since it is not very useful. The following XML configuration snippet within the *PropertySet* element configures TET PDF IFilter to emit information about all digital signatures in a document. Since we don't know the number of signatures in advance we supply the asterisk character '*' as wildcard:

```
<n:Property friendlyName="Signature" suppressValue="none" identifier="5">
        <n:Source pdfObject="signaturefields[*]/sigtype"/>
</n:Property>
```

This property reports all signature types in a document. Note that multiple signatures of the same type are reported only once since multiple occurrences of the same property value are not possible. In order to use the *Signature* property with Windows Search you must configure and register it in a Windows property description file with the following *.propdesc* snippet:

```
<propertyDescription name="Signature"
        formatID="{E544AFE6-13E2-40F1-A702-DCEBE8FB7B03}" propID="5">
           <typeInfo type="String" multipleValues="true" isInnate="true"
            isViewable="true"/>
           <labelInfo label="Signature"/>
           <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>
```

Now you can display the signature status in a new column in an Explorer window and search for signed documents with the following query which checks whether the property has a non-empty value:

```
Signature: <>[]
```

Possible values for the pCOS object *sigtype* are *none* (for unsigned fields), *approval, certification,* and *doctimestamp* (see pCOS Path Reference). The expression above tests whether the signature type is different from *none*.

pCOS property for user-defined document info fields. Most standard PDF document info entries, e.g. *Title, Subject, Author*, are automatically mapped to the corresponding shell properties (see Section A.1, »Shell Property Set Collection«, page 75). In addition to these standard entries you can emit user-defined info entries which may be present in the indexed PDF documents. The following XML configuration snippet within the *PropertySet* element defines the *invoicenumber* property which is populated from an info entry called *InvoiceNumber:*

```
<n:Property friendlyName="invoicenumber" identifier="6">
        <n:Source pdfObject="/Info/InvoiceNumber"/>
</n:Property>
```

In order to use this property with Windows Search you must configure and register it with the following *.propdesc* snippet:

```
<propertyDescription name="invoicenumber"
        formatID="{E544AFE6-13E2-40F1-A702-DCEBE8FB7B03}" propID="6">
        <typeInfo type="String" isInnate="true" isViewable="true" isQueryable="true"/>
        <labelInfo label="invoicenumber"/>
        <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>
```

Now you can display the invoice number in a new column in Explorer windows and search for invoice number with the following query:

```
invoicenumber: =R123456
```

XMP property for identifying ZUGFeRD invoices. The ZUGFeRD and Factur-X standards[1] use PDF/A documents with custom XMP properties for identifying invoices as ZUGFeRD-compliant. For example, the following XMP fragment describes a basic ZUGFeRD invoice:

```
<rdf:Description
        rdf:about=""
        xmlns:zf="urn:ferd:pdfa:CrossIndustryDocument:invoice:1p0#"
        zf:ConformanceLevel="BASIC"
        zf:DocumentFileName="ZUGFeRD-invoice.xml"
        zf:DocumentType="INVOICE"
        zf:Version="1.0"/>
```

TET PDF IFilter can easily be configured so that you can identify ZUGFeRD-conforming documents in your queries based on the corresponding ZUGFeRD XMP properties.

The following XML fragment declares the required ZUGFeRD namespace URI and prefix as specified in the standard:

```
<n:PrefixDeclarations>
        <n:PrefixDeclaration
            uri="urn:ferd:pdfa:CrossIndustryDocument:invoice:1p0#" prefix="zf"/>
</n:PrefixDeclarations>
```

The following XML configuration snippet within the *PropertySet* element configures TET PDF IFilter to emit the XMP property *zf:DocumentType* in the custom property *zugferd*. For ZUGFeRD-compliant documents it must contain the value *INVOICE:*

```
<n:Property friendlyName="Zugferd" identifier="7">
        <n:Source xmpName="zf:DocumentType"/>
</n:Property>
```

In order to use the new *zugferd* property with Windows Search you must configure and register it in a Windows property description file with the following *.propdesc* snippet:

```
<propertyDescription name="Zugferd"
        formatID="{E544AFE6-13E2-40F1-A702-DCEBE8FB7B03}" propID="7">
           <typeInfo type="String" isInnate="true" isViewable="true"/>
           <labelInfo label="Zugferd"/>
           <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>
```

1. More information about ZUGFeRD can be found on the PDFlib Web site.

Now you can display the ZUGFeRD conformance status in a new column in Explorer windows and search for ZUGFeRD documents with the following query:

```
Zugferd:=INVOICE
```

2.7 Index Properties as Text

Most text retrieval engines support property queries in some way. However, querying for properties may not be possible with retrieval products which support only full-text search, e.g. SQL Server. In another scenario, explicitly searching for properties may not be desired if you want to search for any occurrence of the search term, regardless of whether it occurs in the document content or some property. In both situations you can instruct TET PDF IFilter to include all properties in the full-text index. In order to distinguish the property from actual document content, TET PDF IFilter can optionally prefix the property values with a string which makes it easier to identify properties as such in situations where the search engine does not directly support property searches. The canonical property name is used as prefix for the predefined properties listed in Appendix A, »Predefined Metadata Properties«.

While indexing properties as text allows limited property searches in environments which otherwise do not support property searches at all, you should be aware of the fact that the property searches are limited. For example, Boolean or other expressions are not available for property values.

XML configuration for indexing metadata as text. To index metadata properties as text set the *metadataHandling* attribute of the *Filtering* element to *propertyAndText* (to transparently blend the properties into the main text) or *propertyAndPrefixedText* (to identify the properties with a prefix):

```
<Filtering metadataHandling="PropertyAndPrefixedText">
```

The optional prefix which will be prepended to custom properties when filtering the document can be specified for custom properties in the *textIndexPrefix* attribute of the *Property* element:

```
<Property friendlyName="Title" identifier="7" textIndexPrefix="TITLE_">
...
</Property>
```

The prefixes which will be prepended to predefined properties are constructed according to the following scheme:

```
TET_<property name>_
```

where period characters '.' in the property name are replaced with underscore characters '_' (see Table 2.2 for examples).

Table 2.2 Prefixes for indexing metadata properties as text

Property set collection	*sample property name*	*prefix for indexing metadata as text*
Shell	`System.Author`	`TET_System_Author_`
TET	`PDFlib.TET.fullpdfversion`	`TET_fullpdfversion_`
XMP	`dc:contributor`	`TET_dc_contributor_`
Image	`photoshop:DateCreated`	`TET_photoshop_DateCreated_`

Scenario 1: Transparently blend metadata properties into the main text. If metadata properties contains sufficiently distinctive text which identifies the target document(s), it will suffice to include the properties in the full-text index and include it in standard full-text queries. For example, if you query for a specific article number, it doesn't matter whether the number occurs in the main text of the document or in a metadata property, as long as only one particular document talks about the article number in question. In other words, if it doesn't matter whether the text occurs in the main text or some metadata property, you must simply enable the indexing of properties as full-text, without any additional steps.

Use the following XML configuration to transparently blend metadata into the main text:

```
<Filtering metadataHandling="propertyAndText">
```

Scenario 2: Distinguish metadata from the main text. In other situations it may be relevant whether the text occurs in the main document or in some metadata property. For example, it makes a big difference whether you search for documents authored by *Doyle*, or documents which include the term *Doyle* in the main text. In this scenario you must not only enable the indexing of properties as full-text, but also include suitable prefixes for each property which make it possible to distinguish between text in the main document contents and text in metadata properties.

The value of the predefined property *System.Author* will be prepended by the prefix *TET_System_Author_*. For example, you can emulate a property-based search for *System.Author=Doyle* with a full-text search for *TET_System_Author_Doyle*. Since *System.Author* is a predefined property, the corresponding XML configuration does not require any property-specific entries, but must simply enable indexing of properties as prefixed text:

```
<Filtering metadataHandling="propertyAndPrefixedText">
```

In order to emulate a property-based search for documents with the article number *XY123456* with a full-text search for *ArticleNumber_XY123456* use the following XML configuration:

```
<Filtering metadataHandling="propertyAndPrefixedText">

<PropertySet guid="404e8a40-2e85-11dd-97f6-0002a5d5c51b">
        <Property identifier="2" textIndexPrefix="ArticleNumber_">
                <Source pdfObject="/Info/ArticleNumber"/>
        </Property>
<PropertySet>
```

2.8 Ignore Page Contents in Favor of Properties

In some situations users may want to initiate searches based on metadata properties only instead of the page content, i.e. completely ignore page contents in the indexing step and exclusively rely on metadata properties. Conceivable reasons for such a scenario are one or more of the following:

- More control over queries issued by users: don't waste sifting through long result lists when you can get an exact result based on metadata properties.
- The searched documents contain roughly the same words, but in different combinations, e.g. invoices or other transactional documents.
- The actual page contents are not really relevant for the search since they are well-known in advance. However, the kind of assembled pages for a particular document is of interest, e.g. insurance transactions which are associated with a variable number of contracts: not the exact contract wording is relevant for the search, but the number and kind of assembled contract documents.
- The searched documents don't contain any searchable text at all, e.g. scanned documents without any OCR performed.
- The documents contain information which does not contribute to the index in any reasonable way, e.g. long financial documents which contain only numbers, or technical plans without text (or only text within captions in the drawings).
- Performance optimization in any of the cases above: don't waste time indexing documents if the contents are known to be unhelpful for the search.

XML configuration for disabling page content indexing. To completely disable page content indexing set the *indexPageContents* attribute of the *Filtering* element to *false:*

```
<Filtering indexPageContents="false" metadataHandling="property">
```

3 Metadata Properties in IFilter Clients

3.1 Metadata Properties in Windows Search

Note Even if you ultimately want to work with SharePoint or another IFilter Client, you can use Windows Search to get started with metadata handling.

3.1.1 Predefined Properties

Configuration. The Windows operating system supports a powerful property system for handling metadata properties defined by the system (called shell properties) and user-defined (custom) properties. Windows Search uses the Windows property system to access descriptions of system (shell) properties and user-defined metadata properties.

Many properties are already built into TET PDF IFilter (see Section 2.3, »Predefined Properties«, page 22, and Appendix A, »Predefined Metadata Properties«); some of these are fed to Windows shell properties. If you want to work with predefined properties the configuration is rather simple:

- Descriptions of the predefined properties must be made available to Windows. This is done automatically if you selected *Windows Search* as IFilter client during the installation of TET PDF IFilter. Otherwise you must manually register the file *predefined_properties.propdesc* (see below).
- Most property set collections in TET PDF IFilter are enabled by default. Only if you want to work with document XMP or image XMP properties you must enable the respective collection in the XML configuration file (see Section , »XML configuration for enabling property set collections«, page 22).

Once you configured a property in Windows and TET PDF IFilter you can use it for searching (see »Property queries«, page 10).

The option *--list _all* dumps all properties which are registered in the system. The display names of all predefined non-shell properties are shortened versions of the canonical names, e.g. the display name of *PDFlib.TET.pdfa* is *pdfa* (regardless of the UI language).

3.1.2 Custom Properties

In order to search with custom metadata properties in Windows Search you must configure them in both the Windows Property System and TET PDF IFilter. Note that custom properties are implicitly enabled as soon as they are specified in the XML configuration file.

Property description files. In order to use custom metadata properties with Windows Search you must prepare a property description file *(.propdesc)* which specifies the names, data types, and GUIDs of properties, as well as other property attributes. The property descriptions must match the corresponding property descriptions in the XML configuration file for TET PDF IFilter. Property descriptions must be specified as XML files according to the syntax described at the following location:

msdn.microsoft.com/en-us/library/bb773879(VS.85).aspx

An XML schema description of the *.propdesc* format is available in *WDS_Property_Description_Schema.xsd*. The following fragment demonstrates some property descriptions:

```
<propertyDescription name="fontcount" formatID="{5eac0060-1ba4-11dd-92c4-0002a5d5c51b}"
        propID="2">
  <typeInfo type="Int32" isInnate="true" isViewable="true"/>
  <labelInfo label="fontcount"/>
  <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>

<propertyDescription name="weblink"
  formatID="{5eac0060-1ba4-11dd-92c4-0002a5d5c51b}" propID="6">
  <typeInfo type="String" isInnate="true" multipleValues="true" isViewable="true"/>
  <labelInfo label="weblink"/>
  <searchInfo inInvertedIndex="true" isColumn="true"/>
</propertyDescription>
```

The *multipleValues="true"* attribute is important for multivalued properties (see Section , »Multivalued properties«, page 26). Note that Windows restricts property names to a maximum length of 64 characters.

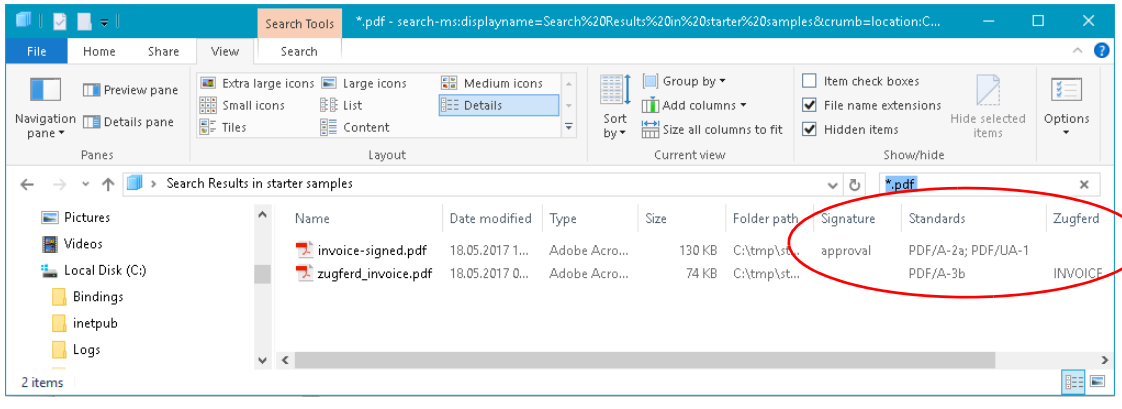
Note *The file* `starter_samples.propdesc` *in the TET PDF IFilter distribution may serve as a starting point. The file* `starter_samples.propdesc` *contains property descriptions for all examples presented later in this chapter.*

Register property descriptions in Windows. You must register custom property descriptions with the command-line utility *proptool.exe* which is installed with TET PDF IFilter. Supply the name of a property description file to register it with the Windows property system (make sure to enclose the path with double quotes if it contains space characters):

```
proptool --register acme.propdesc
```

Now you can configure Windows Explorer to display the new properties (see »Property queries«, page 10). The Windows property system stores the names of property description files in the following registry keys (and increasing numbers in the last component):

Fig. 3.1
Search results in Windows Explorer and additional columns with custom properties



```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\PropertySystem\PropertySchema\0000
```

The *proptool* utility emits an error message and a *HRESULT* error value if the property file could not be registered successfully, e.g. if a duplicate property was detected. In case of an error you can check the application event log for more details:

- *Start, Settings, Control Panel, Administrative Tools, Event Viewer*
- Click on *Application* in the left pane.
- In case of a problem with property registration there will be an entry with source *Microsoft-Windows-propsys*. Double-click on the line containing the entry and examine the error message (e.g. *Omitted duplicate property)*.

Alternatively, you can use the *HRESULT* value emitted by *proptool* to analyze the problem. A list of *HRESULT* values and explanations can be found at

msdn.microsoft.com/en-us/library/cc231198.aspx

Additional error numbers related to XML parsing can be found at

https://msdn.microsoft.com/en-us/library/ms753129%28v=vs.85%29.aspx

The following requirements must be met when registering property description files for Windows Search:

- Property descriptions must use the file name suffix *.propdesc* since other suffixes are not accepted by the Windows Property system.
- In order for custom properties to become available for searching, you must register the property description, stop and restart the search service, and force a rebuild via the Windows Search options:

```
proptool --register "acme.propdesc"
net stop wsearch
net start wsearch
...rebuild the catalog
   (see »Starting and stopping the Windows Search service«, page 8)...
```

- Registered properties must not conflict in GUID+ID or GUID+name with existing Windows properties (including the shell properties). A list of Windows properties can be found at

  msdn.microsoft.com/en-us/library/windows/desktop/dd561977(v=vs.85).aspx

- You need Administrator rights (see »Running privileged commands«, page 6) to write to the *HKEY_LOCAL_MACHINE* registry hive to run the *proptool* tool.
- Since the registry stores only the name of the property description file (but not its contents), the *propdesc* file must remain at the registered location to make sure that the property descriptions will be available for searches.
- The *propdesc* file must be readable for all users.
- You can register multiple property description files. However, two property description files must not include descriptions of the same property (as characterized by GUID+ID).
- If you apply changes in property description files and repeatedly register and unregister a *propdesc* file with the same name (which is typical for testing), rebooting may be required to ensure that Windows correctly picks up all property descriptions.

Use the *--unregister* option to remove a previously registered property description:

```
proptool --unregister "c:\property\acme.propdesc"
```

3.1.3 Properties in TET PDF IFilter

Predefined TET PDF IFilter properties. A property description file *predefined_properties.propdesc* for all predefined properties is available. If you selected *Windows Search* as IFilter client in the installer it is already registered for Windows Search. Although the property definitions are built into TET PDF IFilter you must enable the desired property set collections in the XML configuration file (see Section 2.3, »Predefined Properties«, page 22) before you can use them in queries.

XML configuration for Windows Search. Table 3.1 lists requirements and recommendations related to the XML configuration for TET PDF IFilter when working with Windows Search. The sample configuration file *Windows Search config.xml* implements these requirements and can be used as a starting point.

Table 3.1 XML configuration for Windows Search

element	*attribute*	*requirements and recommendation*
Filtering	***useIdentifier***	*Must be* `true` *since Windows Search supports only the GUID+ID method for identifying properties, but not GUID+name (see »XML configuration for GUID+name treatment of properties«, page 25).*
PropertySet-Collection	***shell***	*Should be* `true` *to support the shell properties known to Windows Search (note that* `true` *is the default anyway).*
Property	***identifier***	*Required since Windows Search supports only the GUID+ID method for identifying properties, but not GUID+name (see »XML configuration for GUID+name treatment of properties«, page 25).*

3.2 Metadata Properties in SharePoint

You can programmatically control metadata handling in SharePoint using the Enterprise Search Administration namespace *Microsoft.Office.Server.Search.Administration*. The following concepts are used for handling metadata properties in SharePoint:

- *Crawled properties* are created by TET PDF IFilter when indexing (crawling) PDF documents. Crawled properties can have multiple values. They are controlled via the TET PDF IFilter configuration file. SharePoint will search crawled properties like other fields, but advanced features for property queries are not be available.
- *Managed properties* can be used in search queries and advanced search, and displayed in search results.

More details on managed properties in the Advanced Search Page can be found at

msdn.microsoft.com/en-us/library/bb428648.aspx
msdn.microsoft.com/en-us/library/bb608302.aspx

XML configuration for SharePoint. Table 3.2 lists requirements and recommendations related to the XML configuration for TET PDF IFilter when working with SharePoint. The sample configuration file *SharePoint config.xml* implements these requirements and can be used as a starting point.

Table 3.2 XML configuration for SharePoint

element	*attribute*	*requirements and recommendation*
Filtering	***useIdentifier***	*Set to* `false` *to force GUID+name treatment of all properties including the predefined properties.*
Property	***emitAsVector***	*Must be* `true` *for properties which may have multiple values (see Section , »Multivalued properties«, page 26).*
Property	***friendlyName***	*Since GUID+name treatment facilitates locating properties in the list, we recommend the use of this attribute (see »Property identification and GUIDs«, page 21).*

Configure custom metadata properties. Proceed as follows to prepare custom metadata properties for indexing:

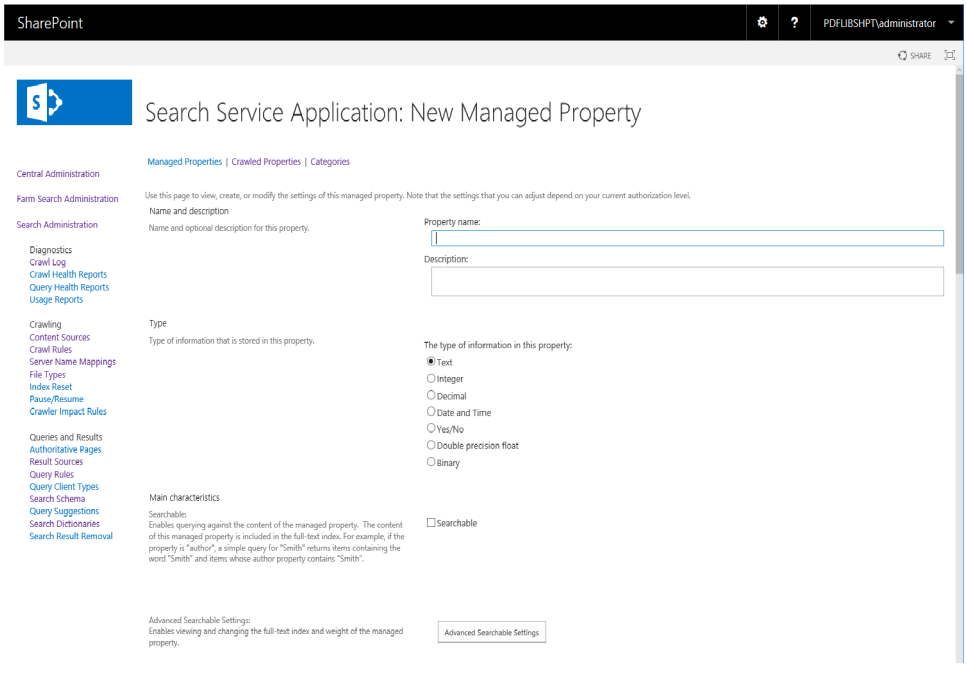
- Run a full crawl. During the crawl TET PDF IFilter reports the properties which are configured in the XML configuration file and found in one or more documents. This is required for SharePoint to pick up the newly found crawled properties. SharePoint generates synthetic names (e.g. *»Category 1«, »Category 2«, etc.)* for all new property categories. All crawled properties with the same GUID are collected in the same new category. You can determine the common GUID by looking at any of the crawled properties in a category. In the SharePoint administration website under *Application Management* click *Manage service applications*, *Search Service Application*. On the page *Search Service Application: Search Administration* on the left-hand side click *Search Schema*. This will display the page *Search Service Application: Managed Properties*. Click on *Categories* at the top of the page, and click on a category with a new synthetic name. Click on any crawled property and look at the Property Set ID to determine the GUID.Compare the GUID to those in *PropertySet/@guid* in the XML configuration file for TET PDF IFilter (if you created custom properties) or the prop-

erty set GUIDs in Appendix A, »Predefined Metadata Properties« (if you use properties from the predefined property set collections).

- Rename the synthetic category names created by SharePoint with user-friendly names after identifying the categories by the GUIDs determined in the previous step:
  SharePoint: In the *Category* page click the drop-down menu for a category name and click *Edit Category*. Edit the name in the *Category name* field as desired and click OK.
- SharePoint: In the *Search Service Application* administration page click on *Managed Properties*.
- Click *New Managed Property* (see Figure 3.2).
- Enter the property's name, description, and data type, and click the button *Add Mapping* which is located in the lower right corner to the right of the field *Mapping to crawled properties*.
- The drop-down menu *Filter on a category* will filter the list of crawled properties which are displayed in the list box titled *Crawled property selection*. The list box displays a list of properties along with their category name and property identifier or friendly name. The choice of identifier or friendly name depends on the property definition in the XML configuration file of TET PDF IFilter, see »Configure custom metadata properties«, page 39. Select a newly crawled property, assign a name to the managed property, and save it.

Preparing SharePoint XML for searches on managed properties. In this step you will prepare the required XML for adding managed properties to Advanced Search. The next section describes how to actually apply this XML. We describe the XML with examples below:

Fig. 3.2
Adding managed properties in SharePoint



For each property you must create a *PropertyDef* element as child of the *PropertyDefs* element:

```
<PropertyDef Name="EbookDateOfBirth" DataType="datetime" DisplayName="Ebook Date of
Birth"/>
```

The *Name* attribute must correspond to the property name, the *DataType* attribute describes the property's type according to Table 3.3, and *DisplayName* contains an arbitrary name which will be presented in the user interface.

In order to make the new managed property available for searches on PDF documents, add the property to the *ResultType* element:

```
<ResultType DisplayName="PDF Documents" Name="pdfdocuments">
        <Query>FileExtension='pdf'</Query>
        <PropertyRef Name="Author"/>
        <PropertyRef Name="Description"/>
        <PropertyRef Name="FileName"/>
        <PropertyRef Name="Size"/>
        <PropertyRef Name="Path"/>
        <PropertyRef Name="Created"/>
        <PropertyRef Name="Write"/>
        <PropertyRef Name="CreatedBy"/>
        <PropertyRef Name="ModifiedBy"/>
        <PropertyRef Name="EbookDateOfBirth"/>
</ResultType>
```

Table 3.3 Property data types for SharePoint

data type in TET PDF IFilter	*data type for SharePoint*
Int32	*integer*
Double	*decimal*
Boolean	*boolean*
DateTime	*datetime*
String	*text*

A complete XML file *(starter_advanced_search.xml)* for all properties in the starter set is installed with TET PDF IFilter. It has been created with Microsoft's Virtual PC image for SharePoint evaluation. You can use the XML for evaluation and testing. Note, however, that XML for production sites must carefully be constructed based on the existing XML of the site and the newly configured properties.

Search for custom metadata properties. In order to implement advanced metadata search you must configure TET PDF IFilter to index the property (see Section 2.5, »Custom Properties«, page 24). This will instruct TET PDF IFilter to create a crawled property. Then you must make a new managed property available in the *Advanced Search* page as follows:

- Log on to the SharePoint site at and navigate to *Advanced Search*. This is the search form where we want to add a new managed property under *Add property restrictions...* at the bottom of the page.

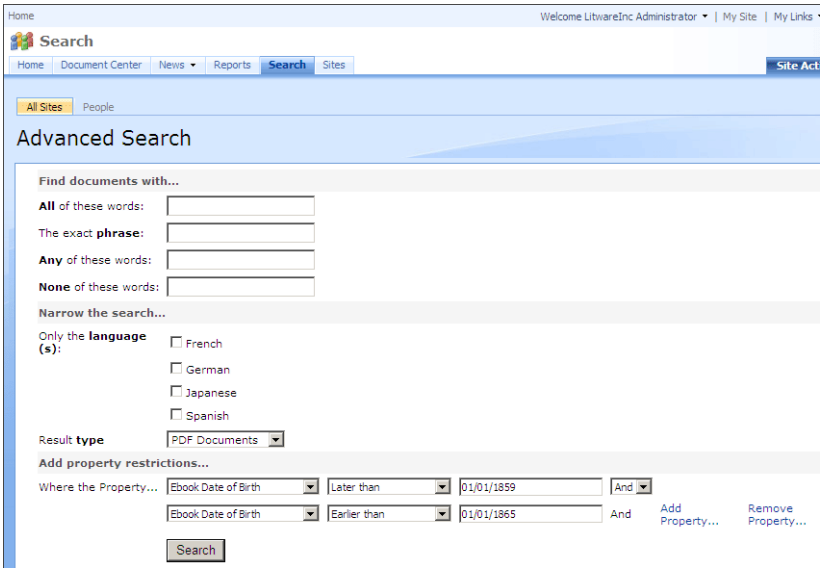
- In the upper-right corner of the page, click *Site Actions*. In the drop-down menu that opens, click *Edit Page*. This will change the look of the page and adds controls for modifying the page.
- In the upper-right corner of the box titled *Advanced Search Box*, click *edit*. In the drop-down menu that opens, select *Modify Shared Web Part*. This surrounds the *Advanced Search Box* with a dashed line, and to the right a new box appears which is also titled *Advanced Search Box*. It contains entries *Search box*, *Scopes*, *Properties*, etc.
- Expand the *Properties* category and look for the field *Properties* which contains XML. Click into the field, and a button with three dots inside and a tool tip *Click to use builder* will appear. Clicking on the button will open a text entry box with XML. This XML must be edited according to the example below; you may want to copy the XML to an XML editor and paste it back after editing.
- After closing the text edit box, click *Ok* at the bottom of box where the editing took place, and click *Check In to Share Draft* at the top of the page. This brings you back to the normal look of the *Advanced Search* form, and if *All Results* is selected in the *Result type* drop-down menu, the new property name will appear in the *(Pick Property)* menu at the bottom of the page under *Add property restrictions...*. It may also be necessary to click *Publish* at the top of the *Advanced Search* form in addition to clicking *Check In to Share Draft* to make the modified form available to all users.

Now you can search for documents with specific entries in the managed property. Figure 3.3 shows the Advanced Search page with the custom property *Ebook Date of Birth* in the lower section.

Note *If you suspect that individual documents are missing from the list of search results, SharePoint may have removed them as duplicates. In order to disable duplicate removal click the »View duplicates« link on the Search Results page.*

Search for metadata properties. Table 3.4 contains examples for property queries. Property queries are constructed according to the following simple scheme:

Fig. 3.3 Advanced Search Page in SharePoint with custom property



```
<property name>:<value>
```

The syntax description for property-based queries can be found at *msdn.microsoft.com/en-us/library/office/ff394509%28v=office.14%29.aspx*

Table 3.4 Metadata query examples for SharePoint

Search term example	***Explanation***
`author:Doyle`	*author contains* `Doyle`
`author:Arthur author:Doyle`	*author contains* `Arthur` *and* `Doyle`
`author:"Arthur Conan Doyle"`	*author contains the exact text* `Arthur Conan Doyle`

3.3 Metadata Properties in SQL Server

SQL Server does not support metadata property indexing or searching. By default, all properties will therefore be ignored. In order to implement metadata queries we recommend to index metadata properties as text (see Section 2.7, »Index Properties as Text«, page 32).

XML configuration for SQL Server. Table 3.5 lists requirements and recommendations related to the XML configuration for TET PDF IFilter when working with SQL Server. The sample configuration file *SQL Server config.xml* implements these requirements and can be used as a starting point.

Table 3.5 XML configuration for SQL Server

element	*attribute*	*requirements and recommendation*
Filtering	***metadataHandling***	*Set this attribute to* `propertyAndText` *or* `propertyAndPrefixedText` *to enable indexing properties as text (see Section 2.7, »Index Properties as Text«, page 32).*
Property	***textIndexPrefix***	*Set the prefix if you want to explicitly search for properties.*

Search for metadata properties. Since there is no dedicated support for searching properties in SQL server, you must query for metadata properties in a full-text query. Once you enabled indexing of properties as text, you can search for documents with the author *Arthur Conan Doyle* as follows:

```
SELECT name FROM DocumentTable WHERE CONTAINS(*,'"TET_System_Author_Arthur Conan Doyle"')
GO
```

Use the following statement to query for documents where the author starts with *Arthur:*

```
SELECT name FROM DocumentTable WHERE CONTAINS(*,'"System_Author_Arthur*"')
GO
```

4 Advanced PDF Indexing

4.1 PDF Versions and Protected Documents

Supported PDF input. TET PDF IFilter has been tested against millions of PDF test files from various sources. It accepts PDF 1.0 up to PDF 1.7 extension level 8 corresponding to Acrobat DC as well as PDF 2.0 (ISO 32000-2) including encrypted documents. TET PDF IFilter attempts to repair various kinds of malformed and damaged PDF documents.

By default, TET PDF IFilter extracts text from the page contents, standard document info entries, annotations, bookmarks and form fields and processes file attachments and PDF portfolios. This behavior can be modified, see Section 4.2, »PDF Document Domains«, page 47).

Note *TET PDF IFilter does not support dynamic XFA forms. XFA is a separate format which is not part of the PDF standard ISO 32000. Since XFA is packaged inside a small PDF wrapper XFA forms are often confused with PDF documents although XFA is actually a completely different file format which requires dedicated software.*

Protected PDF documents. TET PDF IFilter indexes text and metadata from all documents as long as it can open it. This includes the following kinds of PDF documents:

- Unencrypted documents;
- Documents which are encrypted with a master password, but do not require any user password. The status of Acrobat's security setting *Content Copying Allowed/Not Allowed* does not affect documents in this group.

At first glance the second category may look like a violation of the document author's intention for protecting the document. However, it is not, since TET PDF IFilter does not provide any means for actually copying the text; it merely helps the search engine with indexing the document and subsequently locating the document in a search. Once the document is identified in a search and opened in Acrobat, it is still subject to any restrictions regarding content copying which may have been specified for the document.

Encrypted PDF documents which can not be opened are logged with logging level 2 (see Section 6.4, »Debugging Facilities«, page 71). This category includes the following cases:

- Encrypted documents which require a user password, i.e. those which cannot be opened in Acrobat without supplying the corresponding password.
- Encrypted PDF attachments in otherwise unprotected documents.
- Documents which have been encrypted with a user-specific public-key certificate.

Damaged PDF documents. PDF documents may contain damaged data structures, either because of faulty PDF generation software or because of some accidental modification (e.g. caused by failed network transfer). TET PDF IFilter automatically detects damaged PDF documents and attempts to repair such documents in order to allow for successful extraction of text and metadata. This repair mode works automatically as part of the indexing process. In some cases this mode may not be sufficient, and TET PDF IFilter must process the document with a more thorough repair mode. Since it is

more time-consuming, this forced repair mode is only applied for severely damaged PDFs which cannot successfully be processed in automatic repair mode.

If a document can be opened successfully, but contains one or more damaged pages, these pages will be ignored and processing continues with subsequent pages. For each ignored page an entry will be written to the application event log.

4.2 PDF Document Domains

PDF documents may contain text in many other places than only the page contents, such as in annotations and bookmarks. They also make use of metadata in Adobe's XMP form or as classical document info entries. The places in a PDF document which may contain text are referred to as PDF document domains. The list below describes all PDF document domains along with notes how to display the corresponding text in Acrobat. The list also contains the default actions of TET PDF IFilter for all document domains. In short, TET PDF IFilter indexes the text in all relevant locations. As a result, you may get search hits for documents where it is not obvious at first glance why a hit is produced. Since search term highlighting is generally not available in IFilter clients, it is important to know how to locate the search term in the result documents. Remember that the searched text may be present in a location different from the actual page contents, and refer to the list below if you have trouble locating the search text in a PDF document where TET PDF IFilter reports a search hit.

Notes regarding the descriptions below:

- Searching »Multiple PDFs« with Acrobat refers to the following kind of search: *Edit, Advanced Search*, click *Show More Options* (if present) and in the *Look In:* pull-down select a folder of PDF documents (see Figure 4.1).
- Some of the descriptions refer to the property set collections *documentXMP, imageXMP, shell, pdf*, and *internal*. These can be enabled in the XML configuration file (see Section 2.3, »Predefined Properties«, page 22). By default, the *shell, pdf* and *internal* property set collections are enabled, while the *documentXMP* and *imageXMP* property set collections are disabled. See Section 2.3, »Predefined Properties«, page 22, for more details on property set collections.
- The notation *Filtering@indexNestedPdf* refers to an attribute in the XML configuration file (see Section 5.2, »XML Elements and Attributes«, page 61).

In the remaining section we provide information on PDF domain searching. In addition, we summarize how to search these document domains with Acrobat DC. This is important to locate search hits in Acrobat.

Text on the page. Page contents are the main source of text in PDF. Text on a page is rendered with fonts and encoded using one of the many encoding techniques available in PDF.

- How to display with Acrobat: page contents are always visible
- How to search a single PDF with Acrobat DC: *Edit, Find* or *Edit, Advanced Search*. TET PDF IFilter may be able to process the text in documents where Acrobat does not correctly map glyphs to Unicode values. In this situation you can use the TET Plugin which is based on TET. The TET Plugin offers its own search dialog via *Plug-Ins, PDFlib TET Plugin... TET Find*. However, it is not intended as a full-blown search facility.
- How to search multiple PDFs with Acrobat DC: *Edit, Advanced Search* and in *Where would you like to search?* select *All PDF Documents in*, and browse to a folder with PDF documents.
- TET PDF IFilter: page contents are indexed by default. However, in special situations you may want to disable page content indexing with *Filtering@indexPageContents="false"*.

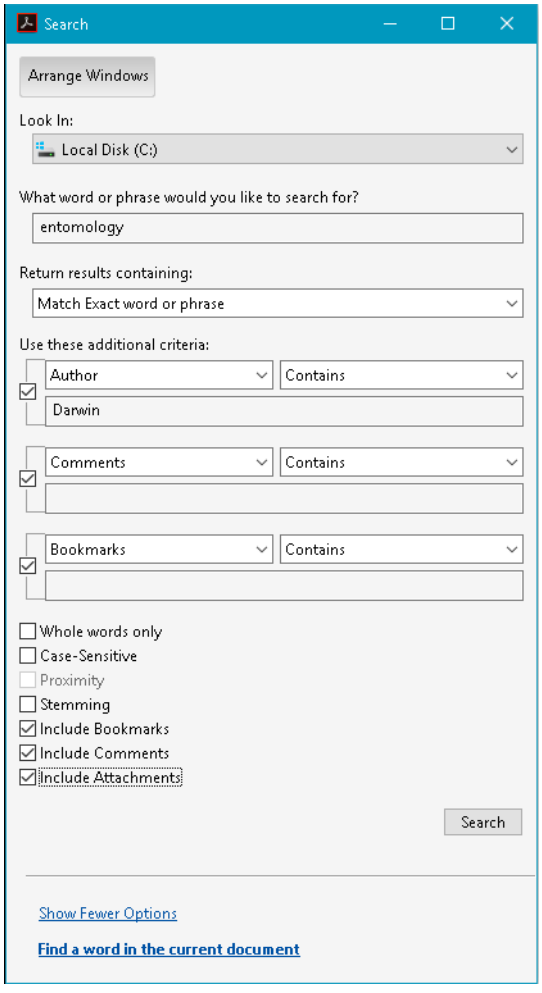


Fig. 4.1
Acrobat's advanced search dialog

Predefined document info entries. standard document info entries are key/value pairs.

- How to display with Acrobat DC: *File, Properties...*
- How to search a single PDF with Acrobat DC: not available
- How to search multiple PDFs with Acrobat DC: click *Edit, [Advanced] Search* and *Show More Options* near the bottom of the dialog. In the *Look In:* pull-down select a folder of PDF documents and in the pull-down menu *Use these additional criteria* select one of *Date Created, Date Modified, Author, Title, Subject, Keywords.*
- TET PDF IFilter: predefined document info entries are indexed if *PropertySetCollection/@shell="true"* (which is default)

Custom document info entries. Custom document info entries can be defined in addition to the standard entries.

- How to display with Acrobat DC: *File, Properties..., Custom* (not available in Acrobat Reader)
- How to search with Acrobat DC: not available

- TET PDF IFilter: custom document info entries are indexed if custom properties are defined based on this document info entry

XMP metadata on document level. XMP metadata consists of an XML stream containing extended metadata.

- How to display with Acrobat DC: *File, Properties..., Description, Additional Metadata.*. (not available in Acrobat Reader)
- TET PDF IFilter: document XMP metadata indexed if *PropertySetCollection/@documentXMP="true"* or custom properties are defined based on document XMP

XMP metadata on image level. XMP metadata can be attached to document components, such as images, pages, fonts, etc. However, XMP is commonly only found on the image level (in addition to document level).

- How to display with Acrobat DC: *View, Show/Hide, Navigation Panes, Content*. Locate the image in the tree structure, right-click on it and select *Show Metadata...* . (not available in Acrobat Reader)
- How to search with Acrobat DC: not available
- TET PDF IFilter: XMP image metadata is indexed if *PropertySetCollection/@imageXMP="true"*

Text in form fields. Form fields are displayed on top of the page. However, technically they are not part of the page contents, but represented by separate data structures.

- How to display with Acrobat DC: *Tools*, *Prepare Form* (not available in Acrobat Reader)
- How to search with Acrobat DC: Acrobat searches the visible contents of form fields
- TET PDF IFilter: form field values are indexed if *Filtering@indexFormFields="true"* (which is default).
  The following additional aspects of form fields are indexed if *Filtering@indexExtendedFormFields="true"*: default value, list items, tooltips.
  If the main page contents (i.e. the form field captions) are not required because they are constant you can disable page content indexing with *Filtering@indexPageContents= "false"*.

Text in comments (annotations). Similar to form fields, annotations (notes, comments, etc.) are layered on top of the page, but are represented by separate data structures. The interesting text contents of an annotation depend on its type. For example, for Web links the interesting part may be the URL, while for other annotation types the visible text contents may be relevant.

- How to display with Acrobat DC: *Tools, Comment, Comments List*
- How to search a single PDF with Acrobat DC: *Edit, Search* and check the box *Include Comments*, or use the *Search Comments* button on the Comments List toolbar
- How to search multiple PDFs with Acrobat DC: click *Edit, Advanced Search* and *Show More Options*. In the *Look In:* pull-down select a folder of PDF documents and in the pull-down menu *Use these additional criteria* select *Comments*.
- TET PDF IFilter: comments (both appearance streams and *Contents* entries) are indexed if *Filtering@indexAnnotations="true"* (which is default).

Text in bookmarks. Bookmarks are not directly page-related, although they may contain an action which jumps to a particular page. Bookmarks can be nested to form a hierarchical structure.

- How to display with Acrobat DC: *View, Show/Hide, Navigation Panes, Bookmarks*
- How to search a single PDF with Acrobat DC: *Edit, Advanced Search* and check the box *Include Bookmarks*
- How to search multiple PDFs with Acrobat DC: click *Edit, [Advanced] Search* and *Show More Options*. In the *Look In:* pull-down select a folder of PDF documents and in the pull-down menu *Use these additional criteria* select *Bookmarks* (not available in Acrobat Reader)
- TET PDF IFilter: form field values are indexed if *Filtering@indexBookmarks="true"* (which is default).

File attachments. PDF documents may contain file attachments (on document or page level) which may themselves be PDF documents.

- How to display with Acrobat DC: *View, Show/Hide, Navigation Panes, Attachments*
- How to search with Acrobat DC: Use *Edit, AdvancedSearch* and check the box *Include Attachments* (not available in Acrobat Reader). Nested attachments are not searched recursively.
- TET PDF IFilter: PDF attachments are indexed recursively if *Filtering@indexNested-Pdf="true"* (which is default)

PDF packages and portfolios. PDF packages and PDF portfolios are file attachments with additional properties.

- How to display with Acrobat DC: Acrobat presents the cover sheet of the package/portfolio and the constituent PDF documents with dedicated user interface elements for PDF packages.
- How to search a single PDF package with Acrobat DC: *Edit, Search Entire Portfolio*
- How to search multiple PDF packages with Acrobat DC: not available
- TET PDF IFilter: PDF documents contained in packages/portfolios are indexed recursively if *Filtering@indexNestedPdf="true"* (which is default)

PDF standards and other PDF properties. This domain does not explicitly contain text, but is used as a container which collects various intrinsic properties of a PDF document, e.g. PDF/X and PDF/A status, Tagged PDF status, etc.

- Acrobat DC: *View, Show/Hide, Navigation Panes, Standards* (only present for standard-conforming PDFs)
- How to search with Acrobat DC: not available
- TET PDF IFilter: PDF properties are indexed if *PropertySetCollection/@pdf="true"* (which is default). Conformance to PDF/A and other standards can be queried with the properties *PDFlib.TET.pdfa* etc.

Tagged PDF and Artifacts. TET reconstructs the layout structure and hierarchy directly from the page contents without using the structure tree which is present in Tagged PDF documents. Text and images which are not required to understand the document but rather are generated for layout purposes or decoration may be marked as Artifacts in Tagged PDF. The most common use of Artifacts is for running headers and footers including page numbers and chapter titles. Depending on the use case it may or may not be desirable to process page contents which are marked as Artifacts.

- How to display with Acrobat DC: *View, Show/Hide, Navigation Panes, Tags;* in the Tags menu click *Find...* and select *Artifacts*. Text, images and vector graphics which are marked as Artifact are highlighted.

Alternatively, you can activate *Tools, Accessibility, Touch Up Reading Order*. This tool highlights the tagged contents on the page with shaded rectangles. Contents which are not highlighted represents Artifacts.

- How to ignore Artifacts when searching with Acrobat DC: not available
- How to ignore Artifacts with TET PDF IFilter: provide the page option *ignoreartifacts*.

Layers. Using layers (technically known as optional content) the page contents can be made visible or invisible. Depending on the use case it may or may not be desirable to process page contents on invisible layers.

- How to display with Acrobat DC: *View, Show/Hide, Navigation Panes, Layers:* layers which are currently visible have an eye symbol in front of the name. Clicking on this symbol controls the visibility of a layer.
- How to search with Acrobat DC: Acrobat searches the contents of all layers. If a search result is found on an invisible layer, Acrobat offers to make the layer visible.
- How to process layers with TET PDF IFilter: the page option *layers=all/visible/invisible* can be used to restrict content extraction to either visible or invisible layers. Alternatively, the contents of all layers can be processed which only makes sense if the layers don't overlap. The default is *visible*.

4.3 Automatic Language Detection

Importance of proper language detection. Several aspects of content indexing can significantly be improved by language-specific postprocessing of the indexed content. While the details vary among different IFilter clients, the following aspects are highly language-dependent:

- A component called wordbreaker is used to break the text content into words.
- Word stemmers extract the base form of an indexed word. This way search queries produce hits even if the search term is used in a slightly modified (inflected) form in the indexed document.
- Thesaurus-based search
- Noise word lists contain words which are considered irrelevant for searches, and are therefore not included in the index, e.g. *the, a, and, at, on*.

Automatic language detection. The features mentioned above can only be implemented if the natural language is known in which the text has been written. By default, IFilter clients will use the system locale for all language-specific processing. As a consequence, indexing Japanese documents on an English system will create mediocre index contents and therefore searches can fail to locate existing target documents. Assigning the proper natural language is especially important for documents with East-Asian language content.

In order to improve all aspects of language-specific processing during the indexing process, TET PDF IFilter automatically determines the natural language for text in the document contents and properties of type *string*. TET PDF IFilter deploys two techniques for determining the language:

- In some cases checking the script (writing system) is sufficient for determining the language. For example, Japanese Hiragana characters are exclusively used for writing text in the Japanese language.
- Many languages share the same writing system with other languages. For example, most Western languages are written in the Latin script. In these cases TET PDF IFilter applies statistical analysis to determine the appropriate language.

Since automatic language detection analyzes each text chunk separately, a document may contain arbitrary mixtures of languages. Each segment will be assigned the corresponding language as follows:

- Automatic script detection splits the text in chunks consisting of the same script.
- If the script does not uniquely identify the language, automatic language detection is used to analyze the text. This process assigns the most likely language for the whole chunk, but will not further split the chunk into smaller chunks for each language. For example, a chunk with Latin text in the English, German, and French languages will be assigned one of these languages for the whole chunk, depending on the actual distribution of the languages.

Manual language assignment. Automatic language detection in TET PDF IFilter can be customized for special applications. This configuration feature makes use of LCIDs (Locale Identifiers). LCIDs specify the natural language and can also distinguish between different national or regional language variants (e.g. UK English vs. US English). Table 4.1 lists some commonly used LCID values. A complete list of LCIDs can be found at

msdn.microsoft.com/en-us/library/ms776294(VS.85).aspx

Note *TET PDF IFilter does not apply any language-specific processing beyond language detection. It is up to the IFilter client to use the LCID information. While some IFilter clients (e.g. SharePoint, SQL Server) include sophisticated LCID treatment, other IFilter clients may completely ignore the LCID information.*

Table 4.1 Common LCID values and the corresponding primary and secondary language

LCID	primary language	secondary language (country)
0x0000	Neutral locale language	Neutral sublanguage
0x0401	Arabic (ar)	Saudi Arabia (SA)
0x0404	Chinese (zh)	Traditional (Hant)
0x0407	German (de)	Germany (DE)
0x0409	English (en)	United States (US)
0x040c	French (fr)	France (FR)
0x0410	Italian (it)	Italy (IT)
0x0411	Japanese (ja)	Japan (JP)
0x0413	Dutch (nl)	Netherlands (NL)
0x0419	Russian (ru)	Russia (RU)
0x0804	Chinese (zh)	Simplified (Hans)
0x0c0a	Spanish (es)	Spain (ES)
0x0800	System default locale language	
0x1000	Unspecified custom locale language	Unspecified custom sublanguage

XML configuration for LCIDs. LCIDs for overriding or supplementing automatic LCID detection can be specified in the *LocaleId* element of the XML configuration file:

```
<LocaleId default="1031" detection="auto" latin="1031" cyrillic="1026" chinese="4100"/>
```

The *detection* attribute can have the values *auto*, *disabled,* and *script*. All other attributes except *default* will be ignored if *detection=disabled*. Default is *auto*. The *script* setting activates script analysis, but disables statistical analysis.

The *default* attribute can be used to specify a global LCID setting which will be used for all text if *detection=disabled*. If this attribute is missing, the system locale will be used.

For all attributes except *detection* a numeric value in decimal or hexadecimal syntax can be specified. Hexadecimal values must start with *0x*. Table 4.2 lists the supported script attributes and their default values. LCIDs for text in all other scripts will be assigned automatically.

Table 4.2 Attributes for specifying script-specific locale IDs (LCIDs)

attribute	***default value***
default	*0x0800 Current system locale (will be used for all text chunks if* `detection=disabled`)
latin	*0x0409 English (US)*
cyrillic	*0x0419 Russian (RU)*
arabic	*0x0401 Arabic (SA)*
chinese	*0x0804 Chinese (People's Republic of China)*

4.4 Unicode Postprocessing

The TET kernel which implements the underlying PDF text extraction engine for TET PDF IFilter provides extensive controls for Unicode postprocessing. These are discussed in detail in the TET manual which is included in the TET PDF IFilter package. Below we mention the most important features.

Unicode postprocessing features are controlled by TET document options. In TET PDF IFilter these options can be provided in the *DocOptions* element of the XML configuration file, e.g.

```
<Tet>
        <DocOptions>decompose={canonical=_all}</DocOptions>
        <PageOptions/>
        <TetOptions/>
</Tet>
```

Unicode Folding. Foldings process one or more Unicode characters and apply a certain action on each of the characters. The following actions are available:

- preserve the character;
- remove the character;
- replace it with a another (fixed) character.

Foldings are not chained: the output of a folding will not be processed again by the available foldings. Foldings affect only the Unicode text output, but not the set of glyphs reported in the *TET_char_info* structure or the <*Glyph*> elements in TETML. For example, if a folding removes certain Unicode characters, the corresponding glyphs which created the initial characters will still be reported.

The following option list specifies multiple foldings:

```
<Tet>
        <DocOptions>fold={ {[:blank:] U+0020 } {_dehyphenation remove} }</DocOptions>
</Tet>
```

Folding examples can be found in the TET manual. The TET manual also documents the default foldings which are applied by TET PDF IFilter.

Unicode Decomposition. Decompositions replace a character with an equivalent sequence of one or more other characters.[1] A Unicode character is called (either compatibility or canonical) equivalent to another character or a sequence of characters if they actually mean the same, but for historical reasons (mostly related to round tripping with legacy encodings) are encoded separately in Unicode. Decompositions destroy information. This is useful if you are not interested in the difference between the original character and its equivalent. If you *are* interested in the difference, however, the respective decomposition should not be applied.

Note *The term »decomposition« is used here as defined in the Unicode standard, although many decompositions do not actually split a character into multiple parts, but convert a single character to another character.*

1. For a full discussion of Unicode decomposition see www.unicode.org/versions/Unicode8.0.0/ch02.pdf (section 2.12) and www.unicode.org/versions/Unicode8.0.0/ch03.pdf (section 3.7).

Canonical decomposition. Characters or character sequences which are canonically equivalent represent the same abstract character and should therefore always have the same appearance and behavior. Common examples include precomposed characters (e.g. Ä U+00C4) vs. combining sequences (e.g. A U+0041 ¨ U+0308): both representations are canonically equivalent. Switching from one representation to the other does not remove information. Canonical decompositions replace one representation with another which is considered the canonical representation.

In the Unicode code charts (see *www.unicode.org/charts/)* (but not the character tables) canonical mappings are marked with the symbol IDENTICAL TO ≡ U+2261 . The decomposition name <*canonical*> is implicitly assumed.

The following document option maps all canonical equivalents to their equivalent counterparts:

```
<Tet>
        <DocOptions>decompose={canonical=_all}</DocOptions>
</Tet>
```

Table 4.3 Canonical decomposition: suboption for the decompose *option (canonically equivalent characters are marked with the symbol IDENTICAL TO* ≡ U+2261 *in the Unicode code charts)*

decomposition name	*description*	*before decomposition*	*after decomposition*
canonical[1]	*Canonical decomposition*	À U+00C0	A U+0041 ` U+0300
		林 U+F9F4	林 U+6797
		Ω U+2126	Ω U+03A9
		ば U+3070	は U+306F ゙ U+3099
		אַ U+FB2F	א U+05D0 ַ U+05B8

1. By default this decomposition is not applied to all characters in order to preserve certain characters; see the TET manual for details.

Compatibility decomposition. Characters which are compatibility equivalent represent the same abstract character, but may differ in appearance or behavior. Examples include isolated forms of Arabic characters (e.g. س U+0633) vs. context-specific shaped forms (e.g. ﺱ U+FEB2 , ﺳ U+FEB4 , ﺲ U+FEB3). Compatibility equivalent characters differ in formatting. Removing this formatting information implies loss of information, but may simplify processing for certain types of applications (e.g. searching). Compatibility decompositions remove the formatting information.

In the Unicode code charts compatibility mappings are marked with the symbol ALMOST EQUAL TO ≈ U+2248 , followed by the decomposition name (or »tag«) in angle

brackets, e.g. <*noBreak*>. If no tag name is provided, <*compat*> is assumed. The tag names are used as option names in the *decompose* option list. As can be seen in some of the examples, the result of a decomposition may convert a single character to a sequence of multiple characters.

Note *Keep in mind that PDF documents may already map glyphs to the decomposed sequence instead of the non-decomposed Unicode value. In this situation the* `decompose` *option will not affect the output.*

Decomposition examples can be found in the TET manual. The decomposition names in the *decompose* option list (e.g. *font, circle, initial, vertical)* and the default decompositions are also listed in the TET manual.

Unicode Normalization. The Unicode standard defines four normalization forms which are based on the notions of canonical equivalence and compatibility equivalence.[1] All normalization forms put combining marks in a specific order and apply decomposition and composition in different ways:

- Normalization Form C (NFC) applies canonical decomposition followed by canonical composition. For example, the composed form C stores *Ä* as a single character Ä (U+00C4). NFC is the preferred format for Unicode text in Windows, on the Web and in most databases.
- Normalization Form D (NFD) applies canonical decomposition. For example, the decomposed form D stores *Ä* as a sequence A ¨ (U+0041 U+0308) of base character and combining diacritical character.
- Normalization Form KC (NFKC) applies compatibility decomposition followed by canonical composition. In other words, some characters are mapped to compatible basic forms, e.g. the ligature fi (U+FB01) is mapped to the sequence f i (U+0066 U+0069).
- Normalization Form KD (NFKD) applies compatibility decomposition. This is similar to form KC, but does not apply canonical composition.

The choice of normalization form depends on the application's requirements.

The normalization forms are specified in Unicode Standard Annex #15 »Unicode Normalization Forms« (see *www.unicode.org/versions/Unicode5.2.0/ch03.pdf#G21796* and *www.unicode.org/reports/tr15/).*

TET PDF IFilter supports all four Unicode normalization forms. Unicode normalization can be controlled via the *normalize* document option, e.g.

```
<Tet>
        <DocOptions>normalize=nfc</DocOptions>
</Tet>
```

TET PDF IFilter does not apply normalization by default. Because of the possible interaction between the *decompose* and *normalize* options, setting the *normalize* option to a value different from *none* disables the default decompositions.

1. The normalization forms are specified in Unicode Standard Annex #15 »Unicode Normalization Forms« (see www.unicode.org/versions/Unicode8.0.0/ch03.pdf#G21796 and www.unicode.org/reports/tr15/).

4.5 Custom Glyph Mapping Tables

Although many workarounds are implemented in TET PDF IFilter, in certain rare cases it may be unable to correctly extract text from PDF documents if crucial information for Unicode mapping is missing from the document. If you have many documents with similar properties (e.g. created with the same software and set of fonts) you can provide auxiliary Unicode mapping tables which can be used to extract text from PDF documents which otherwise could not be indexed.

TET PDF IFilter supports various table formats which are detailed in the documentation of the PDFlib TET product. You can also use the free PDFlib FontReporter and PDFlib TET Plugins for Adobe Acrobat to try such mapping tables. Unicode mapping tables must be configured with appropriate document options in the configuration file, and can be placed in the *resource* directory of the TET PDF IFilter installation directory.

XML configuration for TET options. Option lists for controlling operation of the TET core (e.g. for using custom glyph mapping tables) must be constructed according to the option list syntax described in the TET product documentation, and can be supplied to TET PDF IFilter in the *DocOptions*, *PageOptions*, and *TetOptions* elements of the XML configuration file, which are all children of the *Tet* element:

```
<Tet>
        <DocOptions>glyphmapping {{fontname=T* glyphlist={tex}}}</DocOptions>
        <PageOptions/>
        <TetOptions>searchpath={C:/glyphlists}</TetOptions>
</Tet>
```

5 XML Configuration File

5.1 Working with Configuration Files

Operation of TET PDF IFilter can be controlled with an XML configuration file. Sample configuration files are installed with TET PDF IFilter. Several predefined configuration files are installed with TET PDF IFilter:

- The file *default_config.xml* describes the internal default settings of TET PDF IFilter. It may serve as a starting point for a customized configuration file.
- The file *starter_samples.xml* contains property definitions which can be used with the property samples discussed in Section 2.6, »Examples with Custom Properties«, page 27.
- Additional XML configuration files are available for use with Windows Search, SharePoint, or SQL Server. One of these configuration files can be registered upon user request during the installation process.

Specifying the location of the XML configuration file. The configuration file must be specified in the following registry key which contains a String value with the full path name of the configuration file:

```
HKEY_LOCAL_MACHINE\SOFTWARE\PDFlib\TET PDF IFilter5\configfile
```

Note *It is recommended to place customized XML configuration files outside the installation directory of TET PDF IFilter. This way the configuration survives if the installation directory changes, e.g. after installing an updated version of TET PDF IFilter.*

If this registry entry does not exist or contains an empty string, the default configuration is used. If the configuration file specified in the registry entry cannot be opened or XML parsing of the configuration file fails, an error message is written to the event log and no indexing is performed.

Only a single configuration file can be used for TET PDF IFilter on a machine. However, the 32-bit and 64-bit versions can use different configuration files on the same machine since the registry entry is searched in the 32-bit or 64-bit registry, respectively.

If you changed the configuration file you must rebuild the index for the changes to become active.

XML namespace and schema description. Table 5.1 lists the elements and attributes which are available in the XML configuration file. An XSD schema description for the XML configuration language is installed with TET PDF IFilter, and can also be found at the URI given in the schema file. You can use the schema file with a suitable XML editor to make sure that the generated XML configuration file adheres to the syntax expected by TET PDF IFilter.

Custom data types for XML elements and attributes. Except where a value description is provided, all elements are empty. The following custom data types are used in the XML configuration file:

- LCID: hexadecimal or decimal locale identifier; see

  msdn.microsoft.com/en-us/library/ms776294(VS.85).aspx

The value *0x0800* is translated to the current system default locale.

- GUID *(Globally Unique IDentifier)*, also known as UUID *(Universally Unique IDentifier)*: unique 128-bit identifier in case-insensitive hexadecimal notation according to RFC 4122. The parts must be separated by dash characters »-«. There are various tools available for creating GUIDs, e.g. the online service at

  www.uuidgenerator.net/

  A sample GUID looks as follows: *7a737220-0cd0-11dd-bd75-0002a5d5c51b*.
- pCOS path: extended pCOS path describing a PDF object, see pCOS Reference and the pCOS extensions described in Section »Extended pCOS paths«, page 19
- Option list: string containing an option list according to the syntax specified in the *PDFlib TET Reference Manual*.
- Language identifier: XMP language qualifier according to RFC 1766, or *x-default* which identifies the default language in the document.

5.2 XML Elements and Attributes

Table 5.1 contains details for the elements and attributes of the XML configuration file. More detailed information on the effects controlled by the XML configuration file can be found in the respective sections of this manual. The required and recommended settings for specific IFilter clients are listed in the client-specific sections in Chapter 2, »Indexing Metadata Properties«, page 19.

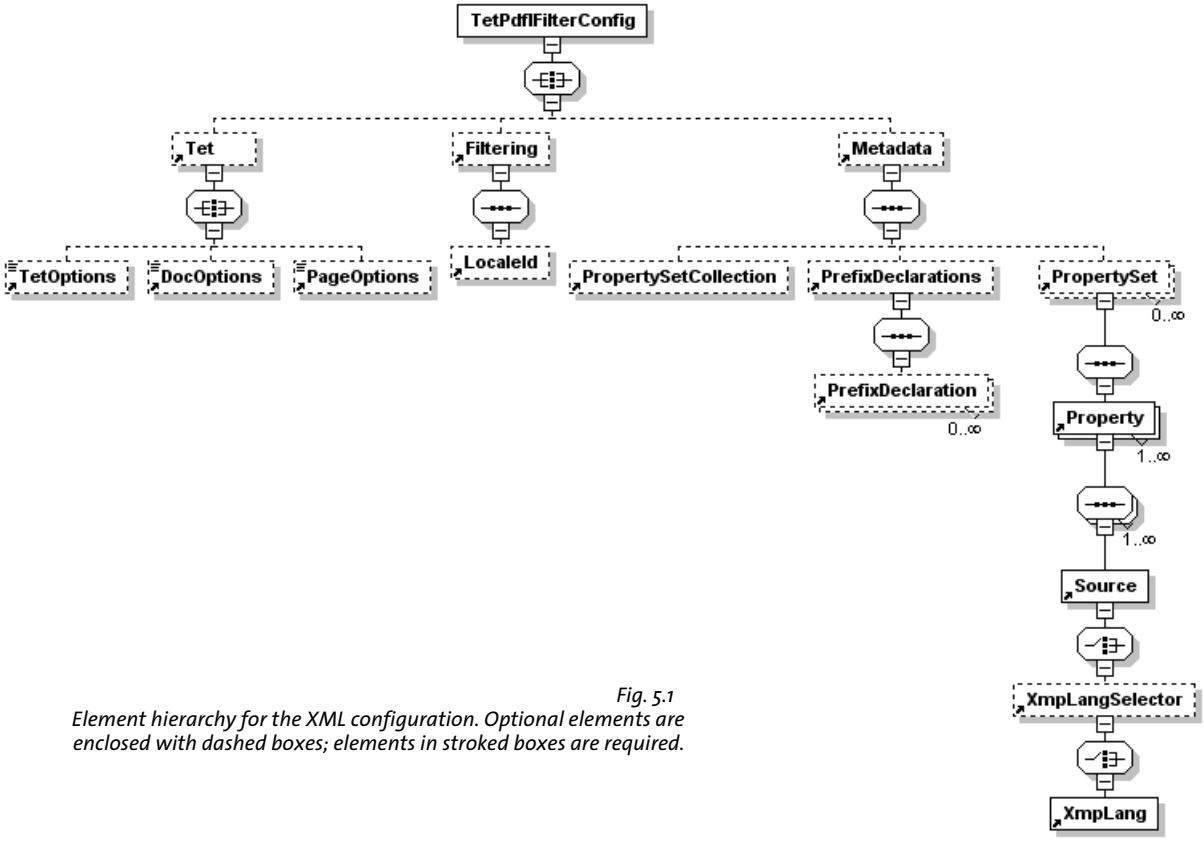


Fig. 5.1
Element hierarchy for the XML configuration. Optional elements are enclosed with dashed boxes; elements in stroked boxes are required.

Table 5.1 XML elements and attributes in the configuration file

element	*description of the element and its attributes*
DocOptions *parent: Tet*	*(May appear zero or one time) Option list for the TET function TET_open_document( ).*
Filtering *parent: TetPdfIFilterConfig*	*(May appear zero or one time) Specify details of the PDF filtering process. Supported attributes:* ***errorIndicator*** *(String; optional) String which is supplied to the IFilter client if a TET function call failed while processing a document. Details regarding the problem may be found in the Windows event log (see »Application event log«, page 71). The error indicator can be useful to identify indexing problems. It is emitted in addition to any (partial) text which may be retrieved from the document. It is recommended to supply a unique string without punctuation characters to make sure that the error indicator doesn't interfere with real index entries, e.g.* `TETPDFIFILTERERROR`. *Default: no error indicator* ***indexAnnotations*** *(Boolean; optional; TET PDF IFilter 5.1) Indicates whether annotations (both appearance streams and* `Contents` *entries) are indexed. Default:* `true` ***indexBookmarks*** *(Boolean; optional; TET PDF IFilter 5.1) Indicates whether bookmarks are indexed. Default:* `true` ***indexExtendedFormFields*** *(Boolean; optional; TET PDF IFilter 5.1) Indicates whether the following aspects of form field are indexed: field name, default value, tooltip, item list of menu fields. Default:* `false` ***indexFormFields*** *(Boolean; optional; TET PDF IFilter 5.1) Indicates whether form field values (i.e. the contents of form fields) are indexed. Default:* `true` ***indexNestedPdf*** *(Boolean; optional) Indicates whether PDF portfolios and attachments are processed recursively (see Section 4.2, »PDF Document Domains«, page 47). Default:* `true` ***indexPageContents*** *(Boolean; optional) Indicates whether the contents of PDF pages are indexed. Disabling page content indexing may be useful in situations where the search is completely based on metadata properties. Default:* `true` ***metadataHandling*** *(Choice; optional) Select the type of metadata handling (see Section 2.7, »Index Properties as Text«, page 32). Default:* `property` ***ignore*** *Drop all metadata properties. This may be useful for debugging or performance optimization in situations where metadata is not required.* ***property*** *Treat metadata as properties.* ***propertyAndPrefixedText*** *In addition to treating metadata as properties, prepend the prefix specified in* `textIndexPrefix` *(if present) for custom properties and the prefixes according to Table 2.2, page 32, for predefined properties, and additionally treat the result as plain text.* ***propertyAndText*** *In addition to treating metadata as properties, treat metadata as plain text.* ***useIdentifier*** *(Boolean; optional) Specify whether* `identifier` *or* `friendlyName` *are used to identify properties if both of these attributes for the* `Property` *element are present. Default:* `true`

Table 5.1 XML elements and attributes in the configuration file

element	***description of the element and its attributes***
LocaleId *parent: Filtering*	*(May appear zero or one time) Configure locale ID detection (see Section 4.3, »Automatic Language Detection«, page 52). Supported attributes:* ***arabic*** *(LCID; optional) LCID for Arabic text. Default: 0x0401 Arabic (SA)* ***chinese*** *(LCID; optional) LCID for Chinese text. Default: 0x0804 Chinese (People's Republic of China)* ***cyrillic*** *(LCID; optional) LCID for Cyrillic text. Default: 0x0419 Russian (RU)* ***default*** *(LCID; optional) Global LCID which will be used for all text chunks if* `detection` *is disabled. Default: 0x0800 (system-locale)* ***detection*** *(Choice; optional) Control automatic LCID detection. Default:* `auto` ***auto*** *Determine LCID based on script and statistical language analysis.* ***disabled*** *Disable LCID detection; all other attributes except* `default` *and* `useCatalogLang` *will be ignored.* ***script*** *Determine LCID based on script.* ***latin*** *(LCID; optional) LCID for Latin text. Default: 0x0409 English (US)* ***useCatalogLang*** *(Boolean; optional) Specify whether the* `Lang` *entry in the document's catalog will be evaluated. If* `true`*, TET PDF IFilter checks the* `Lang` *entry in the PDF document catalog. If present, the* `Lang` *entry will be converted to an LCID. If the conversion is successful the LCID overrides the value of the* `LocaleId/@default` *attribute; if the LCID belongs to one of the Arabic, Chinese, Cyrillic, or Latin scripts it overrides the value of the corresponding attribute of the* `LocaleId` *element. Default:* `true`
Metadata *parent: TetPdfIFilterConfig*	*(May appear zero or one time) Specify metadata properties (see Section 2.5, »Custom Properties«, page 24). If present, this element must appear after* `Filtering` *and* `Tet`*.*
PageOptions *parent: Tet*	*(May appear zero or one time) Option list for the TET function TET_open_page( ).*
PrefixDeclaration *parent: PrefixDeclarations*	*(May appear zero or more times) Declare a namespace prefix which can be used in* `Source/@xmpName`*. Supported attributes:* ***prefix*** *(String which does not contain colon »:« characters; required) Prefix to be used as an abbreviation for the namespace URI.* ***uri*** *(URI; required) Namespace URI*
PrefixDeclarations *parent: Metadata*	*(May appear zero or one time) Declare namespace prefixes for XMP properties in* `xmpName` *attributes of the* `Source` *element.*

Table 5.1 XML elements and attributes in the configuration file

element	***description of the element and its attributes***
Property *parent: PropertySet*	*(May appear one or more times) Specify a metadata property for indexing (see Section 2.5, »Custom Properties«, page 24).* *At least one of* `identifier` *and* `friendlyName` *must be present. If both are supplied,* `identifier` *will be used in the IFilter interface unless* `Filtering/@useIdentifier=false`. *Supported attributes:* ***identifier*** *(Integer >=2; optional) Number which uniquely identifies the property in a* `PropertySet`. ***emitAsVector*** *(Boolean; optional) If* `true`*, the property value is emitted as a single vector entity, regardless of the number of values.* *If* `false`*, the property is emitted as a flat value. If more than one source item was found, multiple flat properties are emitted. Default:* `false` ***friendlyName*** *(String; optional) Name which uniquely identifies the property in a* `PropertySet`. *It can be used to document the property or as an alternative to* `identifier`. *While TET PDF IFilter does not limit the length of property names, Windows imposes a maximum length of 64 characters.* ***precedence*** *(Choice; optional) Specifies precedence for multiple* `Source` *elements (default:* `first-wins`*):* ***first-wins*** *The first non-empty source will be used.* ***try-all*** *All non-empty sources contribute to the property.* ***suppressValue*** *(String; optional; TET PDF IFilter 5.1) If the specified source (single source for* `precedence="first-wins"`*, or all non-empty sources for* `precedence="try-all"`*) delivers the value supplied in this attribute the value is assumed to be non-existent. For properties with* `type="DateTime"` *the value must be supplied in PDF syntax, e.g.* `suppressValue="D:20170116130311+01'00'"`. ***textIndexPrefix*** *(String; optional) String to be prepended to the property value if* `Filtering/@metadataHandling` *is* `propertyAndPrefixedText`. *Default: empty* ***type*** *(Choice; optional) Windows data type of the metadata property. Supported choices are* `Boolean`, `DateTime`, `Double`, `Int32`, *and* `String`. *Default:* `String`
PropertySet *parent: Metadata*	*(May appear zero or more times) Specify filtering of a custom set of properties with the same GUID (see Section 2.5, »Custom Properties«, page 24).* *If present, this element must appear after* `PropertySetCollection` *and* `PrefixDeclarations`. *Supported attributes:* ***guid*** *(GUID; required) Unique 128-bit identifier for the property set in hexadecimal notation (see »Custom data types for XML elements and attributes«, page 59).*
PropertySet-Collection *parent: Metadata*	*(May appear zero or one time) Specify filtering of predefined property set collections (see Section 2.3, »Predefined Properties«, page 22). A list of properties can be found in Appendix A, »Predefined Metadata Properties«. Supported attributes:* ***documentXmp*** *(Boolean; required) Emit document XMP properties. Default:* `false` ***imageXmp*** *(Boolean; required) Emit image XMP properties. Default:* `false` ***internal*** *(Boolean; required) Emit internal properties of TET PDF IFilter. Default:* `true` ***pdf*** *(Boolean; required) Emit PDF-specific properties. Default:* `true` ***shell*** *(Boolean; required) Emit shell properties. Default:* `true`

Table 5.1 XML elements and attributes in the configuration file

element	***description of the element and its attributes***
Source *parent: Property*	*(May appear one or more times) Specify one or more sources for a metadata property.* *The ordering of elements is relevant if* `Property/@precedence` *has the value* `first-wins`*. At least one of the attributes* `pdfObject` *and* `xmpName` *must be provided.* *Supported attributes:* ***pdfObject*** *(pCOS path; optional) Extended pCOS path for one or more PDF objects of type Boolean, number, name or string containing the property. Default:* `/Root/Metadata` *(i.e. document-level XMP)* ***suppressValue*** *(String; optional; TET PDF IFilter 5.1) If the specified source delivers the value supplied in this attribute the value is assumed to be non-existent.* ***xmpName*** *(String consisting of the schema's prefix, a colon »:«, and the property name; optional) Fully qualified XMP property name. A prefix can be used instead of the namespace URI provided it has been declared in a* `PrefixDeclaration` *element. This attribute is only used if* `pdfobject` *is absent or points to one or more XMP streams. Default: empty*
Tet *parent: TetPdfIFilterConfig*	*(May appear zero or one time) Specify processing options for the TET kernel; refer to the TET manual for a description of the option list syntax and available options. Some options will be overridden by TET PDF IFilter.*
TetOptions *parent: Tet*	*(May appear zero or one time) Option list for the TET function TET_set_option( ).*
TetPdfIFilterConfig *parent: (none)*	*(Must appear exactly once as root element) Root element of the XML configuration file. Supported attribute:* ***version*** *(String; optional) Specify the version of TET PDF IFilter for which this configuration was written. New configurations should include this attribute with the appropriate version identifier (*`5.3` *for TET PDF IFilter 5.3).*
XmpLang *parent: XmpLangSelector*	*(Must appear exactly once if* `XmpLangSelector/@languages=subset`*) Specify the language of an XMP property. Supported attribute:* ***lang*** *(Language identifier; required). Name of the language; currently* `x-default` *is the only supported value.*
XmpLangSelector *parent: Xmp*	*(May appear zero or one time) Select a language variant of an XMP property for indexing. This is only relevant for properties with an XMP source of type* `Lang Alt`*. Supported attribute:* ***languages*** *(Choice) Specify language-specific indexing of the property (default:* `all`*):* ***all*** *All available language entries of the property will be indexed.* ***subset*** *Only the languages specified in the* `XmpLang` *element will be indexed.*

5.3 Sample Configuration File

The following listing shows a complete XML configuration file for TET PDF IFilter:

```
<?xml version="1.0" encoding="UTF-8"?>

<n:TetPdfIFilterConfig
    xmlns:n="http://www.pdflib.com/XML/TET_PDF_IFilter3/TET_PDF_IFilter_Config-3.0.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.pdflib.com/XML/TET_PDF_IFilter3/TET_PDF_IFilter_
Config-3.0.xsd http://www.pdflib.com/XML/TET_PDF_IFilter3/TET_PDF_IFilter_Config-3.0.xsd"
        version="5.3">

<n:Tet>
        <n:TetOptions></n:TetOptions>
        <n:DocOptions></n:DocOptions>
        <n:PageOptions></n:PageOptions>
</n:Tet>

<n:Filtering
        indexAnnotations="true"
        indexBookmarks="true"
        indexExtendedFormFields="false"
        indexFormFields="true"
        indexNestedPdf="true"
        metadataHandling="property"
        useIdentifier="true">

        <n:LocaleId
            detection="auto"
            useCatalogLang="true"
            default="0x0800"
            arabic="0x0401"
            chinese="0x0804"
            cyrillic="0x0419"
            latin="0x0409"/>
</n:Filtering>

<n:Metadata>
        <n:PropertySetCollection
            documentXmp="false"
            imageXmp="false"
            internal="true"
            pdf="true"
            shell="true"/>

        <n:PropertySet guid="E544AFE6-13E2-40F1-A702-DCEBE8FB7B03">
            <n:Property identifier="20">
                <n:Source pdfObject="/Info/Producer"/>
            </n:Property>
        </n:PropertySet>
</n:Metadata>

</n:TetPdfIFilterConfig>
```

6 Troubleshooting

6.1 TET PDF IFilter does not work at all

If TET PDF IFilter does not seem to work at all check the items below.

Is TET PDF IFilter correctly registered? You can use the command line tool *FiltReg.exe* to check the correct registration of TET PDF IFilter. The program lists all file extensions that have IFilters associated with them by printing the file extension and the name of the associated IFilter DLL. The *FiltReg.exe* program is installed with Microsoft Visual Studio. For more information see

msdn.microsoft.com/en-us/library/ms692537(VS.85).aspx

Note *For testing the 64-bit edition of TET PDF IFilter the 64-bit version of* `FiltReg.exe` *is required. To get the 64-bit version it is necessary to install the Windows SDK on a 64-bit machine; otherwise the 64-bit tools are not installed.*

If TET PDF IFilter is correctly registered, the output of *FiltReg.exe* includes lines similar to the following:

```
Filters loaded by extension:
...
.pdf --> PDFlib TET PDF IFilter 64-bit (C:\Program Files\PDFlib\TET PDF IFilter 5.3
64-bit\bin\TETPDFIFilter.dll)
```

If TET PDF IFilter is not correctly registered you must register it manually (see Section »Manual installation«, page 6).

If you have Windows Search installed, you can test proper IFilter registration as follows: click *Start*, *Control Panel*, *Indexing Options*, *Advanced, File Types*. This will produce a long list of file types and associated filters. In this list, scroll to *pdf* in the *Extension* column. The corresponding entry in the *Filter Description* column should read *PDFlib TET PDF IFilter 32-bit* (or *64-bit*, as appropriate).

TET PDF IFilter and Adobe Acrobat. If you install Adobe Reader or Adobe Acrobat after TET PDF IFilter (or run Acrobat's automatic repair mode), they will overwrite the TET PDF IFilter registry entries. You can correct the situation by running the TET PDF IFilter installer in repair mode, or by manually registering the TET PDF IFilter DLL according to Section »Manual installation«, page 6.

Is the license key available? While TET PDF IFilter can be used without a commercial license key on Windows 8/10, it requires a license key on Windows Server. If you work on a server system and PDF indexing does not seem to work, the license key for TET PDF IFilter may be missing. In this case TET PDF IFilter will run in evaluation mode, which means it is restricted to small documents.

This situation can be detected by checking the Windows event log (see »Application event log«, page 71). In case of a problem with the license key there will be an entry with source *TET PDF IFilter* and category *TET Error*. Double-click on the line containing the error and examine the error message. The following text indicates that a valid license key could not be found:

```
TET API Error in TetIFilter::Init: open_document:
Invalid license key (error number 1986)
```

If you find this message you must enter the license key in the registry (see Section »Manual installation«, page 6).

6.2 Problems with TET PDF IFilter Operation

If TET PDF IFilter does not seem to work as expected, the analysis methods discussed below may help.

Locked PDF documents are not indexed. If an application locks a PDF file, TET PDF IFilter cannot index the document. In particular, files are locked as long as they are opened in Acrobat. While the IFilter client may retry the locked document later, the index will be incomplete until the locked document is released. We therefore recommend to avoid viewing PDF documents in Acrobat during indexing.

XML configuration file is not read. In order to use an XML configuration file for TET PDF IFilter you must create a suitable registry entry which points to the configuration file (see Chapter 5, »XML Configuration File«, page 59). If the XML configuration file cannot be read or XML parsing fails, an error message is written to the event log (see »Application event log«) and no indexing is performed.

6.3 PDF Documents are not completely indexed

SharePoint is subject to limitations which affect indexing of large documents. Since these limitations are not well explained in Microsoft documentation the following notes collect information based on Microsoft support articles and blogs. These notes are not authoritative; if in doubt please contact Microsoft for guidance.

Limitations in SharePoint. SharePoint imposes several limitations on document indexing. You can find more information about fixed and configurable limits in SharePoint at

technet.microsoft.com/en-us/library/cc262787%28v=office.15%29.aspx (SP 2013)
docs.microsoft.com/en-us/sharepoint/install/software-boundaries-and-limits-0 (SP 2016/2019)

The following SharePoint limitations are imposed upon TET PDF IFilter:

- The maximum file size *(MaxDownloadSize)* specifies the maximum size of documents which will be crawled and indexed. The default value for SharePoint is 64 MB.
- SharePoint 2013: The maximum growth factor *(MaxGrowFactor)* specifies a factor with which the *MaxDownloadSize* value is multiplied to determine the maximum amount of text for an indexed document. This factor is necessary because the text may be compressed inside the file, as is usually the case for PDF documents (unit: none, default: 4).
- The parsed content size specifies how many characters from a document can be indexed. SharePoint has a hard-coded limit of 2 million characters. This limit cannot be modified.

Changing maximum file size and growth factor for SharePoint. Apply the following command in the *SharePoint Management Shell* (add *-id <GUID of SSA>* to the first command if you have multiple search services):

```
$ssa = Get-SPEnterpriseSearchServiceApplication
$ssa.SetProperty("MaxDownloadSize", ...new value...)
```

A similar sequence can be applied to set *MaxGrowFactor*. You can check the current values as follows:

```
$ssa = Get-SPEnterpriseSearchServiceApplication
$ssa.GetProperty("MaxDownloadSize")
```

6.4 Debugging Facilities

If the search results don't match your expectation and you suspect problems with the text contents extracted from the indexed documents the debugging tools discussed below may be helpful.

Application event log. TET PDF IFilter creates entries in the Windows event log for various events. You can check the application event log as follows:

- Windows 8/10: Open the *Event Viewer* (or the corresponding localized term, e.g. *Ereignisprotokolle* in German versions of Windows). In the Event Viewer window click on *Windows Logs, Application*.
- TET PDF IFilter events are listed with source *TET PDF IFilter*. Click on the line containing the message and examine the detailed message.

Entries in the application event log can be enabled for various classes of events by setting the registry value

```
HKEY_LOCAL_MACHINE\SOFTWARE\PDFlib\TET PDF IFilter5\logging
```

or

```
HKEY_LOCAL_MACHINE\SOFTWARE\PDFlib\TET PDF IFilter5\5.3\logging
```

to a DWORD value according to Table 6.1. The logging level is set to 1 by default.

Table 6.1 Logging levels for the Windows event log

level (DWORD)	*summary*	*logged events*
0	*fatal errors*	*Fatal problems which prevent TET PDF IFilter operation, e.g. errors in the XML configuration file, invalid user-supplied TET option lists and licensing problems. No documents can be processed*
1 (default)	*document-specific errors*	*Like level 0, plus: document-specific fatal errors, e.g. out of memory. A single document cannot be processed.*
2	*no text could be extracted*	*Like level 1, plus: documents from which no text could be extracted, e.g. not a PDF document, required password missing, completely damaged PDF, dynamic XFA form. XMP parsing problems are also reported on this level.*
3	*complete file listing*	*Like level 2, plus: each processed document is reported, regardless of the result. This is useful to ensure that a particular document has actually been passed to TET PDF IFilter.*
4	*processing details*	*Like level 3, plus XMP metadata parsing errors and additional details about document processing which may be useful for debugging and support.*

Note If writing to the Windows event log fails (e.g. because of permission problems) some emergency logging information is emitted via the `OutputDebugString()` *facility as fallback. The fallback logging output can be viewed with the Sysinternals application* `Dbgview.exe` *when the option »Capture Global Win32« is turned on in the »Capture« menu. If you need additional diagnosis for particular problem documents you can run* `filtdump.exe` *and examine the generated entries in the Windows event log.*

Check indexing time. TET PDF IFilter stores the processing date and time in a property with the canonical name *PDFlib.TET.indextime* (user interface name *indexname)*. This can be used to check whether or when a document has already been processed by TET PDF IFilter.

For example, if TET PDF IFilter is configured for use with Windows Search you can display the index time property in an Explorer or Search Window via *View, Add columns, Choose columns*, and selecting *indextime*. Now click *View, Details* to display the detailed file information. The new column contains the indexing time for each PDF file processed by TET PDF IFilter. A missing entry in this column indicates that the document has not been processed by TET PDF IFilter.

Identify problematic documents. Depending on the IFilter client in use, event log entries may or may not include the names of affected files, and file names may or may not be useful. For example, SharePoint downloads the documents via HTTP and creates a temporary local copy. In contrast, the crawler used by Windows Search generally provides the name of the indexed file, but doesn't do this in special cases, e.g. when unpacking a compressed ZIP archive. The event log therefore does not always contain a useful file name. As an aid for identifying the affected PDF document the event log entries created by TET PDF IFilter contain the file size in bytes. You can use the search engine itself to identify affected documents.

- In Windows Search you can use the following query expression (assuming 12345 is the file size in bytes):

  ```
  System.Size: = 12345
  ```

- In SharePoint you can identify failed attempts to filter a file in the SharePoint crawl log (Shared Services Administration: *SharedServices, Search Settings, Crawl Log)*. The errors regarding PDF documents listed here correspond to the errors issued by TET PDF IFilter in the Windows application event log. By comparing the file size of the files in the crawl log with the entries in the event log you can identify problematic documents.

Also note the *Filtering/@errorIndicator* configuration attribute which can be used to emit an identifying string for problematic documents in the index (see Section 5.2, »XML Elements and Attributes«, page 61).

Which properties and which text are emitted for a document? In order to see the exact text that TET PDF IFilter extracts from a particular document, the tool *FiltDump.exe* from the Windows SDK can be used. For testing the 64-bit TET PDF IFilter DLL the 64-bit version of this tool is required. For more information see

msdn.microsoft.com/en-us/library/ms692535(VS.85).aspx

With the option *-o* the output of *FiltDump.exe* can be redirected to a UTF-16-encoded file. This makes it possible to see the exact Unicode text and the detected locale (LCID) for the text which is emitted by TET PDF IFilter. Sample invocation:

```
FiltDump.exe -o udhr_japanese.txt udhr_japanese.pdf
```

Sample output:

```
FILE: udhr_japanese.pdf
IFILTER: CLSID == {47A1AF35-C345-475D-AE68-EB07E948BD07}
```

```
IFILTER: Using IPersistStream
IFILTER: IFilter->Init returned IFILTER_FLAGS_OLE_PROPERTIES flag


CHUNK: ------------------------------------------------------------------
    Attribute = {007867F0-C59B-43FC-AB1E-8EEE77057254}\4 (PDFlib.TET.indextime)
    idChunk = 3
    BreakType = 2 (Sentence)
    Flags (chunkstate) =  (Value)
    Locale = 1031 (0x407)
    IdChunkSource = 3
    cwcStartSource = 0
    cwcLenSource = 0

VALUE: ------------------------------------------------------------------
Type = 64 (0x40), VT_FILETIME
Value = "2010/06/10:08:28:04.587"

CHUNK: ------------------------------------------------------------------
    Attribute = {B725F130-47EF-101A-A5F1-02608C9EEBAC}\19 (System.Search.Contents)
    idChunk = 11
    BreakType = 2 (Sentence)
    Flags (chunkstate) =  (Text)
    Locale = 9 (0x9)
    IdChunkSource = 11
    cwcStartSource = 0
    cwcLenSource = 0

TEXT: ------------------------------------------------------------------
UDHR – Japanese

CHUNK: ------------------------------------------------------------------
    Attribute = {B725F130-47EF-101A-A5F1-02608C9EEBAC}\19 (System.Search.Contents)
    idChunk = 12
    BreakType = 2 (Sentence)
    Flags (chunkstate) =  (Text)
    Locale = 17 (0x11)
    IdChunkSource = 12
    cwcStartSource = 0
    cwcLenSource = 0

TEXT: ------------------------------------------------------------------
...text contents of the document...
```

TET kernel logging. You can enable detailed TET logging in order to analyze the behavior of the TET kernel as driven by TET PDF IFilter. TET logging can be activated as follows:

- By setting suitable TET options in the XML configuration file (make sure to specify the file name of the XML configuration file in the registry, see Chapter 5, »XML Configuration File«, page 59):

  ```
  <Tet>
       <TetOptions>logging={filename=C:\debug.log classes={pcos=2}}</TetOptions>
  </Tet>
  ```

  This will create a log file with details about internal calls to TET functions, error messages, etc. Make sure to use a file name which is writable for the service which calls

TET PDF IFilter, and keep in mind that TET logging creates lots of output and slows down the filtering process.

- By setting an environment variable with PowerShell:

```
PS C:\> ${env:TET PDF IFILTERLOGGING} = "filename=tetpdfifilter.log
classes={filesearch=3}"
```

A Predefined Metadata Properties

The predefined properties are known internally to TET PDF IFilter (see Section 2.3, »Predefined Properties«, page 22). The first column in each table lists the canonical property name. The second column in Table A.1 lists the display names (user interface names) of Shell properties which are offered as a column in the Details view of Explorer windows. The display names of other predefined properties can be derived by omitting the *PDFlib.TET* prefix, e.g. the property *PDFlib.TET.pdfa* has the display name *pdfa*.

A.1 Shell Property Set Collection

The Shell property set collection is already known to Windows. Canonical names and display names (without spaces) can be used in queries. To determine non-English display names, e.g. German, see »Canonical property name and localized display name«, page 9).

Table A.1 Predefined properties in the Shell property set collection; these are already known in Windows

canonical property name	English display name (label)	data type	multi-valued	property set GUID/property ID	source: XMP property or pCOS path
System.Document.Contributor	Contributors	String	yes	F334115E-DA1B-4509-9B3D-119504DC7ABB/100	dc:contributor
System.Document.DateCreated	Content created	DateTime	no	F29F85E0-4FF9-1068-AB91-08002B27B3D9/12	xmp:CreateDate, /Info/CreationDate
System.Document.DateSaved[1]	Date last saved	DateTime	no	F29F85E0-4FF9-1068-AB91-08002B27B3D9/13	xmp:ModifyDate, /Info/ModDate
System.Document.DocumentID	Document ID	String	no	E08805C8-E395-40DF-80D2-54F0D6C43154/100	dc:identifier
System.Document.PageCount	Pages	Int32	no	F29F85E0-4FF9-1068-AB91-08002B27B3D9/14	length:pages
System.Document.Version	Version number	String	no	D5CDD502-2E9C-101B-9397-08002B2CF9AE/29	xmpMM:VersionID
System.Search.Contents	n/a	String	yes	B725F130-47EF-101A-A5F1-02608C9EEBAC/19	text contents of PDF pages
System.Title	Title	String	no	F29F85E0-4FF9-1068-AB91-08002B27B3D9/2	dc:title["x-default"], /Info/Title
System.Subject	Subject	String	no	F29F85E0-4FF9-1068-AB91-08002B27B3D9/3	dc:description["x-default"], /Info/Subject
System.Author	Authors	String	yes	F29F85E0-4FF9-1068-AB91-08002B27B3D9/4	dc:creator, pdf:Author, xmp:Author, /Info/Author
System.DateModified[1]	Date modified	DateTime	no	B725F130-47EF-101A-A5F1-02608C9EEBAC/14	xmp:ModifyDate, /Info/ModDate
System.Keywords	Tags	String	yes	F29F85E0-4FF9-1068-AB91-08002B27B3D9/5	pdf:Keywords, /Info/Keywords
System.MIMEType	n/a	String	no	0B63E350-9CCC-11D0-BCDB-00805FCCCE04/5	`application/pdf` *(fixed value)*
System.ApplicationName	Program name	String	no	F29F85E0-4FF9-1068-AB91-08002B27B3D9/18	xmp:CreatorTool, /Info/Creator
System.Kind	Kind	String	no	1E3EE840-BC2B-476C-8237-2ACD1A839B22/3	`Document` *(fixed value)*

1. Some IFilter clients, e.g. Windows Search, override the PDF-based values with file system properties.

A.2 PDF Property Set Collection

The properties in this collection are emitted by default. In order to use these properties with Windows Search they must be registered with the *proptool.exe* utility (see Section 3.1, »Metadata Properties in Windows Search«, page 35). Property descriptions for this collection are available in the file *predefined_properties.propdesc*.

Table A.2 Predefined properties in the PDF property set collection

canonical property name	*data type*	*multi-valued*	*property set GUID/property ID*	*source: XMP property or pCOS path*
PDFlib.TET.pdfversion (PDF version multiplied by 10, e.g. »17«)	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/2*	*pdfversion*
PDFlib.TET.fullpdfversion (PDF version multiplied by 100, e.g. »173«)	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/12*	*fullpdfversion*
PDFlib.TET.pdfa[1]	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/3*	*pdfa*
PDFlib.TET.pdfe[1]	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/13*	*pdfe*
PDFlib.TET.pdfua[1]	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/14*	*pdfua*
PDFlib.TET.pdfvt[1]	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/15*	*pdfvt*
PDFlib.TET.pdfx[1]	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/4*	*pdfx*
PDFlib.TET.font	*String*	*yes*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/5*	*fonts[*]/name*
PDFlib.TET.producer	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/10*	*/Info/Producer*
PDFlib.TET.trapped	*String*	*no*	*E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/11*	*/Info/Trapped*

1. The value none *which is returned by pCOS if the document does not conform to any part of this standard is suppressed.*

A.3 Document XMP Metadata Property Set Collection

The properties in this collection are not emitted by default, but must be enabled with *PropertySetCollection/@documentXmp="true"* in the XML configuration file. In order to use these properties with Windows Search they must be registered with the *proptool.exe* utility (see Section 3.1, »Metadata Properties in Windows Search«, page 35). Property descriptions for this collection are available in the file *predefined_properties.propdesc*.

Table A.3 Predefined properties in the document XMP metadata property set collection (from the XMP specification)

canonical property name	*data type*	*multi-valued*	*property set GUID/property ID*	*source: XMP property or pCOS path*
Dublin Core				
PDFlib.TET.dc.contributor	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/2*	*dc:contributor*
PDFlib.TET.dc.coverage	*String*	*no*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/3*	*dc:coverage*
PDFlib.TET.dc.creator	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/4*	*dc:creator*
PDFlib.TET.dc.date	*DateTime*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/5*	*dc:date*
PDFlib.TET.dc.description	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/6*	*dc:description*
PDFlib.TET.dc.format	*String*	*no*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/7*	*dc:format*
PDFlib.TET.dc.identifier	*String*	*no*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/8*	*dc:identifier*
PDFlib.TET.dc.language	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/9*	*dc:language*
PDFlib.TET.dc.publisher	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/10*	*dc:publisher*
PDFlib.TET.dc.relation	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/11*	*dc:relation*
PDFlib.TET.dc.rights	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/12*	*dc:rights*
PDFlib.TET.dc.source	*String*	*no*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/13*	*dc:source*
PDFlib.TET.dc.subject	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/14*	*dc:subject*
PDFlib.TET.dc.title	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/15*	*dc:title*
PDFlib.TET.dc.type	*String*	*yes*	*D92BB3CA-CE2B-4B9B-972A-5BF54B468171/16*	*dc:type*
XMP Media Management				
PDFlib.TET.xmpMM.DocumentID	*String*	*no*	*743CD065-7F47-4b4d-ACF1-F0A09C92EAA9/2*	*xmpMM:DocumentID*
PDFlib.TET.xmpMM.InstanceID	*String*	*no*	*743CD065-7F47-4b4d-ACF1-F0A09C92EAA9/3*	*xmpMM:InstanceID*
PDFlib.TET.xmpMM.VersionID	*String*	*no*	*743CD065-7F47-4b4d-ACF1-F0A09C92EAA9/4*	*xmpMM:VersionID*
PDFlib.TET.xmpMM.OriginalDocumentID	*String*	*no*	*743CD065-7F47-4b4d-ACF1-F0A09C92EAA9/5*	*xmpMM:OriginalDocumentID*
XMP Basic				
PDFlib.TET.xmp.Advisory	*String*	*yes*	*C60E822A-074F-4BD5-9889-6EBD372F2000/2*	*xmp:Advisory*

Table A.3 Predefined properties in the document XMP metadata property set collection (from the XMP specification)

canonical property name	data type	multi-valued	property set GUID/property ID	source: XMP property or pCOS path
PDFlib.TET.xmp.BaseURL	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/3	xmp:BaseURL
PDFlib.TET.xmp.CreateDate	DateTime	no	C60E822A-074F-4BD5-9889-6EBD372F2000/4	xmp:CreateDate
PDFlib.TET.xmp.CreatorTool	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/5	xmp:CreatorTool
PDFlib.TET.xmp.Identifier	String	yes	C60E822A-074F-4BD5-9889-6EBD372F2000/6	xmp:Identifier
PDFlib.TET.xmp.Label	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/7	xmp:Label
PDFlib.TET.xmp.MetadataDate	DateTime	no	C60E822A-074F-4BD5-9889-6EBD372F2000/8	xmp:MetadataDate
PDFlib.TET.xmp.ModifyDate	DateTime	no	C60E822A-074F-4BD5-9889-6EBD372F2000/9	xmp:ModifyDate
PDFlib.TET.xmp.Nickname	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/10	xmp:Nickname
PDFlib.TET.xmp.Rating	Int32	no	C60E822A-074F-4BD5-9889-6EBD372F2000/11	xmp:Rating
XMP Rights Management				
PDFlib.TET.xmpRights.Certificate	String	no	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/2	xmpRights:Certificate
PDFlib.TET.xmpRights.Marked	Boolean	no	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/3	xmpRights:Marked
PDFlib.TET.xmpRights.Owner	String	yes	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/4	xmpRights:Owner
PDFlib.TET.xmpRights.UsageTerms	String	yes	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/5	xmpRights:UsageTerms
PDFlib.TET.xmpRights.WebStatement	String	no	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/6	xmpRights:WebStatement
XMP Basic Job Ticket				
PDFlib.TET.xmpBJ.JobRef	String	yes	EBC983EF-C1CF-45C8-A29E-993543A0ECFB/2	xmpBJ:JobRef
XMP Paged-Text				
PDFlib.TET.xmpTPg.NPages	Int32	no	7A9EB492-35AB-49FE-B364-A21FC9575C28/2	xmpTPg:NPages
PDFlib.TET.xmpTPg.PlateNames	String	yes	7A9EB492-35AB-49FE-B364-A21FC9575C28/3	xmpTPg:PlateNames
Adobe PDF				
PDFlib.TET.pdf.Keywords[1]	String	no	17EB8447-FC9B-4D4D-81DF-31E9AA770CBF/2	pdf:Keywords
PDFlib.TET.pdf.PDFVersion[1]	String	no	17EB8447-FC9B-4D4D-81DF-31E9AA770CBF/3	pdf:PDFVersion
PDFlib.TET.pdf.Producer[1]	String	no	17EB8447-FC9B-4D4D-81DF-31E9AA770CBF/4	pdf:Producer

1. These XMP properties are rarely used; it is recommended to use the corresponding properties `System.Keywords`, `PDFlib.TET.fullpdfversion` and `PDFlib.TET.producer` instead.

A.4 XMP Image Metadata Property Set Collection

The properties in this collection are not emitted by default, but must be enabled with *PropertySetCollection/@imageXmp="true"* in the XML configuration file. In order to use these properties with Windows Search they must be registered with the *proptool.exe* utility (see Section 3.1, »Metadata Properties in Windows Search«, page 35). Property descriptions for this collection are available in the file *predefined_properties.propdesc*.

Table A.4 Predefined properties in the XMP image metadata property set collection (from the Photoshop *schema in the XMP 2005 specification)*

canonical property name	*data type*	*multi-valued*	*property set GUID/property ID*	*source: XMP property or pCOS path*
PDFlib.TET.images.photoshop.AuthorsPosition	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/2*	*photoshop:AuthorsPosition*
PDFlib.TET.images.photoshop.CaptionWriter	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/3*	*photoshop:CaptionWriter*
PDFlib.TET.images.photoshop.Category	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/4*	*photoshop:Category*
PDFlib.TET.images.photoshop.City	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/5*	*photoshop:City*
PDFlib.TET.images.photoshop.Country	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/6*	*photoshop:Country*
PDFlib.TET.images.photoshop.Credit	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/7*	*photoshop:Credit*
PDFlib.TET.images.photoshop.DateCreated	*DateTime*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/8*	*photoshop:DateCreated*
PDFlib.TET.images.photoshop.Headline	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/9*	*photoshop:Headline*
PDFlib.TET.images.photoshop.Instructions	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/10*	*photoshop:Instructions*
PDFlib.TET.images.photoshop.Source	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/11*	*photoshop:Source*
PDFlib.TET.images.photoshop.State	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/12*	*photoshop:State*
PDFlib.TET.images.photoshop.SupplementalCategories	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/13*	*photoshop:SupplementalCategories*
PDFlib.TET.images.photoshop.TransmissionReference	*String*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/14*	*photoshop:TransmissionReference*
PDFlib.TET.images.photoshop.Urgency	*Int32*	*yes*	*C9F08C60-189D-11DD-8441-0002A5D5C51B/15*	*photoshop:Urgency*

A.5 Internal Property Set Collection

The properties in this collection are emitted by default. In order to use these properties with Windows Search they must be registered with the *proptool.exe* utility (see Section 3.1, »Metadata Properties in Windows Search«, page 35). Property descriptions for this collection are available in the file *predefined_properties.propdesc*.

Table A.5 Predefined properties in the internal property set collection

canonical property name	*data type*	*multi-valued*	*property set GUID/property ID*	*source: XMP property or pCOS path*
PDFlib.TET.version	*String*	*no*	*007867F0-C59B-43FC-AB1E-8EEE77057254/2*	`5.3` *(plus possibly a patchlevel number)*
PDFlib.TET.indextime	*DateTime*	*no*	*007867F0-C59B-43FC-AB1E-8EEE77057254/4*	*date and time of index run*
PDFlib.TET.eval	*Int32*	*no*	*007867F0-C59B-43FC-AB1E-8EEE77057254/5*	*exception number if IFilter runs in evaluation mode*

B Revision History

Revision history of this manual

Date	Changes
March 16, 2021	► Updates for TET PDF IFilter 5.3 (based on TET 5.3)
July 18, 2019	► Updates for TET PDF IFilter 5.2 (based on TET 5.2)
December 04, 2018	► Added information about SharePoint 2019
September 04, 2017	► Added information about SharePoint 2016
May 23, 2017	► Updates for TET PDF IFilter 5.1 (based on TET 5.1)
July 28, 2016	► Documented support for SQL Server 2014 and 2016
October 23, 2015	► Updates for TET PDF IFilter 5.0 (based on TET 5.0)
January 27, 2015	► Updates for TET PDF IFilter 4.4 (based on TET 4.4)
May 26, 2014	► Updates for TET PDF IFilter 4.3 (based on TET 4.3)
May 16, 2013	► Updates for TET PDF IFilter 4.2 (based on TET 4.2)
October 22, 2012	► Added section on Exchange Server 2010 (based on TET 4.1p9)
February 13, 2012	► Updates for TET PDF IFilter 4.1 (based on TET 4.1)
September 22, 2010	► Updates for TET PDF IFilter 4.0p2 (based on TET 4.0p2)
July 22, 2010	► Updates for TET PDF IFilter 4.0 (based on TET 4.0)
August 06, 2008	► Updates for Search Server
June 16, 2008	► TET PDF IFilter 3.0 (based on TET 3.0pre2)
June 6, 2008	► TET PDF IFilter 3.0 beta3 (based on TET 3.0pre2)
May 09, 2008	► Initial version for TET PDF IFilter 3.0 beta2 (based on TET 3.0pre1)

Index

PDFlib GmbH

PDFlib GmbH
Franziska-Bilek-Weg 9
80339 München, Germany
www.pdflib.com
phone +49 • 89 • 452 33 84-0

Licensing contact
sales@pdflib.com

Support
support@pdflib.com *(please include your license number)*