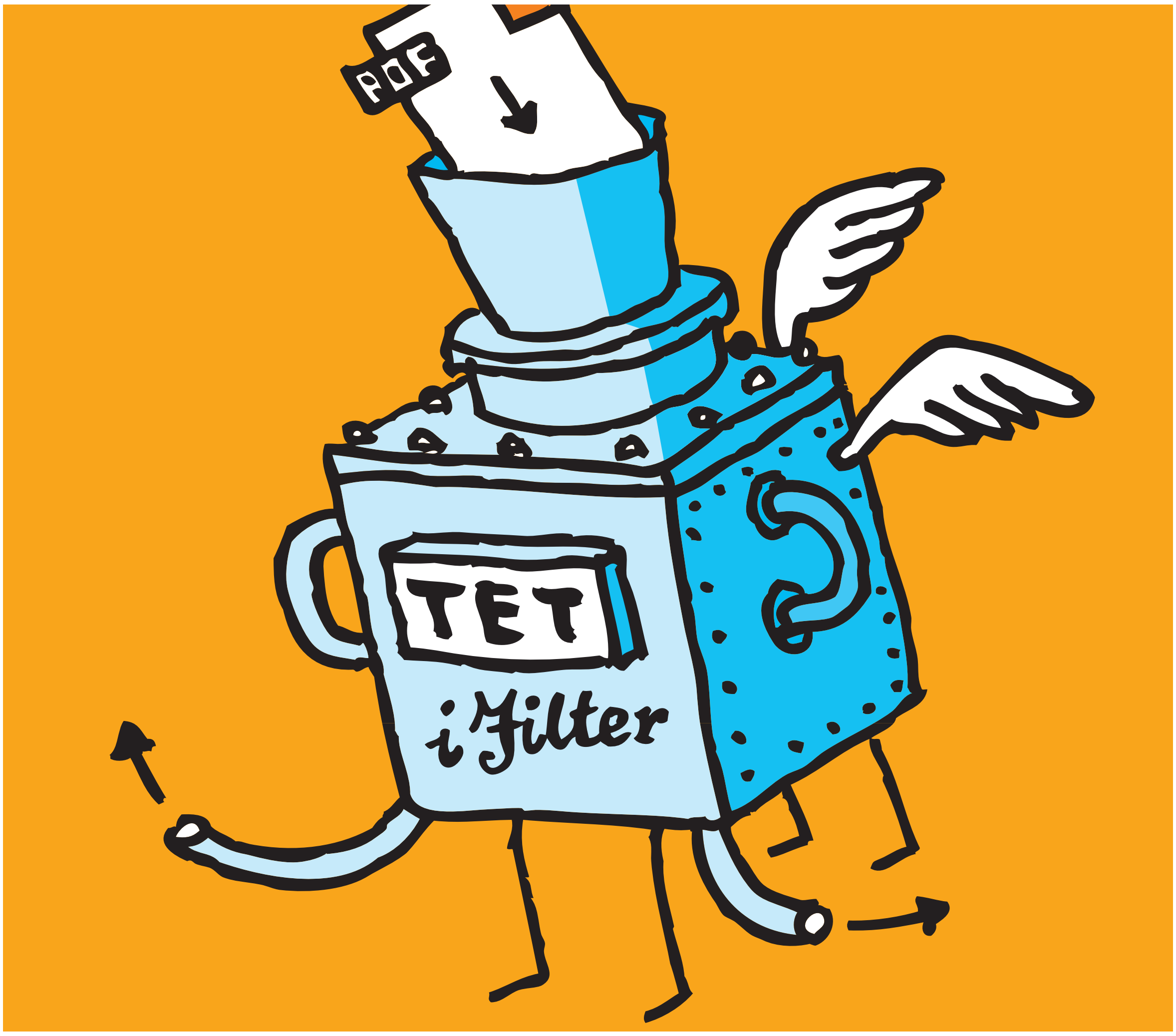
PDFlib GmbH

TET PDF IFilter

Version 5.0r1

Enterprise PDF Search for Windows



PDFlib GmbH
Franziska-Bilek-Weg 9, 80339 München, Germany
www.pdflib.com
phone +49 • 89 • 452 33 84-0
fax +49 • 89 • 452 33 84-99

If you have questions check the PDFlib mailing list and archive at
groups.yahoo.com/neo/groups/pdflib/info

Licensing contact: sales@pdflib.com
Support for commercial PDFlib licensees: support@pdflib.com (please include your license number)

Contents

0 Installing TET PDF IFilter

TET PDF IFilter is delivered as an MSI installer for Windows systems. All TET PDF IFilter packages contain a signed IFilter DLL plus support files, documentation, and samples. Running the MSI installer requires Administrator privileges. The installer will install and register TET PDF IFilter. Additional steps for specific search environments (e.g. Windows Search, SharePoint) as well as custom configuration are discussed elsewhere 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 will refuse to install on 32-bit systems, the 32-bit version works on both 32-bit and 64-bit systems.

Updating to a newer version of TET PDF IFilter. If an older version of TET PDF IFilter is already installed on the machine, you may, at your option, uninstall the older version before installing a newer 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 will enter the license key when installing TET PDF IFilter with the MSI 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 will only process PDF documents with up to 10 pages and 1 MB size. Evaluation versions of TET PDF IFilter must not be used for production purposes, but only for evaluating the product. Using TET PDF IFilter for production purposes requires a valid license.

Free desktop version for non-commercial use. TET PDF IFilter for desktop systems, i.e. Windows XP/Vista/7/8/10 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 SharePoint Foundation 2013

- SharePoint Server 2010, SharePoint Foundation 2010 with Search Server or Search Server Express (note that SharePoint Foundation 2010 standalone does not support the addition of file types for indexing beyond the built-in set of file types, which does not include PDF)
- Windows SharePoint Services 3.0, SharePoint Portal Services 2003, Office SharePoint Server 2007, SharePoint Server 2010, FAST Search Server 2010 for SharePoint
- SQL Server 2005, 2008, 2012, 2014, 2016
- Search Server 2008 and Search Server Express 2008, Search Server 2010 and Search Server Express 2010
- Exchange Server 2007 and 2010
- Site Server
- Windows Search is integrated in the Explorer of Windows Vista/7/8/10

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.

Post-installation steps. 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 5.0 64-bit\bin\TETPDFIFilter.dll"
```

Make sure to run this command from a command prompt with administrator privilege (click *Start*, type *Command Prompt,* right-click on *Command Prompt* and click *Run as Administrator).*

If there are IFilter clients which were 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 *registerpropdesc* 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 will open which runs with administrator privileges.

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 3, »Indexing Metadata«, page 35. The full TET PDF IFilter installation includes a set of sample PDF documents which can be used for testing the installation and getting familiar with its powerful metadata features.

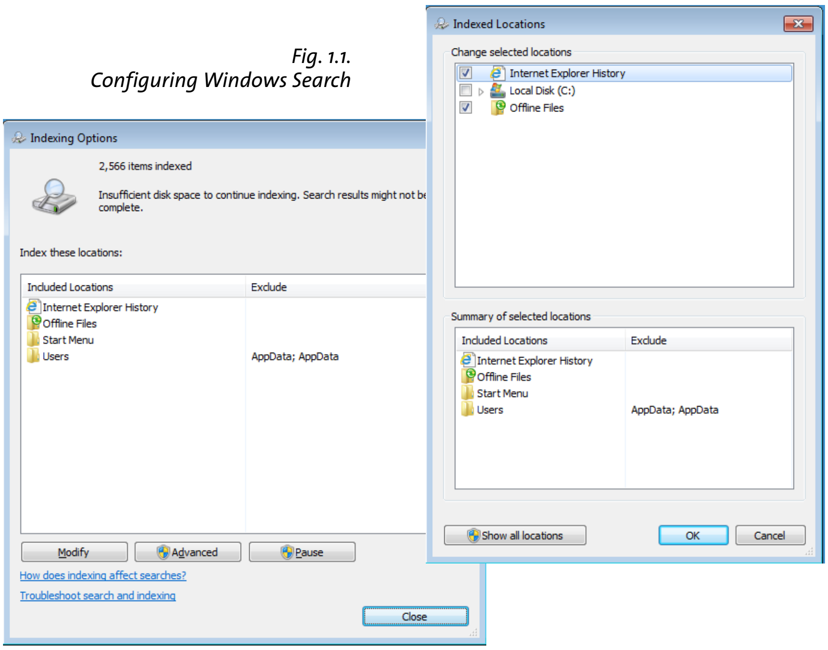
1.1 Windows Search

System requirements. Windows Desktop Search (WDS) 3 and its successor Windows Search replace the legacy Indexing Service. It brings performance and extensibility advantages and supports a property system for metadata which is supported by TET PDF IFilter. Windows Search 4 is available for Windows Vista/7/8/10 and Windows Server 2008/2012.

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

- Click *Start*, *Control Panel*, *Indexing Options*, and *Modify*.
- In the *Change selected locations* section, select the locations that you want, and then click OK.
- Click *Advanced* and *Rebuild* to force the documents to be indexed immediately.

Fig. 1.1.
Configuring Windows Search


Indexing Options
Index these locations:
Modify
Advanced
Pause
How does indexing affect searches?
Troubleshoot search and indexing
Close
Indexed Locations
Change selected locations
Summary of selected locations
Show all locations
OK
Cancel

2,566 items indexed

Insufficient disk space to continue indexing. Search results might not be complete.

Included Locations	Exclude
Internet Explorer History	
Offline Files	
Start Menu	
Users	AppData; AppData

Change selected locations:
- [✓] Internet Explorer History
- [ ] Local Disk (C:)
- [✓] Offline Files

Included Locations	Exclude
Internet Explorer History	
Offline Files	
Start Menu	
Users	AppData; AppData

Starting/Stopping Windows Search. 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 WDS will automatically start the indexing service in certain situations.

Simple text search. You can use several methods to search for text:

- WDS displays a search box in the *Start* menu. Type the search terms into this search box and press *Enter*.
- Open the Search window with Windows-F.
- In a Windows Explorer Window use the *Search* box at the top right.

Regardless of the method you choose you can type search terms in the corresponding window, and WDS will display a hit list of documents which contain the search term (if any). If a PDF viewer (e.g. Adobe Acrobat or Adobe Reader) is installed on the system the contents of the selected document is displayed in the results window.

Advanced text search. In addition to typing simple search terms you can use advanced query operators to narrow your search. Table 1.1 contains some examples. A comprehensive list of syntax elements for advanced queries is available at

http://windows.microsoft.com/en-us/windows7/advanced-tips-for-searching-in-windows

See Section 4.1, »Metadata in Windows Search«, page 47, for metadata queries.

Table 1.1 Query syntax examples for Windows Search 3.0 and above

Search term example	*Explanation*
Hol	*document contents or any property contain words starting with* `Hol`
Holmes AND Watson *Holmes + Watson* *Holmes Watson* *(Holmes Watson)*	*document contents contain both* `Holmes` *and* `Watson`
"Sherlock Holmes"	*document contents contain the exact phrase* `Sherlock Holmes`
Holmes OR Watson	*document contents contain either* `Holmes` *or* `Watson`
Holmes NOT Mowgli *Holmes -Mowgli*	*document contents contain the term* `Holmes`*, but not the term* `Mowgli`

Programmatically querying the index. In addition to interactive queries you can also query the WDS index programmatically. Accessing the search index can be accomplished via the AQS *(Advanced Query Syntax)* interface and SQL syntax extensions which present the search index through a database-like programming interface. See »SQL queries for metadata properties«, page 50, for WDS metadata queries with SQL.

Limitations. In our testing we ran into the following intrinsic limitations of WDS:

- The following limitation applies only to Windows XP/Vista: The generated catalog entry for a document will not exceed a size of ca. 1 MB. If a document creates larger catalog entries WDS will ignore the remaining text. At a typical size of ca. 3 KB text per page this amounts to more than 300 pages of text. However, there is not a strict limit so the results may vary. The only known workaround for this limitation in WDS is to split large documents into multiple smaller parts.
- PDF documents will not be indexed as long as they are opened in Acrobat. We recommend to close Acrobat before creating the search index.
- Searching for custom metadata properties interactively does not work, while programmatic searches (e.g. with PowerShell) do work.

1.2 SharePoint and FAST Search

1.2.1 System Requirements

TET PDF IFilter works with the following SharePoint configurations:

- SharePoint Server 2013 and SharePoint Foundation 2013; hotfix KB2883000 or July 8, 2014 Cumulative Update for SharePoint Server 2013 is required.
- SharePoint Server 2010, SharePoint Foundation 2010 with Search Server or Search Server Express (note that SharePoint Foundation 2010 standalone does not support the addition of file types for indexing beyond the built-in set of file types, which does not include PDF)
- FAST Search Server 2010 for SharePoint
- Microsoft Office SharePoint Server 2007
- SharePoint Portal Server 2003
- Windows SharePoint Services 3.0

You can find more information about SharePoint at
msdn.microsoft.com/en-us/sharepoint/default.aspx

1.2.2 Installation for SharePoint 2013

Proceed as follows to configure TET PDF IFilter for use with SharePoint 2013:

- Install TET PDF IFilter from the MSI package.
- Open a *SharePoint 2013 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 2013 search service:

```
net stop OSearch15
net stop SPSearchHostController
net start SPSearchHostController
```

Starting the *SPSearchHostController* service implicitly starts the *OSearch15* service as well.

1.2.3 Installation for SharePoint 2010 and earlier Versions

Install and restart. Proceed as follows to configure TET PDF IFilter for use with SharePoint 2010 and earlier versions; for SharePoint Foundation 2010 with Search Server or Search Server Express configure the PDF icon as described below, and configure TET PDF IFilter itself as described in Section 1.3, »Search Server«, page 13:

- Install TET PDF IFilter from the MSI package.
- If SharePoint is running while TET PDF IFilter is installed, the SharePoint crawler will not know about the newly installed PDF IFilter until the SharePoint search service is restarted. If you don't restart it you will experience the following error message from the crawler:

  ```
  The filtering process could not load the item.
  This is possibly caused by an unrecognized item format or item corruption.
  ```

  To restart the SharePoint search service for SharePoint 2010 apply the following command:

  ```
  net stop osearch14
  net start osearch14
  ```

  To restart the SharePoint search service for SharePoint Server2007 and earlier apply the following command:

  ```
  net stop osearch
  net start osearch
  ```

- Open the SharePoint administration web page: *Start, All Programs, Microsoft Office Server, SharePoint 3.0 Central Administration*.
- This will open a web page *Shared Services Administration: SharedServices1*. Under *Search*, click on *Search Settings*. Click on *File Types*.
- On the page that opens click *New File Type*, and create *pdf* file extension.
- Optionally configure a PDF file icon as described in the next section.

Add a PDF file icon. In order to visually represent PDF documents in the result list of a SharePoint query you can configure a PDF icon. This icon will be displayed for PDF documents. Proceed as follows:

- Download the PDF icon from

  www.adobe.com/misc/linking.html#pdficon

  and save it as *pdficon_small.gif*.
- For SharePoint Server 2010 and SharePoint Foundation 2010 copy the icon to the following directory:

  ```
  C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\IMAGES
  ```

  For SharePoint Server 2007 and Windows SharePoint Services 3.0 copy the icon file to the following directory:

  ```
  C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\Template\Images
  ```

  For SharePoint Portal Server 2003 copy the icon file to the following directory:

  ```
  C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\60\Template\Images
  ```

- For SharePoint Server 2010 and SharePoint Foundation 2010 load the *Docicon.xml* configuration file in the following directory in a text editor:

  ```
  C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\XML
  ```

  For SharePoint Server 2007 and Windows SharePoint Services 3.0 load the *Docicon.xml* configuration file in the following directory in a text editor:

  ```
  C:\Program Files\Common Files\Microsoft Shared\Web server extensions\12\Template\Xml
  ```

  For SharePoint Portal Server 2003 load the *Docicon.xml* configuration file in the following directory in a text editor:

  ```
  C:\Program Files\Common Files\Microsoft Shared\Web server extensions\60\Template\Xml
  ```

- Add the following line as a sub-element of the *<ByExtension>* element:

  ```
  <Mapping Key="pdf" Value="pdficon_small.gif"/>
  ```

- Restart the IIS Web server.

1.2.4 Simple and advanced Text Search

Several methods are available for building search queries in SharePoint:
- Query keywords
- SQL queries
- Queries encoded in URLs

For more information see

msdn.microsoft.com/en-us/library/ms497338.aspx

See Section 4.2, »Metadata in SharePoint and Search Server«, page 53, for metadata queries.

1.3 Search Server

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

- Search Server 2008 and Search Server 2008 Express
- Search Server 2010 and Search Server 2010 Express

Setup and configuration. Search Server configuration is very similar to SharePoint configuration. Proceed as follows to configure TET PDF IFilter for use with Search Server:

- Install TET PDF IFilter from the MSI package.
- Apply the post-installation steps described below.
- Open the Search administration web page: *Start, All Programs, Microsoft Search Server, Search Server 2008 Administration*.
- This will open a web page *Search Administration*. Under *Crawling*, click on *File Types*.
- On the page that opens click *New File Type*, and create *pdf* file extension.
- Optionally configure a PDF file icon as described for SharePoint in »Add a PDF file icon«, page 11.

Post-installation steps. If Search Server is running while TET PDF IFilter is installed, the Search Server crawler will not know about the newly installed PDF IFilter until the search service is restarted. If you don't restart it you will experience an error message from the crawler.

To restart the Search Server apply the following command:

```
net stop spsearch
net start spsearch
```

Simple and advanced text search. Configuring queries in Search Server works the same way as in SharePoint. For more information see *msdn.microsoft.com/en-us/library/aa981100.aspx*.

1.4 Exchange Server

System requirements. TET PDF IFilter works with Microsoft Exchange Server 2010. It may work with older versions of Exchange Servers, but these combinations have not been tested.

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

- Install TET PDF IFilter from the MSI package.
- 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 each update.

After TET PDF IFilter has been registered the PDF attachments of all new mails will be 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 to rebuild the Exchange full-text index:
technet.microsoft.com/en-us/library/aa995966(v=EXCHG.80).aspx

1.5 SQL Server

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

- SQL Server 2005 Workgroup, Standard, and Enterprise editions
- SQL Server 2005 Express Edition
- SQL Server 2008, 2012, 2014, 2016

You can find more information about SQL Server at

msdn.microsoft.com/en-us/library/bb418498.aspx

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 from the MSI package.
- 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 5.0 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:\Program Files\PDFlib\TET PDF IFilter 5.0
64-bit\PDF samples\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 4.3, »Metadata in SQL Server«, page 58, for metadata queries.

2 Indexing PDF Contents

2.1 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.

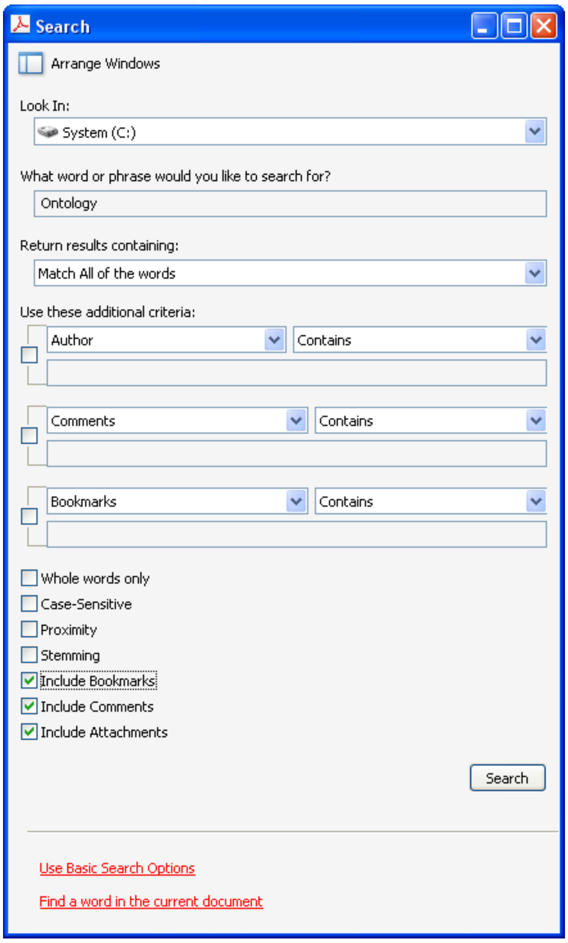
Notes regarding the descriptions below:

- 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.
- 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 3.3, »Predefined Metadata Properties«, page 39). By default, the *shell* and *internal* property set collections are enabled, while the *pdf, documentXMP,* and *imageXMP* property set collections are disabled. See Section 3.3, »Predefined Metadata Properties«, page 39, for more details on property set collections.
- The notation *@indexNestedPdf* refers to an attribute in the XML configuration file (see Section 6.2, »XML Elements and Attributes«, page 69).

In the remaining section we provide information on PDF domain searching. In addition, we summarize how to search these document domains with Acrobat X/XI/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 X/XI/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 X/XI/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.


Search

Arrange Windows

Look In: System (C:)

What word or phrase would you like to search for? Ontology

Return results containing: Match All of the words

Use these additional criteria:

- [ ] Author — Contains
- [ ] Comments — Contains
- [ ] Bookmarks — Contains

- [ ] Whole words only
- [ ] Case-Sensitive
- [ ] Proximity
- [ ] Stemming
- [x] Include Bookmarks
- [x] Include Comments
- [x] Include Attachments

Search

Use Basic Search Options

Find a word in the current document

Fig. 2.1
Acrobat's advanced search dialog

- TET PDF IFilter: page contents are indexed by default. However, in special situations you may want to disable page content indexing with *@indexPageContents=false*.

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

- How to display with Acrobat X/XI/DC: *File, Properties...*
- How to search a single PDF with Acrobat X/XI/DC: not available
- How to search multiple PDFs with Acrobat X/XI/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 property set collection *shell* is enabled

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

- How to display with Acrobat X/XI/DC: *File, Properties..., Custom* (not available in the free Adobe Reader)

- How to search with Acrobat X/XI/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 X/XI/DC: *File, Properties..., Description, Additional Metadata.*. (not available in the free Adobe Reader)
- How to search a single PDF with Acrobat X/XI: not available
- How to search multiple PDFs with Acrobat X/XI: 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 *XMP Metadata* (not available in the free Adobe Reader).
- TET PDF IFilter: document XMP metadata indexed if property set collection *documentXMP* is enabled 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 X: *Tools, Content, Edit Object,* select image, right-click, *Show Metadata...* (not available in the free Adobe Reader)
- How to display with Acrobat XI/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 the free Acrobat Reader)
- How to search with Acrobat X/XI/DC: not available
- TET PDF IFilter: XMP image metadata is indexed if property set collection *imageXMP* is enabled

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 X/XI: *Tools, Forms, Edit* (not available in the free Adobe Reader)
- How to display with Acrobat DC: *Tools*, *Prepare Form* (not available in the free Acrobat Reader)
- How to search with Acrobat X/XI/DC: not available
- TET PDF IFilter: form fields are indexed if custom properties are defined based on pCOS pseudo object *fields*. 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 the page option *skipengines={text image}*. The following excerpt shows the relevant parts of the XML configuration file (see Chapter 6, »XML Configuration File«, page 67) for this situation:

```
<n:Tet>
    <n:TetOptions></n:TetOptions>
    <n:DocOptions></n:DocOptions>
    <n:PageOptions>skipengines={text image}</n:PageOptions>
</n:Tet>

<n:Filtering metadataHandling="propertyAndText"/>

<n:Metadata>
```

```
            <n:PropertySet guid="E9CDA960-D09A-43bc-AAAA-BBBBBBBBBBBB">
                <n:Property friendlyName="Formfield" identifier="2">
                    <n:Source pdfObject="fields[*]/V"/>
                </n:Property>
            </n:PropertySet>
        </n:Metadata>
```

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 X/XI: *Comment, Comments List*
- How to display with Acrobat DC: *Tools, Comment, Comments List*
- How to search a single PDF with Acrobat X/XI/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 X/XI/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 are indexed if property set collection *pdf* is enabled

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 X/XI/DC: *View, Show/Hide, Navigation Panes, Bookmarks*
- How to search a single PDF with Acrobat X/XI/DC: *Edit, Advanced Search* and check the box *Include Bookmarks*
- How to search a single PDF with Acrobat X/XI/DC: *Edit, Search* and check the box *Include Bookmarks*
- How to search multiple PDFs with Acrobat X/XI/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 the free Adobe Reader)
- TET PDF IFilter: bookmarks are indexed if property set collection *pdf* is enabled

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

- How to display with Acrobat X/XI/DC: *View, Show/Hide, Navigation Panes, Attachments*
- How to search with Acrobat X/XI/DC: Use *Edit, AdvancedSearch* and check the box *Include Attachments* (not available in the free Adobe Reader). Nested attachments are not searched recursively.
- TET PDF IFilter: PDF attachments are indexed recursively if *@indexNestedPdf=true*

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

- How to display with Acrobat X/XI/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 X/XI/DC: *Edit, Search Entire Portfolio*
- How to search multiple PDF packages with Acrobat X/XI/DC: not available

- TET PDF IFilter: PDF documents contained in packages/portfolios are indexed recursively if *@indexNestedPdf=true*

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 X/XI/DC: *View, Show/Hide, Navigation Panes, Standards* (only present for standard-conforming PDFs)
- How to search with Acrobat X/XI/DC: not available
- TET PDF IFilter: PDF properties are indexed if property set collection *pdf* is enabled

Tagged PDF. TET reconstructs the layout structure and hierarchy directly from the page contents without using the structure tree which is present in Tagged PDF documents. Page contents 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 XI/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 X/XI/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 XI/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 X/XI/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* 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.

2.2 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 2.1 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 2.1 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 2.2 lists the supported script attributes and their default values. LCIDs for text in all other scripts will be assigned automatically.

Table 2.2 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)*

2.3 PDF Versions and Protected Documents

PDF versions. TET PDF IFilter accepts all PDF versions up to PDF 1.7 extension level 8, the file format of Acrobat DC. This includes various PDF-based ISO standards, e.g. PDF/A, PDF/E, and PDF/X. Note that ISO 32 000-1 is technically equivalent to PDF 1.7 and therefore also supported, as well as the encryption method specified in ISO 32 000-2.

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 will be logged. 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 security 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.

2.4 Unicode Postprocessing

The TET kernel, which provides 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.

The 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>
```

2.4.1 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.

In order to improve readability the examples in the tables below list isolated suboptions of the *fold* option list. Keep in mind that these suboptions must be combined to a single large fold option list if you want to apply multiple foldings; do not supply the *fold* option more than once. For example, the following is wrong:

```
fold={ {[:blank:] U+0020} } fold={ {_dehyphenation remove} }        WRONG!
```

The following option list shows the correct syntax for multiple foldings:

```
fold={ {[:blank:] U+0020 } {_dehyphenation remove} }
```

Folding examples. Table 2.3 lists examples for the *fold* option which demonstrate various folding applications. The sample options must be supplied in the document option list. TET can apply foldings to a selected subset of all Unicode characters. These are called Unicode sets; their syntax is discussed in the TET manual.

Table 2.3 Examples for the `fold` *option*

description and option list	***before folding***	***after folding***
Remove all characters in a Unicode set		
Keep only characters in ISO 8859-1 (Latin-1) in the output, i.e. remove all characters outside the Basic Latin Block: `fold={{[^U+0020-U+00FF] remove}}`	Ą U+0104	*n/a*

Remove all non-alphabetic characters (e.g. punctuation, numbers):
`fold={{[:Alphabetic=No:] remove}}`
	7
	U+0037
	A
	U+0041
		n/a
		
		A
		U+0041

Remove all characters except numbers: `fold={{[^[:General_Category=Decimal_Number:]] remove}}`	7 U+0037 A U+0041	7 U+0037 *n/a*
Remove all unknown characters, i.e. PUA characters and characters for which no usable Unicode value could be determined (the remaining default foldings are re-enabled): `fold={{[:Private_Use:] remove} {[U+FFFD] remove} default}`	U+FFFF	*n/a*
Remove all dashed punctuation characters: `fold={{[:General_Category=Dash_Punctuation:] remove}}`	- U+002D	*n/a*
Remove all Bidi control characters: `fold={{[:Bidi_Control:] remove}}`	U+200E	*n/a*
Remove all variation selectors for Standard or Ideographic Variation Sequences (IVS) while keeping the builtin foldings: `fold={{[[\uFE00-\uFE0F][\U000E0100-\U000E01EF]] remove} default}`	≨ FE00 U+2268 U+FE00	≨ U+2268
Replace all characters in a Unicode set with a specific character		
Space folding: map all variants of Unicode space characters to U+0020: `fold={{[:blank:] U+0020}}`	U+00A0	U+0020
Dashes folding: map all variants of Unicode dash characters to U+002D: `fold={{[:Dash:] U+002D}}`	- U+2011	- U+002D
Replace all unassigned characters (i.e. Unicode code points to which no character is assigned) with U+FFFD: `fold={{[:Unassigned:] U+FFFD}}`	U+03A2	� U+FFFD
Special handling for individual characters		
Preserve all hyphen characters at line breaks while keeping the remaining default foldings. Since these characters are identified internally in TET (as opposed to having a fixed Unicode property) the keyword `_dehyphenation` *is used to identify the folding's domain:* `fold={{_dehyphenation preserve}}`	- U+002D	- U+002D
Preserve Arabic Tatweel characters (which are removed by default): `fold={{[U+0640] preserve}}`	ـ U+0640	ـ U+0640
Replace various punctuation characters with their ASCII counterparts: `fold={ {[U+2018] U+0027} {[U+2019] U+0027} {[U+201C] U+0022} {[U+201D] U+0022}}`	“ U+201C	" U+0022

2.4.2 Unicode Decomposition

Decompositions replace a character with an equivalent sequence of one or more other characters. 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. 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)

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.*

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[1] (but not the character tables) canonical mappings are marked with the symbol IDENTICAL TO ≡ U+2261. The decomposition name <*canonical*> is implicitly assumed. Table 2.4 contains several examples.

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

```
decompose={canonical=_all}
```

Table 2.4 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 ゙ U+306F U+3099
		אָ U+FB2F	א ָ U+05D0 U+05B8

1. See www.unicode.org/charts/

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 identical to the option names in Table 2.5. 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 *While all entries in Table 2.5 describe compatibility decompositions, the* `compat` *tag includes only* `other` *compatibility decompositions, i.e. those without a specific name.*

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. Decompositions in TET can be controlled with the document option *decompose*. A decomposition can be restricted to operate only on some, but not all Unicode characters. The subset on which a decomposition operates is called its domain. Table 2.5 lists the suboptions for all Unicode decompositions along with examples.

The following examples for the *decompose* option must be supplied in the option list for *TET_open_document( )*. The decomposition names in the *decompose* option list are taken from Table 2.5.

Disable all decompositions:

```
decompose={none}
```

Preserve wide (double-byte or zenkaku) and hankaku (narrow) characters:

```
decompose={wide=_none narrow=_none}
```

Map all canonical equivalents to their counterparts:

```
decompose={canonical=_all}
```

The following option list enables the *circle* decomposition, but disables all other decompositions:

Table 2.5 Compatibility decomposition: suboptions for the decompose *option (canonically equivalent characters are marked with the symbol ALMOST EQUAL TO* ≈ U+2248 *in the Unicode code charts)*

decomposition name	**description**	**before decomposition**	**after decomposition (in logical order)**
circle	*Encircled characters*	㉑ U+3251	2 1 U+0032 U+0031
compat[1]	*Other compatibility decompositions, e.g. common ligatures*	ﬁ U+FB01	f i U+0066 U+0069
final	*Final presentation forms, especially Arabic*	ﺲ U+FEB2	س U+0633
font	*Font variants, e.g. mathematical set letters, Hebrew ligatures*	ℂ U+2102	C U+0043
fraction[1]	*Vulgar fraction forms*	¼ U+00BC	1 ⁄ 4 U+0031 U+2044 U+0034
initial	*Initial presentation forms, especially Arabic*	ﺳ U+FEB3	س U+0633
isolated	*Isolated presentation forms, especially Arabic*	ﴎ U+FD0E	س ر U+0633 U+0631
medial	*Medial presentation forms, especially Arabic*	ﺴ U+FEB4	س U+0633
narrow	*Narrow (hankaku) compatibility characters*	ｦ U+FF66	ヲ U+30F2
nobreak	*Non-breaking characters*	U+00A0	U+0020
none	*Disable all decompositions which are not explicitly specified in the* decompose *option list.*	*(leaves all characters unmodified)*	
small	*Small forms for CNS 11643 compatibility*	﹐ U+FE50	, U+002C
square	*CJK squared font variants*	㌔ U+3314	キ ロ U+30AD U+30ED
sub[1]	*Subscript forms*	₁ U+2081	1 U+0031
super[1]	*Superscript forms*	ª U+00AA	a U+0061
		™ U+2122	T M U+0054 U+004D
vertical	*Vertical layout presentation forms*	︷ U+FE37	{ U+007B
wide	*Wide (zenkaku) compatibility forms*	￡ U+FFE1	£ U+00A3

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

```
decompose={none circle=_all}
```

In contrast, the following option list enables all decompositions (since omitting the other options activates the default):

```
decompose={circle=_all}
```

2.4.3 Unicode Normalization

The Unicode standard defines four normalization forms which are based on the notions of canonical equivalence and compatibility equivalence. 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.
- Normalization Form D (NFD) applies canonical decomposition.
- Normalization Form KC (NFKC) applies compatibility decomposition followed by canonical composition.
- Normalization Form KD (NFKD) applies compatibility decomposition.

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.

```
normalize=nfc
```

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.

The choice of normalization form depends on the application's requirements. For example, some databases expect text in NFC which is also the preferred format for Unicode text on the Web. Table 2.6 demonstrates the effect of Normalization on various characters.

Table 2.6 Unicode normalization forms: examples

before normalization	***NFC***	***NFD***	***NFKC***	***NFKD***
Ä U+00C4	Ä U+00C4	A ̈ U+0041 U+0308	Ä U+00C4	A ̈ U+0041 U+0308
A ̈ U+0041 U+0308	Ä U+00C4	A ̈ U+0041 U+0308	Ä U+00C4	A ̈ U+0041 U+0308
̈ A U+0308 U+0041	̈ A U+0308 U+0041	̈ A U+0308 U+0041	̈ A U+0308 U+0041	̈ A U+0308 U+0041
ﬁ U+FB01	ﬁ U+FB01	ﬁ U+FB01	f i U+0066 U+0069	f i U+0066 U+0069
3 ⁵ U+0033 U+2075	3 ⁵ U+0033 U+2075	3 ⁵ U+0033 U+2075	3 5 U+0033 U+0035	3 5 U+0033 U+0035
Å U+212B	Å U+00C5	A ̊ U+0041 U+030A	Å U+00C5	A ̊ U+0041 U+030A
™ U+2122	™ U+2122	™ U+2122	T M U+0054 U+004D	T M U+0054 U+004D

Table 2.6 Unicode normalization forms: examples

before normalization	***NFC***	***NFD***	***NFKC***	***NFKD***
Ⅳ U+2163	Ⅳ U+2163	Ⅳ U+2163	I V U+0049 U+0056	I V U+0049 U+0056
רּ U+FB48	ר ּ U+05E8 U+05BC	ר ּ U+05E8 U+05BC	ר ּ U+05E8 U+05BC	ר ּ U+05E8 U+05BC
가 U+AC00	가 U+AC00	ᄀ ᅡ U+1100 U+1161	가 U+AC00	ᄀ ᅡ U+1100 U+1161
ぢ U+3062	ぢ U+3062	ち ゙ U+3061 U+3099	ぢ U+3062	ち ゙ U+3061 U+3099
㋉ U+32C9	㋉ U+32C9	㋉ U+32C9	1 0 月 U+0031 U+0030 U+6708	1 0 月 U+0031 U+0030 U+6708

2.5 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>
```

3 Indexing Metadata

3.1 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 following document info entries are predefined in PDF:

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

In addition to predefined document info 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 an increasing number of 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 practise 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 will usually be 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. More information on XMP can be found at

www.adobe.com/devnet/xmp

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.

XMP metadata can be displayed and edited in Acrobat with custom XMP panels. These are descriptions of a custom user interface for specific XMP metadata. Sample XMP panels are installed with TET PDF IFilter; more information and installation instructions can be found at

www.pdflib.com/knowledge-base/xmp-metadata/xmp-panels/

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

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, 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 3.1 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 3.1 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	the number of fonts in a document
length:images	string	the 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
fields[*]/V	string	text contents (value) of all form fields on all pages in the document

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"/>
```

3.2 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 *Globally Unique Identifier* GUID (sometimes also called *Universally Unique Identifier*, UUID), a unique 128-bit identifier which has the same value for all properties in a property set. Details on GUIDs can be found at

  www.itu.int/ITU-T/studygroups/com17/oid/X.667-E.pdf

  There are various tools available for creating GUIDs; you can also use online services, e.g. the one which is available at

  www.itu.int/ITU-T/asn1/uuid.html

  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 41, 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 3.4, »Custom Metadata Properties«, page 40.

3.3 Predefined Metadata Properties

The following 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. Common examples are *System.Author* and *System.Title*, and *System.Document.DateCreated*. For a list and description of shell properties see

  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. Examples are the PDF version number, bookmark contents, or page size.
- *Document XMP* properties cover all predefined document properties in the XMP 2005 specification, see

  www.adobe.com/devnet/xmp

- *Image XMP* properties cover XMP properties which are attached to images in the document. Image properties are also derived from the XMP 2005 specification.
- *Internal properties* are auxiliary properties which are not intended for production use, but as development and debugging aids. They include software version numbers and the time of indexing.

A full list of predefined properties along with their GUIDs and user-friendly names can be found in Appendix A, »Predefined Metadata Properties«.

Properties from these collections do not have to be configured individually. However, they must be enabled as required since not all collections are enabled by default (see below). You can override predefined properties by assigning the corresponding GUID+ID combination to a custom property.

XML configuration for enabling property set collections. Some or all of the 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* and *Internal* property set collections are enabled, while the *documentXMP, imageXMP,* and *PDF* property set collections are disabled. Custom properties are enabled implicitly if one or more custom properties are specified.

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

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

3.4 Custom Metadata 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.

Planning custom metadata properties. In order to specify custom properties you must consider the following aspects (see »Property identification and GUIDs«, page 38, 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 the GUID.
- The property *identifier* is a unique integer which identifies the property within its property set. Property identifiers in a set start at 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 it can be used instead of the identifier, friendly names do not work with all IFilter clients.
- Property *source:* properties can be populated from document metadata or general PDF information according to Section 3.1, »Sources of Metadata in PDF«, page 35.
- 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 will have precedence (i.e. subsequent sources will be ignored), or whether data from all non-empty sources will be collected.
- Specify whether the property will be emitted as a *vector*, i.e. multiple values will be handed to the IFilter interface in an array structure instead of a flat value (see Section 3.5, »Multivalued Properties«, page 42).
- A *prefix* which will be prepended to the property name if properties are indexed as part of the full text (see Section 3.6, »Index Metadata Properties as Text«, page 43).

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:201109231858              GMT
D:201109231058-08'00'       same instant expressed in U.S. Pacific Standard Time
D:201109231958+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 2005 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:

```
2011-09-23T18:58:30Z          GMT
2011-09-23T13:58:30-05:00     same instant expressed in US Eastern Standard Time
2011-09-23T19:58:30+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="33333333-5354-4c72-992C-5DF2AA4E7CBA">
        <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 3.2, except that the leading *TET_* and the trailing underscore '_' will be 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.

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

3.5 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 document keywords. The multiplicity of a property may have several reasons:

- 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 will be expanded to any number of individual entries, e.g. *bookmarks[*]/Title*.
- The property is populated from more than one source, and the *precedence* attribute of the property has the value *try-all*, e.g. *pdf:Keywords* and */Info/Keywords*. However, the default *precedence=first-wins* processes only the first non-empty property source.

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

Alternatively, multivalued properties can be provided as vectors to the IFilter client. This means that a single array entity will be emitted which can hold 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>
```

3.6 Index Metadata 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. Fixed prefixes will be used 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 3.2 for examples).

Table 3.2 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.TETPDFIFilter.pdfversion	TET_pdfversion_
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>
```

3.7 Ignore Page Contents in Favor of Metadata

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">
```

4 Metadata Handling in IFilter Clients

4.1 Metadata in Windows Search

Create a property description file. Windows Search accesses the Windows property system which holds descriptions of predefined and custom metadata properties. In order to use custom metadata properties with Windows Search you must prepare a property description file (often called *.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. 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

Sample property description files are installed with TET PDF IFilter. The following (incomplete) example demonstrates a few property descriptions:

```
<propertyDescription name="PDFlib.TETPDFIFilter.fontcount"
  formatID="{5eac0060-1ba4-11dd-92c4-0002a5d5c51b}" propID="2">
  <typeInfo type="Int32" isInnate="true" isViewable="true" isQueryable="true"/>
  <labelInfo label="fontcount" sortDescription="LowestHighest"/>
  <searchInfo inInvertedIndex="true" isColumn="true" columnIndexType="OnDisk"/>
</propertyDescription>
<propertyDescription name="PDFlib.TETPDFIFilter.weblink"
  formatID="{5eac0060-1ba4-11dd-92c4-0002a5d5c51b}" propID="6">
  <typeInfo type="String" isInnate="true" multipleValues="true" isViewable="true"
    isQueryable="true"/>
  <labelInfo label="weblink" sortDescription="AToZ"/>
  <searchInfo inInvertedIndex="true" isColumn="true" columnIndexType="OnDisk"/>
</propertyDescription>
```

Note that the *multipleValues="true"* attribute is crucial for multivalued properties (see Section 3.5, »Multivalued Properties«, page 42).

Predefined properties. A property description file *predefined_properties.propdesc* for all predefined properties (see Appendix A, »Predefined Metadata Properties«) is installed with TET PDF IFilter. 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 3.3, »Predefined Metadata Properties«, page 39), and register the property descriptions (see »Register metadata properties in Windows«, page 47) before you can use them in queries.

XML configuration for Windows Search. Table 4.1 lists requirements and recommendations related to the XML configuration for TET PDF IFilter when working with Windows Search.

Register metadata properties in Windows. In order to use property queries in Windows Search the corresponding property descriptions must be registered in the Windows property system. You can register property descriptions with the command-line

Table 4.1 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 41).*
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 41).*
PropertySetCollection	***shell***	*Should be* `true` *to support the shell properties known to Windows Search (note that* `true` *is the default anyway).*

program *registerpropdesc.exe* which is installed with TET PDF IFilter. Note that this tool works only if Windows Search is installed on the system. Supply the file name of a property description file to this tool to register it with the Windows property system (make sure to enclose the path with double quotes if it contains space characters):

```
registerpropdesc "predefined_properties.propdesc"
```

See also the section »Running privileged commands«, page 6. The Windows property system stores the names of property description files in the following registry keys (and increasing numbers in the last component of the key):

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

The *registerpropdesc* tool will emit an error message and a *HRESULT* error value if the property file could not successfully be registered (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 *registerpropdesc.exe* to analyze the problem. A list of *HRESULT* values and explanations can be found at

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

A shared source command-line tool called *prop.exe* offers functionality similar to *registerpropdesc*, plus additional features for dealing with the Windows Property System:

prop.codeplex.com

Requirements for registering property descriptions. Note the following important requirements when registering property description files for Windows Search:

- 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:

```
registerpropdesc "predefined_properties.propdesc"
net stop wsearch
net start wsearch
```

```
...rebuild the catalog (see »Starting/Stopping Windows Search«, page 8)...
```

- Registered properties must not conflict in GUID+ID or name with existing Windows properties (including the shell properties listed in Appendix A, »Predefined Metadata 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 proper privileges to write into the *HKEY_LOCAL_MACHINE* registry hive to run the *registerpropdesc* 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 files. However, two property description files must not include descriptions of the same property (as characterized by GUID+ID).

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

```
registerpropdesc -u "c:\property\acme.propdesc"
```

Search for metadata properties. Once you configured custom metadata properties, you can search for properties. Table 4.2 contains examples for property queries. The names of all predefined properties for Windows Search are listed in Appendix A, »Predefined Metadata Properties«.

Shell properties can be queried interactively according to the samples shown in Table 4.2. The Advanced Query Syntax (AQS) for interactive searches supports custom properties only in Windows 7 and above. In Windows XP/Vista you must index metadata properties as text (see Section 3.6, »Index Metadata Properties as Text«, page 43) in order to include custom metadata in interactive searches, or use SQL-based searches (see below).

Table 4.2 Metadata query examples for Windows Search 3.0 and above

Search term example	***Explanation***
`author:Doyle`	*author contains* `Doyle`
`author:"Conan Doyle"`	*author contains the words* `Conan Doyle`
`author:Doy`	*author starts with* `Doy`
`size: < 500000`	*document size is less than 500,000 bytes*
`date: <=4/7/12`	*modification date on or before April 7, 2012 (date format depends on system settings)*
`System.Document.DateCreated: = 09/23/2002`	*creation date is September 23, 2002 (date format depends on system settings)*
`System.MIMEType: ="application/pdf"`	*list all PDF documents in the index*
`System.Document.PageCount: = 144`	*document contains 144 pages*
`PDFlib.TETPDFIFilter.bookmark:Index`	*document contains a bookmark with the text* `Index`

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

PowerShell for submitting SQL queries. Sample PowerShell query scripts are installed with TET PDF IFilter. The following PowerShell script lists all bookmarks 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.TETPDFIFilter.bookmark`" FROM SYSTEMINDEX ",
$objConnection)

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

The following PowerShell script lists all documents where at least one bookmark contains the text *alpha:*

```
$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.TETPDFIFilter.bookmark`" = SOME ARRAY ['alpha']", $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.TETPDFIFilter.width`"* instead of *"PDFlib.TETPDFIFilter.width"*. Many examples below use array queries for vector properties (see Section 3.5, »Multivalued Properties«, page 42). 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.TETPDFIFilter.pdfa" =
  'PDF/A-1:2005'
  ```

- List all documents where at least one of the bookmarks begins with *alph:*

  ```
  SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE "PDFlib.TETPDFIFilter.bookmark"
  LIKE SOME ARRAY ['alph%']
  ```

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

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

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

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

- List all documents with at least one page which has *width*=595. Note the single quote characters around 595 since *width* has type *Double*; the quotes are not required for type *Int32:*

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE "PDFlib.TETPDFIFilter.width" =
SOME ARRAY ['595']
```

- List all documents with at least one page with *width*=200 and at least one page with *height*=150:

```
SELECT System.ItemPathDisplay FROM SYSTEMINDEX WHERE "PDFlib.TETPDFIFilter.width" =
SOME ARRAY ['200'] AND "PDFlib.TETPDFIFilter.height" = SOME ARRAY ['150']
```

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

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

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

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

4.2 Metadata in SharePoint and Search Server

Metadata property handling in Search Server is almost identical to metadata handling in SharePoint. This section covers configuration steps for both products. In a few cases product-specific details are mentioned explicitly.

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 TET PDF IFilter use with SharePoint. Table 4.3 lists requirements and recommendations related to the XML configuration for TET PDF IFilter when working with SharePoint.

Table 4.3 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 3.5, »Multivalued Properties«, page 42).*
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 38).*

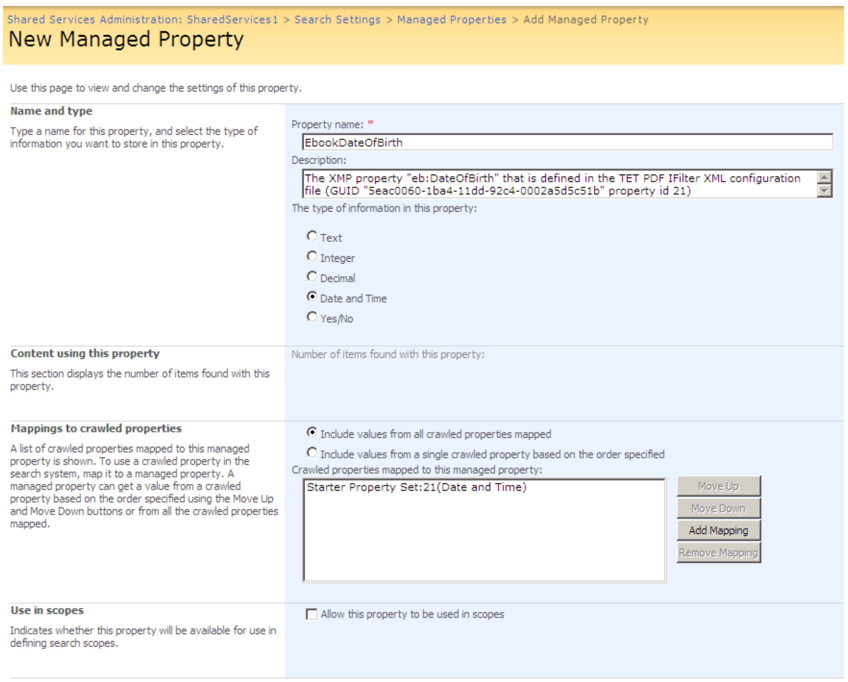
Configure custom metadata properties. Proceed as follows to prepare custom metadata properties for indexing:

- Run a full crawl. During the crawl TET PDF IFilter will report 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 will generate synthetic names (e.g. *»Category 1«, »Category 2«, etc.)* for all new property categories. A PowerShell script which is installed with TET PDF IFilter will create a list of category names and the corresponding GUIDs. Run the script as follows:

  ```
  list_categories.ps1 <site URL>
  ```

  In the resulting list output you can identify categories via their 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 property 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 returned by the *list_categories.ps1* script:
  SharePoint: Click *Start, SharePoint 3.0 Central Administration*. In the column on the left-hand side click *Shared Services Administration: SharedServices1* (or similar), *Search Settings, Metadata Property Mappings*. Again in the column on the left-hand side click *Crawled Properties*. In the *Crawled Properties View* open the drop-down menu that appears for a category name and click *Edit Category*. In the *Edit Category* dialog change the name of the category.
- SharePoint: Click *Start, SharePoint 3.0 Central Administration*. In the column on the left-hand side click *Shared Services Administration: SharedServices1* (or similar), *Search Settings, Managed Properties*
  Search Server: Click *Start, All Programs, Microsoft Search Server, Search Server 2008 Administration*. This will open a page titled *Metadata Property Mappings*.
- Click *New Managed Property* (see Figure 4.1).
- 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 *Crawled properties mapped to this managed property*.
- The drop-down menu *Select a category* will filter the list of crawled properties which are displayed in the list box titled *Select a crawled property*. 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 Appendix , »Configure custom


Shared Services Administration: SharedServices1 > Search Settings > Managed Properties > Add Managed Property

New Managed Property

Use this page to view and change the settings of this property.

Name and type

Type a name for this property, and select the type of information you want to store in this property.

Property name: *

EbookDateOfBirth

Description:

The XMP property "eb:DateOfBirth" that is defined in the TET PDF IFilter XML configuration file (GUID "5eac0060-1ba4-11dd-92c4-0002a5d5c51b" property id 21)

The type of information in this property:

Text / Integer / Decimal / [✓ Date and Time] / Yes/No

Content using this property

This section displays the number of items found with this property.

Number of items found with this property:

Mappings to crawled properties

A list of crawled properties mapped to this managed property is shown. To use a crawled property in the search system, map it to a managed property. A managed property can get a value from a crawled property based on the order specified using the Move Up and Move Down buttons or from all the crawled properties mapped.

[✓ Include values from all crawled properties mapped]

Include values from a single crawled property based on the order specified

Crawled properties mapped to this managed property:

Starter Property Set:21(Date and Time)

Use in scopes

Indicates whether this property will be available for use in defining search scopes.

☐ Allow this property to be used in scopes

Fig. 4.1 Adding managed properties in SharePoint

metadata properties«. Select a newly crawled property, assign a name to the managed property, and save it.

Sample PowerShell script output. The following listing shows the sequence of running the PowerShell scripts for the *litwaredemo* site:

```
PS C:\> & "C:\Program Files\PDFlib\TET PDF IFilter 5.0 64-bit\IFilter clients\Sharepoint\
   list_property_categories.ps1" http://litwaredemo

"Category 1" 007867f0-c59b-43fc-ab1e-8eee77057254
"Category 2" 1e3ee840-bc2b-476c-8237-2acd1a839b22
"Category 5" c60e822a-074f-4bd5-9889-6ebd372f2000
"Category 6" 17eb8447-fc9b-4d4d-81df-31e9aa770cbf
"Category 7" 5eac0060-1ba4-11dd-92c4-0002a5d5c51b
"Dublin Core" d92bb3ca-ce2b-4b9b-972a-5bf54b468171
...
```

The generated list shows that *»Category 7«* belongs to the property set defined in *starter_advanced_search.xml*. Rename it to *»Starter Property Set«* as described above.

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 will describe the XML with examples below:

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 4.4, 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>
```

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,

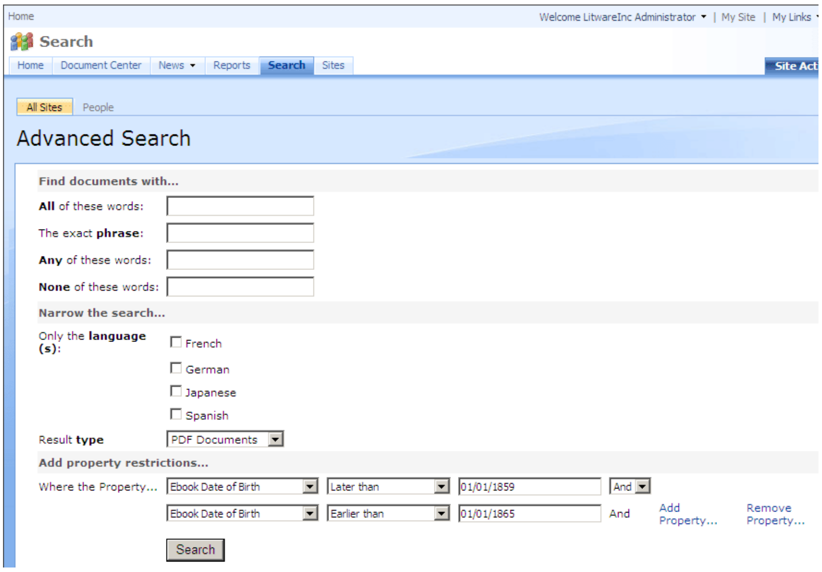
Table 4.4 *Property data types for SharePoint*

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

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 3.4, »Custom Metadata Properties«, page 40). 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.



Home   Welcome LitwareInc Administrator ▾ | My Site | My Links ▾

Search

Home | Document Center | News ▾ | Reports | Search | Sites   Site Act

All Sites | People

Advanced Search

Find documents with...

All of these words:
The exact **phrase**:
Any of these words:
None of these words:

Narrow the search...

Only the **language(s)**: ☐ French ☐ German ☐ Japanese ☐ Spanish

Result **type**: PDF Documents

Add property restrictions...

Where the Property... Ebook Date of Birth | Later than | 01/01/1859 | And
Ebook Date of Birth | Earlier than | 01/01/1865 | And   Add Property...   Remove Property...

Search

Fig. 4.2
Advanced Search Page in SharePoint with custom property

- 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 4.2 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 4.5 contains examples for property queries. Property queries are constructed according to the following simple scheme:

```
<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 4.5 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`

4.3 Metadata 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 3.6, »Index Metadata Properties as Text«, page 43).

XML configuration for SQL Server. Table 4.6 lists requirements and recommendations related to the XML configuration for TET PDF IFilter when working with SQL Server.

Table 4.6 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 3.6, »Index Metadata Properties as Text«, page 43).*
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
```

5 Troubleshooting

5.1 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 and is also included in the Windows SDK for Windows 7. 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 will not be 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 32-bit (C:\Program Files\PDFlib\TET PDF IFilter 5.0
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 XP/Vista/7/8/10, 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 64). 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_mem:
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).

5.2 Problems with TET PDF IFilter Operation

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

Identifying problematic documents. Depending on the IFilter client in use, logging entries may or may not include the names of affected files, and file names may or may not be useful. For example, Indexing Server does include file names, while Windows Search does not include them. On the other hand, SharePoint downloads the documents via HTTP and creates a temporary local copy. The event log contains the temporary local file names which are unrelated to the original URLs. As an aid for identifying the affected PDF documents the event log entries contain the file size in bytes. You can use the search engine itself to quickly identify affected documents.

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

  ```
  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 6.2, »XML Elements and Attributes«, page 69).

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.

5.3 PDF Documents are not or not completely indexed

SharePoint Server and Search Server are 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.

5.3.1 Limitations in SharePoint 2010 and SharePoint 2013

SharePoint imposes several limitations on document indexing. You can find more information about fixed and configurable limits in SharePoint 2013 at

https://technet.microsoft.com/en-us/library/cc262787%28v=office.15%29.aspx

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 2013 is 64 MB.
- 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 2013 has a hard-coded limit of 2 million characters. This limit cannot be modified.

Changing maximum file size and growth factor for SharePoint 2010 and 2013. Apply the following command in the *SharePoint 2013 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")
```

5.3.2 Limitations in earlier SharePoint Versions

The default value of *MaxDownloadSize* is 16 MB, the default value of *MaxGrowFactor* is 4 (see Section 5.3.1, »Limitations in SharePoint 2010 and SharePoint 2013«, page 62, for an explanation of these values), resulting in a maximum of 64 MB of extracted text per document. Depending on the exact product and version the *MaxDownloadSize* and *MaxGrowFactor* registry entries can be found under the following keys in the Windows registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office Server\<version>
        \Search\Applications\<GUID>\Gathering Manager

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Shared Tools\Web Server Extensions\<version>
        \Search\Applications\<GUID>\Gathering Manager
```

```
HKEY_LOCAL_MACHINE\Software\Microsoft\SPSSearch\Gathering Manager
```

The *GUID* will vary from installation to installation.

Chunk buffer size. Another limitation affects the total number of unique words per document which can be indexed. The value *CB_ChunkBufferSizeInMegaBytes* determines the space which is reserved for the collection of unique words per document (unit: MB, default: 8).

Bytes reserved for document. The value *CB_MinBytesReservedForDoc* depends on the *CB_ChunkBufferSizeInMegaBytes* value. It should be 2 MB less than the value of *CB_ChunkBufferSizeInMegaBytes*, although this relation is not true for the default values (unit: bytes, default: 3,145,728).

Depending on the exact product and version the *CB_ChunkBufferSizeInMegaBytes* and *CB_MinBytesReservedForDoc* registry entries can be found under the following keys in the Windows registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office Server\<version>
        \Search\Global\Gathering Manager

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Shared Tools\Web Server Extensions\<version>
        \Search\Global\Gathering Manager
```

A Microsoft support article which describes these values for Microsoft Office SharePoint Server 2007 can be found here:

support.microsoft.com/kb/970776/EN-US

5.3.3 Memory Limits for Search Server

The crawling process for Search Server 2008 is subject to certain memory limits which may prevent successful crawling of some documents. These limits can be controlled with registry entries:

- The registry key *DedicatedFilterProcessMemoryQuota* places a limit on memory usage.
- If an IFilter uses an amount of memory that exceeds the value of the registry key *FilterProcessMemoryQuota*, the crawler kills the process. Microsoft recommends to increase the default value if a 64-bit build of Search Server is used and the index server has more than 4GB of physical memory.

The registry entries mentioned above can be found under the following key in the Windows registry:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office Server\<version>
        \Search\Global\Gathering Manager
```

For more details please refer to the following article:

technet.microsoft.com/en-us/library/dd630760.aspx

5.4 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 Vista/7: Click *Start*, in the *Start Search* box type *Event Viewer* (or the corresponding localized term, e.g. *Ereignisanzeige* in German versions of Windows), click on the *Event Viewer* program. In the Event Viewer window click on *Windows Logs, Application*.
  Windows 8/10: Open the Windows *Search charm* by pressing the Windows key and *F*, select *Settings* under the Search text entry field and enter *View event logs* (or the corresponding localized term, e.g. *Ereignisprotokolle* in German versions of Windows) in the Search text entry field. Click on the *View event logs* icon that turns up as the search result. In the Event Viewer window click on *Windows Logs, Application*.
- TET PDF IFilter filtering events create an entry with source *TET PDF IFilter*. Double-click on the line containing the error and examine the error 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\5.0\logging
```

to a DWORD value according to Table 5.1. The logging level is set to 1 by default. Note that a PDF document may result in more than one error message in the event log, e.g. each damaged page may generate an entry.

Table 5.1 Logging levels for the Windows event log

Level	*Summary*	*Logged events*
0	*silent*	*None: all error messages are suppressed*
1 (default)	*errors*	*All failed TET function calls and exceptions thrown by TET, e.g. invalid or missing license key, encrypted PDFs which require a user password, severely damaged PDF documents which cannot be repaired; problems reading the registry; problems parsing the configuration file.*
2	*activity*	*Like 1, plus all* `Load()` *and* `Init()` *IFilter interface calls; XML configuration file reading*
3	*details*	*Like 2, plus details on IFilter interface calls for retrieving text and properties including LCID. This level creates a large number of log entries.*

Which properties and which text are emitted by TET PDF IFilter 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. Again, 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}\3 (Unknown)
    idChunk = 1
    BreakType = 2 (Sentence)
    Flags (chunkstate) =  (Value)
    Locale = 1031 (0x407)
    IdChunkSource = 1
    cwcStartSource = 0
    cwcLenSource = 0

VALUE: ---------------------------------------------------------------
Type = 31 (0x1f), VT_LPWSTR
Value = "4.0"

CHUNK: ---------------------------------------------------------------
    Attribute = {007867F0-C59B-43FC-AB1E-8EEE77057254}\4 (Unknown)
    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 6, »XML Configuration File«, page 67):

  ```
  <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 will be 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=tet.log classes={filesearch=3}"
  ```

6 XML Configuration File

6.1 Working with Configuration Files

The operation of TET PDF IFilter can be controlled with an XML configuration file. Sample configuration files are installed with TET PDF IFilter.

Specifying the location of the configuration file. The configuration file can 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 XML configuration files outside the installation directory of TET PDF IFilter. This way the configuration will survive 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 will be used. If the configuration file specified in the registry entry cannot be opened, a warning will be written to the application event log, and the default configuration will be used for indexing. If XML parsing of the configuration file fails, a warning will be written to the event log, and no indexing will be done.

Note The installer does not create a registry entry for a configuration file. This must be done by the user if required.

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 above registry entry will be searched in the 32-bit and 64-bit registry, respectively.

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

Predefined XML configuration files. Several predefined configuration files are installed with TET PDF IFilter:

- The file *default.xml* describes the internal default settings of TET PDF IFilter. It may serve as a starting point for creating a customized configuration file.
- The file *starter.xml* contains property definitions which can be used with the starter samples that are installed with TET PDF IFilter.

XML namespace and schema description. Table 6.1 lists the elements and attributes which are available in the XML configuration file. The namespace URI for the XML configuration is

```
http://www.pdflib.com/XML/TET_PDF_IFilter3/TET_PDF_IFilter_Config-3.0.xsd
```

Note The schema namespace URI and download location contain the version number 3 since the current schema is compatible with the schema used in TET PDF IFilter 3.0.

An XSD schema description for the XML configuration language is installed with TET PDF IFilter, and can also be found at the URI above which serves as namespace identifier. 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 will be translated to the current system default locale.
- GUID: unique 128-bit identifier in hexadecimal notation according to ITU-T Rec. X.667 (see *www.itu.int/ITU-T/studygroups/com17/oid/X.667-E.pdf)*. The parts must be separated by dash characters »-«. There are various tools available for creating GUIDs; you can also use online services, e.g. the one which is available at

  www.itu.int/ITU-T/asn1/uuid.html
- pCOS path: extended pCOS path describing a PDF object, see pCOS Reference and the pCOS extensions described in Section »Extended pCOS paths«, page 36
- 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.

6.2 XML Elements and Attributes

Table 6.1 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 3, »Indexing Metadata«, page 35.

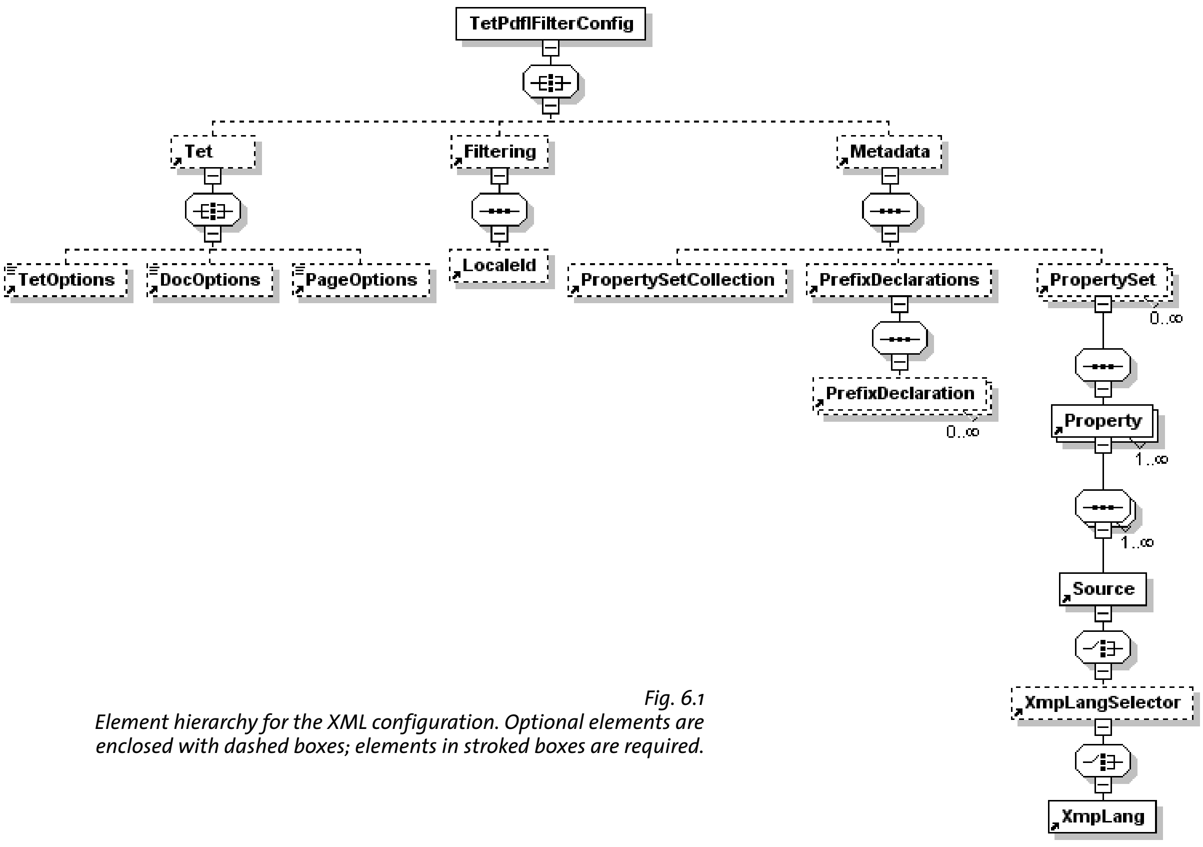


Fig. 6.1
Element hierarchy for the XML configuration. Optional elements are enclosed with dashed boxes; elements in stroked boxes are required.

Table 6.1 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 64). 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* ***indexNestedPdf*** *(Boolean; optional) Process PDF attachments recursively (see Section 2.1, »PDF Document Domains«, page 17). Default:* `true` ***indexPageContents*** *(Boolean; optional) Indicates whether or not the contents of PDF pages will be indexed. Disabling page content indexing may be useful in scenarios where the search is exclusively driven by metadata properties. Default:* `true` ***metadataHandling*** *(Choice; optional) Select the type of metadata handling (see Section 3.6, »Index Metadata Properties as Text«, page 43). 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 3.2, page 43, 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` *will be used to identify properties if both of these attributes for the* `Property` *element are present. Default:* `true`

Table 6.1 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 2.2, »Automatic Language Detection«, page 22). 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** (TET PDF IFilter 4.0) Determine LCID based on script. **latin** (LCID; optional) LCID for Latin text. Default: 0x0409 English (US) **useCatalogLang** (Boolean; optional; TET PDF IFilter 4.0) 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 3.4, »Custom Metadata Properties«, page 40). 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 6.1 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 3.4, »Custom Metadata Properties«, page 40).* *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 will be emitted as a single vector entity, regardless of the number of values.* *If* `false`*, the property will be emitted as a flat value. If more than one source item was found, multiple flat properties will be 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`. ***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.* ***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 time) Specify filtering of a custom set of properties with the same GUID (see Section 3.4, »Custom Metadata Properties«, page 40).* *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.*
PropertySet-Collection *parent: Metadata*	*(May appear zero or one time) Specify filtering of predefined property set collections (see Section 3.3, »Predefined Metadata Properties«, page 39). 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:* `false` ***shell*** *(Boolean; required) Emit shell properties. Default:* `true`

Table 6.1 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 below 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)* ***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 will only be used if* `pdfobject` *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 the root element) Root element of the XML configuration file. Supported attribute:* ***version*** *(String; optional; TET PDF IFilter 4.0) Specify the version of TET PDF IFilter for which this configuration was written. Since the TET PDF IFilter 3 configuration file didn't support this attribute the default is* `3.0`*. New configurations should include this attribute with the appropriate version identifier (*`4.0` *for TET PDF IFilter 4). Default:* `3.0`
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.*

6.3 Sample Configuration File

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
        XML configuration file for TET PDF IFilter
        (c) PDFlib GmbH 2008-2015 www.pdflib.com
        This file must be configured in the following registry key:
        HKEY_LOCAL_MACHINE\SOFTWARE\PDFlib\TET PDF IFilter5\configfile
-->

<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="4.0">

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

<n:Filtering 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="false"
            shell="true"/>

        <n:PropertySet guid="b01ca440-1b9f-11dd-8b87-0002a5d5c51b">
            <n:Property identifier="2">
                <n:Source pdfObject="/Info/Producer"/>
            </n:Property>
        </n:PropertySet>
</n:Metadata>

</n:TetPdfIFilterConfig>
```

A Predefined Metadata Properties

The properties listed in Table A.1 are built into TET PDF IFilter, and therefore do not require any custom configuration. Although the actual property definitions are built into TET PDF IFilter, you must enable the desired property set collections in the XML configuration file (see Section 3.3, »Predefined Metadata Properties«, page 39) in order to use these properties. For Windows Search you must register the properties with *registerpropdesc.exe* tool.

Table A.1 Property handling in TET PDF IFilter: predefined property set collections

property name for Windows Search; also used to derive a prefix if the property is indexed as text	***data type***	***multi-valued***	***property set GUID/property ID***	***source: XMP property or pCOS path***
Shell property set collection				
System.Document.Contributor	*String*	*yes*	*F334115E-DA1B-4509-9B3D-119504DC7ABB/100*	*dc:contributor*
System.Document.DateCreated	*DateTime*	*no*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/12*	*xmp:CreateDate, /Info/CreationDate*
System.Document.DateSaved	*DateTime*	*no*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/13*	*xmp:ModifyDate, /Info/ModDate*
System.Document.DocumentID	*String*	*no*	*E08805C8-E395-40DF-80D2-54F0D6C43154/100*	*dc:identifier*
System.Document.PageCount	*Int32*	*no*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/14*	*length:pages*
System.Document.Version	*String*	*no*	*D5CDD502-2E9C-101B-9397-08002B2CF9AE/29*	*xmpMM:VersionID*
System.Search.Contents	*String*	*yes*	*B725F130-47EF-101A-A5F1-02608C9EEBAC/19*	*text contents of PDF pages*
System.Title	*String*	*no*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/2*	*dc:title["x-default"], /Info/Title*
System.Subject	*String*	*no*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/3*	*dc:description["x-default"], /Info/Subject*
System.Author	*String*	*yes*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/4*	*dc:creator, pdf:Author, xmp:Author, /Info/Author*
System.Keywords	*String*	*yes*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/5*	*pdf:Keywords, /Info/Keywords*
System.MIMEType	*String*	*no*	*0B63E350-9CCC-11D0-BCDB-00805FCCCE04/5*	`application/pdf` *(fixed)*
System.DateModified (IS: Write)	*DateTime*	*no*	*B725F130-47EF-101A-A5F1-02608C9EEBAC/14*	*xmp:ModifyDate, /Info/ModDate*
System.ApplicationName	*String*	*no*	*F29F85E0-4FF9-1068-AB91-08002B27B3D9/18*	*xmp:CreatorTool, /Info/Creator*
System.Kind	*String*	*no*	*1E3EE840-BC2B-476C-8237-2ACD1A839B22/3*	`Document` *(fixed)*

Table A.1 Property handling in TET PDF IFilter: predefined property set collections

property name for Windows Search; also used to derive a prefix if the property is indexed as text	data type	multi-valued	property set GUID/property ID	source: XMP property or pCOS path
PDF property set collection				
PDFlib.TETPDFIFilter.pdfversion (contains the PDF version multiplied by 10, e.g. »16«)	String	no	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/2	pdfversion
PDFlib.TETPDFIFilter.pdfa	String	no	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/3	pdfa
PDFlib.TETPDFIFilter.pdfx	String	no	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/4	pdfx
PDFlib.TETPDFIFilter.font	String	yes	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/5	fonts[*]/name
PDFlib.TETPDFIFilter.bookmark	String	yes	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/6	bookmarks[*]/Title
PDFlib.TETPDFIFilter.annotation	String	yes	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/7	pages[*]/annots[*]/Contents
PDFlib.TETPDFIFilter.width	Double	yes	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/8	pages[*]/width
PDFlib.TETPDFIFilter.height	Double	yes	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/9	pages[*]/height
PDFlib.TETPDFIFilter.producer	String	no	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/10	/Info/Producer
PDFlib.TETPDFIFilter.trapped	String	no	E544AFE6-13E2-40F1-A702-DCEBE8FB7B02/11	/Info/Trapped
XMP document metadata property set collection (from the XMP 2005 specification)				
Dublin Core				
PDFlib.TETPDFIFilter.dc.contributor	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/2	dc:contributor
PDFlib.TETPDFIFilter.dc.coverage	String	no	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/3	dc:coverage
PDFlib.TETPDFIFilter.dc.creator	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/4	dc:creator
PDFlib.TETPDFIFilter.dc.date	DateTime	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/5	dc:date
PDFlib.TETPDFIFilter.dc.description	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/6	dc:description
PDFlib.TETPDFIFilter.dc.format	String	no	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/7	dc:format
PDFlib.TETPDFIFilter.dc.identifier	String	no	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/8	dc:identifier
PDFlib.TETPDFIFilter.dc.language	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/9	dc:language
PDFlib.TETPDFIFilter.dc.publisher	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/10	dc:publisher
PDFlib.TETPDFIFilter.dc.relation	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/11	dc:relation

Table A.1 Property handling in TET PDF IFilter: predefined property set collections

property name for Windows Search; also used to derive a prefix if the property is indexed as text	data type	multi-valued	property set GUID/property ID	source: XMP property or pCOS path
PDFlib.TETPDFIFilter.dc.rights	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/12	dc:rights
PDFlib.TETPDFIFilter.dc.source	String	no	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/13	dc:source
PDFlib.TETPDFIFilter.dc.subject	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/14	dc:subject
PDFlib.TETPDFIFilter.dc.title	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/15	dc:title
PDFlib.TETPDFIFilter.dc.type	String	yes	D92BB3CA-CE2B-4B9B-972A-5BF54B468171/16	dc:type
XMP Basic				
PDFlib.TETPDFIFilter.xmp.Advisory	String	yes	C60E822A-074F-4BD5-9889-6EBD372F2000/2	xmp:Advisory
PDFlib.TETPDFIFilter.xmp.BaseURL	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/3	xmp:BaseURL
PDFlib.TETPDFIFilter.xmp.CreateDate	DateTime	no	C60E822A-074F-4BD5-9889-6EBD372F2000/4	xmp:CreateDate
PDFlib.TETPDFIFilter.xmp.CreatorTool	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/5	xmp:CreatorTool
PDFlib.TETPDFIFilter.xmp.Identifier	String	yes	C60E822A-074F-4BD5-9889-6EBD372F2000/6	xmp:Identifier
PDFlib.TETPDFIFilter.xmp.Label	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/7	xmp:Label
PDFlib.TETPDFIFilter.xmp.MetadataDate	DateTime	no	C60E822A-074F-4BD5-9889-6EBD372F2000/8	xmp:MetadataDate
PDFlib.TETPDFIFilter.xmp.ModifyDate	DateTime	no	C60E822A-074F-4BD5-9889-6EBD372F2000/9	xmp:ModifyDate
PDFlib.TETPDFIFilter.xmp.Nickname	String	no	C60E822A-074F-4BD5-9889-6EBD372F2000/10	xmp:Nickname
PDFlib.TETPDFIFilter.xmp.Rating	Int32	no	C60E822A-074F-4BD5-9889-6EBD372F2000/11	xmp:Rating
XMP Rights Management				
PDFlib.TETPDFIFilter.xmpRights.Certificate	String	no	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/2	xmpRights:Certificate
PDFlib.TETPDFIFilter.xmpRights.Marked	Boolean	no	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/3	xmpRights:Marked
PDFlib.TETPDFIFilter.xmpRights.Owner	String	yes	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/4	xmpRights:Owner
PDFlib.TETPDFIFilter.xmpRights.UsageTerms	String	yes	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/5	xmpRights:UsageTerms
PDFlib.TETPDFIFilter.xmpRights.WebStatement	String	no	0DE7A11C-E2C5-4EFA-8017-BECD888E7EC9/6	xmpRights:WebStatement
XMP Basic Job Ticket				
PDFlib.TETPDFIFilter.xmpBJ.JobRef	String	yes	EBC983EF-C1CF-45C8-A29E-993543A0ECFB/2	xmpBJ:JobRef

Table A.1 Property handling in TET PDF IFilter: predefined property set collections

property name for Windows Search; also used to derive a prefix if the property is indexed as text	data type	multi-valued	property set GUID/property ID	source: XMP property or pCOS path
XMP Paged-Text				
PDFlib.TETPDFIFilter.xmpTPg.NPages	Int32	no	7A9EB492-35AB-49FE-B364-A21FC9575C28/2	xmpTPg:NPages
PDFlib.TETPDFIFilter.xmpTPg.PlateNames	String	yes	7A9EB492-35AB-49FE-B364-A21FC9575C28/3	xmpTPg:PlateNames
Adobe PDF				
PDFlib.TETPDFIFilter.pdf.Keywords	String	no	17EB8447-FC9B-4D4D-81DF-31E9AA770CBF/2	pdf:Keywords
PDFlib.TETPDFIFilter.pdf.PDFVersion	String	no	17EB8447-FC9B-4D4D-81DF-31E9AA770CBF/3	pdf:PDFVersion
PDFlib.TETPDFIFilter.pdf.Producer	String	no	17EB8447-FC9B-4D4D-81DF-31E9AA770CBF/4	pdf:Producer
XMP image metadata property set collection (from the XMP 2005 specification)				
Photoshop				
PDFlib.TETPDFIFilter.images.photoshop.AuthorsPosition	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/2	photoshop:AuthorsPosition
PDFlib.TETPDFIFilter.images.photoshop.CaptionWriter	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/3	photoshop:CaptionWriter
PDFlib.TETPDFIFilter.images.photoshop.Category	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/4	photoshop:Category
PDFlib.TETPDFIFilter.images.photoshop.City	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/5	photoshop:City
PDFlib.TETPDFIFilter.images.photoshop.Country	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/6	photoshop:Country
PDFlib.TETPDFIFilter.images.photoshop.Credit	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/7	photoshop:Credit
PDFlib.TETPDFIFilter.images.photoshop.DateCreated	DateTime	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/8	photoshop:DateCreated
PDFlib.TETPDFIFilter.images.photoshop.Headline	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/9	photoshop:Headline
PDFlib.TETPDFIFilter.images.photoshop.Instructions	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/10	photoshop:Instructions
PDFlib.TETPDFIFilter.images.photoshop.Source	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/11	photoshop:Source
PDFlib.TETPDFIFilter.images.photoshop.State	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/12	photoshop:State
PDFlib.TETPDFIFilter.images.photoshop.SupplementalCategories	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/13	photoshop:SupplementalCategories
PDFlib.TETPDFIFilter.images.photoshop.TransmissionReference	String	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/14	photoshop:TransmissionReference
PDFlib.TETPDFIFilter.images.photoshop.Urgency	Int32	yes	C9F08C60-189D-11DD-8441-0002A5D5C51B/15	photoshop:Urgency

Table A.1 Property handling in TET PDF IFilter: predefined property set collections

property name for Windows Search; also used to derive a prefix if the property is indexed as text	**data type**	**multi-valued**	**property set GUID/property ID**	**source: XMP property or pCOS path**
Internal property set collection				
PDFlib.TETPDFIFilter.version	String	no	007867F0-C59B-43FC-AB1E-8EEE77057254/2	`5.0` (fixed)
PDFlib.TETPDFIFilter.tetversion	String	no	007867F0-C59B-43FC-AB1E-8EEE77057254/3	`5.0` (fixed)
PDFlib.TETPDFIFilter.indextime	DateTime	no	007867F0-C59B-43FC-AB1E-8EEE77057254/4	date and time of index run
PDFlib.TETPDFIFilter.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***
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
fax +49 • 89 • 452 33 84-99

If you have questions check the PDFlib mailing list
and archive at groups.yahoo.com/neo/groups/pdflib/info

Licensing contact
sales@pdflib.com

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