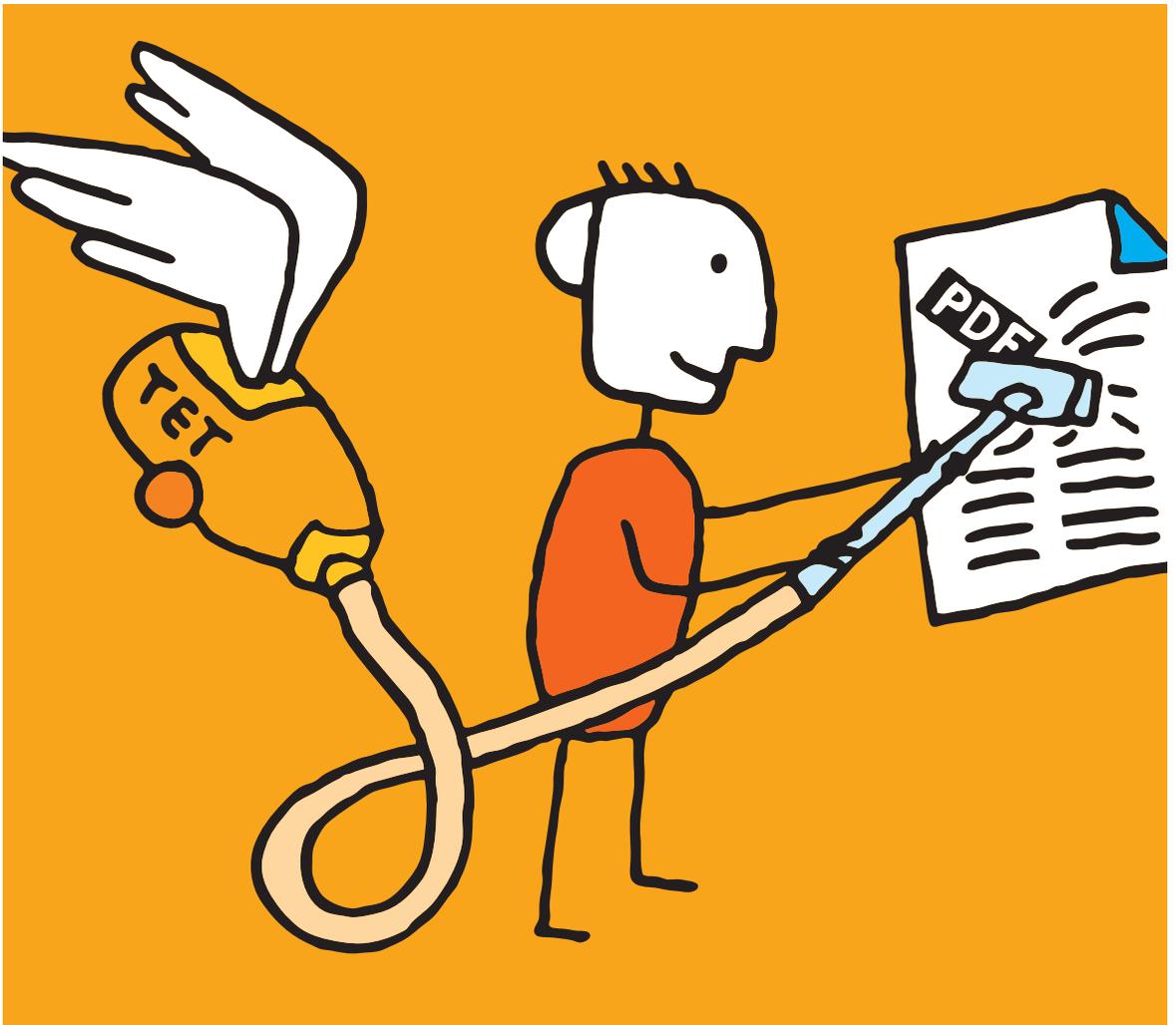


# Text and Image Extraction Toolkit (TET)

Version 5.0

**Toolkit zur Extraktion von Text, Bildern  
und anderen Elementen aus PDF**



Copyright © 2002–2015 PDFlib GmbH und Thomas Merz. Alle Rechte vorbehalten.  
Geschützt durch europäische und US-amerikanische Patente.

PDFlib GmbH  
Franziska-Bilek-Weg 9, D-80339 München  
www.pdflib.com  
Tel. +49 • 89 • 452 33 84-0  
Fax +49 • 89 • 452 33 84-99

Bei Fragen können Sie die PDFlib-Mailing-Liste abonnieren und sich deren Archiv ansehen unter:  
[groups.yahoo.com/neo/groups/pdflib/info](mailto:groups.yahoo.com/neo/groups/pdflib/info).

Vertriebsinformationen: [sales@pdflib.com](mailto:sales@pdflib.com)  
Support für Inhaber einer kommerziellen PDFlib-Lizenz: [support@pdflib.com](mailto:support@pdflib.com) (geben Sie bitte immer Ihre Lizenznummer an)

Der Inhalt dieser Dokumentation wurde mit größter Sorgfalt erstellt. PDFlib GmbH gibt jedoch keine Gewähr oder Garantie hinsichtlich der Richtigkeit oder Genauigkeit der Angaben in dieser Dokumentation und übernimmt keinerlei juristische Verantwortung oder Haftung für Schäden, die durch Fehler in dieser Dokumentation entstehen. Alle Warenbezeichnungen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen.

PDFlib und das PDFlib-Logo sind eingetragene Warenzeichen der PDFlib GmbH. PDFlib-Lizenznehmer sind dazu berechtigt, den Namen PDFlib und das PDFlib-Logo in ihrer Produktdokumentation zu verwenden. Dies ist jedoch nicht zwingend erforderlich.

Adobe, Acrobat, PostScript und XMP sind Warenzeichen von Adobe Systems Inc. AIX, IBM, OS/390, WebSphere, iSeries und zSeries sind Warenzeichen von International Business Machines Corporation. ActiveX, Microsoft, OpenType und Windows sind Warenzeichen von Microsoft Corporation. Apple, Macintosh und TrueType sind Warenzeichen von Apple Computer, Inc. Unicode und das Unicode-Logo sind Warenzeichen von Unicode, Inc. Unix ist ein Warenzeichen von The Open Group. Java und Solaris sind Warenzeichen von Sun Microsystems, Inc. HKS ist eine eingetragene Marke des HKS Warenzeichenverbands e.V.: Hostmann-Steinberg, K+E Printing Inks, Schmincke. Die Namen von anderen Produkten und Diensten können Warenzeichen von Unternehmen oder Organisationen sein, die hier nicht angeführt sind.

TET enthält modifizierte Bestandteile folgender Software anderer Hersteller:

Zlib Compression Library, Copyright © 1995-2012 Jean-loup Gailly und Mark Adler  
TIFFlib Image Library, Copyright © 1988-1997 Sam Leffler, Copyright © 1991-1997 Silicon Graphics, Inc.  
Kryptografische Software von Eric Young, Copyright © 1995-1998 Eric Young ([eyay@cryptsoft.com](mailto:eyay@cryptsoft.com))  
JPEG-Software der Independent JPEG Group, Copyright © 1991-1998, Thomas G. Lane  
Kryptografische Software, Copyright © 1998-2002 The OpenSSL Project ([www.openssl.org](http://www.openssl.org))  
XML-Parser Expat, Copyright © 1998, 1999, 2000 Thai Open Source Software Center Ltd  
ICU International Components for Unicode, Copyright © 1995-2012 International Business Machines Corporation und andere  
Reference sRGB ICC Farbprofil-Daten, Copyright (c) 1998 Hewlett-Packard Company

TET enthält den Message-Digest-Algorithmus MD5 von RSA Security, Inc.



# Inhaltsverzeichnis

## o Erste Schritte mit TET 7

- o.1 Installation der Software 7
- o.2 Eingabe des TET-Lizenzschlüssels 8

## 1 Einführung 11

- 1.1 Funktionsumfang von TET 11
- 1.2 Anwendungsmöglichkeiten von TET 14
- 1.3 Roadmap für Dokumentation und Beispiele 14
- 1.4 Was ist neu in TET 5.0? 16

## 2 TET-Kommandozeilen-Tool 19

- 2.1 Kommandozeilen-Optionen 19
- 2.2 Erstellen von TET-Kommandozeilen 22
- 2.3 Kommandozeilen-Beispiele 24
  - 2.3.1 Textextraktion 24
  - 2.3.2 Bildextraktion 24
  - 2.3.3 Erzeugung von TETML 25
  - 2.3.4 Erweiterte Optionen 25

## 3 Sprachbindungen für die TET-Bibliothek 27

- 3.1 Verarbeitung von Exceptions 27
- 3.2 C-Sprachbindung 29
- 3.3 C++-Sprachbindung 32
- 3.4 COM-Sprachbindung 34
- 3.5 Java-Sprachbindung 35
- 3.6 .NET-Sprachbindung 37
- 3.7 Objective-C-Sprachbindung 38
- 3.8 Perl-Sprachbindung 40
- 3.9 PHP-Sprachbindung 41
- 3.10 Python-Sprachbindung 43
- 3.11 REALbasic/Xojo-Sprachbindung 44
- 3.12 Ruby-Sprachbindung 45
- 3.13 RPG-Sprachbindung 47

## 4 TET-Konnektoren 49

- 4.1 Kostenloses TET Plugin für Adobe Acrobat 49
- 4.2 TET-Konnektor für die Suchmaschine Lucene 51

- 4.3 TET-Konnektor für den Solr Search Server 54
- 4.4 TET-Konnektor für Oracle 56
- 4.5 TET PDF IFilter für Produkte von Microsoft 59
- 4.6 TET-Konnektor für das Apache Tika-Toolkit 62
- 4.7 TET-Konnektor für MediaWiki 64

## 5 Konfiguration 67

- 5.1 Extrahieren von Inhalten aus geschützten PDF-Dokumenten 67
- 5.2 Ressourcenkonfiguration und Dateisuche 70
- 5.3 Empfehlungen für häufige Anwendungsfälle 74

## 6 Textextraktion 79

- 6.1 PDF-Dokumentdomänen 79
- 6.2 Geometrie von Seite und Text 84
- 6.3 Textfarbe 91
- 6.4 Chinesischer, japanischer und koreanischer Text 93
  - 6.4.1 CJK-Encodings und CMaps 93
  - 6.4.2 Wortgrenzen für CJK-Text 93
  - 6.4.3 Vertikale Schreibrichtung 93
  - 6.4.4 CJK-Dekomposition: Narrow, wide, vertical usw. 94
- 6.5 Bidirektionaler arabischer und hebräischer Text 96
  - 6.5.1 Allgemeine Bidi-Themen 96
  - 6.5.2 Nachbearbeitung von arabischem Text 96
- 6.6 Inhaltsanalyse 99
- 6.7 Layout-Analyse 103
- 6.8 Überprüfen auf leere Bereiche 106

## 7 Fortgeschrittene Unicode-Verarbeitung 107

- 7.1 Wichtige Unicode-Konzepte 107
- 7.2 Unicode-Vorbereitung (Filtern von Text) 110
  - 7.2.1 Filter für alle Granularitätsstufen 110
  - 7.2.2 Filter für die Granularität Wort oder höher 111
- 7.3 Unicode-Nachbearbeitung 112
  - 7.3.1 Unicode-Folding 112
  - 7.3.2 Unicode-Dekomposition 116
  - 7.3.3 Unicode-Normalisierung 120
- 7.4 Zeichen außerhalb der BMP und Surrogatpaare 122
- 7.5 Unicode-Zuordnung für Glyphen 123

## 8 Extraktion von Rasterbildern 131

- 8.1 Grundlagen der Bildextraktion 131

- 8.2 Bildextraktion 134**
  - 8.2.1 Platzierte Bilder und Bild-Ressourcen 134
  - 8.2.2 Seitenbasierte und ressourcenbasierte Bildextraktion 135
  - 8.2.3 Geometrie von platzierten Bildern 136
- 8.3 Zusammensetzen fragmentierter Bilder 139**
- 8.4 Filtern von Bildfragmenten 141**
- 8.5 Bildfarben und Masken 142**
  - 8.5.1 Farbräume 142
  - 8.5.2 Bildmasken und Transparenzmasken 144

## **9 TET Markup Language (TETML) 147**

- 9.1 Erzeugen von TETML 147**
- 9.2 TETML-Beispiele 149**
- 9.3 Steuerung von TETML-Informationen 153**
- 9.4 TETML-Elemente und das TETML-Schema 157**
- 9.5 TETML-Transformationen mit XSLT 165**
- 9.6 XSLT-Beispiele 169**

## **10 API-Referenz für die TET-Bibliothek 173**

- 10.1 Optionslisten 173**
  - 10.1.1 Syntax von Optionslisten 173
  - 10.1.2 Einfache Datentypen 176
  - 10.1.3 Geometrische Typen 178
  - 10.1.4 Encoding-Namen 179
- 10.2 Allgemeine Funktionen 180**
  - 10.2.1 Umgang mit Optionen 180
  - 10.2.2 Setup 182
  - 10.2.3 PDFlib Virtual Filesystem (PVF) 183
  - 10.2.4 Funktion zur Unicode-Konvertierung 185
  - 10.2.5 Verarbeitung von Exceptions 187
  - 10.2.6 Logging 188
- 10.3 Dokumentfunktionen 190**
- 10.4 Seitenfunktionen 199**
- 10.5 Funktionen zur Abfrage von Text- und Glyphendetails 209**
- 10.6 Funktionen zur Abfrage von Bildern 215**
- 10.7 Funktionen für TET Markup Language (TETML) 219**
- 10.8 pCOS-Funktionen 222**

## **A TET-Kurzreferenz 227**

## **B Änderungen an diesem Handbuch 229**

## **Index 231**



# o Erste Schritte mit TET

## o.1 Installation der Software

TET wird als MSI- oder komprimiertes Paket für Windows ausgeliefert oder als komprimiertes Archiv für alle anderen unterstützten Betriebssysteme. Alle TET-Pakete enthalten das TET-Kommandozeilen-Tool und die TET-Bibliothek/Komponente sowie Hilfsdateien, Dokumentation und Beispiele. Nachdem Sie TET installiert bzw. entpackt haben, ist zunächst folgendes zu beachten:

- ▶ Das TET-Kommandozeilen-Tool kann sofort ausgeführt werden. Die unterstützten Optionen werden in Abschnitt 2.1, »Kommandozeilen-Optionen«, Seite 19 beschrieben und werden außerdem angezeigt, wenn Sie das Tool ohne Optionen starten.
- ▶ Beim Einsatz der TET-Bibliothek/Komponente sollten Sie sich die für Ihre Entwicklungsumgebung relevanten Abschnitte von Kapitel 3, »Sprachbindungen für die TET-Bibliothek«, Seite 27 sowie die installierten Beispiele ansehen.

Wenn Sie eine kommerzielle TET-Lizenz erworben haben, müssen Sie den TET-Lizenzschlüssel eingeben, wie in Abschnitt 0.2, »Eingabe des TET-Lizenzschlüssels«, Seite 8 beschrieben.

**CJK-Konfiguration.** Zur Extraktion von chinesischem, japanischem oder koreanischem (CJK) Text benötigt TET entsprechende CMap-Dateien zur Umsetzung von CJK-Encodings in Unicode. Die CMap-Dateien sind in allen TET-Paketen enthalten und werden im Unterverzeichnis *resource/cmap* des TET-Installationsverzeichnisses installiert.

Auf anderen Systemen als Windows müssen Sie die CMap-Dateien manuell konfigurieren:

- ▶ Beim TET-Kommandozeilen-Tool kann der Name des Verzeichnisses für die CMap-Dateien mit der Option `--searchpath` übergeben werden.
- ▶ Bei der TET-Bibliothek/Komponente kann *searchpath* zur Laufzeit gesetzt werden:

```
tet.set_option("searchpath={/path/to/resource/cmap}");
```

Alternativ dazu können Sie den Zugriff auf die CJK-CMap-Dateien konfigurieren, indem Sie die Umgebungsvariable *TETRESOURCEFILE* auf eine UPR-Konfigurationsdatei setzen, die die passende *searchpath*-Definition enthält.

**Einschränkungen der Evaluierungsversion.** Das TET-Kommandozeilen-Tool sowie die Bibliothek können auch ohne kommerzielle Lizenz als voll funktionsfähige Evaluierungsversion verwendet werden. Ohne gültigen Lizenzschlüssel unterstützt TET alle Funktionen, verarbeitet aber nur PDF-Dokumente mit bis zu 10 Seiten und einer Größe bis zu 1 MB. Nicht lizenzierte TET-Versionen dürfen nicht im produktiven Einsatz, sondern nur für die Evaluierung des Produkts verwendet werden. Zum produktiven Einsatz von TET benötigen Sie einen gültigen Lizenzschlüssel.

## o.2 Eingabe des TET-Lizenzschlüssels

Zum produktiven Einsatz von TET benötigen Sie einen gültigen Lizenzschlüssel. Wenn Sie eine TET-Lizenz erworben haben, müssen Sie zur Verarbeitung von großen Dokumenten den Lizenzschlüssel eingeben. Verwenden Sie zur Eingabe des Lizenzschlüssels eine der folgenden Methoden.

*Hinweis* TET-Lizenzschlüssel sind plattformabhängig und können nur auf der Plattform eingesetzt werden, für die sie erworben wurden.

**Windows-Installationsroutine.** Wenn Sie die Windows-Installationsroutine verwenden, können Sie den Lizenzschlüssel bei der Produktinstallation eingeben. Dieser wird dann zur Registry hinzugefügt (siehe unten).

**Verwendung einer Lizenzdatei.** PDFlib-Produkte lesen den Lizenzschlüssel aus einer Lizenzdatei, bei der es sich um eine Textdatei gemäß dem unten angegebenen Format handelt. Sie können das in allen TET-Paketen enthaltene Template *licensekeys.txt* verwenden. Mit einem '#'-Zeichen beginnende Zeilen enthalten Kommentare und werden ignoriert. Die zweite Zeile enthält Informationen zur Version der Lizenzdatei selbst:

```
# Lizenz-Information für Produkte der PDFlib GmbH
PDFlib license file 1.0
TET 5.0 ...Ihr Lizenzschlüssel...
```

Sie können Lizenzschlüssel für verschiedene Produkte der PDFlib GmbH in die Lizenzdatei aufnehmen, wobei jeder Schlüssel in einer eigenen Zeile stehen muss. Sie können auch Lizenzschlüssel für verschiedene Plattformen aufnehmen, so dass die Lizenzdatei für mehrere Plattformen gemeinsam benutzt werden kann. Sie können Lizenzdateien folgendermaßen konfigurieren:

- ▶ Eine Datei namens *licensekeys.txt* wird an allen vorgegebenen Stellen gesucht (siehe »Standard-Dateisuchpfade«, Seite 9).
- ▶ Sie können die Option *licensefile* mit der API-Funktion *set\_option()* angeben:

```
tet.set_option("licensefile={/Pfad/zu/licensekeys.txt}");
```

Die Option *licensefile* muss unmittelbar nach der Instantiierung eines TET-Objekts gesetzt werden, das heißt nach dem Aufruf von *TET\_new()* (in C) oder der Erzeugung eines TET-Objekts.

- ▶ Übergeben Sie die Option *--teto*pt des TET-Kommandozeilen-Werkzeugs und die Option *licensefile* mit dem Namen einer Lizenzdatei:

```
tet --teto
```

pt "licensefile=/Pfad/zu licensekeys.txt" ...

Wenn der Pfadname Leerzeichen enthält, müssen Sie den Pfad mit Klammern umschließen:

```
tet --teto
```

pt "licensefile={/Pfad/zu/Ihrer Lizenzdatei.txt}" ...

- ▶ Sie können eine Umgebungsvariable anlegen, die auf die Lizenzdatei verweist. Unter Windows öffnen Sie die Systemsteuerung und wählen *System, Erweiterte System-einstellungen, Erweitert, Umgebungsvariablen*. Unter Unix verwenden Sie einen ähnlichen Befehl wie den folgenden:

```
export PDFLIBLICENSEFILE="/Pfad/zu/Ihren/Lizenzschlüsseln.txt"
```

Unter i5/iSeries kann die Lizenzdatei folgendermaßen angegeben werden (dieses Kommando kann im Startup-Programm *QSTRUP* angegeben werden und gilt für alle Produkte der PDFlib GmbH):

```
ADDENVVAR ENVVAR(PDFLIBLICENSEFILE) VALUE(<... path ...>) LEVEL(*SYS)
```

**Lizenzschlüssel in der Registry.** Unter Windows können Sie den Namen der Lizenzdatei auch in den folgenden Registry-Wert eintragen:

```
HKLM\SOFTWARE\PDFlib\PDFLIBLICENSEFILE
```

Alternativ können Sie den Lizenzschlüssel auch direkt in einen der folgenden Registry-Werte eintragen:

```
HKLM\SOFTWARE\PDFlib\TET5\license  
HKLM\SOFTWARE\PDFlib\TET5\5.0\license
```

Die MSI-Installationsroutine schreibt den bei der Installation übergebenen Lizenzschlüssel in den letzten dieser Einträge.

*Hinweis* Seien Sie vorsichtig beim manuellen Zugriff auf die Registry von 64-Bit-Windows-Systemen: wie üblich arbeiten 64-Bit-Programme mit der 64-Bit-Ansicht der Windows-Registry, während 32-Bit-Programme auf einem 64-Bit-System mit der 32-Bit-Ansicht der Registry arbeiten. Wenn Sie Registry-Werte für ein 32-Bit-Produkt manuell eintragen müssen, achten Sie darauf, die 32-Bit-Version des Werkzeugs *regedit* zu verwenden. Sie können es folgendermaßen mittels Start, Ausführen... aufrufen:

```
%systemroot%\syswow64\regedit
```

**Standard-Dateisuchpfade.** Unter Unix, Linux, OS X und i5/iSeries werden per Voreinstellung einige Verzeichnisse standardmäßig sogar ohne Angabe von Pfad und Verzeichnisnamen nach Dateien durchsucht. Bevor die UPR-Datei (welche zusätzliche Suchpfade enthalten kann) durchsucht und gelesen wird, werden folgende Verzeichnisse durchsucht:

```
<rootpath>/PDFlib/TET/5.0/resource/cmap  
<rootpath>/PDFlib/TET/5.0/resource/codelist  
<rootpath>/PDFlib/TET/5.0/resource/glyphlst  
<rootpath>/PDFlib/TET/5.0  
<rootpath>/PDFlib/TET  
<rootpath>/PDFlib
```

Unter Unix, Linux und OS X wird *<rootpath>* zuerst durch */usr/local* und dann durch das HOME-Verzeichnis ersetzt. Auf i5/iSeries ist *<rootpath>* leer.

**Standard-Dateinamen für Lizenz- und Ressourcendateien.** Standardmäßig werden die folgenden Dateinamen in den Standard-Verzeichnissuchpfaden gesucht:

```
licensekeys.txt          (Lizenzdatei)  
tet.upr                  (Ressource-Datei)
```

Mit dieser Funktion lässt sich eine Lizenzdatei ohne die Angabe einer Umgebungsvariablen oder Laufzeit-Option verwenden.

**Lizenzschlüssel in einer Option für das TET-Kommandozeilen-Tool setzen.** Mit dem TET-Kommandozeilen-Tool können Sie eine Option übergeben, die den Namen der Lizenzdatei oder den Lizenzschlüssel selbst enthält:

```
tet --tetopt "license ...Ihr Lizenzschlüssel..." ...weitere Optionen...
```

**Lizenzschlüssel mit einem TET-API-Aufruf setzen.** Mit dem TET-API können Sie einen API-Aufruf zu Ihrem Skript oder Programm hinzufügen, der den Lizenzschlüssel zur Laufzeit setzt:

- ▶ In COM/VBScript:

```
oTET.set_option "license=...Ihr Lizenzschlüssel..."
```

- ▶ In C:

```
TET_set_option(tet, "license=...Ihr Lizenzschlüssel...");
```

- ▶ In C++, Java, .NET/C# und Ruby:

```
tet.set_option("license=...Ihr Lizenzschlüssel...");
```

- ▶ In Perl, Python und PHP:

```
tet->set_option("license=...Ihr Lizenzschlüssel...");
```

- ▶ In RPG:

```
d licensekey      s          20
d licenseval      s          50
c                  eval      licenseopt='license=... Ihr Lizenzschlüssel ...'+x'00'
c                  callp     TET_set_option(TET:licenseopt:0)
```

Die Option *license* muss unmittelbar nach der Instantiierung eines TET-Objekts gesetzt werden, das heißt nach dem Aufruf von *TET\_new()* (in C) oder der Erzeugung eines TET-Objekts.

**Lizenzvarianten.** Es gibt verschiedene Lizenzierungsmöglichkeiten für die Verwendung von TET auf einem oder mehreren Computern und für die Weitergabe von TET in eigenen Produkten. Wir bieten außerdem Support- und Wartungsverträge an. Einzelheiten zur Lizenzierung sowie das Bestellformular finden Sie im TET-Paket. Bitte wenden Sie sich an uns, wenn Sie Fragen haben oder eine kommerzielle Lizenz beziehen möchten:

PDFlib GmbH, Lizenzabteilung  
Franziska-Bilek-Weg 9, D-80339 München  
[www.pdflib.com](http://www.pdflib.com)  
Tel. +49 • 89 • 452 33 84-0  
Fax +49 • 89 • 452 33 84-99  
Vertrieb: [sales@pdflib.com](mailto:sales@pdflib.com)  
Support für PDFlib-Lizenznehmer: [support@pdflib.com](mailto:support@pdflib.com)

# 1 Einführung

Das PDFlib Text and Image Extraction Toolkit (TET) wurde zur Extraktion von Textinhalten und Bildern aus PDF-Dokumenten entwickelt, kann aber auch zum Abfragen anderer Informationen aus PDF-Dokumenten verwendet werden. TET kann als Basiskomponente für folgende Aufgaben verwendet werden:

- ▶ Durchsuchen eines PDFs nach Textinhalten
- ▶ Erstellung einer Liste aller in einem PDF enthaltenen Wörter (Konkordanz)
- ▶ Implementierung einer Suchmaschine zur Verarbeitung vieler PDF-Dateien
- ▶ Extraktion von Text aus einer PDF-Datei zur Speicherung, Übersetzung oder anderweitigen Verwendung
- ▶ Erstellung von Software zur Konvertierung von Textinhalten von PDF in andere Formate
- ▶ Verarbeitung oder Erweiterung von PDFs auf Basis ihrer Inhalte
- ▶ Vergleich der Textinhalte von mehreren PDF-Dokumenten
- ▶ Extraktion von Rasterbildern aus einer PDF-Datei
- ▶ Extraktion von Metadaten und anderen Informationen aus einer PDF-Datei

TET ist eine eigenständige Anwendung, die keine Fremdsoftware benötigt. Sie ist äußerst stabil und für den Multithreaded-Einsatz auf einem Server geeignet.

## 1.1 Funktionsumfang von TET

**Unterstützte PDF-Eingabe.** TET wurde mit Millionen von PDF-Dateien aus verschiedensten Quellen getestet. Es verarbeitet PDF 1.0 bis PDF 1.7 Extension Level 8 und PDF 2.0, was Acrobat 1-DC einschließlich verschlüsselter Dokumente entspricht. TET versucht, verschiedene Arten von fehlerhaften oder beschädigten PDF-Dokumenten zu reparieren.

*Hinweis* TET verarbeitet keine dynamischen XFA-Formulare. XFA ist ein separates Format, das kein Bestandteil des PDF-Standards ISO 32000-1 ist. Da XFA in eine dünne PDF-Schicht verpackt ist, werden XFA-Formulare oft mit PDF-Dokumenten verwechselt, obwohl es sich bei XFA um ein vollkommen anderes Dateiformat handelt, das spezielle Software erfordert.

**Unicode-Unterstützung.** TET enthält eine Vielzahl von Algorithmen und Daten, um den gesamten Text zuverlässig nach Unicode zu konvertieren. Da Text in PDF normalerweise nicht in Unicode kodiert ist, wandelt PDFlib TET den Text in einem PDF-Dokument in Unicode um:

- ▶ TET konvertiert alle Textinhalte nach Unicode. In C wird der Text im Format UTF-8 oder UTF-16 zurückgegeben, in anderen Sprachbindungen als native Unicode-Strings.
- ▶ Ligaturen und andere Glyphen, die mehrere Unicode-Zeichen repräsentieren, werden in eine Folge eben dieser Zeichen umgesetzt.
- ▶ Herstellerspezifische Unicode-Zuordnungen (*Corporate Use Subarea, CUS*) werden identifiziert und, sofern möglich, auf Unicode-Zeichen mit klar definierter Bedeutung abgebildet.
- ▶ Glyphen ohne Unicode-Zuordnung werden erkannt und auf ein konfigurierbares Ersatzzeichen abgebildet.

- ▶ UTF-16-Surrogatwerte (Ersatzpaare) für Zeichen außerhalb der Basic Multilingual Plane (BMP) werden erkannt und beibehalten. Surrogatpaare und UTF-32-Werte können in allen Sprachbindungen abgefragt werden.

Manche PDF-Dokumente enthalten nicht genügend Informationen, um zuverlässig nach Unicode konvertiert werden zu können. Um den Text trotzdem erfolgreich zu extrahieren, kann das Verhalten von TET durch zahlreiche Optionen konfiguriert werden, die unterstützende Informationen für die einwandfreie Unicode-Konvertierung liefern. Um das Zusammenstellen der erforderlichen Konvertierungstabellen zu erleichtern, stellen wir PDFlib FontReporter, ein kostenloses Plugin für Adobe Acrobat, zur Verfügung. Dieses Plugin kann zur Analyse der im PDF enthaltenen Fonts, Encodings und Glyphen verwendet werden.

**CJK-Unterstützung.** TET bietet umfassende Unterstützung für die Extraktion von chinesischem, japanischem und koreanischem Text:

- ▶ Alle vordefinierten CJK-CMaps (Encodings) werden erkannt; CJK-Text wird nach Unicode konvertiert. CMap-Dateien für die Konvertierung von CJK-Encodings werden mit der TET-Distribution ausgeliefert.
- ▶ Besondere Zeichenformen wie breite, schmale oder vorrotierte Glyphen für vertikalen Text können optional in die entsprechenden regulären Formen umgewandelt werden (Folding).
- ▶ Horizontale und vertikale Schreibrichtung werden unterstützt.
- ▶ CJK-Fontnamen werden in Unicode dargestellt.

**Unterstützung von bidirektionalem hebräischem oder arabischem Text.** TET bietet die folgenden Funktionen zur Verarbeitung von Bidi-Text:

- ▶ Neuordnung von linksläufigem Text und Bidi-Text in logischer Reihenfolge
- ▶ Ermitteln der Haupt-Schreibrichtung des Textes auf der Seite
- ▶ Normalisierung arabischer Präsentationsformen und Zerlegung von Ligaturen
- ▶ Entfernen von arabischen Tatweel-Zeichen, die zur Streckung von Wörtern verwendet werden

**Unicode-Nachbearbeitung.** TET unterstützt die Unicode-Nachbearbeitung unter anderem auf folgende Arten:

- ▶ Folding: Erhalten, Ersetzen oder Entfernen von ein oder mehreren Zeichen; betroffene Zeichen können komfortabel als Unicode-Sets festgelegt werden;
- ▶ Zerlegung (Dekomposition): optionale kanonische oder Kompatibilitätszerlegung gemäß Unicode-Standard. Dadurch lässt sich der Text in manchen Umgebungen besser verwenden. Sie können Buchstaben mit Akzentzeichen, Brüche oder Symbole wie z.B. das Trademark-Symbol als Ganzes erhalten oder zerlegen.
- ▶ Normalisierung: Konvertierung der Ausgabe in die Unicode-Normalisierungsformate NFC, NFD, NFKC oder NFKD gemäß Unicode-Standard. Auf diese Weise lässt sich mit TET genau das für manche Umgebungen wie Datenbanken oder Suchmaschinen als Eingabe erforderliche Format erzeugen.

**Extraktion von Rasterbildern.** TET unterstützt das Extrahieren von Rasterbildern aus PDF-Dokumenten. Nebeneinander liegende Teile eines fragmentierten Bildes werden zur vereinfachten Nachbearbeitung und Wiederverwendung zusammengesetzt (zum Beispiel die von manchen Anwendungen erzeugten Bilder vom Typ »multi-strip«). Klei-

ne Bilder lassen sich herausfiltern, damit die Ausgabe nicht durch nutzlose kleine Bildfragmente überfrachtet wird. Wenn ein Rasterbild mit einer Maske versehen ist, kann diese ebenfalls extrahiert werden.

Rasterbilder lassen sich im Format TIFF, JPEG, JPEG 2000 oder JBIG2 extrahieren.

**Geometrie.** TET liefert präzise geometrische Angaben zum Text, z.B. die Position auf der Seite, Glyphenbreiten und Textrichtung. Für die Textextraktion können Seitenbereiche einbezogen oder ausgeschlossen werden, zum Beispiel um Kopf- und Fußzeilen oder Seitenränder zu übergehen.

Für Rasterbilder stehen Angaben zu Pixelgröße, physikalischer Größe und Farbraum sowie zu Position und Winkel zur Verfügung.

**Textfarbe.** TET liefert Informationen über die Farbe der Glyphen. Die Farbräume für Füll- und Linienfarbe sowie die zugehörigen Farbwerte lassen sich abfragen. Eine bequeme Abkürzung ist für den einfachen Vergleich von Farben mehrerer Glyphen verfügbar, ohne sich mit der Komplexität der PDF-Farbräume beschäftigen zu müssen.

**Worterkennung und Inhaltsanalyse.** TET dient zur Ermittlung von elementaren Glypheninformationen, enthält darüber hinaus aber auch ausgefeilte Algorithmen zur Analyse von Seiteninhalt und Layout:

- ▶ Ermitteln von Wörtern durch Erkennung von Wortgrenzen (Wordfinder).
- ▶ Zusammenfügen der Silben eines getrennten Wortes (Enttrennung).
- ▶ Entfernen von doppeltem Text, zum Beispiel bei Schatteneffekten oder simulierter Fettschrift.
- ▶ Umstellen von Absätzen in Lesereihenfolge.
- ▶ Umordnen von über die Seite verstreutem Text.
- ▶ Rekonstruktion von Textzeilen.
- ▶ Erkennen von Tabellenstrukturen auf der Seite.
- ▶ Erkennen von hoch- und tiefgestellten Zeichen oder mehrzeiligen Anfangsbuchstaben am Anfang eines Absatzes (Dropcaps).

**TET Markup Language (TETML).** Die aus einem PDF-Dokument extrahierten Informationen können in einem XML-Format ausgegeben werden, der TET Markup Language oder TETML, die sich mit Standard-XML-Werkzeugen verarbeiten lässt. TETML enthält Informationen zu Text, Rasterbildern und Metadaten und optional auch Informationen zu Fonts und Geometrie. TETML enthält außerdem Angaben zu Anmerkungen, Formularfeldern, Lesezeichen und anderen interaktiven Elementen.

**pCOS-Schnittstelle zum einfachen Zugriff auf PDF-Objekte.** TET beinhaltet pCOS (*PDFlib Comprehensive Object System*) zum Abfragen beliebiger PDF-Objekte. Mit pCOS können Sie PDF-Metadaten, interaktive Elemente (z.B. Lesezeichentext, Inhalte von Formularfeldern) und beliebige andere Informationen mit einer einfachen Abfrageschnittstelle aus einem PDF-Dokument extrahieren. Die pCOS-Pfadsyntax wird in der pCOS Pfadreferenz gesondert beschrieben.

**Was ist Text?** TET verarbeitet zwar ein sehr großes Spektrum von PDF-Dokumenten, jedoch kann sichtbarer Text in manchen Fällen nicht extrahiert werden. Der Text muss mit den Text- und Encoding-Funktionen von PDF kodiert sein (d.h. er muss auf einem Font basieren). Folgende Arten von Text sind zwar auf der Seite sichtbar, lassen sich mit TET aber nicht extrahieren:

- ▶ Gerasterter Text (Bitmap), zum Beispiel eingescannte Seiten;
- ▶ Als Vektorelement dargestellter Text ohne Font.

Beachten Sie, dass Metadaten und Text in interaktiven Elementen (wie Lesezeichen, Formularfelder, Notizen und Kommentare) mit TETML oder der pCOS-Schnittstelle abgefragt werden können; für weitere Informationen siehe Abschnitt 6.1, »PDF-Dokumentdomänen«, Seite 79. TET kann manchmal aber auch Text extrahieren, der auf der Seite *nicht* sichtbar ist. Dies kann in folgenden Fällen vorkommen:

- ▶ Text, der mit dem PDF-Attribut *unsichtbar* versehen ist (es gibt jedoch eine Option, um diese Art Text von der Extraktion auszuschließen)
- ▶ Text, der durch andere Elemente auf der Seite (zum Beispiel ein Bild) verdeckt wird.

## 1.2 Anwendungsmöglichkeiten von TET

TET steht als Programmierbibliothek (Komponente) für verschiedene Entwicklungsumgebungen zur Verfügung sowie als Kommandozeilen-Tool für Stapelverarbeitung. Ihre Funktionalität ist zwar nahezu identisch, die Einsatzbereiche unterscheiden sich jedoch. Sowohl die TET-Bibliothek als auch das Kommandozeilen-Tool kann die Ausgabe als TETML darstellen, dem XML-basierten Ausgabeformat von TET.

- ▶ Die TET-Programmierbibliothek eignet sich zur Integration in Desktop- oder Serveranwendungen. Es wird eine Vielzahl von Programmiersprachen unterstützt. Beispiele für den Einsatz der TET-Bibliothek mit allen unterstützten Sprachbindungen finden Sie im TET-Paket.
- ▶ Das TET-Kommandozeilen-Tool eignet sich zur Stapelverarbeitung von PDF-Dokumenten. Es erfordert keine Programmierung, sondern kann über Kommandozeilen-Optionen gesteuert und damit in komplexe Abläufe integriert werden.
- ▶ TETML-Ausgabe ist für XML-basierte Workflows und für Entwickler geeignet, die mit dem breiten Spektrum von XML-Werkzeugen und -sprachen wie XSLT vertraut sind.
- ▶ Mit den TET-Konnektoren lässt sich TET in verschiedene gängige Software-Pakete integrieren, wie zum Beispiel Datenbanken und Suchmaschinen.
- ▶ Das TET-Plugin ist eine frei erhältliche Erweiterung für Adobe Acrobat, die eine interaktive Nutzung von TET ermöglicht (für weitere Informationen siehe Abschnitt 4.1, »Kostenloses TET Plugin für Adobe Acrobat«, Seite 49).

## 1.3 Roadmap für Dokumentation und Beispiele

**Minibeispiele für die TET-Bibliothek.** Das TET-Paket enthält Beispielcode für alle unterstützten Sprachbindungen. Die Minibeispiele dienen in erster Linie zum Testen Ihrer TET-Installation und als Ausgangspunkt für die Programmierung Ihrer eigenen Anwendungen. Sie enthalten Sourcecode für die folgenden Anwendungen:

- ▶ Das Beispiel *extractor* zeigt die grundlegende Schleife für die Textextraktion aus einem PDF-Dokument.
- ▶ Im Beispiel *images\_per\_page* werden Rasterbilder auf jeder Seite extrahiert sowie die zugehörige Geometrie und weitere Eigenschaften abgefragt.

- ▶ Das Beispiel *image\_resources* zeigt die grundlegende Schleife für die Extraktion von Rasterbildern aus einem PDF-Dokument auf eine Ressourcen-orientierte Weise (geometrische Informationen sind nicht verfügbar).
- ▶ Das Beispiel *dumper* zeigt die Verwendung der integrierten pCOS-Schnittstelle zur Abfrage von allgemeinen Informationen über das PDF-Dokument.
- ▶ Das Beispiel *fontfilter* zeigt die Verarbeitung von fontbezogenen Informationen wie Fontnamen und Schriftgröße.
- ▶ Das Beispiel *glyphinfo* zeigt die Abfrage detaillierter Informationen zu Glyphen (Font, Größe, Position usw.) sowie zu Textattributen (*dropcap*, *shadow*, *hyphenation* usw.). Es verdeutlicht auch den Zugriff auf die Textfarbe.
- ▶ Das Beispiel *tetml* zeigt, wie aus einem PDF-Dokument TETML erzeugt werden kann (TETML ist das XML-basierte Ausgabeformat von TET für PDF-Inhalte).
- ▶ Das Beispiel *get\_attachments* zeigt die Verarbeitung von PDF-Dateianhängen, d.h. von PDF-Dokumenten, die in einem anderen PDF-Dokument eingebettet sind (dieses Beispiel ist nicht für alle Sprachbindungen verfügbar).

**XSLT-Beispiele.** Das TET-Paket enthält mehrere XSLT-Stylesheets, die die Verarbeitung von TETML für verschiedene Zwecke demonstrieren:

- ▶ *concordance.xsl*: Liste aller eindeutigen Wörter in einem Dokument, sortiert nach absteigender Häufigkeit.
- ▶ *fontfilter.xsl*: Liste aller Wörter in einem Dokument, die einen bestimmten Font verwenden, der größer ist als der angegebene Wert.
- ▶ *fontfinder.xsl*: Liste aller Fonts im Dokument, mit Seitenangaben und Positionsinformationen zu jeder Fundstelle eines Fonts.
- ▶ *fontstat.xsl*: Statistik zu Fonts und Glyphen.
- ▶ *index.xsl*: Alphabetischer Index (wie am Ende eines Buches).
- ▶ *metadata.xsl*: Extraktion ausgewählter Properties aus XMP-Metadaten auf Dokumentebene, die in TETML enthalten sind.
- ▶ *solr.xsl*: Erzeugen von Eingabe für den Solr Enterprise Search Server.
- ▶ *table.xsl*: Extraktion einer Tabelle in eine CSV-Datei (komma-separierte Werte).
- ▶ *tetml2html.xsl*: Konvertierung von TETML in einfaches HTML.
- ▶ *textonly.xsl*: Extraktion von reinen Textdaten aus der TETML-Eingabe.

**TET Cookbook.** Das TET Cookbook ist eine Sammlung von Beispielcode zur Lösung von spezifischen Anwendungsproblemen mit der TET-Bibliothek. Die meisten Cookbook-Beispiele sind in Java implementiert, lassen sich aber leicht an andere Programmiersprachen anpassen, da sich das TET-API in den verschiedenen unterstützten Sprachbindungen kaum unterscheidet. Einige Cookbook-Beispiele sind in der Sprache XSLT geschrieben. Das TET Cookbook ist folgendermaßen aufgebaut:

- ▶ Text: Beispiele zur Textextraktion
- ▶ Font: Beispiele für Text mit besonderem Fokus auf Fonteigenschaften
- ▶ Rasterbilder: Beispiele zur Extraktion von Rasterbildern
- ▶ TET & PDFlib+PDI: Beispiele zur Extraktion von Informationen aus einem PDF-Dokument mit TET sowie zur Konstruktion eines neuen PDF-Dokuments auf Basis des Original-PDFs und den extrahierten Informationen. Diese Beispiele erfordern zusätzlich zu TET das Produkt PDFlib+PDI.
- ▶ TETML: XSLT-Beispiele zur Verarbeitung von TETML
- ▶ Spezial: weitere Beispiele

Das TET Cookbook steht unter folgender Adresse zur Verfügung:  
[www.pdfplib.com/tet-cookbook](http://www.pdfplib.com/tet-cookbook).

**pCOS Cookbook.** Das *pCOS Cookbook* ist eine Sammlung von Codefragmenten für die pCOS-Schnittstelle, die in TET enthalten ist. Es steht unter folgender Adresse zur Verfügung:  
[www.pdfplib.com/pcos-cookbook](http://www.pdfplib.com/pcos-cookbook).

Die pCOS-Schnittstelle wird in der pCOS Pfadreferenz beschrieben, die im TET-Paket enthalten ist.

## 1.4 Was ist neu in TET 5.0?

Folgende Features sind neu oder wurden für TET 5.0 erheblich verbessert:

Extrahieren von Text:

- ▶ Abfragen von Füll- und Linienfarbe des Textes
- ▶ Verbesserte Layout-Erkennung
- ▶ Berücksichtigung von Vektorgrafiken bei der Erkennung von Seiten- und Tabellen-Layout
- ▶ Unterstützung vertikaler Fontmetrik für CJK-Text

Extraktion von Bildern:

- ▶ Deutlich verbessertes Zusammensetzen fragmentierter Bilder, z.B. von rotierten Bildern
- ▶ Verbesserte Bildverarbeitung für viele Spezialfälle und seltene PDF-Bildvarianten
- ▶ Extraktion von Bildmasken und Transparenzmasken
- ▶ Zusammensetzen und Konvertieren von JPEG-2000-komprimierten Bildern
- ▶ Erhalten von Schmuckfarben in extrahierten TIFF-Bildern
- ▶ Beschränkung der Bildextraktion auf einen vom Benutzer ausgewählten Bereich
- ▶ Berücksichtigung von XMP-Metadaten für Bilder, die von InDesign nicht an den eigentlich vorgesehenen Stellen in PDF gespeichert wurden

Seitenverarbeitung:

- ▶ Berücksichtigung von Beschneidungspfaden, um die Extraktion von unsichtbaren Inhalten zu vermeiden
- ▶ Berücksichtigung von Ebenen (optionale Inhalte), um die Extraktion von unsichtbaren Inhalten zu vermeiden
- ▶ Wahlweises Ignorieren von Artefakten (irrelevanten Inhalten) in Tagged PDF
- ▶ Prüfen, ob ein Bereich auf der Seite leer ist oder Text-, Bild- bzw. Vektorgrafik enthält

TETML:

- ▶ TETML enthält Füll- und Linienfarbe der Glyphen
- ▶ TETML enthält Informationen über interaktive Elemente wie Anmerkungen, Formularfelder, Lesezeichen, Aktionen, JavaScript, Unterschriften usw.
- ▶ TETML enthält Farbraum und ICC-Profildetails
- ▶ TETML enthält Informationen über Ebenen und Seiten-Labels

Abfragen von Informationen aus PDF-Dokumenten:

- ▶ pCOS-Pseudo-Objekte für ICC-Profildetails und Bildmaskierungseigenschaften
- ▶ pCOS-Pseudo-Objekte für Formularfelder

Weitere Verbesserungen:

- ▶ Zusätzliche Prüfungen und Heuristiken für beschädigte und nicht-konforme PDF-Eingabe
- ▶ Aktualisierte TET-Sprachbindungen, Programmierbeispiele und TET-Konnektoren
- ▶ Neue Optionen für verbesserte Steuerung der PDF-Verarbeitung
- ▶ Zahlreiche Verbesserungen in bestehenden TET-Features



# 2 TET-Kommandozeilen-Tool

## 2.1 Kommandozeilen-Optionen

Das TET-Kommandozeilen-Tool ermöglicht es, Text und Bilder aus einem oder mehreren PDF-Dokumenten ohne Programmieraufwand zu extrahieren. Ausgabe kann im Klartextformat (Unicode) oder in TETML generiert werden, dem XML-basierten Ausgabeformat von TET. TET kann über verschiedene Kommandozeilen-Optionen gesteuert werden. Das Programm fügt Leerzeichen (U+0020) nach jedem Wort, U+000A nach jeder Zeile und U+000C nach jeder Seite ein. Es wird für eine oder mehrere PDF-Eingabedateien folgendermaßen aufgerufen:

```
tet [<options>] <filename>...
```

Das TET-Kommandozeilen-Tool baut auf der TET-Bibliothek auf. Mit den Optionen `--docopt`, `--teto`, `--imageopt` und `--pageopt` können Sie Optionen gemäß der Optionslisten-Tabellen in Kapitel 10, »API-Referenz für die TET-Bibliothek«, Seite 173 an die Bibliothek übergeben. Tabelle 2.1 führt alle TET-Kommandozeilen-Optionen auf (diese Liste wird auch angezeigt, wenn Sie das TET-Programm ohne Optionen ausführen).

*Hinweis* Zur Extraktion von CJK-Text müssen Sie den Zugriff auf die CMap-Dateien konfigurieren, die mit TET ausgeliefert werden (siehe Abschnitt 0.1, »Installation der Software«, Seite 7).

Tabelle 2.1 TET-Kommandozeilen-Optionen

Option	Parameter	Funktion
--		Beendet die Optionsliste; nützlich, wenn Dateinamen mit dem Zeichen »-« beginnen.
@filename <sup>1</sup>		Legt eine Response-Datei mit Optionen fest; für eine Syntaxbeschreibung siehe »Response-Dateien«, Seite 22. Response-Dateien werden nur vor der Option -- und vor dem ersten Dateinamen erkannt. Response-Dateien müssen komplette Kombinationen aus Option/Parameter enthalten, können jedoch nicht den Parameter für eine andere Option ersetzen.
--docopt	<Optionsliste>	Zusätzliche Optionsliste für <code>TET_open_document()</code> (siehe Tabelle 10.8, Seite 191). Die Unteroption <code>filename</code> der Option <code>tetml</code> kann hier nicht verwendet werden.
--firstpage -f	<integer>   last	(Wird bei <code>--imageLoop resource ignoriert</code> ) Nummer der Seite, auf der mit der Extraktion begonnen werden soll. Die letzte Seite kann mit dem Schlüsselwort <code>last</code> festgelegt werden, die vorletzte Seite mit dem Schlüsselwort <code>last-1</code> usw. Standardwert: 1
--format	utf8   utf16	Legt das Format der Textausgabe fest (Standardwert: utf8): <b>utf8</b> UTF-8 mit BOM (Byte Order Mark) <b>utf16</b> UTF-16 in nativer Bytereihenfolge mit BOM Diese Option hat keine Auswirkung auf die TETML-Ausgabe, die immer im Format UTF-8 erzeugt wird.
--help, -? (oder keine Option)		Anzeige der Hilfe mit einer Übersicht über die verfügbaren Optionen

Tabelle 2.1 TET-Kommandozeilen-Optionen

Option	Parameter	Funktion
<b>--image<sup>2</sup></b> <b>-i</b>		Extraktion von Bildern aus dem gesamten Dokument (mit <code>--image loop resource</code> ) oder den ausgewählten Seiten (mit <code>--image loop page</code> ). Das Namensschema für extrahierte Bilder ist abhängig von der Option <code>--image loop</code> .
<b>--image loop</b>	page   resource	<p>Legt die Art der Auflistung für die Bildextraktion mit der Option <code>--image fest</code> (Standardwert: <code>page</code>, wird aber auf <code>resource</code> gesetzt, falls <code>--tetml</code> angegeben ist):</p> <p><b>page</b> Extrahieren aller Bilder auf den ausgewählten Seiten. Mehrfach platzierte Bildressourcen werden mehrfach extrahiert. Extrahierte Bilder werden nach folgendem Muster benannt:  <code>&lt;filename&gt;_p&lt;pagenumber&gt;_&lt;imagenumber&gt;.[tif jpg jp2 jpf j2k jbig2]</code>                      Bilder, die als Transparenzmasken oder Masken für ein anderes Bild verwendet werden, werden nach folgendem Muster benannt:  <code>&lt;filename&gt;_p&lt;pagenumber&gt;_&lt;imagenumber&gt;_mask.[tif jpg jp2 jpf j2k jbig2]</code>, wobei <code>imagenumber</code> die Nummer des maskierten Bildes auf der Seite ist.</p> <p>Berechnete Auflösungswerte auf Basis der Größe des platzierten Bildes werden in die erzeugten TIFF-Bilder eingebettet.</p> <p><b>resource</b> Extrahieren aller einfachen und zusammengesetzten Bildressourcen im Dokument einschließlich Masken und Transparenzmasken. Jede Bildressource wird einmal extrahiert, unabhängig davon, wie oft sie im Dokument vorkommt. Extrahierte Bilder (einschließlich der Bilder, die als Transparenzmasken oder als Masken für andere Bilder verwendet werden) werden nach folgendem Muster benannt:  <code>&lt;filename&gt;_I&lt;imageid&gt;.[tif jpg jp2 jpf j2k jbig2]</code>                      Der selbe Bilddateiname wird mit dem TETML-Attribut <code>Image/@filename</code> ausgegeben.</p> <p>Da für Bildressourcen keine Größenangaben verfügbar sind, wird ein Dummywert von 72 dpi in die erzeugten TIFF-Bilder eingebettet.</p>
<b>--imageopt</b>	<Optionsliste>	Zusätzliche Optionsliste für <code>TET_write_image_file()</code> (siehe Tabelle 10.20, Seite 217)
<b>--lastpage</b> <b>-l</b>	<integer>   last	(Wird bei <code>--image loop resource</code> ignoriert) Nummer der Seite, auf der die Extraktion enden soll. Die letzte Seite kann mit dem Schlüsselwort <code>last</code> festgelegt werden, die vorletzte Seite mit dem Schlüsselwort <code>last-1</code> usw. Standardwert: <code>last</code>
<b>--outfile</b> <b>-o</b>	<Dateiname>	(Nicht erlaubt, wenn mehrere Eingabedateinamen übergeben werden) Name der Text- oder TETML-Ausgabedatei. Der Dateiname »-« bezeichnet die Standardausgabe, sofern nur eine einzige Eingabedatei übergeben wurde. Standardwert: Name der Eingabedatei, wobei <code>.pdf</code> oder <code>.PDF</code> bei Textausgabe durch <code>.txt</code> und bei TETML-Ausgabe durch <code>.tetml</code> ersetzt wird.
<b>--pagecount</b>		Druckt die Anzahl der Seiten im Dokument, d.h. den Wert des <code>pCOS</code> -Pfadens <code>length:pages</code> , nach <code>stdout</code> oder in die mit <code>--outfile</code> angegebene Datei.
<b>--pageopt</b>	<Optionsliste>	Zusätzliche Optionsliste, die an <code>TET_open_page()</code> übergeben wird, wenn Textausgabe erzeugt wird, oder an <code>TET_process_page()</code> , wenn TETML-Ausgabe erzeugt wird (siehe Tabelle 10.10, Seite 199 und Tabelle 10.21, Seite 219). Für Textausgabe wird die Option <code>granularity</code> immer auf <code>page</code> gesetzt.
<b>--password,</b> <b>-p</b>	<Kennwort>	Benutzer-, Master oder Dateianlagen-Kennwort für verschlüsselte Dokumente. In manchen Fällen kann das Feature <code>shrug</code> für die Indizierung verschlüsselter Dokumente ohne Kennwortübergabe verwendet werden (siehe Abschnitt 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67).
<b>--samedir</b>		Ausgabedatei(en) werden im selben Verzeichnis wie die Eingabe erzeugt.

Tabella 2.1 TET-Kommandozeilen-Optionen

Option	Parameter	Funktion
<b>--searchpath<sup>1</sup></b> <b>-s</b>	<Pfad>...	Name eines oder mehrerer Verzeichnisse, wo Dateien (z.B. CMaps) gesucht werden. Standardwert: abhängig von der Installation
<b>--targetdir</b> <b>-t</b>	<Verzeichnisname>	Ausgabeverzeichnis für die erzeugten Text-, TETML- und Bilddateien. Das Verzeichnis muss existieren. Wurde --samedir angegeben, wird diese Option ignoriert. Standardwert: . (d.h. das aktuelle Arbeitsverzeichnis)
<b>--tetml</b> <b>-m</b>	glyph   image   word   wordplus   line   page	(Kann nicht mit --text kombiniert werden) Erzeugt TETML-Ausgabe mit Angaben zu Text, Bildern und interaktiven Elementen. TETML wird im UTF-8-Format erzeugt. Mit dem übergebenen Parameter wird eine von mehreren Varianten ausgewählt (siehe Abschnitt 9.3, »Steuerung von TETML-Informationen«, Seite 153): <b>glyph</b> Glyph-basiertes TETML mit Glyphen-Geometrie und Fontinformationen <b>image</b> TETML mit Bildinformationen, aber ohne Informationen zu Text und interaktiven Elementen <b>line</b> Zeilenbasiertes TETML <b>page</b> Seitenbasiertes TETML <b>word</b> Wortbasiertes TETML mit Wortboxen <b>wordplus</b> Wortbasiertes TETML mit Wortboxen sowie allen Glyph-Informationen
<b>--teto</b>	<Optionsliste>	Zusätzliche Optionsliste für <b>TET_set_option()</b> (siehe Tabelle 10.2, Seite 180). Die Option <b>outputformat</b> wird ignoriert (verwenden Sie stattdessen --format).
<b>--text<sup>2</sup></b>		(Kann nicht mit --tetml kombiniert werden) Textextraktion aus dem Dokument (standardmäßig aktiviert)
<b>--verbose</b> <b>-v</b>	0   1   2   3	Umfang der ausgegebenen Fehlermeldungen (Standardwert: 1): <b>0</b> keine Ausgabe <b>1</b> nur Fehler werden ausgegeben <b>2</b> Fehler und Dateinamen werden ausgegeben <b>3</b> detaillierter Report
<b>--version, -V</b>		Ausgabe der TET-Versionsnummer.

1. Diese Option kann mehrfach übergeben werden.

2. Mit der Option --image wird die Textextraktion standardmäßig deaktiviert. Diese Option kann jedoch mit --text und --tetml kombiniert werden.

## 2.2 Erstellen von TET-Kommandozeilen

Beim Erstellen von TET-Kommandozeilen sind folgende Regeln zu beachten:

- ▶ Eingabedateien werden in allen Verzeichnissen gesucht, die im *searchpath* festgelegt wurden.
- ▶ Für manche Optionen stehen Kurzformen zur Verfügung, die mit langen Optionen kombiniert werden können.
- ▶ Lange Optionsnamen dürfen abgekürzt werden, sofern sie eindeutig bleiben.
- ▶ Abhängig vom Verschlüsselungsstatus der Eingabedatei wird ein Benutzer- oder Master-Kennwort benötigt, um den Text erfolgreich zu extrahieren. Dieses wird mit der Option *--password* übergeben. TET überprüft, ob das Kennwort zur Inhaltsextraktion ausreicht und gibt gegebenenfalls eine Fehlermeldung aus.

Vor der Verarbeitung einer Datei prüft TET die gesamte Kommandozeile. Liegt in den Optionen der Kommandozeile ein Fehler vor, werden keinerlei Dateien verarbeitet.

**Dateinamen.** Dateinamen mit Leerzeichen müssen beim Einsatz mit Kommandozeilen-Tools wie TET besonders behandelt werden. Um einen Dateinamen mit Leerzeichen zu verarbeiten, sollten Sie den gesamten Dateinamen in doppelte Anführungszeichen " setzen. Wildcards können auf die übliche Art verwendet werden. Zum Beispiel umfasst *\*.pdf* alle Dateinamen mit dem Suffix *.pdf* im Dateinamen. Beachten Sie, dass manche Systeme nach Groß- und Kleinschreibung unterscheiden (d.h. *\*.pdf* kann von *\*.PDF* verschieden sein). Beachten Sie außerdem, dass unter Windows keine Wildcards für Dateinamen mit Leerzeichen verwendet werden können. Wildcards werden nur im aktuellen Verzeichnis und nicht für ein beliebiges *searchpath*-Verzeichnis ausgewertet.

Unter Windows akzeptieren alle Dateinamen-Optionen Unicode-Strings, z.B. wenn Dateien aus dem Windows Explorer in ein Kommandozeilen-Fenster gezogen werden.

**Response-Dateien.** Optionen können direkt in der Kommandozeile oder in einer Response-Datei übergeben werden. Die Inhalte einer Response-Datei werden an der Stelle in der Kommandozeile eingefügt, an der die Option *@filename* gefunden wurde.

Eine Response-Datei ist eine einfache Textdatei mit Optionen und Parametern. Es müssen folgende Syntaxregeln eingehalten werden:

- ▶ Optionswerte müssen durch Leerraum, also Leerzeichen, Zeilenumbruch oder Tabulatorzeichen voneinander getrennt werden.
- ▶ Werte mit Leerzeichen müssen in doppelte Anführungszeichen eingeschlossen werden: "
- ▶ Doppelte Anführungszeichen am Anfang und Ende eines Wertes werden übersprungen.
- ▶ Ein doppeltes Anführungszeichen muss mit einem Backslash maskiert werden, wenn es als Anführungszeichen verwendet werden soll: \"
- ▶ Ein Backslash muss mit einem weiteren Backslash maskiert werden, wenn er als solcher verwendet werden soll: \\

Response-Dateien können verschachtelt werden, das heißt, die Syntax *@filename* kann in einer anderen Response-Datei verwendet werden.

Response-Dateien können Unicode-Strings für Dateinamen-Argumente enthalten. Response-Dateien können im Format UTF-8, EBCDIC-UTF-8 oder UTF-16 kodiert sein und müssen mit dem zugehörigen BOM beginnen. Wird kein BOM gefunden, werden

die Inhalte der Response-Datei unter zSeries als EBCDIC interpretiert und auf allen anderen Plattformen als ISO 8859-1 (Latin-1).

**Rückgabewerte.** Das TET-Kommandozeilen-Tool gibt einen Wert zurück, mit dem geprüft werden kann, ob die angeforderten Operationen erfolgreich ausgeführt werden konnten:

- ▶ Rückgabewert 0: alle Kommandozeilen-Optionen konnten erfolgreich und vollständig verarbeitet werden.
- ▶ Rückgabewert 1: Bei der Dateiverarbeitung sind Fehler aufgetreten, die Verarbeitung wurde aber fortgeführt.
- ▶ Rückgabewert 2: In den Kommandozeilen-Optionen wurde ein Fehler gefunden. Die Verarbeitung wurde bei der fehlerhaften Option abgebrochen; es wurden keine Eingabedateien verarbeitet.

## 2.3 Kommandozeilen-Beispiele

Die folgenden Beispiele zeigen einige nützliche Kombinationen von TET-Kommandozeilen-Optionen.

### 2.3.1 Textextraktion

Extrahiert den Text aus einem PDF-Dokument *file.pdf* im UTF-8-Format und speichert ihn in *file.txt*:

```
tet file.pdf
```

Schließt die erste und letzte Seite von der Textextraktion aus:

```
tet --firstpage 2 --lastpage last-1 file.pdf
```

Übergibt ein Verzeichnis, in dem sich die CJK-CMaps befinden (erforderlich für die CJK-Textextraktion):

```
tet --searchpath /usr/local/cmaps file.pdf
```

Extrahiert den Text aus einem PDF-Dokument im UTF-16-Format und speichert ihn in *file.utf16*:

```
tet --format utf16 --outfile file.utf16 file.pdf
```

Extrahiert den Text aus allen PDF-Dokumenten eines Verzeichnisses und speichert die erzeugten *\*.txt*-Dateien in einem anderen (bereits existierenden) Verzeichnis:

```
tet --targetdir out in/*.pdf
```

Extrahiert den Text aus allen PDF-Dokumenten aus zwei Verzeichnissen und speichert die erzeugten *\*.txt*-Dateien im selben Verzeichnis wie das zugehörige Eingabedokument:

```
tet --samedir dir1/*.pdf dir2/*.pdf
```

Beschränkt die Textextraktion auf einen bestimmten Bereich auf der Seite:

```
tet --pageopt "includebox={{0 0 200 200}}" file.pdf
```

Verwendet eine Response-Datei mit verschiedenen Kommandozeilen-Optionen und verarbeitet alle PDF-Dokumente im aktuellen Verzeichnis (die Datei *options* enthält Kommandozeilen-Optionen):

```
tet @options *.pdf
```

### 2.3.2 Bildextraktion

Extrahiert Bilder seitenbasiert aus *file.pdf* und speichert sie im Verzeichnis *out*:

```
tet --targetdir out --image file.pdf
```

Extrahiert Bilder ressourcenbasiert aus *file.pdf* und speichert sie im Verzeichnis *out*:

```
tet --targetdir out --image --imagemap resource file.pdf
```

Extrahiert Bilder aus *file.pdf* ohne Bildzusammenführung; dies kann durch Übergabe einer Liste von Seitenoptionen für die Bildverarbeitung erreicht werden:

```
tet --targetdir out --image --pageopt "imageanalysis={merge={disable}}" file.pdf
```

### 2.3.3 Erzeugung von TETML

Erzeugt TETML-Ausgabe im Wort-Modus für ein PDF-Dokument *file.pdf* und speichert sie in *file.tetml*:

```
tet --tetml word file.pdf
```

Erzeugt TETML-Ausgabe ohne Elemente vom Typ *Options*; dies kann durch Übergabe einer geeigneten Liste von Dokumentoptionen erreicht werden:

```
tet --docopt "tetml={elements={options=false}}" --tetml word file.pdf
```

Erzeugt TETML-Ausgabe im Wort-Modus mit allen Glyphen-Informationen und speichert sie in *file.tetml*:

```
tet --tetml word --pageopt "tetml={glyphdetails={all}}" file.pdf
```

Extrahiert Bilder und erzeugt TETML mit Text und Bildinformationen:

```
tet --image --tetml word file.pdf
```

Extrahiert Bilder und erzeugt Bildinformationen in TETML, aber keinen Text:

```
tet --tetml image --image file.pdf
```

Erzeugt TETML-Ausgabe mit Topdown-Koordinaten:

```
tet --tetml word --pageopt "topdown={output}" file.pdf
```

Erzeugt TETML-Ausgabe mit verbesserter Tabellenerkennung:

```
tet --tetml word --pageopt "vectoranalysis={structures=tables}" file.pdf
```

### 2.3.4 Erweiterte Optionen

Übergibt die Dokumentoption *checkglyphlists* zur Verbesserung der Unicode-Konvertierung für bestimmte Arten von TeX-generierten PDF-Dokumenten:

```
tet --docopt checkglyphlists file.pdf
```

Wendet Unicode-Foldings an, z.B. Leerzeichen-Folding: alle Varianten von Unicode-Leerzeichen werden auf *U+0020* abgebildet:

```
tet --docopt "fold={{[:blank:] U+0020}}" file.pdf
```

Deaktiviert Satzzeichen als Wortgrenze:

```
tet --pageopt "contentanalysis={punctuationbreaks=false}" file.pdf
```



# 3 Sprachbindungen für die TET-Bibliothek

In diesem Kapitel werden die für die TET-Bibliothek unterstützten Sprachbindungen behandelt. Die TET-Distribution enthält für jede unterstützte Sprachbindung vollständigen Beispielcode verschiedener kleiner TET-Anwendungen.

## 3.1 Verarbeitung von Exceptions

Eine bestimmte Art von Fehlern wird in vielen Sprachen zurecht als Ausnahme (Exception) bezeichnet – es handelt sich um echte Ausnahmesituationen, die während der Laufzeit eines Programms nicht allzu häufig auftreten werden. Im Allgemeinen werden für Funktionsaufrufe, die häufig fehlschlagen, herkömmliche Reportmechanismen verwendet (sprich: Fehlerrückgabecodes), und nur für seltenere Fehler ein spezieller Exception-Mechanismus, um den Code nicht mit Fehlerabfragen zu überfrachten. Genau diesen Weg geht auch TET: Manche Operationen können ziemlich oft fehlschlagen, so zum Beispiel:

- ▶ Versuch, ein PDF-Dokument ohne korrektes Kennwort zu öffnen (siehe auch das Feature *shrug* in Abschnitt 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67);
- ▶ Versuch, ein PDF-Dokument mit falschem Dateinamen zu öffnen;
- ▶ Versuch, ein PDF-Dokument mit beschädigter Dateistruktur zu öffnen.

TET gibt für diese Fehler den Wert `-1` zurück, siehe hierzu auch die API-Referenz. Andere Ereignisse gelten als problematisch, treten aber nur selten auf, zum Beispiel:

- ▶ zu wenig virtueller Speicher;
- ▶ Übergabe falscher Funktionsparameter (z.B. ein ungültiges Dokument-Handle);
- ▶ Übergabe fehlerhafter Optionslisten;
- ▶ eine erforderliche Ressource (z.B. eine CMap-Datei für die Extraktion von CJK-Text) kann nicht gefunden werden.

Stößt TET auf eine solche Ausnahmesituation, so wird kein spezieller Fehlerwert an die aufrufende Funktion zurückgegeben, sondern eine Exception ausgelöst. Bei Sprachen, die über eine interne Verarbeitung von Exceptions verfügen, werden die Standardmechanismen der jeweiligen Sprache oder Umgebung angewendet. Für die C-Sprachbindung bietet TET eine spezielle Behandlung von Exceptions an, die von der Anwendung verwendet werden muss (siehe Abschnitt 3.2, »C-Sprachbindung«, Seite 29).

Denken Sie unbedingt daran, die Verarbeitung eines Dokuments zu beenden, wenn eine Exception ausgelöst wurde. Die einzigen Methoden, die nach einer Exception noch aufgerufen werden dürfen, sind `delete()`, `get_apiname()`, `get_errnum()` und `get_errmsg()`. Wenn andere Methoden nach einer Exception aufgerufen werden, kann dies zu unerwünschten Ergebnissen führen. In der Exception sind folgende Informationen enthalten:

- ▶ eine eindeutige Fehlernummer;
- ▶ der Name der Funktion, die die Exception ausgelöst hat;
- ▶ eine genaue Beschreibung des Problems;

**Abfragen der Gründe für einen fehlerhaften Funktionsaufruf.** Einige TET-Funktionsaufrufe, wie *open\_document()* oder *open\_page()*, können ohne das Auslösen einer Exception scheitern (im Fehlerfall wird dann -1 zurückgegeben). Um in solchen Fällen die Fehlerursache zu ermitteln, können Sie direkt nach dem gescheiterten Aufruf die Funktionen *get\_errnum()*, *get\_errmsg()* und *get\_apiname()* ausführen.

## 3.2 C-Sprachbindung

TET ist in C geschrieben, mit einigen C++-Modulen. Um die C-Sprachbindung zu nutzen, können Sie eine statische oder eine dynamische Bibliothek (DLL unter Windows und MVS) verwenden. Außerdem benötigen Sie die zentrale TET-Include-Datei *tetlib.h* zur Einbindung in die Quellmodule Ihrer Anwendung.

*Hinweis* Anwendungen, die die C-Sprachbindung von TET verwenden, müssen mit einem C++-Linker gebunden werden, da die TET-Bibliothek einige in C++ implementierte Teile enthält. Die Verwendung eines C-Linkers kann zu offenen externen Verweisen führen, sofern die Anwendung nicht mit den benötigten C++-Laufzeit-Bibliotheken gebunden wird.

**Verarbeitung von Exceptions.** Das TET-API bietet einen Mechanismus für die Verarbeitung von Exceptions, die von der Bibliothek ausgelöst werden, um das Fehlen einer integrierten Verarbeitung von Exceptions in C zu kompensieren. Mit den *TET\_TRY()*- und *TET\_CATCH()*-Makros lässt sich Client-Code so einrichten, dass ein dediziertes Codefragment für Fehlerbehandlung und Cleanup aufgerufen wird, wenn eine Exception auftritt. Diese Makros erzeugen zwei Codeabschnitte: den TRY-Block mit Code, der eine Exception auslösen kann, und den CATCH-Block mit dem Code, der eine Exception verarbeitet. Wenn eine der im TRY-Block aufgerufenen Funktionen eine Exception auslöst, wird das Programm direkt an der ersten Anweisung des CATCH-Blocks fortgesetzt. Folgende Regeln müssen im TET-Client-Code beachtet werden:

- ▶ *TET\_TRY()* und *TET\_CATCH()* müssen immer paarweise aufgerufen werden.
- ▶ *TET\_new()* löst nie eine Exception aus; da ein TRY-Block nur mit einem gültigen TET-Objekt-Handle gestartet werden kann, kann *TET\_new()* nur außerhalb eines TRY-Blocks aufgerufen werden.
- ▶ *TET\_delete()* löst nie eine Exception aus und kann daher immer außerhalb eines TRY-Blocks aufgerufen werden. Es kann auch in einem CATCH-Block aufgerufen werden.
- ▶ Besondere Aufmerksamkeit ist beim Umgang mit Variablen erforderlich, die sowohl im TRY-Block als auch im CATCH-Block verwendet werden. Da der Compiler nichts vom Sprung zwischen den Blöcken weiß, erzeugt er in einer solchen Situation unter Umständen fehleranfälligen Code (z.B. durch Optimierung des Variablenzugriffs über Register).

Zum Glück gibt es eine einfache Regel zur Vermeidung dieses Problems: Variablen, die im TRY- und im CATCH-Block verwendet werden, müssen als *volatile* deklariert werden. Das Schlüsselwort *volatile* signalisiert dem Compiler, keine riskanten Optimierungen mit der Variablen durchzuführen.

- ▶ Wird ein TRY-Block verlassen (zum Beispiel mit einer *return*-Anweisung, ohne dass das entsprechende Makro *TET\_CATCH()* zur Ausführung kommt), muss der Exception-Mechanismus mit dem Makro *TET\_EXIT\_TRY()* darüber informiert werden.
- ▶ Die Dokumentverarbeitung muss beendet werden, wenn eine Exception ausgelöst wurde.

Das folgende Codefragment veranschaulicht diese Regeln durch eine typische Sequenz für den Umgang mit TET-Exceptions im Client-Code (eine vollständiges Beispiel finden Sie im TET-Paket):

```
volatile int pageno;
...
if ((tet = TET_new()) == (TET *) 0)
{
```

```

        printf("out of memory\n");
        return(2);
    }
    TET_TRY(tet)
    {
        for (pageno = 1; pageno <= n_pages; ++pageno)
        {
            /* Verarbeitung der Seite */

            if (/* ein Fehler ist aufgetreten */)
            {
                TET_EXIT_TRY(tet);
                return -1;
            }
        }
        /* direkter oder indirekter Aufruf von API-Funktionen */
    }
    TET_CATCH(tet)
    {
        printf("Error %d in %s() on page %d: %s\n",
            TET_get_errnum(tet), TET_get_apiname(tet), pageno, TET_get_errmsg(tet));
    }
    TET_delete(tet);

```

**Unicode-Verarbeitung von Name-Strings.** Die Programmiersprache C bietet erst in Version C11 native Unicode-Unterstützung. Da diese Version aber noch nicht allgemein verbreitet ist, bietet TET Unicode-Unterstützung auf Basis des traditionellen Datentyps *char*. Einige String-Parameter für Funktionen sind als *Name-Strings* deklariert. Diese werden abhängig vom Parameter *length* und dem Vorhandensein eines BOM am Anfang des Strings verarbeitet. Wenn der Parameter *length* in C ungleich 0 ist, wird der String als UTF-16 interpretiert. Wenn der Parameter *length* gleich 0 ist, wird der String als UTF-8 interpretiert, sofern er mit einem UTF-8-BOM beginnt, oder als EBCDIC-UTF-8, sofern er mit einem EBCDIC-UTF-8-BOM beginnt, oder aber als Encoding *auto*, sofern kein BOM gefunden werden konnte (oder *ebcdic* auf EBCDIC-basierten Plattformen).

**Unicode-Verarbeitung für Optionslisten.** Strings innerhalb von Optionslisten erfordern besondere Beachtung, da sie sich nicht als Unicode-Strings im UTF-16-Format ausdrücken lassen, sondern nur als Byte-Arrays. Daher wird für Unicode-Optionen UTF-8 verwendet. Anhand der Suche nach einem BOM am Anfang einer Option entscheidet TET, wie sie zu interpretieren ist. Anhand des BOM wird das Format des Strings bestimmt. Genauer gesagt, wird eine String-Option wie folgt interpretiert:

- ▶ Wenn die Option mit einem UTF-8-BOM beginnt ( $\backslash\text{EF}\backslash\text{BB}\backslash\text{BF}$ ), wird sie als UTF-8 interpretiert.
- ▶ Wenn sie mit einem EBCDIC-UTF-8-BOM ( $\backslash\text{x57}\backslash\text{x8B}\backslash\text{xAB}$ ) beginnt, wird sie als EBCDIC-UTF-8 interpretiert.
- ▶ Wird kein BOM gefunden, wird der String als *winansi* (oder *ebcdic* auf EBCDIC-basierten Plattformen) interpretiert.

*Hinweis* Mit der Hilfsfunktion `TET_convert_to_unicode()` lassen sich aus UTF-16-Strings UTF-8-Strings erzeugen, was für die Erzeugung von Optionslisten mit Unicode-Werten nützlich ist.

**Einsatz von TET als DLL, die zur Laufzeit geladen wird.** Die meisten Clients werden TET als statisch gebundene oder dynamische Bibliothek einsetzen, die beim Linken gebun-

den wird. Sie können die DLL aber auch zur Laufzeit laden und sich dynamisch Zeiger auf alle API-Funktionen besorgen. Dies ist insbesondere sinnvoll, wenn die DLL erst bei Bedarf geladen wird, und unter MVS, wo die Bibliothek üblicherweise als DLL zur Laufzeit geladen wird, ohne die Applikation überhaupt mit der TET-Bibliothek zu linken. Zur Verwendung dieser Methode gehen Sie wie folgt vor:

- ▶ Inkludieren Sie *tetlibdl.h* statt *tetlib.h*.
- ▶ Verwenden Sie *TET\_new\_dl()* und *TET\_delete\_dl()* statt *TET\_new()* und *TET\_delete()*.
- ▶ Verwenden Sie *TET\_TRY\_DL()* und *TET\_CATCH\_DL()* statt *TET\_TRY()* und *TET\_CATCH()*.
- ▶ Arbeiten Sie bei allen anderen TET-Aufrufen mit Funktionszeigern.
- ▶ Kompilieren Sie zusätzlich das Hilfsmodul *tetlibdl.c* und binden Sie die entsprechende Objektdatei mit Ihrer Anwendung.

Das dynamische Laden wird im Beispiel *extractordl.c* veranschaulicht.

## 3.3 C++ -Sprachbindung

*Hinweis* Für in C++ geschriebene .NET-Anwendungen empfehlen wir, auf die .NET-DLL von TET direkt zuzugreifen, anstatt über die C++-Sprachbindung (plattformübergreifende Anwendungen sollten allerdings die C++-Sprachbindung verwenden). Das TET-Paket enthält C++-Beispielcode für .NET CLI (Common Language Infrastructure), der diese Kombination veranschaulicht.

Neben der C-Include-Datei *tetlib.h* wird für TET-Clients ein objektorientierter Wrapper für C++ mitgeliefert. Dieser erfordert die Include-Datei *tet.hpp*, die wiederum *tetlib.h* inkludiert. Da die Implementierung von *tet.hpp* auf Templates basiert, wird kein entsprechendes *tet.cpp*-Modul benötigt. Der C++-Wrapper ersetzt den funktionalen Ansatz (mit Funktionen und Präfix *TET\_* in allen TET-Funktionsnamen) durch einen stärker objektorientierten Ansatz.

**String-Behandlung in C++.** Mit dem Template-basierten Ansatz in TET wird die String-Verarbeitung folgendermaßen unterstützt:

- ▶ Strings vom Typ *std::wstring* der C++-Standard-Bibliothek werden als grundlegender String-Typ verwendet. Sie können UTF-16- oder UTF-32-kodierte Unicode-Zeichen enthalten. Dies ist das voreingestellte Verhalten und die empfohlene Vorgehensweise für neue Anwendungen, es sei denn, benutzerdefinierte Datentypen bieten einen erheblichen Vorteil gegenüber *wstrings* (siehe nächster Punkt).
- ▶ Benutzerdefinierte Datentypen können für die String-Verarbeitung verwendet werden, sofern der benutzerdefinierte Datentyp eine Instantiierung des Klassen-Templates *basic\_string* ist und mit vom Client übergebenen Konvertierungsmethoden von und nach Unicode konvertiert werden kann. Diese Methode wird im Beispiel *glyphinfo.cpp* in der TET-Distribution dargestellt.

Die Standard-Schnittstelle geht davon aus, dass alle an TET-Methoden übergebenen und von TET-Methoden zurückgegebenen Strings native *wstrings* sind. Abhängig von der Größe des Datentyps *wchar\_t* müssen *wstrings* Unicode-Strings enthalten, die als UTF-16 (2-Byte-Zeichen) oder UTF-32 (4-Byte-Zeichen) kodiert sind. Literalen Strings im Quellcode muss ein *L* vorangestellt werden, um sie als Wide Strings zu kennzeichnen. Unicode-Zeichen in Literalen können mit der Syntax *\u* und *\U* erstellt werden. Obwohl diese Syntax Teil der ISO-Norm von C++ ist, wird sie von einigen Compilern nicht unterstützt. In diesem Fall müssen literale Unicode-Zeichen mit Hex-Ziffern erstellt werden.

*Hinweis* Auf EBCDIC-basierten Systemen erfordert das Formatieren der Optionslisten-Strings für die auf *wstring* basierende Schnittstelle eine zusätzliche Konvertierung, um eine Mischung aus EBCDIC- und UTF-16-*wstrings* in Optionslisten zu vermeiden. Beispiel-Code sowie Anweisungen für diese Konvertierung finden Sie im Hilfsmodul *utf16num\_ebcdic.hpp*.

**Fehlerbehandlung in C++.** TET-API-Funktionen lösen im Fehlerfall eine C++-Exception aus. Diese Exceptions müssen im Client-Code mit den üblichen *try/catch*-Anweisungen von C++ abgefangen werden. Für ausführlichere Fehlerinformationen stellt die Klasse TET die öffentliche (*public*) Klasse *TET::Exception* mit Methoden zur Verfügung, die die genaue Fehlermeldung, die Exception-Nummer sowie den Namen der API-Funktion liefern, die die Exception ausgelöst hat.

Native C++-Exceptions, die durch TET-Routinen ausgelöst wurden, verhalten sich wie erwartet. Das folgende Codefragment fängt Exceptions ab, die von TET ausgelöst werden:

```

try {
    ...TET-Anweisungen...
} catch (TET::Exception &ex) {
    wcerr << L"Error " << ex.get_errnum()
    << L" in " << ex.get_apiname()
    << L"(): " << ex.get_errmsg() << endl;
}

```

**Einsatz von TET als DLL, die zur Laufzeit geladen wird.** Ähnlich wie bei der C-Sprachbindung kann TET mit der C++-Sprachbindung dynamisch zur Laufzeit in Ihre Anwendung geladen werden (siehe »Einsatz von TET als DLL, die zur Laufzeit geladen wird«, Seite 30). Das dynamische Laden beim Kompilieren des Anwendungsmoduls, das *tet.hpp* inkludiert, können Sie folgendermaßen aktivieren:

```
#define TETCPP_DL 1
```

Kompilieren Sie zusätzlich das Hilfsmodul *tetlibdl.c* und binden Sie die entsprechende Objektdatei mit Ihrer Anwendung. Da die Details des dynamischen Ladens im TET-Objekt versteckt sind, ist das C++-API davon nicht betroffen: alle Methodenaufrufe sehen gleich aus, unabhängig davon, ob dynamisches Laden aktiviert ist. Das dynamische Laden wird im Beispiel *extractordl* im mitgelieferten Makefile veranschaulicht.

## 3.4 COM-Sprachbindung

**Installation der COM-Edition von TET.** TET kann in allen Umgebungen eingesetzt werden, die COM-Komponenten unterstützen. Die Installation von TET ist einfach zu bewerkstelligen. Beachten Sie dabei Folgendes:

- ▶ Wenn Sie TET auf einer NTFS-Partition installieren, benötigen alle TET-Benutzer die Lesen-Berechtigung für das Installationsverzeichnis und die Ausführen-Berechtigung für  
... \TET5.0 32-bit\bind\COM\bin\tet\_com.dll.
- ▶ Der installierende Benutzer benötigt die Schreiben-Berechtigung für die Registry des Systems. Administrator-Berechtigungen oder Berechtigungen der Gruppe »Hauptbenutzer« reichen in der Regel aus.

**Verarbeitung von Exceptions.** Die Verarbeitung von Exceptions für die TET-COM-Komponente erfolgt entsprechend der COM-Konventionen: Tritt eine TET-Exception auf, so wird eine COM-Exception ausgelöst, die eine Klartextbeschreibung des Fehlers enthält. Außerdem wird der vom TET-Objekt belegte Speicher freigegeben. Die COM-Exception kann im TET-Client abgefangen und auf die Art bearbeitet werden, die die Client-Umgebung für COM-Fehler vorsieht.

**Einsatz der COM-Edition von TET mit .NET.** Alternativ zu TET.NET (siehe Abschnitt 3.6, »*.NET-Sprachbindung*«, Seite 37) kann die COM-Edition von TET auch mit .NET verwendet werden. Dazu müssen Sie aus der COM-Edition von TET mit dem Hilfsprogramm *tlbimp.exe* eine .NET-Assembly erstellen:

```
tlbimp tet_com.dll /namespace:tet_com /out:Interop.tet_com.dll
```

Diese Assembly verwenden Sie dann in Ihrer .NET-Anwendung. Wenn Sie innerhalb von Visual Studio .NET eine Referenz auf *tet\_com.dll* hinzufügen, wird automatisch eine Assembly erzeugt. Das folgende Codefragment zeigt den Einsatz der COM-Edition von TET mit C#:

```
using TET_com;
...
static TET_com.ITET tet;
...
tet = New TET();
...
```

Der übrige Code ist der gleiche wie bei der .NET-Edition von TET.

## 3.5 Java-Sprachbindung

**Installation der Java-Edition von TET.** TET ist als Java-Paket namens *com.pdflib.TET* zusammengefasst. Dieses Paket benötigt eine native JNI-Bibliothek; beide Teile müssen entsprechend konfiguriert werden.

Um die JNI-Bibliothek verfügbar zu machen, sind die folgenden plattformabhängigen Schritte durchzuführen:

- ▶ Unter Unix muss die Bibliothek *libtet\_java.so* (unter OS X: *libtet\_java.jnilib*) in eines der Standardverzeichnis für dynamisch ladbare Bibliotheken oder in ein entsprechend konfiguriertes Verzeichnis kopiert werden.
- ▶ Unter Windows muss die Bibliothek *tet\_java.dll* ins Windows-Systemverzeichnis oder in ein Verzeichnis kopiert werden, das in der Umgebungsvariablen *PATH* aufgeführt ist.

Das Java-Paket von TET befindet sich in der Datei *TET.jar*. Zur Verwendung dieses Pakets in Ihrer Anwendung müssen Sie *TET.jar* in die Umgebungsvariable *CLASSPATH* eintragen, die Option *-classpath TET.jar* in die Aufrufe des Java-Compilers aufnehmen oder entsprechende Schritte in Ihrer Java IDE durchführen. Im JDK können Sie die Java-VM so konfigurieren, dass sie ein vorgegebenes Verzeichnis nach nativen Bibliotheken durchsucht. Dazu weisen Sie der Property *java.library.path* den Namen des gewünschten Verzeichnisses zu, zum Beispiel:

```
java -Djava.library.path=. extractor
```

Der Wert dieser Property lässt sich wie folgt überprüfen:

```
System.out.println(System.getProperty("java.library.path"));
```

**Einsatz von TET in J2EE-Applikationsservern und Servlet-Containern.** TET eignet sich hervorragend für serverseitige Java-Anwendungen. Das TET-Paket enthält Beispielcode und Konfigurationen für die Verwendung von TET in J2EE-Umgebungen. Beachten Sie dabei die folgenden Konfigurationsaspekte:

- ▶ Das Verzeichnis, in dem der Server die nativen Bibliotheken erwartet, ist je nach Anbieter unterschiedlich. Üblich sind Systemverzeichnisse, Verzeichnisse der zugrunde liegenden Java-VM oder lokale Server-Verzeichnisse. Einzelheiten hierzu finden Sie in der Dokumentation, die vom Hersteller des Servers bereitgestellt wird.
- ▶ Applikationsserver und Servlet-Container verwenden oft einen speziellen Klassenloader, der möglicherweise Einschränkungen unterliegt oder einen bestimmten Klassenpfad verwendet. Bei manchen Servern muss ein besonderer Engine-Klassenpfad festgelegt werden, damit das TET-Paket gefunden werden kann.

Ausführlichere Hinweise zum Einsatz von TET mit verschiedenen Servlet-Engines und Applikationsservern finden Sie in der Dokumentation im Verzeichnis J2EE der TET-Distribution.

**Konvertierung von Unicode und anderen Encodings.** Um TET-Anwendern die Arbeit zu erleichtern, zeigen wir im Folgenden einige nützliche Methoden zur String-Konvertierung. Weitere Informationen hierzu finden Sie in der Java-Dokumentation.

Der folgende Konstruktor erzeugt einen Unicode-String aus einem Byte-Array, wobei das Standard-Encoding der Plattform verwendet wird:

```
String(byte[] bytes)
```

Der folgende Konstruktor erzeugt einen Unicode-String aus einem Byte-Array, wobei das im Parameter *enc* übergebene Encoding (z.B. *SJIS*, *UTF8*, *UTF-16*) verwendet wird:

```
String(byte[] bytes, String enc)
```

Die folgende Methode der Klasse `String` konvertiert einen Unicode-String anhand des im Parameter *enc* definierten Encodings in einen `String`:

```
byte[] getBytes(String enc)
```

**Javadoc-Dokumentation für TET.** Das TET-Paket enthält Javadoc-Dokumentation für TET. Javadoc enthält nur abgekürzte Beschreibungen aller TET-API-Methoden; für weitere Informationen siehe Abschnitt 10, »API-Referenz für die TET-Bibliothek«, Seite 173.

Um Javadoc für TET in Eclipse zu konfigurieren, gehen Sie folgendermaßen vor:

- ▶ Klicken Sie im Paket-Explorer mit der rechten Maustaste auf das Java-Projekt und wählen Sie *Javadoc Location*.
- ▶ Klicken Sie auf *Browse...* und wählen Sie den Pfad, wo sich das Javadoc-Paket befindet (Bestandteil des TET-Pakets).

Dann können Sie die Javadoc für TET durchsuchen, z.B. mit der *Java-Browsing*-Perspektive oder über das Hilfe-Menü.

**Verarbeitung von Exceptions.** In der Java-Sprachbindung von TET werden native Java-Exceptions der Klasse `TETException` ausgelöst. Der TET-Client-Code muss die Standard-syntax für Java-Exceptions verwenden:

```
TET tet = null;

try {
    ...TET-Methodenaufrufe...

} catch (TETException e) {
    System.err.print("TET-Exception aufgetreten:\n");
    System.err.print("[ " + e.get_errnum() + " ] " + e.get_apiname() + ": " +
        e.get_errmsg() + "\n");
} catch (Exception e) {
    System.err.println(e.getMessage());
} finally {
    if (tet != null) {
        tet.delete();           /* TET-Objekt löschen */
    }
}
```

Da TET passende *throws*-Klauseln deklariert, muss der Client-Code entweder alle möglichen Exceptions abfangen oder diese selbst deklarieren.

## 3.6 .NET-Sprachbindung

*Hinweis* Ausführliche Informationen zu den verschiedenen Ausprägungen und Möglichkeiten für die Verwendung von TET mit dem .NET-Framework finden Sie im Dokument *PDFlib-in-.NET-HowTo.pdf*, das in den Produktpaketen enthalten und auch über die *PDFlib-Website* verfügbar ist.

Die .NET-Edition von TET unterstützt alle wesentlichen .NET-Konzepte. Technisch gesehen handelt es sich bei der .NET-Edition von TET um eine C++-Klasse (mit einem managed Wrapper um die unmanaged TET-Kernbibliothek), die unter Kontrolle des .NET-Frameworks abläuft. Diese Klasse wird als statische Assembly mit einem starken Namen (*strong name*) ausgeliefert. Die TET-Assembly (*TET\_dotnet.dll*) enthält die Bibliothek selbst sowie zusätzliche Meta-Informationen.

**Installation der TET-Edition für .NET.** Die MSI-Installationsroutine von TET.NET installiert die TET-Assembly einschließlich der zugehörigen Hilfsdateien, Dokumentation und Beispiele interaktiv auf dem Rechner. Außerdem wird TET registriert, so dass Sie auf der Registerkarte .NET im Dialogfeld *Add Reference* von Visual Studio .NET sofort darauf zugreifen können.

**Fehlerbehandlung.** TET.NET unterstützt .NET-Exceptions und löst eine Exception mit detaillierter Fehlermeldung aus, sobald ein Laufzeitproblem auftritt. Der Client ist für das Abfangen der Exception und eine angemessene Reaktion zuständig. Andernfalls fängt das .NET-Framework die Exception ab, was gewöhnlich zum Abbruch der Anwendung führt.

Um Informationen über die Exception zu übermitteln, definiert TET eine eigene Exception-Klasse namens *TET\_dotnet.TETException* mit den Members *get\_errnum*, *get\_errmsg* und *get\_apiname*.

**Einsatz von TET mit C++ und CLI.** In C++ geschriebene .NET-Anwendungen (basierend auf der *Common Language Infrastructure CLI*) können ohne die C++-Sprachbindung von TET direkt auf die TET.NET-DLL zugreifen. Dazu muss TET im Quellcode folgendermaßen referenziert werden:

```
using namespace TET_dotnet;
```

## 3.7 Objective-C-Sprachbindung

Obwohl die C- und C++-Sprachbindungen mit Objective-C verwendet werden können, bieten wir auch eine genuine Sprachbindung für Objective-C an. Das TET-Framework ist in den folgenden Ausprägungen erhältlich:

- *TET* für OS X
- *TET\_ios* für iOS

Beide Frameworks enthalten Sprachbindungen für C, C++ und Objective-C.

**Installation der TET-Edition für Objective-C unter OS X.** Um TET in Ihrer Anwendung einsetzen zu können, kopieren Sie *TET.framework* oder *TET\_ios.framework* in das Verzeichnis */Library/Frameworks*. Sie können das TET-Framework auch an einem anderen Ort installieren, benötigen dazu aber das *install\_name\_tool* von Apple, das hier nicht beschrieben wird. Die Header-Datei *TET\_objc.h* mit den Methoden-Deklarationen von TET müssen Sie im Quellcode Ihrer Anwendung importieren:

```
#import "TET/TET_objc.h"
```

oder

```
#import "TET_ios/TET_objc.h"
```

**Namenskonventionen für Parameter.** Für TET-Methodenaufrufe müssen Sie die Parameter gemäß der folgenden Konventionen übergeben:

- Der Wert des ersten Parameters wird direkt nach dem Methodennamen, durch einen Doppelpunkt getrennt, angegeben.
- Für jeden weiteren Parameter muss der Parametername mit seinem Wert (wiederum jeweils getrennt durch einen Doppelpunkt) angegeben werden. Die Parameternamen finden Sie in Kapitel 10, »API-Referenz für die TET-Bibliothek«, Seite 173 und in der Datei *TET\_objc.h*.

Die folgende Zeile aus der API-Funktionsbeschreibung:

```
int open_page(int doc, int pagenumber, String optlist)
```

entspricht der folgenden Objective-C-Methode:

```
-(NSInteger) open_page: (NSInteger) doc pagenumber: (NSInteger) pagenumber optlist: (NSString *) optlist;
```

Ihre Anwendung muss daher ungefähr folgenden Aufruf absetzen:

```
page = [tet open_page:doc pagenumber:pageno optlist:pageoptlist];
```

Xcode Code Sense kann zur Code-Vervollständigung mit dem TET-Framework verwendet werden.

**Fehlerbehandlung in Objective-C.** Die Objective-C-Sprachbindung übersetzt TET-Exceptions in native Objective-C-Exceptions. Bei einem Laufzeitproblem löst TET eine native Objective-C-Exception der Klasse *TETException* aus. Diese Exceptions können mit der üblichen *try/catch*-Methode behandelt werden:

```

@try {
    ...TET-Anweisungen...
}
@catch (TETException *ex) {
    NSString * errorMessage =
        [NSString stringWithFormat:@"TET error %d in '%@': %@",
        [ex get_errnum], [ex get_apiname], [ex get_errmsg]];
    UIAlertView *alert = [[UIAlertView alloc] init];
    [alert setMessageText: errorMessage];
    [alert runModal];
    [alert release];
}
@catch (NSEException *ex) {
    UIAlertView *alert = [[UIAlertView alloc] init];
    [alert setMessageText: [ex reason]];
    [alert runModal];
    [alert release];
}
@finally {
    [tet release];
}

```

Außer der Methode *get\_errmsg* können Sie auch das Feld *reason* des Exception-Objekts verwenden, um Fehlermeldungen zu erhalten.

## 3.8 Perl-Sprachbindung

Der TET-Wrapper für Perl besteht aus einer C-Wrapperdatei und zwei Perl-Paketmodulen, eins zur Bereitstellung eines Perl-Äquivalents für jede TET-API-Funktion und ein weiteres für das TET-Objekt. Das C-Modul wird zum Aufbau einer dynamischen Bibliothek verwendet, die vom Perl-Interpreter unter Zuhilfenahme der Paketdatei zur Laufzeit geladen wird. Perl-Skripte referenzieren das Bibliotheksmodul mit einer *use*-Anweisung.

**Installation der TET-Edition für Perl.** Der Erweiterungsmechanismus von Perl lädt dynamische Bibliotheken zur Laufzeit mittels des DynaLoader-Moduls. Perl selbst muss mit Unterstützung dynamischer Bibliotheken kompiliert worden sein (das ist bei den meisten Perl-Konfigurationen der Fall).

Damit die TET-Sprachbindung funktioniert, benötigt der Perl-Interpreter Zugriff auf den TET-Perl-Wrapper und die Module *tetlib\_pl.pm* und *PDFlib/TET.pm*. Zusätzlich zu den unten beschriebenen plattformspezifischen Methoden können Sie mit der Perl-Kommandozeilenoption *-I* ein Verzeichnis zum Modulsuchpfad *@INC* hinzufügen, zum Beispiel:

```
perl -I/path/to/tet extractor.pl
```

**Unix.** Perl sucht *tetlib\_pl.so* (unter OS X: *tetlib\_pl.bundle*), *tetlib\_pl.pm* und *PDFlib/TET.pm* im aktuellen Verzeichnis oder in dem Verzeichnis, das mit folgendem Befehl ausgegeben wird:

```
perl -e 'use Config; print $Config{sitearchexp};'
```

Perl durchsucht außerdem das Unterverzeichnis *auto/tetlib\_pl*. Der obige Befehl liefert eine Ausgabe, die in etwa wie folgt aussieht:

```
/usr/lib/perl5/site_perl/5.16/i686-linux
```

**Windows.** Die DLL *tetlib\_pl.dll* und die Module *tetlib\_pl.pm* und *PDFlib/TET.pm* werden im aktuellen Verzeichnis gesucht oder im Verzeichnis, das mit folgendem Perl-Befehl ausgegeben wird:

```
perl -e "use Config; print $Config{sitearchexp};"
```

Der obige Befehl liefert eine Ausgabe, die in etwa wie folgt aussieht:

```
C:\Programme\Perl5.16\site\lib
```

**Verarbeitung von Exceptions in Perl.** Wenn eine TET-Exception auftritt, wird eine Perl-Exception ausgelöst. Sie kann mit einer *eval*-Sequenz abgefangen und verarbeitet werden:

```
eval {  
    ...TET-Anweisungen...  
};  
die "Exception caught: $@" if $@;
```

## 3.9 PHP-Sprachbindung

*Hinweis* Ausführliche Informationen über die verschiedenen Möglichkeiten des Einsatzes von TET mit PHP finden Sie in der Datei *PDFlib-in-PHP-HowTo.pdf*, das in den Produktpaketen enthalten und auch über die *PDFlib-Website* verfügbar ist. Obwohl es vor allem den Einsatz von *PDFlib* mit PHP betrifft, gilt die Diskussion auch für den Einsatz von TET mit PHP.

**Installation der TET-Edition für PHP.** TET ist als C-Bibliothek implementiert, die dynamisch in PHP eingebunden werden kann. TET unterstützt verschiedene PHP-Versionen. Abhängig von der verwendeten PHP-Version müssen Sie die entsprechende TET-Bibliothek aus dem entpackten TET-Archiv auswählen.

Sie müssen PHP per Konfiguration über die externe TET-Bibliothek informieren. Dazu gibt es zwei Möglichkeiten:

- ▶ Fügen Sie in *php.ini* eine der folgenden Zeilen ein:

```
extension=php_tet.dll      ; für Windows
extension=php_tet.so      ; für Unix und OS X
extension=php_tet.sl      ; für HP-UX
```

PHP sucht die Bibliothek in dem Verzeichnis, das unter Unix in der Variablen *extension\_dir* in der Datei *php.ini* verzeichnet ist. Unter Windows werden außerdem die Standardsystemverzeichnisse durchsucht. Mit dem folgenden einzeiligen PHP-Skript können Sie ermitteln, welche Version der PHP-Sprachbindung von TET Sie installiert haben:

```
<?phpinfo()?>
```

Angezeigt wird eine lange Info-Seite über Ihre aktuelle PHP-Konfiguration. Suchen Sie auf der Seite nach dem Abschnitt *tet*. Wenn dieser Abschnitt den Satz enthält

```
PDFlib TET Support          enabled
```

(plus der TET-Versionsnummer), haben Sie TET für PHP erfolgreich installiert.

- ▶ Alternativ laden Sie TET zur Laufzeit, wobei Sie eine der folgenden Zeilen an den Anfang Ihres Skripts stellen müssen:

```
dl("php_tet.dll");        # für Windows
dl("php_tet.so");         # für Unix und OS X
dl("php_tet.sl");         # für HP-UX
```

**Behandlung von Dateinamen in PHP.** Nicht qualifizierte Dateinamen (also solche ohne Pfadangabe) sowie relative Dateinamen werden in der Unix- und der Windows-Version von PHP unterschiedlich behandelt:

- ▶ Unter Unix sucht PHP Dateien ohne Pfadangabe in dem Verzeichnis, in dem sich das Skript befindet.
- ▶ Unter Windows sucht PHP Dateien ohne Pfadangabe nur in dem Verzeichnis, in dem sich die PHP-DLL befindet.

**Verarbeitung von Exceptions.** Da PHP strukturierte Ausnahmebehandlung unterstützt, werden TET-Exceptions als PHP-Exceptions weitergegeben. TET-Exceptions können also mit der üblichen Kombination aus *try/catch* abgefangen werden:

```
try {
```

...TET-Anweisungen...

```
} catch (TETException $e) {  
    print "TET-Exception aufgetreten:\n";  
    print "[" . $e->get_errnum() . "]" " " . $e->get_apiname() . ": "  
        $e->get_errmsg() . "\n";  
}  
catch (Exception $e) {  
    print $e;  
}
```

**Entwicklung mit Eclipse und Zend Studio.** Die PHP Development Tools (PDT) unterstützen die PHP-Entwicklung mit Eclipse und Zend Studio. Für PDT kann kontextsensitive Hilfe mit den unten beschriebenen Schritten konfiguriert werden.

Fügen Sie TET zu den Eclipse-Voreinstellungen hinzu, um es bei allen PHP-Projekten bekannt zu machen:

- ▶ Wählen Sie *Window, Preferences, PHP, PHP Libraries, New...* Ein Assistent wird gestartet.
- ▶ Fügen Sie unter *User library name* das Wort *TET* ein, klicken Sie auf *Add External folder...* und wählen Sie das Verzeichnis *bind\php\Eclipse PDT*.

In einem bestehenden oder neuen PHP-Projekt können Sie folgendermaßen einen Verweis auf die TET-Bibliothek legen:

- ▶ Klicken Sie im PHP-Explorer mit der rechten Maustaste auf das PHP-Projekt und wählen Sie *Include Path, Configure Include Path...*
- ▶ Gehen Sie zur Registermarke *Libraries*, klicken Sie auf *Add Library...* und wählen Sie *User Library, TET*.

Dann können Sie in der PHP-Explorer-Ansicht die Liste der TET-Methoden unter dem Knoten *PHP Include Path/TET/TET* durchsuchen. Beim Schreiben von neuem PHP-Code bietet Eclipse mit Code-Vervollständigung und kontextsensitiver Hilfe Unterstützung für alle TET-Methoden.

## 3.10 Python-Sprachbindung

**Installation der TET-Edition für Python.** Der Erweiterungsmechanismus von Python lädt dynamische Bibliotheken zur Laufzeit. Damit die TET-Sprachbindung funktioniert, benötigt der Python-Interpreter Zugriff auf die TET-Bibliothek für Python, nach der in den Verzeichnissen gesucht wird, die in der Umgebungsvariable PYTHONPATH aufgeführt sind. Der Name des Python-Wrappers ist plattformabhängig:

- ▶ Unix und OS X: *tetlib\_py.so*
- ▶ Windows: *tetlib\_py.pyd*

**Fehlerbehandlung in Python.** Die Python-Sprachbindung übersetzt TET-Exceptions in native Python-Exceptions. Die Python-Exceptions können mit der üblichen Kombination aus TRY und EXCEPT behandelt werden:

```
try:
    ...TET-Anweisungen...
except TETException:
    print("TET-Exception aufgetreten:\n[%d] %s: %s" %
          ((tet.get_errnum()), tet.get_apiname(), tet.get_errmsg()))
```

## 3.11 REALbasic/Xojo-Sprachbindung

**Installation der TET-Edition für REALbasic/Xojo.** Das TET-Plugin für REALbasic/Xojo (*TET.rbx*) muss in den Ordner *Plugins* kopiert werden, der sich im selben Ordner wie die REALbasic/Xojo-Anwendung befindet. Für REALbasic/Xojo enthält TET Varianten für OS X, Windows und Linux. Das bedeutet, dass Sie jede Plattformversion von REALbasic/Xojo verwenden können, um Anwendungen für alle unterstützten Plattformen zu erzeugen. Beim Erstellen einer eigenständigen Applikation sucht sich REALbasic/Xojo aus dem TET-Plugin die benötigten Bestandteile heraus und bettet die plattformspezifischen Teile in die generierte Anwendung ein.

**Zusätzliche REALbasic/Xojo-Klassen.** TET erweitert die Objekthierarchie von REALbasic/Xojo um zwei neue Klassen:

- ▶ Die Klasse *TET* enthält alle TET-API-Methoden.
- ▶ Mit der Klasse *TETException*, die von *RuntimeException* abgeleitet ist, lassen sich von TET ausgelöste Exceptions verarbeiten (siehe unten).

Mit TET können sowohl GUI-Anwendungen als auch Anwendungen für die Kommandozeile erstellt werden. Da TET kein Steuerelement ist, wird auch kein neues Symbol in der Steuerelement-Palette installiert. Sobald das TET-Plugin verfügbar ist, ist die Klasse *TET* mit ihren Methoden jedoch in REALbasic/Xojo bekannt. So funktionieren zum Beispiel die Befehlsergänzung oder Parameterüberprüfung für TET-API-Methoden in vollem Umfang.

**Fehlerbehandlung in REALbasic/Xojo.** Bei einem Laufzeitproblem löst TET eine native REALbasic-Exception der Klasse *TETException* aus. TET-Exceptions lassen sich mit einem gewöhnlichen *try/catch*-Block behandeln.

## 3.12 Ruby-Sprachbindung

**Installation der Ruby-Edition von TET.** Der Erweiterungsmechanismus von Ruby lädt eine dynamische Bibliothek zur Laufzeit. Damit die TET-Sprachbindung funktioniert, benötigt der Ruby-Interpreter Zugriff auf die TET-Erweiterungsbibliothek für Ruby. Diese Bibliothek (unter Windows und Unix: *TET.so*; unter OS X: *TET.bundle*) wird normalerweise im Unterverzeichnis *site\_ruby* des lokalen Ruby-Installationsverzeichnis installiert, das heißt in einem Verzeichnis mit etwa folgendem Namen:

```
/usr/local/lib/ruby/site_ruby/<version>/
```

Ruby durchsucht aber auch andere Verzeichnisse nach Erweiterungen. Mit folgendem Ruby-Aufruf erhalten Sie eine Liste dieser Verzeichnisse:

```
ruby -e "puts $:"
```

Diese Liste enthält in der Regel auch das aktuelle Verzeichnis, so dass Sie die TET-Erweiterungsbibliothek und die Skripten zum Testen einfach ins gleiche Verzeichnis stellen können.

**Fehlerbehandlung in Ruby.** Die Ruby-Sprachbindung installiert einen Error-Handler, der TET-Exceptions in native Ruby-Exceptions übersetzt. Die Ruby-Exceptions können mit der üblichen *rescue*-Technik verarbeitet werden:

```
begin
  ...TET-Anweisungen...
rescue TETException => pe
  print pe.backtrace.join("\n") + "\n"
  print "Error [" + pe.get_errnum.to_s + "]" + pe.get_apiname + ": " + pe.get_errmsg
  print " on page pageno" if (pageno != 0)
  print "\n"
rescue Exception => e
  print e.backtrace.join("\n") + "\n" + e.to_s + "\n"
ensure
  tet.delete() if tet
end
```

**Ruby on Rails.** Ruby on Rails ist ein Open-Source-Framework, das die Webentwicklung mit Ruby erleichtert. Die TET-Erweiterung für Ruby ist auch mit Ruby on Rails einsetzbar. Um die TET-Beispiele für Ruby on Rails auszuführen, gehen Sie wie folgt vor:

- ▶ Installieren Sie Ruby und Ruby on Rails.
- ▶ Richten Sie einen neuen Controller von der Kommandozeile aus ein:

```
$ rails new tetdemo
$ cd tetdemo
$ cp <TET dir>/bind/ruby/<version>/TET.so vendor/ # eins von .so/.dll/.bundle
$ cp <TET dir>/bind/data/TET-datasheet.pdf .
$ rails generate controller home demo
$ rm public/index.html
```

- ▶ Editieren Sie *config/routes.rb*:

```
...
# Vergessen Sie nicht, public/index.html zu löschen
```

```
root :to => "home#demo"
```

- ▶ Editieren Sie *app/controllers/home\_controller.rb* wie unten beschrieben und fügen Sie TET-Code zur Extraktion des PDF-Inhalts ein. Als Ausgangspunkt können Sie den Code aus dem Beispiel *extractor-rails.rb* verwenden:

```
class HomeController < ApplicationController
  def demo
    require "TET"
    begin
      p = TET.new
      doc = tet.open_document(infilename, doctype)
      ...TET-Anwendungscode, siehe extractor-rails.rb...
      ...
      # abgefragter Text wird angezeigt
      send_data text, :type => "text/plain", :disposition => "inline"
      rescue TETException => pe
      # Fehlerbehandlung
    end
  end
end
```

- ▶ Um die Installation zu testen, starten Sie den WEBrick-Server mit folgendem Kommando:

```
$ rails server
```

Geben Sie im Browser *http://0.0.0.0:3000* ein. Der aus dem PDF-Dokument extrahierte Text wird im Browser angezeigt.

**Lokale TET-Installation.** Wenn Sie TET nur mit Ruby on Rails einsetzen möchten und nicht global zum allgemeinen Einsatz mit Ruby installieren können, können Sie TET auch lokal im Verzeichnis *vendors* in der Rails-Hierarchie installieren. Dies ist insbesondere dann nützlich, wenn Sie nicht über die Berechtigung verfügen, allgemeine Ruby-Erweiterungen zu installieren, aber in Rails mit TET arbeiten möchten.

## 3.13 RPG-Sprachbindung

TET bietet ein */copy*-Modul, das alle Prototypen sowie einige nützliche Konstanten definiert, die zur Kompilierung von ILE-RPG-Programmen mit eingebetteten TET-Funktionen benötigt werden.

**Unicode-String-Verarbeitung.** Da alle in TET verfügbaren Funktionen Unicode-Strings variabler Länge als Parameter verwenden, müssen Sie einen Ein-Byte-String mit der integrierten Funktion `%UCS2` in einen Unicode-String konvertieren. Alle von TET-Funktionen zurückgegebenen Strings sind Unicode-Strings variabler Länge. Mit der integrierten Funktion `%CHAR` konvertieren Sie Unicode-Strings in Ein-Byte-Strings.

*Hinweis* Die Funktionen `%CHAR` und `%UCS2` konvertieren Strings anhand der `CCSID` des aktuellen Jobs von und nach Unicode. Die mit TET mitgelieferten Beispiele basieren auf `CCSID 37` (US EBCDIC). Wenn Sie die Beispiele mit anderen Codepages ausführen, werden unter Umständen nicht alle Sonderzeichen in Optionslisten (z.B. { [ ] } ) korrekt umgesetzt.

Da alle Strings als Strings variabler Länge übergeben werden, dürfen Sie keinen Längensparameter in jenen Funktionen angeben, die explizite Übergabe der String-Länge erwarten (die Länge eines Strings variabler Länge wird in den ersten beiden Bytes des Strings gespeichert).

**Kompilieren und Binden von RPG-Programmen für TET.** Zum Einsatz von TET-Funktionen mit RPG ist das kompilierte Serviceprogramm TET erforderlich. Um die TET-Definitionen zur Kompilierzeit einzubinden, müssen Sie ihren Namen in den *D*-Anweisungen Ihres ILE-RPG-Programms angeben:

```
d/copy QRPGLSRC,TETLIB
```

Wenn sich die TET-Quelldateibibliothek nicht am Anfang Ihrer Bibliotheksliste befindet, müssen Sie zudem die Bibliothek angeben:

```
d/copy tetsrclib/QRPGLSRC,TETLIB
```

Bevor Sie mit der Kompilierung des ILE-RPG-Programms beginnen, müssen Sie ein Sprachbindungsverzeichnis anlegen, das das mit TET ausgelieferte Serviceprogramm TETLIB enthält. Das folgende Beispiel zeigt, wie Sie in der Bibliothek TETLIB das Bindungsverzeichnis TETLIB erstellen:

```
CRTBNDDIR BNDDIR(TETLIB/TETLIB) TEXT('TETlib Binding Directory')
```

Nach dem Anlegen des Sprachbindungsverzeichnisses müssen Sie das Serviceprogramm TETLIB zu Ihrem Sprachbindungsverzeichnis hinzufügen. Das folgende Beispiel zeigt, wie Sie das Serviceprogramm TETLIB in der Bibliothek TETLIB zum bereits erzeugten Bindungsverzeichnis hinzufügen:

```
ADDBNDDIRE BNDDIR(TETLIB/TETLIB) OBJ((TETLIB/TETLIB *SRVPGM))
```

Sie können Ihr Programm nun mit dem Befehl `CRTBNDRPG` (oder der Option 14 in PDM) kompilieren:

```
CRTBNDRPG PGM(TETLIB/EXTRACTOR) SRCFILE(TETLIB/QRPGLSRC) SRCMBR(*PGM) DFTACTGRP(*NO)  
BNDDIR(TETLIB/TETLIB)
```

**Fehlerbehandlung in RPG.** In ILE-RPG geschriebene TET-Clients können zur Fehlerbehandlung *monitor/on-error/endmon* von ILE-RPG verwenden. Außerdem können Exceptions auch mit dem globalen Unterprogramm *\*PSSR* von ILE-RPG überwacht werden. Wenn eine Exception auftritt, zeigt das Job-Protokoll die Fehlernummer, die fehlerhafte Funktion und den Grund für die Exception an. TET sendet eine Escape-Meldung an das aufrufende Programm.

```
c      eval      p=TET_new
*
c      monitor
*
c      callp     TET_set_option(tet:globaloptlist)
c      eval      doc=TET_open_document(tet:%ucs2(%trim(parm1)):docoptlist)
:
:
*      Fehlerbehandlung
c      on-error
*      Fehler wird verarbeitet
*      Vergessen Sie nicht, das TET-Objekt freizugeben
c      callp     TET_delete(tet)
c      endmon
```

# 4 TET-Konnektoren

TET-Konnektoren bieten den notwendige Verbindungscode für die Verbindung von TET mit anderer Software. Sie basieren auf der TET-Bibliothek oder dem TET-Kommandozeilen-Tool.

## 4.1 Kostenloses TET Plugin für Adobe Acrobat

In diesem Abschnitt wird das TET Plugin vorgestellt, eine frei verfügbare TET-Anwendung, die zu Testzwecken in Adobe Acrobat sowie für die interaktive Nutzung von TET mit jedem beliebigen PDF-Dokument verwendet werden kann. Das TET Plugin kann mit Acrobat X-DC Standard, Pro und Pro Extended verwendet werden (nicht jedoch mit dem kostenlosen Adobe Reader). Es kann unter folgender Adresse kostenlos heruntergeladen werden:

[www.pdflib.com/products/tet-plugin](http://www.pdflib.com/products/tet-plugin).

**Was bietet das TET Plugin?** Das TET Plugin erlaubt den einfachen interaktiven Zugriff auf TET. Obwohl es als Acrobat-Plugin konzipiert ist, basiert die Inhaltsextraktion vollständig auf der TET-Technologie und nutzt hierfür keine Acrobat-Funktionen. Das TET Plugin wird kostenlos zur Verfügung gestellt, um die leistungsstarke Funktionalität von PDFlib TET zu verdeutlichen. TET bietet sich als sinnvoller Ersatz für die Kopier- und Suchfunktionalität von Acrobat an, da es wesentlich mächtiger ist als die in Acrobat integrierten Werkzeuge zur Text- und Bildextraktion und zudem eine Reihe von praktischen Bedienmöglichkeiten bietet. PDFlib TET kann viele Dokumente erfolgreich verarbeiten, aus denen Acrobat keinen brauchbaren Text extrahieren kann. Das TET Plugin bietet die folgenden Funktionen:

- ▶ Kopieren von Text aus einem PDF-Dokument in die Zwischenablage oder eine Datei.
- ▶ Konvertieren eines PDF-Dokuments nach TETML und Speichern in der Zwischenablage oder einer Datei.
- ▶ Kopieren der XMP-Metadaten des Dokuments in die Zwischenablage.
- ▶ Finden von Wörtern im Dokument.
- ▶ Hervorheben aller Instanzen eines Suchbegriffs auf einer Seite gleichzeitig.
- ▶ Extrahieren von Bildern als TIFF, JPEG, JPEG 2000 oder JBIG2.
- ▶ Anzeigen von Farbraum- und Positionsinformationen von Bildern.
- ▶ Text- und Bildextraktion lassen sich durch Konfiguration genau an die eigenen Anforderungen anpassen. Die Einstellungen lassen sich speichern und wiederverwenden.

**Vorteile gegenüber der Kopierfunktion in Acrobat.** Die Kopierfunktion des TET Plugins ist in vielen Punkten der in Acrobat integrierten Kopierfunktion überlegen:

- ▶ Die Ausgabe kann so angepasst werden, dass sie unterschiedlichen Anforderungen entspricht.
- ▶ TET ist in vielen Fällen in der Lage den Text richtig zu interpretieren, in denen Acrobat keinen brauchbaren Text extrahiert.
- ▶ Unbekannte Glyphen (für die keine brauchbare Unicode-Zuordnung vorliegt) werden rot markiert und können durch ein benutzerdefiniertes Zeichen (etwa ein Fragezeichen) ersetzt werden.

- ▶ TET verarbeitet Dokumente wesentlich schneller als Acrobat.
- ▶ Bilder können interaktiv für den Export ausgewählt werden. Alternativ lassen sich auch alle Bilder einer Seite oder eines Dokuments extrahieren.
- ▶ Kleine Bildfragmente werden zu brauchbaren Bildern zusammengesetzt.

## 4.2 TET-Konnektor für die Suchmaschine Lucene

Lucene ist eine Open-Source-Suchmaschine. Lucene ist in erster Linie ein Java-Projekt, eine Version für .NET ist jedoch ebenfalls verfügbar. Für weitere Informationen zu Lucene siehe [lucene.apache.org](http://lucene.apache.org).

*Hinweis* Geschützte Dokumente können unter bestimmten Bedingungen mit der Option `shrug` indiziert werden (für weitere Informationen siehe Kapitel 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67). Dies wird zwar in den Konnektor-Dateien vorbereitet, Sie müssen diese Option jedoch manuell aktivieren.

**Voraussetzungen und Installation.** Die TET-Distribution enthält einen TET-Konnektor, mit dem das Indizieren von PDF-Dokumenten in Lucene Java aktiviert werden kann. Im folgenden wird der Konnektor für Lucene Java unter Annahme folgender Voraussetzungen ausführlicher beschrieben:

- ▶ JDK 1.7 oder höher für Lucene 5.2.x
- ▶ Eine funktionierende Installation des Build-Tools *Ant*
- ▶ Die Lucene-Distribution mit der Core-JAR-Datei von Lucene. Die mit TET ausgelieferte *Ant*-Datei erwartet die Dateien *lucene-core-x.x.x.jar*, *lucene-analyzers-common-x.x.x.jar* und *lucene-queryparser-x.x.x.jar*, die in der Lucene-Distribution enthalten sind.
- ▶ Ein installiertes TET-Paket für Unix, Linux, OS X oder Windows

Um den TET-Konnektor für Lucene zu implementieren, führen Sie folgende Schritte in der Eingabeaufforderung durch:

- ▶ Wechseln Sie in das Verzeichnis `<TET install dir>/connectors/lucene`.
- ▶ Kopieren Sie die Dateien *lucene-core-x.x.x.jar*, *lucene-analyzers-common-x.x.x.jar* und *lucene-queryparser-x.x.x.jar* in dieses Verzeichnis.
- ▶ Passen Sie optional in *TetReader.java* die Einstellungen durch Hinzufügen von globalen, Dokument- und Seitenoptionen von TET an. Mit der globalen Optionsliste können Sie zum Beispiel einen geeigneten Suchpfad für Ressourcen angeben (wenn die CJK-CMaps z.B. nicht im Standardinstallationsverzeichnis installiert sind). Das Modul *PdfDocument.java* zeigt, wie PDF-Dokumente zu verarbeiten sind, die entweder in einer Datei oder im Speicher abgelegt sind, wo sie zum Beispiel von einem Web-Crawler abgelegt wurden.
- ▶ Führen Sie das Kommando `ant index` aus. Damit wird der Sourcecode kompiliert und der Indexer für die PDF-Dateien angestoßen, die sich im Verzeichnis `<TET install dir>/bin/data` befinden.
- ▶ Führen Sie das Kommando `ant search` aus, um die Kommandozeilen-orientierte Suchanwendung zu starten, in die Abfragen in der Lucene-eigenen Abfragesprache eingegeben werden können.

### Testen von TET und Lucene mit der Kommandozeilen-orientierten Suchanwendung.

Das folgende Beispiel veranschaulicht Kommandos und Ausgabe für die Indizierung mit TET und Lucene und für die Prüfung des generierten Index mit dem Kommandozeilen-Tool von Lucene. Der Prozess wird mit dem Kommando `ant index` gestartet:

```
amira (1)$ ant index
Buildfile: build.xml
...
index:
```

```
[echo] Indexing PDF files in directory "../bind/data"
[java] adding ../bind/data/Whitepaper-Technical-Introduction-to-PDFA.pdf
[java] adding ../bind/data/Whitepaper-XMP-metadata-in-PDFlib-products.pdf
[java] adding ../bind/data/PDFlib-datasheet.pdf
[java] adding ../bind/data/TET-datasheet.pdf
[java] 662 total milliseconds
```

```
BUILD SUCCESSFUL
Total time: 1 second
```

```
amira (1)$ ant search
Buildfile: build.xml
```

```
compile:
```

```
search:
```

```
[java] Enter query:
PDFlib
[java] Searching for: pdflib
[java] 4 total matching documents
[java] 1. ../bind/data/PDFlib-datasheet.pdf
[java] Title: PDFlib, PDFlib+PDI, Personalization Server data sheet
[java] Font : PDFlibLogo-Regular
[java] Font : TheSans-Plain
...
[java] 2. ../bind/data/Whitepaper-XMP-metadata-in-PDFlib-products.pdf
[java] Title: Whitepaper: XMP Metadata support in PDFlib products
[java] Font : PDFlibLogo-Regular
[java] Font : TheSansLight-Italic
...
[java] 3. ../bind/data/Whitepaper-Technical-Introduction-to-PDFA.pdf
[java] Title: Whitepaper: A Technical Introduction to PDF/A
[java] Font : PDFlibLogo-Regular
[java] Font : TheSansLight-Italic
...
[java] 4. ../bind/data/TET-datasheet.pdf
[java] Title: PDFlib TET datasheet
[java] Subject: PDFlib TET reliably extracts text, images, and metadata from PDF
documents
[java] Font : TheSans-Plain
[java] Font : PDFlibLogo-Regular
...
[java] Press (q)uit or enter number to jump to a page.
q
[java] Enter query:
title:XMP
[java] Searching for: title:xmp
[java] 1 total matching documents
[java] 1. ../bind/data/Whitepaper-XMP-metadata-in-PDFlib-products.pdf
[java] Title: Whitepaper: XMP Metadata support in PDFlib products
[java] Font : PDFlibLogo-Regular
[java] Font : TheSansLight-Italic
...
```

Zwei Abfragen wurden durchgeführt: eine für das Wort *PDFlib* im Text und eine zweite für das Wort *XMP* im Feld *title*. Beachten Sie, dass *q* eingegeben werden muss, um den Ergebnismodus zu verlassen, bevor die nächste Abfrage gestartet werden kann.

Alle Pfade und Dateinamen in der Ant-Datei *build.xml* werden über Properties definiert, so dass die Datei in verschiedenen Umgebungen verwendet werden kann; dabei können die Properties entweder über die Kommandozeile oder durch Überschreiben der Properties in einer Datei *build.properties* oder sogar plattformspezifisch in den Dateien *windows.properties* oder *unix.properties* übergeben werden. Um das Beispiel mit einer Lucene-eigenen JAR-Datei auszuführen, die unter */tmp* installiert ist, können Sie die Ant-Datei folgendermaßen aufrufen:

```
ant -Dlucene-core.jar=/tmp/lucene-core-x.x.x.jar -Dlucene-analyzers-common.jar=/tmp/lucene-analyzers-common-x.x.x.jar -Dlucene-queryparser.jar=/tmp/lucene-queryparser-x.x.x.jar index
```

**Indizieren von Metadaten-Feldern.** Der TET-Konnektor für Lucene indiziert folgende Metadaten-Felder:

- ▶ *path (StringField)*: Pfadname des Dokuments
- ▶ *modified (DateLongField)*: Datum der letzten Änderung (gemäß dem PDF-Zeitstempel und nicht den PDF-Metadaten)
- ▶ *contents (ReaderTextField)*: vollständige Textinhalte des Dokuments
- ▶ Alle Standard- und benutzerdefinierten Dokument-Infofelder des PDF-Dokuments, wie Title, Subject, Author usw. Dokument-Infofelder können mit der pCOS-Schnittstelle abgefragt werden, die in TET integriert ist (für weitere Informationen zu pCOS siehe die pCOS Path Reference), z.B.

```
String objType = tet.pcos_get_string(tetHandle, "type:/Info/Subject");
if (!objType.equals("null")) {
    doc.add(new TextField("summary",
        tet.pcos_get_string(tetHandle, "/Info/Subject"),
        Field.Store.YES));
}
```

- ▶ *font*: Namen aller Fonts im PDF-Dokument

Sie können Metadaten-Felder durch Veränderung der indizierten Dokument-Infofelder oder durch Hinzufügen weiterer Informationen anpassen, die auf den pCOS-Pfaden in *PdfDocument.java* basieren.

**PDF-Dateianhänge.** Der Lucene-Konnektor für TET verarbeitet alle PDF-Dateianhänge in einem Dokument rekursiv und liefert für jeden Anhang Text und Metadaten zur Indizierung an die Lucene-Suchmaschine. Auf diese Weise werden Suchtreffer generiert, auch wenn der gesuchte Text nur in einem Anhang und nicht im Hauptdokument vorkommt. Das rekursive Durchsuchen von Anhängen ist besonders wichtig bei PDF-Paketen und -Portfolios.

## 4.3 TET-Konnektor für den Solr Search Server

Solr ist ein leistungsstarker Open-Source Enterprise-Suchserver auf der Basis der Lucene-Bibliothek. Es bietet eine XML-/HTTP-Schnittstelle und APIs für JSON/Python/Ruby, sowie Hit-Highlighting, Facettierte Suche, Caching, Replikation und ein Web-Interface für die Administration. Solr läuft direkt in einem Java-Servlet-Container (siehe [lucene.apache.org/solr](http://lucene.apache.org/solr)).

Solr implementiert eine zusätzliche Schicht um die Core-Engine von Lucene. Indizierte Daten werden in einem einfachen XML-Format erwartet. Solr-Eingabe kann sehr leicht basierend auf TETML erzeugt werden, dem XML-basierten Ausgabeformat von TET. Der TET-Konnektor für Solr besteht aus einem XSLT-Stylesheet, das TETML in das von Solr benötigte XML-Format konvertiert. TETML-Eingabe für dieses Stylesheet kann mit der TET-Bibliothek oder dem TET-Kommandozeilen-Tool erzeugt werden (siehe Abschnitt 9.1, »Erzeugen von TETML«, Seite 147).

*Hinweis* Geschützte Dokumente können unter bestimmten Bedingungen mit der Option `shrug` indiziert werden (für weitere Informationen siehe Kapitel 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67). Um geschützte Dokumente zu indizieren, müssen Sie diese Option in der TET-Bibliothek oder dem TET-Kommandozeilen-Tool manuell aktivieren, wenn Sie TETML-Eingabe für Solr erzeugen wollen.

**Indizieren von Metadaten-Feldern.** Der TET-Konnektor für Solr indiziert alle standardmäßigen Dokument-Infofelder. Das Schlüsselwort eines Feldes wird dabei als Feldname verwendet.

**PDF-Dateianhänge.** Der TET-Konnektor für Solr verarbeitet alle PDF-Dateianhänge in einem Dokument rekursiv und liefert für jeden Anhang Text und Metadaten zur Indizierung an die Suchmaschine. Auf diese Weise werden Suchtreffer generiert, auch wenn der gesuchte Text nur in einem Anhang und nicht im Hauptdokument vorkommt. Das rekursive Durchsuchen von Anhängen ist besonders wichtig bei PDF-Paketen und -Portfolios.

**XSLT-Stylesheet für die Konvertierung von TETML.** Das Stylesheet `solr.xsl` erwartet TETML-Eingabe in einem beliebigen Modus außer `glyph`. Es erzeugt das erforderliche XML, mit dem die Eingabedaten an den Suchserver übergeben werden können. Dokumentinfo-Einträge werden als Felder übergeben, die den Namen des Info-Eintrags tragen (plus das Suffix `_s` zur Kennzeichnung eines String-Werts) und der Haupttext wird in einer Reihe von Textfeldern übergeben. PDF-Anhänge im Dokument (einschließlich PDF-Paketen und -Portfolios) werden rekursiv verarbeitet:

```
<?xml version="1.0" encoding="UTF-8"?><add>
<doc>
<field name="id">TET-datasheet.pdf</field>
<field name="Author_s">PDFlib GmbH</field>
<field name="CreationDate_s">2015-08-04T23:45:46+02:00</field>
<field name="Creator_s">Adobe InDesign CS6 (Windows)</field>
<field name="ModDate_s">2015-08-04T23:45:46+02:00</field>
<field name="Producer_s">Adobe PDF Library 10.0.1</field>
<field name="Subject_s">PDFlib TET: Text and Image Extraction Toolkit (TET)</field>
<field name="Title_s">PDFlib TET datasheet</field>
<field name="text">PDFlib</field>
<field name="text">datasheet</field>
```

```
<field name="text">PDFlib</field>
<field name="text">TET</field>
<field name="text">5</field>
...
```

## 4.4 TET-Konnektor für Oracle

Der TET-Konnektor für Oracle bindet TET an eine Oracle-Datenbank, so dass PDF-Dokumente indiziert und mit Oracle Text abgefragt werden können. PDF-Dokumente können über ihre Pfadnamen in der Datenbank referenziert oder direkt als BLOBs in der Datenbank abgelegt werden.

**Hinweis** Geschützte Dokumente können unter bestimmten Bedingungen mit der Option `shrug` indiziert werden (für weitere Informationen siehe Kapitel 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67). Dies wird zwar in den Konnektor-Dateien vorbereitet, Sie müssen diese Option jedoch manuell aktivieren.

**Voraussetzungen und Installation.** Der TET-Konnektor wurde mit Oracle 10i und Oracle 11g getestet. Um den TET-Konnektor zu nutzen, müssen Sie den Zeichensatz `AL32UTF8` angeben, wenn Sie die Datenbank erstellen. Dies gilt nur, wenn Sie die Universal Edition von Oracle Express verwenden (bei der Westeuropäischen Edition ist dies nicht nötig). `AL32UTF8` ist der von Oracle empfohlene Datenbank-Zeichensatz und erzielt auch für die Indizierung von PDF-Dokumenten mit TET die besten Ergebnisse. Es ist jedoch auch möglich, TET mit Oracle Text unter Verwendung anderer Zeichensätze gemäß einer der folgenden Methoden zu verbinden:

- ▶ Beginnend mit Oracle Text 11.1.0.7 kann die Datenbank die Konvertierung der erforderlichen Zeichensätze durchführen. Für weitere Informationen siehe den Abschnitt »Using USER\_FILTER with Charset and Format Columns« in der Dokumentation zu Oracle Text 11.1.0.7, die unter folgender Adresse verfügbar ist [docs.oracle.com/cd/B28359\\_01/text.111/b28304/cdatadic.htm#sthref497](http://docs.oracle.com/cd/B28359_01/text.111/b28304/cdatadic.htm#sthref497).
- ▶ Mit Oracle Text 11.1.0.6 oder früheren Versionen muss der vom TET-Filterskript erzeugte UTF-8-Text in die Datenbank-Zeichensätze konvertiert werden. Dazu muss ein Kommando für die Konvertierung der Zeichensätze zu `tetfilter.sh` hinzugefügt werden  
Unix: Rufen Sie `iconv` auf (Open-Source-Software) oder `uconv` (Teil der kostenlosen ICU-Unicode-Bibliothek)  
Windows: Rufen Sie einen geeigneten Codepage-Konverter in `tetfilter.bat` auf.

Um die Vorteile des TET-Konnektors für Oracle nutzen zu können, müssen Sie Oracle das TET-Filterskript wie folgt zur Verfügung stellen:

- ▶ Kopieren Sie das TET-Filterskript in ein für Oracle zugängliches Verzeichnis:  
Unix: Kopieren Sie `connectors/Oracle/tetfilter.sh` nach `$ORACLE_HOME/ctx/bin`  
Windows: Kopieren Sie `connectors/Oracle/tetfilter.bat` nach `%ORACLE_HOME%\bin`
- ▶ Die Variable `TETDIR` im TET-Filterskript (`tetfilter.sh` oder entsprechend `tetfilter.bat`) sollte auf das TET-Installationsverzeichnis zeigen.
- ▶ Falls erforderlich, können Sie weitere globale, Seiten- oder Dokumentoptionen von TET in den Variablen `TETOPT`, `DOCOPT` und `PAGEOPT` übergeben (für Informationen zu den Optionslisten siehe Kapitel 10, »API-Referenz für die TET-Bibliothek«, Seite 173). Dies ist besonders nützlich zur Übergabe der TET-Lizenzschlüssel, z.B.:

```
TETOPT="license=aaaaaa-bbbbbb-ccccc-ddddd-eeeeee"
```

Für weitere Informationen zur Übergabe von TET-Lizenzschlüsseln siehe Abschnitt 0.2, »Eingabe des TET-Lizenzschlüssels«, Seite 8.

**Berechtigungen für den Oracle-Benutzer erteilen.** Die folgenden Beispiele gehen von einem Oracle-Benutzer mit den entsprechenden Berechtigungen zur Erzeugung und Abfrage eines Index aus. Die folgenden Befehle erteilen entsprechende Berechtigungen für den Benutzer *HR* (Diese Befehle müssen als *system* eingegeben und gegebenenfalls angepasst werden):

```
SQL> GRANT CTXAPP TO HR;
SQL> GRANT EXECUTE ON CTX_CLS TO HR;
SQL> GRANT EXECUTE ON CTX_DDL TO HR;
SQL> GRANT EXECUTE ON CTX_DOC TO HR;
SQL> GRANT EXECUTE ON CTX_OUTPUT TO HR;
SQL> GRANT EXECUTE ON CTX_QUERY TO HR;
SQL> GRANT EXECUTE ON CTX_REPORT TO HR;
SQL> GRANT EXECUTE ON CTX_THES TO HR;
```

**Beispiel A: Speichern der Pfadnamen von PDF-Dokumenten in der Datenbank.** In diesem Beispiel werden Dateinamen-Verweise auf die indizierten PDF-Dokumente in der Datenbank gespeichert. Dazu gehen Sie wie folgt vor:

- ▶ Wechseln Sie in der Eingabeaufforderung in das folgende Verzeichnis:

```
<TET-Installationsverzeichnis>/connectors/Oracle
```

- ▶ Passen Sie die Variable *tetpath* im Skript *tetsetup\_a.sql* so an, dass sie auf das TET-Installationsverzeichnis zeigt.
- ▶ Bereiten Sie die Datenbank vor: erzeugen Sie mit dem Oracle-Programm *sqlplus* die Tabelle *pdftable\_a*, füllen Sie diese mit Pfadnamen von PDF-Dokumenten und erzeugen Sie den Index *tetindex\_a* (beachten Sie, dass sich die Inhalte vom Skript *tetsetup\_a.sql* je nach Plattform in der Pfadsyntax unterscheiden):

```
SQL> @tetsetup_a.sql
```

- ▶ Fragen Sie die Datenbank mit folgendem Index ab:

```
SQL> select * from pdftable_a where CONTAINS(pdffile, 'Whitepaper', 1) > 0;
```

- ▶ Aktualisieren Sie den Index (erforderlich, nachdem weitere Dokumente hinzugefügt wurden):

```
SQL> execute ctx_ddl.sync_index('tetindex_a')
```

- ▶ Bereinigen Sie optional die Datenbank (entfernt Index und Tabelle):

```
SQL> @tetcleanup_a.sql
```

**Beispiel B: Speichern von PDF-Dokumenten als BLOBs in der Datenbank und Hinzufügen von Metadaten.** In diesem Beispiel werden die eigentlichen PDF-Dokumente in der Datenbank gespeichert. Zusätzlich zu den PDF-Daten werden Metadaten mit der pCOS-Schnittstelle extrahiert und in speziellen Datenbankspalten gespeichert. Das Java-Programm *tet\_pdf\_loader* speichert die PDF-Dokumente als BLOBs in der Datenbank. Um die Verarbeitung von Metadaten zu zeigen, extrahiert das Programm mit der pCOS-Schnittstelle den Dokumenttitel (über den pCOS-Pfad */Info/Title*) und die Anzahl der Seiten im Dokument (über den pCOS-Pfad *length:pages*). Titel und Seitenzahl werden in getrennten Spalten in der Datenbank gespeichert. Gehen Sie zur Ausführung dieses Beispiels wie folgt vor:

- ▶ Wechseln Sie in der Eingabeaufforderung in das folgende Verzeichnis:

<TET installation directory>/connectors/Oracle

- ▶ Bereiten Sie die Datenbank vor: erzeugen Sie mit dem Oracle-Programm *sqlplus* die Tabelle *pdftable\_b* und den Index *tetindex\_b*:

```
SQL> @tetsetup_b.sql
```

- ▶ Füllen Sie die Datenbank: füllen Sie die Tabelle mit PDF-Dokumenten und Metadaten über JDBC (beachten Sie, dass dies nicht mit Stored Procedures möglich ist). Die im TET-Paket enthaltene Ant-Build-Datei erwartet die Datei *ojdbc14.jar* für den Oracle-Treiber JDBC im selben Verzeichnis wie den Sourcecode *tet\_pdf\_loader.java*. Geben Sie mit dem Kommando *ant* einen geeigneten JDBC-Konnektor-String an. Die Build-Datei enthält eine Beschreibung aller Eigenschaften, mit denen Optionen für den Ant-Build angegeben werden können. Sie können Werte für diese Optionen über die Kommandozeile eingeben. Im folgenden Beispiel wird *localhost* als Host-Name, Portnummer 1521, *xe* als Datenbankname und *HR* als Benutzername und -kennwort verwendet (passen Sie die Werte entsprechend Ihrer Datenbank-Konfiguration an):

```
ant -Dtet.jdbc.connection=jdbc:oracle:thin:@localhost:1521:xe ←  
-Dtet.jdbc.user=HR -Dtet.jdbc.password=HR
```

- ▶ Aktualisieren Sie den Index (erforderlich zu Beginn und nachdem weitere Dokumente hinzugefügt wurden):

```
SQL> execute ctx_ddl.sync_index('tetindex_b')
```

- ▶ Fragen Sie die Datenbank mit folgendem Index ab:

```
SQL> select * from pdftable_b where CONTAINS(pdffile, 'Whitepaper', 1) > 0;
```

- ▶ Bereinigen Sie optional die Datenbank (entfernt Index und Tabelle):

```
SQL> @tetcleanup_b.sql
```

## 4.5 TET PDF IFilter für Produkte von Microsoft

Dieser Abschnitt beschreibt TET PDF IFilter, ein auf PDFlib TET basierendes, separates Produkt. Weitere Informationen und Distributionspakete für TET PDF IFilter finden Sie unter [www.pdfli.com/products/tet-pdf-ifilter](http://www.pdfli.com/products/tet-pdf-ifilter).

TET PDF IFilter ist für den nicht-kommerziellen Einsatz auf Desktop-Systemen kostenlos verfügbar; für den kommerziellen Einsatz auf Desktop-Systemen benötigen Sie eine kommerzielle Lizenz.

**Was bietet PDFlib TET PDF IFilter?** TET PDF IFilter extrahiert Text und Metadaten aus PDF-Dokumenten, um sie Retrieval-Produkten unter Windows zugänglich zu machen. Damit haben Sie die Möglichkeit, die PDF-Dokumente auf Ihrem Desktop-Computer, dem Enterprise-Server oder im Web zu durchsuchen. TET PDF IFilter basiert auf dem patentierten Entwicklungswerkzeug PDFlib Text Extraction Toolkit (TET), mit dem sich Text zuverlässig aus PDF-Dokumenten extrahieren lässt.

TET PDF IFilter ist eine stabile Implementierung der Microsoft IFilter-Schnittstelle zur Volltextindizierung und arbeitet mit allen Produkten zur Textabfrage zusammen, die die IFilter-Schnittstelle unterstützen, z.B. SharePoint oder SQL Server. Diese Produkte verwenden für jedes Dateiformat, z.B. HTML, ein anderes formatspezifisches Filterprogramm, das IFilter genannt wird. TET PDF IFilter ist ein solches Filterprogramm für PDF-Dokumente. Die Benutzerschnittstelle zum Durchsuchen der Dokumente kann Windows Explorer, ein Web- oder Datenbank-Frontend, ein Abfrageskript oder eine selbst entwickelte Anwendung sein. Alternativ zur interaktiven Suche über die Benutzeroberfläche lassen sich Anfragen auch über eine Programmierschnittstelle absetzen.

**Besondere Vorteile.** TET PDF IFilter bietet folgende Vorteile:

- ▶ Unterstützt westlichen, chinesischen, japanischen und koreanischen (CJK-)Text, sowie von rechts nach links laufende Sprachen wie Arabisch und Hebräisch:
- ▶ Indiziert auch geschützte Dokumente und extrahiert Text sogar aus PDFs, bei denen Acrobat scheitert
- ▶ Unterstützt Unicode-Nachbearbeitung durch Folding, Dekomposition und Normalisierung
- ▶ Leistung: thread-sicher, schnell und stabil, 32- und 64-Bit-Varianten
- ▶ Automatische Erkennung von Sprache und Schriftsystem für verbesserte Suche

**Unternehmensweite Suche in PDF-Dokumenten.** TET PDF IFilter ist in thread-sicheren nativen 32- und 64-Bit-Versionen verfügbar. Unternehmensweite Lösungen zur Textsuche lassen sich in Kombination mit folgenden Produkten implementieren:

- ▶ Microsoft SharePoint Server und FAST Server für SharePoint
- ▶ Microsoft Search Server
- ▶ Microsoft SQL Server
- ▶ Microsoft Exchange Server
- ▶ Microsoft Site Server

TET PDF IFilter ist mit allen Produkten von Microsoft und anderen Anbietern einsetzbar, die die IFilter-Schnittstelle unterstützen.

**Desktop-Suche in PDF-Dokumenten.** TET PDF IFilter lässt sich auch zur Suche nach PDF-Dokumenten auf dem Desktop-Computer einsetzen, zum Beispiel mit Windows Search, das in Windows Vista/7/8/10 integriert ist.

TET PDF IFilter ist für den nicht-kommerziellen Einsatz auf Desktop-Systemen kostenlos verfügbar und bietet damit eine bequeme Basis zum Testen und Evaluieren.

**Akzeptierte PDF-Eingabe.** TET PDF IFilter verarbeitet alle gängigen Varianten von PDF-Dokumenten:

- ▶ Alle PDF-Versionen bis Acrobat DC einschließlich ISO 32000-1 und ISO 32000-2
- ▶ Geschützte PDFs, die zum Öffnen kein Kennwort benötigen
- ▶ Beschädigte PDF-Dokumente werden repariert

**Unicode-Nachbearbeitung.** TET PDF IFilter unterstützt Nachbearbeitung zur Verbesserung der Suchergebnisse:

- ▶ Foldings erhalten, entfernen oder ersetzen Zeichen, um z.B. Interpunktionszeichen oder Zeichen aus einem irrelevanten Schriftsystem zu filtern.
- ▶ Dekomposition (Zerlegung) ersetzt ein Zeichen durch eine äquivalente Folge von einem oder mehreren anderen Zeichen, z.B. beim Ersetzen eines chinesischen Zeichen durch das kanonisch äquivalente Unicode-Zeichen.
- ▶ Text kann in alle vier Unicode-Normalformen konvertiert werden, z.B. um Texte in NFC-Form auszugeben, damit sie den Anforderungen einer Datenbank entsprechen.

**Internationalisierung.** Neben westlichem Text unterstützt TET PDF IFilter chinesischen, japanischen und koreanischen (CJK-)Text. Alle CJK-Kodierungen werden erkannt; horizontale und vertikale Schreibrichtung werden korrekt behandelt. Die automatische Erkennung der ID für die Spracheinstellung des Textes (Sprach- und Regionskürzel, Locale ID) verbessert die Ergebnisse von Microsofts Algorithmen zur Bestimmung von Wortgrenzen und Wortstämmen, was insbesondere bei ostasiatischem Text wichtig ist.

Linkslaufende Schriften wie Hebräisch und Arabisch werden auch unterstützt. Dabei normalisiert TET PDF IFilter kontextabhängige Zeichenformen und sortiert den Text in logische Reihenfolge um.

**PDF enthält mehr als nur Seiten.** TET PDF IFilter behandelt PDF-Dokumente als Container für weit mehr als nur die Seiteninhalte und indiziert alle relevanten Elemente eines PDF-Dokuments:

- ▶ Seiteninhalte
- ▶ Text in Lesezeichen
- ▶ Metadaten (siehe unten)
- ▶ Eingebettete PDFs und PDF-Pakete/-Portfolios werden rekursiv verarbeitet, so dass sich auch Text in PDF-Dateianhängen durchsuchen lässt.

**XMP-Metadaten und Dokument-Infelder.** Die leistungsfähige Metadaten-Implementierung von TET PDF IFilter unterstützt das Property-System von Windows für Metadaten. TET PDF IFilter indiziert XMP-Metadaten sowie Standard- und benutzerdefinierte Dokument-Infelder. Die Indizierung der Metadaten lässt sich auf verschiedenen Ebenen konfigurieren:

- ▶ Dokument-Infelder, Dublin-Core-Felder und andere gängige XMP-Properties werden auf entsprechende Windows-Properties wie *Title*, *Subject*, *Author* abgebildet.

- ▶ TET PDF IFilter ergänzt nützliche PDF-spezifische Pseudo-Properties wie Seitengröße, PDF/A-Konformitätslevel oder Fontnamen.
- ▶ Nach allen relevanten vordefinierten XMP-Properties kann gesucht werden.
- ▶ Die Suche umfasst auch benutzerdefinierte XMP-Properties wie firmenspezifische Klassifizierungen oder PDF/A-Extension-Schemas.

TET PDF IFilter bietet optional die Möglichkeit, Metadaten in den indizierten Rohtext zu integrieren. Damit können auch Volltextsuchmaschinen ohne Metadaten-Unterstützung (z.B. SQL Server) nach Metadaten suchen.

## 4.6 TET-Konnektor für das Apache Tika-Toolkit

Tika ist ein »Open-Source-Toolkit zur Erkennung und Extraktion von Metadaten und strukturiertem Text aus einer Reihe von Dokumentformaten mit Hilfe bestehender Parser-Bibliotheken«. Für weitere Informationen zu Tika siehe [tika.apache.org](http://tika.apache.org). Der TET-Konnektor für Tika ersetzt den in Tika konfigurierten Standard-PDF-Parser für das PDF-Format durch TET. Der TET-Konnektor übergibt folgende Elemente an Tika:

- den unformatierten Text aller Seiten
- vor- und benutzerdefinierte Dokument-Infofelder
- Anzahl der Seiten im Dokument

*Hinweis* Geschützte Dokumente können unter bestimmten Bedingungen mit der Option `shrug` indiziert werden (für weitere Informationen siehe Kapitel 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67). Dies wird zwar in den Konnektor-Dateien vorbereitet, Sie müssen diese Option jedoch manuell aktivieren. `TETPDFParser.java` bietet außerdem eine Methode zur Übergabe eines Kennworts, falls die Option `shrug` nicht ausreicht.

**Voraussetzungen und Installation.** Die TET-Distribution enthält einen TET-Konnektor für das Tika-Toolkit. In der folgenden Beschreibung steht `<tet-dir>` für das Verzeichnis, in dem das TET-Paket entpackt wurde. Folgende Voraussetzungen müssen erfüllt sein:

- JDK 1.5 oder höher
- Eine funktionierende Installation des Build-Tools *Ant*
- Ein installiertes TET-Paket für Unix, Linux, OS X oder Windows.
- Eine vorgefertigte JAR-Datei für Tika namens `tika-app-1.x.jar`. Informationen zum Download dieser Datei finden Sie unter folgender Adresse:

[tika.apache.org/download.html](http://tika.apache.org/download.html)

Im Allgemeinen kann Tika 1.8 oder höher verwendet werden. Tika 1.9 hat jedoch einen Fehler, der das Überschreiben des integrierten PDF-Parsers verhindert. Der TET-Konnektor kann daher nur mit Tika 1.9 verwendet werden, wenn einige Anpassungen am Tika Quellcode vorgenommen werden oder wenn ein Mechanismus wie die XML-Konfigurationsdatei von Tika verwendet wird.

**Erstellen und Testen des TET-Konnektors für Tika.** Zum Erstellen und Testen des TET-Konnektors für Tika gehen Sie folgendermaßen vor:

- Kopieren Sie `tika-app-1.x.jar` ins Verzeichnis `<tet-dir>/connectors/Tika`.
- Wechseln Sie zu `<tet-dir>/connectors/Tika` und erstellen Sie den TET-Konnektor für Tika:

```
ant
```

Wenn Ihre Tika-JAR-Datei einen anderen Namen als `tika-app-1.x.jar` hat, müssen Sie den Namen der JAR-Datei in einer Kommandozeile übergeben:

```
ant -Dtika-app.jar=tika-app-1.x.jar
```

- Die Build-Datei enthält ein Target zum Testen des TET-Konnektors für Tika:

```
ant test
```

Dieses Kommando erzeugt die Inhalte des Testdokuments als XHTML auf der Standard-Ausgabe. Für den Test mit einer PDF-Datei geben Sie die Ant-Property `test.inputfile` über die Kommandozeile wie folgt ein:

```
ant -Dtest.inputfile=/path/to/your/file.pdf test
```

Die erfolgreiche Übergabe eines Kennworts für ein geschütztes Dokument können Sie folgendermaßen testen:

```
ant -Dtest.inputfile=<protected file.pdf> -Dtest.outputfile=<output file name> ←  
-Dtest.password=<password> api-test
```

- ▶ Um zu prüfen, ob der TET-Konnektor für Tika für den MIME-Typ *application/pdf* tatsächlich verwendet wird, führen Sie unter Unix und OS X den folgenden Befehl im Verzeichnis *<tet-dir>/connectors/Tika* aus:

```
java -Djava.library.path=<tet-dir>/bind/java -classpath ←  
<tet-dir>/bind/java/TET.jar:tika-app-1.x.jar:tet-tika.jar ←  
org.apache.tika.cli.TikaCLI --list-parser-details
```

Unter Windows:

```
java -Djava.library.path=<tet-dir>/bind/java -classpath ←  
<tet-dir>/bind/java/TET.jar;tika-app-1.x.jar;tet-tika.jar ←  
org.apache.tika.cli.TikaCLI --list-parser-details
```

In der Ausgabe sollte folgendes Fragment erscheinen:

```
com.pdflib.tet.tika.TETPDFParser  
application/pdf
```

- ▶ Um die Tika-Benutzeroberfläche mit dem TET-Konnektor zu aktivieren, führen Sie folgenden Befehl aus: *<tet-dir>/connectors/Tika*:

Unix und OS X:

```
java -Djava.library.path=<tet-dir>/bind/java -classpath ←  
<tet-dir>/bind/java/TET.jar:tika-app-1.x.jar:tet-tika.jar ←  
org.apache.tika.cli.TikaCLI
```

Unter Windows:

```
java -Djava.library.path=<tet-dir>\bind\java -classpath ←  
<tet-dir>\bind\java\TET.jar;tika-app-1.x.jar;tet-tika.jar ←  
org.apache.tika.cli.TikaCLI
```

**Anpassen des TET-Konnektors für Tika.** Im Source-Modul *TETPDFParser.java* können Sie den Tika-Konnektor wie folgt anpassen:

- ▶ Fügen Sie der Variable *DOC\_OPT\_LIST* Dokumentoptionen hinzu, z.B. die Option *shrug* zur Verarbeitung geschützter Dokumente;
- ▶ Fügen Sie der Variable *PAGE\_OPT\_LIST* Seitenoptionen hinzu;
- ▶ Passen Sie den Suchpfad für Ressourcen wie CJK-CMaps in der Variable *SEARCHPATH* an. Alternativ können Sie zur Verarbeitung von PDF-Dokumenten die Property *tet.searchpath* übergeben.

## 4.7 TET-Konnektor für MediaWiki

MediaWiki ist eine frei verfügbare Wiki-Software zum Betrieb von Wikipedia und anderer Community-Websites. Für weitere Informationen zu MediaWiki siehe [www.mediawiki.org/wiki/MediaWiki](http://www.mediawiki.org/wiki/MediaWiki).

*Hinweis* Geschützte Dokumente können unter bestimmten Bedingungen mit der Option `shrug` indiziert werden (für weitere Informationen siehe Kapitel 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67). Dies wird zwar in den Konnektor-Dateien vorbereitet, Sie müssen diese Option jedoch manuell aktivieren.

**Voraussetzungen und Installation.** Die TET-Distribution enthält einen TET-Konnektor zur Indizierung von PDF-Dokumenten, die auf eine MediaWiki-Website hochgeladen werden. MediaWiki unterstützt intern keine PDF-Dokumente, erlaubt aber das Hochladen von PDF-Dokumenten als »Bilder«. Der TET-Konnektor für MediaWiki indiziert alle PDF-Dokumente beim Hochladen. Im MediaWiki bereits vorhandene PDF-Dokumente werden nicht indiziert. Folgende Voraussetzungen müssen erfüllt sein:

- ▶ MediaWiki 22 oder höher
- ▶ Ein installiertes TET-Paket mit der TET-Sprachbindung für PHP unter Unix, Linux, OS X oder Windows.

Um den TET-Konnektor für MediaWiki zu implementieren, führen Sie folgende Schritte durch:

- ▶ Installieren Sie die TET-Sprachbindung für PHP wie in Abschnitt 3.9, »PHP-Sprachbindung«, Seite 41 beschrieben.
- ▶ Kopieren Sie `<TET install dir>/connectors/MediaWiki/PDFIndexer.php` nach `<MediaWiki install dir>/extensions/PDFIndexer/PDFIndexer.php`.
- ▶ Wenn Sie Unterstützung für CJK-Text benötigen, kopieren Sie die CMap-Dateien von `<TET install dir>/resource/cmap` nach `<MediaWiki install dir>/extensions/PDFIndexer/resource/cmap`.
- ▶ Fügen Sie der Konfigurationsdatei des MediaWiki `LocalSettings.php` folgende Zeilen hinzu:

```
# Indizieren hochgeladener PDF-Dateien, um sie durchsuchbar zu machen
include("extensions/PDFIndexer/PDFIndexer.php");
```

- ▶ Um Warnungen beim Hochladen von PDF-Dokumenten zu vermeiden, empfehlen wir, der Datei `<MediaWiki install dir>/includes/DefaultSettings.php` folgende Zeilen hinzuzufügen, um den Dateityp und die Extension `.pdf` bekannt zu machen:

```
/**
 * Dies ist die Liste der bevorzugten Extensionen für hochgeladene Dateien. *
 * Hochgeladene Dateien, deren Extension in der Liste fehlt,
 * lösen eine Warnung aus.
 */
$wgFileExtensions = array( 'png', 'gif', 'jpg', 'jpeg', 'pdf' );
```

**Wie funktioniert der TET-Konnektor für MediaWiki?** Der TET-Konnektor für MediaWiki besteht aus dem PHP-Modul `PDFIndexer.php`. Mit einem der in MediaWiki vordefinierten Hooks wird das PHP-Modul ausgeführt, sobald ein neues PDF-Dokument hochgeladen wird. Text und Metadaten werden aus dem PDF-Dokument extrahiert und an den optionalen Benutzerkommentar für das hochgeladene Dokument angehängt. Der

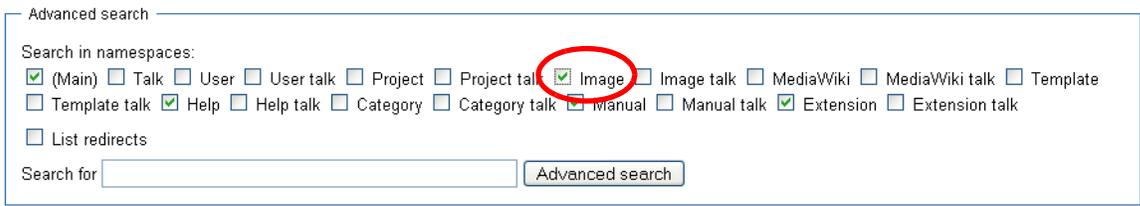


Abb. 4.1 Suche nach PDF-Dokumenten in MediaWiki

Text wird in einem HTML-Kommentar versteckt und ist für Benutzer nicht sichtbar, wenn sie sich den Kommentar zum Dokument ansehen. Da MediaWiki den kompletten Inhalt des Kommentars einschließlich des versteckten Volltexts indiziert, werden die Textinhalte des PDF-Dokuments ebenfalls indiziert. Der Text für den Index ist wie folgt aufgebaut:

- ▶ Der TET-Konnektor speist den Wert aller Dokument-Infofelder in den Index ein.
- ▶ Die Textinhalte aller Seiten werden extrahiert und verkettet.
- ▶ Bleibt die Größe des extrahierten Textes unter einem bestimmten Grenzwert, wird er dem Index komplett zugeführt. Der Vorteil hierbei ist, dass Suchergebnisse den Suchbegriff in seinem Kontext anzeigen.
- ▶ Wenn die Größe des extrahierten Textes einen bestimmten Grenzwert überschreitet, wird der Text auf eindeutige Wörter reduziert (d.h. mehrere Instanzen des gleichen Wortes werden auf eine einzelne Instanz des Wortes reduziert).
- ▶ Bleibt die Größe des reduzierten Textes unter einem bestimmten Grenzwert, wird er dem Index zugeführt. Andernfalls wird er abgeschnitten, d.h. Text zum Ende des Dokuments wird nicht mehr indiziert.

Der vordefinierte Grenzwert ist 512 KB, er kann jedoch in *PDFIndexer.php* angepasst werden. Wenn einer der oben beschriebenen Größentests den Grenzwert erreicht, wird eine Warnung in *DebugLogFile* von MediaWiki ausgegeben, sofern die Protokollierung in MediaWiki aktiviert ist.

**Suche nach PDF-Dokumenten.** Da PDF-Dokumente von MediaWiki als Bilder behandelt werden, müssen Sie nach Ihnen im Namensraum *image* suchen. Aktivieren Sie hierzu im Dialog *Advanced search* das Kontrollkästchen *Image* in der Liste der Namensräume (siehe Abbildung 4.1). Der Namensraum *Image* wird standardmäßig nicht durchsucht. Dies kann jedoch in den Voreinstellungen unter *LocalSettings.php* wie folgt aktiviert werden:

```
$wgNamespacesToBeSearchedDefault = array(
    NS_MAIN          => true,
    NS_IMAGE         => true,
)
```

Als Suchergebnis wird eine Liste der Dokumente angezeigt, die den Suchbegriff enthalten. Wenn der gesamte Text indiziert wurde (im Gegensatz zu der verkürzten Wortliste für lange Dokumente), werden einige zusätzliche Begriffe vor und hinter dem Suchbegriff angezeigt, um einen Kontext zu schaffen. Da die PDF-Textinhalte im HTML-Format in den MediaWiki-Index eingespeist werden, werden vor dem Text Zeilennummern angezeigt. Sie sind für PDF-Dokumente irrelevant und können ignoriert werden.

**Indizieren von Metadaten-Feldern.** Der TET-Konnektor für MediaWiki indiziert alle standardmäßigen Dokument-Infofelder. Der Wert jedes Feldes wird dem Index zugeführt, damit er bei der Suche verwendet werden kann. Da MediaWiki die Suche nach Metadaten nicht unterstützt, können Sie nicht direkt nach Dokument-Infofeldern suchen, sondern nur nach Info-Einträgen als Teil des Volltextes.

# 5 Konfiguration

## 5.1 Extrahieren von Inhalten aus geschützten PDF-Dokumenten

**Sicherheitsfunktionen von PDF.** PDF-Dokumente können mit Kennwortschutz versehen werden, welcher folgende Sicherheitsfunktionen bietet:

- ▶ Das Benutzerkennwort (auch Kennwort zum Öffnen des Dokuments) ist zum Öffnen der Datei erforderlich.
- ▶ Das Master-Kennwort (auch Berechtigungskennwort) ist zum Ändern von Sicherheitseinstellungen wie Berechtigungen, Benutzer- oder Master-Kennwort erforderlich. Dateien mit Benutzer- und Master-Kennwort können mit einem von beiden Kennwörtern geöffnet werden.
- ▶ Berechtigungen beschränken die mit dem PDF-Dokument erlaubten Aktionen, wie zum Beispiel das Drucken oder das Extrahieren von Text.
- ▶ Dateianhänge können separat verschlüsselt werden, ohne dass das Dokument selbst verschlüsselt ist.

Verwendet ein PDF-Dokument eine dieser Sicherheitsfunktionen, wird es verschlüsselt. Zum Anzeigen oder Ändern der Sicherheitseinstellungen eines Dokuments klicken Sie in Acrobat auf *Datei, Eigenschaften, Sicherheit, Details anzeigen* bzw. *Einstellungen ändern*.

TET berücksichtigt die Sicherheitseinstellungen eines PDF-Dokuments. Der Kennwort- und Berechtigungsstatus lässt sich mit den pCOS-Pfaden *encrypt/master, encrypt/user, encrypt/nocopy* usw. abfragen, wie im *dumper*-Beispiel gezeigt. Mit dem pCOS-Pseudo-Objekt *pcosmode* lassen sich auch die Aktionen festlegen, die für ein bestimmtes Dokument erlaubt sind.

**Status der Extraktion von Inhalten.** TET erlaubt die Text- und Bildextraktion, sofern sich das Dokument öffnen lässt (dies gilt nicht mehr, wenn die Option *requiredmode* von *TET\_open\_document()* übergeben wurde). Abhängig von den in *nocopy* eingestellten Berechtigungen kann die Extraktion von Inhalten im eingeschränkten pCOS-Modus erlaubt oder nicht erlaubt sein (im vollständigen pCOS-Modus ist sie immer erlaubt). Mit folgender Bedingung lässt sich prüfen, ob sie erlaubt ist:

```
if ((int) tet.pcos_get_number(doc, "encrypt/nocopy") == 0)
{
    /* Inhaltsextraktion erlaubt */
}
```

**Gründe für die Verarbeitung geschützter Dokumente.** PDF-Berechtigungen helfen Autoren, ihre Rechte als Urheber von Inhalten durchzusetzen und Benutzer von PDF-Dokumenten müssen bei der Extraktion von Text- oder Bildinhalten diese Urheberrechte einhalten. TET arbeitet bei solchen geschützten Dokumenten standardmäßig im eingeschränkten Modus, d.h. Inhalte lassen sich nicht extrahieren. Jedoch bedeutet das Extrahieren von Inhalten nicht in allen Fällen automatisch eine Verletzung der Urheberrechte. In folgenden Fällen ist die Extraktion von Inhalten durchaus zu vertreten:

- ▶ Es werden nur geringfügige Mengen für Zitate extrahiert (»fair use«).

- ▶ Unternehmen möchten ein- und ausgehende Dokumente nach bestimmten Stichwörtern durchsuchen, ohne die Inhalte weiterzuverwenden (*document screening*).
- ▶ Der Autor des Dokuments hat das Master-Kennwort vergessen.
- ▶ Suchmaschinen indizieren geschützte Dokumente, ohne den Benutzern die Dokumentinhalte direkt verfügbar zu machen (nur indirekt durch einen Link auf das Original-PDF).

Das letzte Beispiel ist besonders wichtig: Auch wenn Benutzer die Inhalte eines geschützten PDF-Dokuments nicht extrahieren dürfen, sollten sie in der Lage sein, das Dokument in einer unternehmensweiten oder Web-basierten Suche zu lokalisieren. Es kann vertretbar sein, Inhalte zu extrahieren, wenn diese für Benutzer nicht direkt verfügbar sind, sondern nur dem Index einer Suchmaschine zugeführt werden, damit ein Dokument gefunden werden kann. Da die Benutzer nur Zugriff auf das geschützte Original-PDF erhalten (nachdem die Suchmaschine die Inhalte indiziert und in der Trefferliste einen Link zum PDF ausgegeben hat), schützen die internen PDF-Berechtigungen das Dokument wie gewohnt, wenn ein Benutzer es abrufen.

**Das Feature »shrug« für geschützte Dokumente.** Mit dem Feature *shrug* lassen sich in TET Text und Bilder aus geschützten Dokumenten extrahieren, vorausgesetzt der TET-Benutzer übernimmt die Verantwortung für die Beachtung der Urheberrechte des Dokuments. Das Feature *shrug* funktioniert folgendermaßen: mit der Übergabe der Option *shrug* an `TET_open_document()` versichert der Benutzer, dass er oder sie die Urheberrechte des Dokuments nicht verletzt. Die Allgemeinen Geschäftsbedingungen von PDFlib GmbH verlangen dies.

Wenn alle folgenden Voraussetzungen erfüllt sind, wird *shrug* aktiviert:

- ▶ Die Option *shrug* wurde an `TET_open_document()` übergeben.
- ▶ Das Dokument verlangt ein Master-Kennwort, aber dieses wurde nicht an `TET_open_document()` übergeben.
- ▶ Verlangt das Dokument ein Benutzer-Kennwort zum Öffnen, muss dieses an `TET_open_document()` übergeben worden sein.
- ▶ Die Berechtigungen des Dokuments erlauben keine Textextraktion, d.h. *nocopy=true*.

Das Feature *shrug* hat folgende Wirkung:

- ▶ Die Extraktion von Inhalten aus dem Dokument ist erlaubt, auch wenn *nocopy=true* ist. Der Benutzer übernimmt die Verantwortung für die Beachtung der Urheberrechte des Dokuments.
- ▶ Das pCOS-Pseudo-Objekt *shrug* wird auf *true/1* gesetzt.
- ▶ pCOS läuft im Vollmodus (statt im eingeschränkten Modus), das heißt, das Pseudo-Objekt *pcosmode* wird auf 2 gesetzt.

Das Pseudo-Objekt *shrug* kann gemäß folgendem Schema verwendet werden, um festzustellen, ob der Inhalt dem Benutzer direkt zur Verfügung gestellt werden kann oder nur für die Indizierung und ähnliche indirekte Zwecke verwendet werden darf:

```
int doc = tet.open_document(filename, "shrug");
...
if ((int) tet.pcos_get_number(doc, "shrug") == 1)
{
    /* Nur Indizierung erlaubt */
}
else
```

```
{  
  /* Inhalt kann an Benutzer übergeben werden */  
}
```

## 5.2 Ressourcenkonfiguration und Dateisuche

**UPR-Dateien und Ressourcenkategorien.** In manchen Fällen benötigt TET Zugriff auf Ressourcen wie Encoding-Definitionen oder Zuordnungstabellen für Glyphnamen. Um die Verarbeitung von Ressourcen plattformunabhängig und benutzerdefinierbar zu machen, kann eine Konfigurationsdatei angegeben werden, in der die verfügbaren Ressourcen gemeinsam mit ihren Dateinamen aufgeführt sind. Außerdem kann die Konfiguration dynamisch zur Laufzeit durch Hinzufügen von Ressourcen mit `TET_set_option()` erfolgen. Für die Konfigurationsdatei wird ein einfaches Textformat namens *Unix PostScript Resource* (UPR) verwendet. Das von TET verwendete UPR-Dateiformat wird unten beschrieben. Tabelle 5.1 zeigt alle von TET unterstützten Ressourcenkategorien.

Tabelle 5.1 Ressourcenkategorien (alle Dateinamen müssen in UTF-8 vorliegen)

Kategorie	Format <sup>1</sup>	Beschreibung
<code>cmap</code>	<code>key=value</code>	Ressourcenname und Dateiname einer CMap
<code>codelist</code>	<code>key=value</code>	Ressourcenname und Dateiname einer Codelist
<code>encoding</code>	<code>key=value</code>	Ressourcenname und Dateiname eines Encodings
<code>glyphlist</code>	<code>key=value</code>	Ressourcenname und Dateiname einer Glyphliste
<code>glyphmapping</code>	<code>option list</code>	Optionsliste mit einer Methode zur Glyphenzuordnung gemäß Tabelle 10.9, Seite 196. Diese Ressource wird in <code>TET_open_document()</code> ausgewertet und das Ergebnis nach den in der Option <code>glyphmapping</code> von <code>TET_open_document()</code> angegebenen Zuordnungen aufgelistet.
<code>hostfont</code>	<code>key=value</code>	Name einer im System installierten Font-Ressource, die für einen nicht eingebetteten Font verwendet wird (key ist der PDF-Fontname; value ist der in UTF-8 kodierte Systemfontname)
<code>fontoutline</code>	<code>key=value</code>	Font- und Dateiname eines TrueType- oder OpenType-Fonts, der für einen nicht eingebetteten Font verwendet wird
<code>searchpath</code>	<code>value</code>	Relativer oder absoluter Pfadname von Verzeichnissen mit Datenfiles

1. Während in der UPR-Syntax ein Gleichheitszeichen '=' zwischen Name und Wert erwartet wird, ist dieses Zeichen bei der Angabe von Ressourcen mit `TET_set_option()` weder erforderlich noch erlaubt.

**Das UPR-Dateiformat.** UPR-Dateien liegen im Textformat vor und sind sehr einfach aufgebaut, so dass sie problemlos manuell im Texteditor oder auch automatisch erstellt werden können. Beginnen wir mit der Syntax:

- ▶ Eine Zeile besteht aus maximal 255 Zeichen.
- ▶ Ein Gegenschrägstrich '\ ' am Zeilenende bewirkt, dass die Zeile auch nach dem Newline-Zeichen fortgeführt wird. Dies kann zur Zeilenverlängerung verwendet werden.
- ▶ Ein einzelner Punkt ' .' dient als Abschnittsende.
- ▶ Kommentare beginnen mit einem Prozentzeichen '%' und enden am Zeilenende.
- ▶ Leer- und Tabulatorzeichen werden ignoriert, außer in Ressourcen- und Dateinamen.

UPR-Dateien bestehen aus folgenden Komponenten:

- ▶ Eine Kopfzeile zur Identifizierung der Datei, die folgendermaßen aussieht:

```
PS-Resources-1.0
```

- ▶ Ein Abschnitt, der alle Ressourcenkategorien auflistet, die in der Datei beschrieben werden. Jede Zeile beschreibt eine Kategorie. Diese Liste wird durch eine Zeile mit einem einzelnen Punkt abgeschlossen.
- ▶ Ein Abschnitt für jede der Ressourcenkategorien, die am Dateianfang aufgeführt wurden. Jeder Abschnitt beginnt mit einer Zeile für die Ressourcenkategorie, gefolgt von einer beliebigen Anzahl von Zeilen, die die verfügbaren Ressourcen beschreiben. Diese Liste wird durch eine Zeile mit einem einzelnen Punkt abgeschlossen. Jede Ressourcenzeile besteht aus dem Namen der Ressource (bei Gleichheitszeichen sind Anführungszeichen erforderlich). Erfordert die Ressource einen Dateinamen, muss dieser nach einem Gleichheitszeichen angefügt werden. TET berücksichtigt die Option *searchpath* (siehe unten) bei der Suche nach Dateien, deren Namen als Ressourcen eingetragen sind.

**UPR-Beispieldatei.** Das folgende Listing zeigt ein Beispiel für eine UPR-Konfigurationsdatei:

```
PS-Resources-1.0
searchpath
glyphlist
codelist
encoding
.
searchpath
/usr/local/lib/cmmaps
/users/kurt/myfonts
.
glyphlist
myglyphlist=/usr/lib/sample.gl
.
codelist
mycodelist=/usr/lib/sample.cl
.
encoding
myencoding=sample.enc
.
```

**Dateisuche und Ressourcenkategorie *searchpath*.** Neben relativen und absoluten Pfadnamen können Sie Dateinamen auch ohne jede Pfadangabe an TET übergeben. Dazu definieren Sie mit der Ressourcenkategorie *searchpath* eine Liste von Pfadnamen für die Verzeichnisse, die die benötigten Dateien enthalten. Beim Öffnen einer Datei versucht TET zuerst, diese mit genau dem übergebenen Dateinamen zu öffnen. Schlägt dieser Versuch fehl, sucht TET nacheinander in den Verzeichnissen, die in der Ressourcenkategorie *searchpath* aufgeführt sind. Mehrere *searchpath*-Einträge können zusammen zu einer Liste aufgebaut werden, die in umgekehrter Reihenfolge durchsucht wird (später hinzugefügte Pfade werden also zuerst durchsucht). Um diesen Suchmechanismus zu unterbinden, geben Sie in den jeweiligen TET-Funktionsaufrufen vollständige Pfadnamen an.

Unter Windows initialisiert TET die Ressourcenkategorie *searchpath* mit dem Inhalt der folgenden Registry-Einträge:

```
HKLM\SOFTWARE\PDFlib\TET5\5.0\SearchPath
HKLM\SOFTWARE\PDFlib\TET5\SearchPath
HKLM\SOFTWARE\PDFlib\SearchPath
```

Diese Registry-Einträge können eine Liste von Pfadnamen enthalten, die durch Semikolon ';' getrennt sind. Die Windows-Installationsroutine initialisiert den Registry-Eintrag `SearchPath` mit dem Namen des `resource`-Verzeichnisses im TET-Standardinstallationsverzeichnis.

*Hinweis* Seien Sie vorsichtig beim manuellen Zugriff auf die Registry von 64-Bit-Windows-Systemen: wie üblich arbeiten 64-Bit-Programme mit der 64-Bit-Ansicht der Windows-Registry, während 32-Bit-Programme auf einem 64-Bit-System mit der 32-Bit-Ansicht der Registry arbeiten. Wenn Sie Registry-Werte für ein 32-Bit-Produkt manuell eintragen müssen, achten Sie darauf, die 32-Bit-Version des Werkzeugs `regedit` zu verwenden. Sie können es folgendermaßen mittels Start, Ausführen... aufrufen:

```
%systemroot%\syswow64\regedit
```

**Standard-Dateisuchpfade.** Unter Unix, Linux, OS X und i5/iSeries werden per Voreinstellung einige Verzeichnisse standardmäßig sogar ohne Angabe von Pfad und Verzeichnisnamen nach Dateien durchsucht. Bevor die UPR-Datei (welche zusätzliche Suchpfade enthalten kann) durchsucht und gelesen wird, werden folgende Verzeichnisse durchsucht:

```
<rootpath>/PDFlib/TET/5.0/resource/cmap
<rootpath>/PDFlib/TET/5.0/resource/codelist
<rootpath>/PDFlib/TET/5.0/resource/glyphlst
<rootpath>/PDFlib/TET/5.0/resource/fonts
<rootpath>/PDFlib/TET/5.0/resource/icc
<rootpath>/PDFlib/TET/5.0
<rootpath>/PDFlib/TET
<rootpath>/PDFlib
```

Unter Unix, Linux und OS X wird `<rootpath>` zuerst durch `/usr/local` und dann durch das HOME-Verzeichnis ersetzt. Auf i5/iSeries ist `<rootpath>` leer.

**Standard-Dateinamen für Lizenz- und Ressourcendateien.** Standardmäßig werden die folgenden Dateinamen in den Standard-Verzeichnissuchpfaden gesucht:

<code>licensekeys.txt</code>	(Lizenzdatei)
<code>pdflib.upr</code>	(Ressourcendatei)

Mit dieser Funktion lässt sich eine Lizenzdatei ohne die Angabe einer Umgebungsvariablen oder Laufzeit-Option verwenden.

**Suchen der UPR-Ressourcendatei.** Sollen Ressourcendateien verwendet werden, so können Sie diese mit `TET_set_option()` (siehe unten) oder in einer UPR-Ressourcendatei festlegen. TET liest diese Datei automatisch, sobald die erste Ressource benötigt wird. Im Einzelnen wird wie folgt vorgegangen:

- ▶ Ist die Umgebungsvariable `TETRESOURCEFILE` definiert, verwendet TET deren Wert als Name für die zu lesende UPR-Datei. Kann diese Datei nicht gelesen werden, wird eine Exception ausgelöst.
- ▶ Ist die Umgebungsvariable `TETRESOURCEFILE` nicht definiert, versucht TET eine Datei mit folgendem Namen zu öffnen:

```
upr (auf MVS; ein Dataset wird erwartet)
tet.upr (unter Windows, Unix und allen anderen Systemen)
```

Kann diese Datei nicht gelesen werden, wird keine Exception ausgelöst.

- ▶ Unter Windows versucht TET zudem, den folgenden Registry-Eintrag zu lesen:

```
HKLM\SOFTWARE\PDFlib\TET5\5.0\resourcefile
```

Der Wert dieses Eintrags (der mit dem Wert `<installdir>/tet.upr` bei der TET-Installation erzeugt wird, aber auch manuell belegt werden kann) wird als Name der zu lesenden Ressourcendatei verwendet. Kann diese Datei nicht gelesen werden, wird eine Exception ausgelöst.

- ▶ Durch explizites Setzen der Option `resourcefile` kann der Client TET veranlassen, eine Ressourcendatei zur Laufzeit einzulesen:

```
set_option("resourcefile=/path/to/tet.upr");
```

Dieser Aufruf kann beliebig oft wiederholt werden; die Ressourceneinträge werden akkumuliert.

**Konfiguration von Ressourcen zur Laufzeit.** Statt eine UPR-Datei zur Konfiguration zu verwenden, können Sie einzelne Ressourcen mit `TET_set_option()` direkt zur Laufzeit konfigurieren. Diese Funktion erhält den Namen einer Ressourcenkategorie sowie Name/Wert-Paare, wie sie auch im entsprechenden Abschnitt in der UPR-Datei erscheinen, zum Beispiel:

```
set_option("glyphlist={myglyphnames=/usr/local/glyphnames.gl}");
```

In der Optionsliste für eine Ressourcenkategorie-Option können mehrere Ressourcennamen angegeben werden (eine Ressourcenkategorie-Option darf in einem Aufruf von `TET_set_option()` aber nicht mehrfach vorkommen). Alternativ lassen sich mit mehreren Aufrufen Ressourcen-Einstellungen akkumulieren.

**Escape-Sequenzen für Textdateien.** Escape-Sequenzen werden von allen Textdateien außer UPR- und CMap-Dateien unterstützt. Mittels spezieller Zeichenfolgen können nicht druckbare Zeichen in Textdateien eingefügt werden. Solche Sequenzen beginnen prinzipiell mit dem Gegenschrägstrich `\`:

- ▶ `\x` leitet eine Folge von zwei Hexadezimalzeichen ein (`0-9`, `A-F`, `a-f`), z.B. `\x0D`
- ▶ `\nnn` steht für eine Folge von drei Oktalzahlen (`0-7`), z.B. `\015`. Die Sequenz `\000` wird ignoriert.
- ▶ Die Sequenz `\\` bezeichnet einen einfachen Gegenschrägstrich.
- ▶ Ein Gegenschrägstrich am Zeilenende setzt das Zeilenende-Zeichen außer Kraft.

## 5.3 Empfehlungen für häufige Anwendungsfälle

TET bietet zahlreiche Optionen zur Steuerung verschiedener Aspekte der Textextraktion. In diesem Abschnitt möchten wir einige Ratschläge für häufige Anwendungsfälle von TET geben. Weitere Informationen zu den hier erwähnten Funktionen und Optionen finden Sie in Kapitel 10, »API-Referenz für die TET-Bibliothek«, Seite 173.

**Optimierung der Verarbeitungsgeschwindigkeit.** In einigen Situationen, die insbesondere Suchmaschinen betreffen, ist es von entscheidender Bedeutung, wie schnell sich der Text extrahieren lässt. Die optimale Ausgabe tritt dann in den Hintergrund. Die Standardeinstellungen von TET erzielen auf eine bestmögliche Ausgabe ab, sie lassen sich jedoch auch für eine möglichst schnelle Verarbeitung anpassen. Um den Durchsatz bei der Textextraktion zu erhöhen, können Sie Optionen in `TET_open_page()` und `TET_open_document()` wie folgt nutzen:

- ▶ `docstyle=searchengine`

Mit dieser Seitenoption werden mehrere interne Parameter so eingestellt, dass die Verarbeitung durch Reduzierung der Ausgabequalität beschleunigt wird, ohne dass die Indizierung für Suchmaschinen davon negativ beeinflusst wird.

- ▶ `engines={image=false textcolor=false}`

Sind weder Bildextraktion noch Erkennung der Textfarbe erforderlich, können mit dieser Dokumentoption zur Beschleunigung der Verarbeitung interne Verarbeitungsschritte deaktiviert werden.

- ▶ `contentanalysis={merge=0}`

Mit dieser Seitenoption wird das aufwändige Zusammenführen von Streifen und Zonen deaktiviert, wodurch sich die Verarbeitungszeit für typische Dateien auf ca. 60% im Vergleich zu den Standardeinstellungen reduziert. Bei Dokumenten jedoch, deren Inhalt beliebig über die Seiten verstreut ist, kann mancher Text unter Umständen nicht in logischer (Lese-)Reihenfolge extrahiert werden.

- ▶ `contentanalysis={shadowdetect=false}`

Mit dieser Seitenoption wird die Erkennung von redundanten Schatten und künstlicher Fettschrift deaktiviert, was die Verarbeitungszeit ebenfalls reduziert.

- ▶ Bei der Erzeugung von TETML kann mit der folgenden Dokumentoption die Erzeugung von TETML-Elementen für verschiedene interaktive PDF-Features deaktiviert werden:

```
tetml={elements={annotations=false bookmarks=false destinations=false fields=false  
javascripts=false}}
```

**Wörter vs. Zeilenstruktur vs. Fließtext.** Je nach Anwendung kann eine andere Art von Ausgabe wünschenswert sein (getrennte Wörter werden bei den folgenden Einstellungen immer entrennt):

- ▶ Einzelne Wörter (Layout wird ignoriert): Eine Suchmaschine interessiert sich oft nur für die im Text enthaltenen Wörter und nicht für Layout-relevante Aspekte. In diesem Fall verwenden Sie `granularity=word` von `TET_open_page()`, um pro Aufruf von `TET_get_text()` genau ein Wort zu extrahieren.
- ▶ Erhalt der Zeilenstruktur: Verwenden Sie `granularity=page` von `TET_open_page()`, um den gesamten Textinhalt auf der Seite mit einem einzigen Aufruf von `TET_get_text()` zu extrahieren. Um die vorhandene Zeilenstruktur zu erhalten, werden Textzeilen durch ein Zeilenvorschub-Zeichen U+00A0 (*linefeed, LF*) getrennt.

- ▶ Fließtext: Um Zeilenumbrüche zu verhindern und das spätere Umbrechen des extrahierten Textes zu erleichtern, verwenden Sie die Dokumentoption *lineseparator=U+0020* und die Seitenoption *granularity=page*. Der gesamte Textinhalt auf der Seite lässt sich mit einem einzigen Aufruf von *TET\_get\_text()* extrahieren. Absätze werden standardmäßig durch U+000A getrennt. Ein anderer Absatz-Separator kann durch Angabe der Dokumentoption *paraseparator=U+2029* (oder einen anderen geeigneten Unicode-Wert) angewendet werden.

**Aufbau einer Suchmaschine oder eines Indexers.** Indexer interessieren sich in der Regel nicht für die Position des Textes auf der Seite (außer sie markieren die Suchbegriffe). In vielen Fällen tolerieren sie Fehler, die bei der Unicode-Zuordnung auftreten, und verarbeiten einfach den zurückgegebenen Text. Empfehlungen:

- ▶ Verwenden Sie *granularity=word* von *TET\_open\_page()*.
- ▶ Wenn die Anwendung Satzzeichen verarbeiten kann, können Sie diese in den angrenzenden Text aufnehmen, indem Sie die folgende Seitenoption setzen:  
*contentanalysis={punctuationbreaks=false}*

**Geometrie.** Geometrische Eigenschaften können in manchen Anwendungen sinnvoll genutzt werden:

- ▶ Die Schnittstelle *TET\_get\_char\_info()* ist nur erforderlich, wenn Sie die Position des Textes auf der Seite, die jeweiligen Fontnamen, die Textfarbe oder andere Details benötigen. Wenn Sie keine Textkoordinaten benötigen, ist der Aufruf von *TET\_get\_text()* ausreichend.
- ▶ Wenn Sie genauere Informationen über das Seitenlayout besitzen, können Sie die Optionen *includebox* und/oder *excludebox* von *TET\_open\_page()* verwenden, um Kopf- und Fußzeilen und andere nicht zum Haupttext gehörende Objekte zu entfernen.

**Komplexe Layouts.** Manche Dokumentklassen verwenden ein recht komplexes Seitenlayout. Bei Zeitschriften und Illustrierten kann TET zum Beispiel die Beziehung zwischen den Spalten auf der Seite eventuell nicht richtig erkennen. In solchen Fällen ist es möglich, den extrahierten Text auf Kosten der Verarbeitungszeit zu verbessern. Für diesen Zweck geeignete Optionen finden Sie in Abschnitt 6.7, »Layout-Analyse«, Seite 103. Für weitere Informationen zu den relevanten Optionen siehe Tabelle 10.2, Seite 180.

**Juristische Texte.** Wenn Sie mit rechtlich relevanten Dokumenten arbeiten, ist meist keinerlei Toleranz gegenüber falschen Unicode-Zuordnungen erlaubt, da diese den Inhalt oder die Interpretation eines Dokuments verfälschen könnten. In vielen Fällen ist die Textposition nicht erforderlich, und der Text muss wortweise extrahiert werden. Empfehlungen:

- ▶ Verwenden Sie die Option *granularity=word* von *TET\_open\_page()*.
- ▶ Verwenden Sie die Option *password* von *TET\_open\_document()* mit dem geeigneten Dokumentkennwort, wenn Sie Dokumente verarbeiten, die zum Öffnen ein Kennwort benötigen, oder die Option *shrug*, wenn in den Sicherheitseinstellungen keine Textextraktion erlaubt ist und Sie berechtigt sind, Text aus dem Dokument zu extrahieren (siehe »Das Feature »shrug« für geschützte Dokumente«, Seite 68).
- ▶ Für absolute Texttreue: Brechen Sie die Verarbeitung ab, sobald das Feld *unknown* in der von *TET\_get\_char\_info()* zurückgegebenen *charinfo*-Struktur gleich 1 oder das Unicode-Ersatzzeichen U+FFFD in dem String enthalten ist, der von *TET\_get\_text()* zu-

rückgegeben wird. In TETML mit einem der Textmodi *glyph* oder *wordplus* können Sie diesen Fall an folgendem Attribut im Element *Glyph* erkennen:

```
unknown="true"
```

Setzen Sie die Option *unknownchar* nicht auf ein häufig verwendetes Zeichen, da Sie es dann ohne Überprüfung des Feldes *unknown* nicht mehr von korrekt zugeordneten Zeichen unterscheiden können.

- ▶ Um Texttreue zu gewährleisten, können Sie außerdem einstellen, dass nur Text extrahiert wird, der auf der Seite auch sichtbar ist:

```
ignoreinvisibletext=true
```

**Verarbeitung von Dokumenten mit PDFlib+PDI.** Wenn Sie PDF-Dokumente mit PDFlib+PDI seitenweise verarbeiten, können Sie TET integrieren, um das Zerlegen und Zusammensetzen des Dokuments zu steuern. Sie könnten ein PDF-Dokument beispielsweise auf Basis des Seiteninhalts aufspalten. Wenn Sie die Kontrolle über den Erstellungsvorgang haben, können Sie Trennseiten mit geeigneten Verarbeitungsanweisungen im Text einfügen. Beispiele für die Analyse von Dokumenten mit TET und der anschließenden Verarbeitung mit PDFlib+PDI finden Sie im TET Cookbook.

**Veraltete PDF-Dokumente mit fehlenden Unicode-Werten.** In manchen Fällen müssen PDF-Dokumente verarbeitet werden, die aus veralteten Anwendungen stammen. Das PDF enthält dann unter Umständen nicht genug Informationen für eine korrekte Unicode-Zuordnung und TET ist anhand der Standardeinstellungen nicht in der Lage, den Text ganz oder teilweise zu extrahieren. Empfehlungen:

- ▶ Beginnen Sie die Textextraktion mit Standardeinstellungen und analysieren Sie die Ergebnisse. Ermitteln Sie die Fonts, denen Informationen für eine korrekte Unicode-Zuordnung fehlen.
- ▶ Schreiben Sie eigene Encoding-Tabellen und Glyphnamenlisten, um die problematischen Fonts in den Griff zu bekommen. Verwenden Sie das Plugin PDFlib FontReporter zur Analyse der Fonts und Vorbereitung der Unicode-Zuordnungstabellen.
- ▶ Konfigurieren Sie die selbst erstellten Zuordnungstabellen und extrahieren Sie den Text erneut mit einer größeren Anzahl von Dokumenten. Wenn immer noch nicht alle Glyphen zugeordnet werden können, so passen Sie die Zuordnungstabellen entsprechend an.
- ▶ Wenn Sie eine größere Anzahl von Dokumenten mit problematischen Glyphen besitzen, kann Sie PDFlib GmbH dabei unterstützen, geeignete Zuordnungstabellen zu erstellen.

**Konvertierung von PDF-Dokumenten in ein anderes Format.** Wenn Sie die Seiteninhalte von PDF-Dokumenten in Ihre Anwendung importieren und dabei möglichst viele Informationen erhalten möchten, benötigen Sie die genaue Zeichenposition. Empfehlungen:

- ▶ Verwenden Sie *TET\_get\_char\_info()* zur Abfrage der Zeichenposition und Fontnamen. Auch wenn Sie mit dem Feld *uv* die Unicode-Werte einzelner Zeichen abfragen, müssen Sie mit *TET\_get\_text()* dennoch die Struktur *char\_info* füllen.
- ▶ Verwenden Sie je nach den Anforderungen Ihrer Anwendung *granularity=glyph* oder *word* in *TET\_open\_page()*. Bei *granularity=glyph* kann es zu Konflikten zwischen dem visuellen Layout des Textes und dem von TET erzeugten, verarbeiteten logischen

Text kommen (z.B. könnten die von einer Ligatur-Glyphe erzeugten zwei Zeichen mehr Platz benötigen als die Ligatur selbst).

**Firmschriften mit individuell kodierten Logos.** In vielen Fällen enthalten Firmschriften, die ein Firmenlogo enthalten, fehlende oder falsche Unicode-Zuordnungen für die Logos. Wenn Sie häufig mit solchen PDF-Dokumenten zu tun haben, sollten Sie eine benutzerdefinierte Zuordnungstabelle mit korrekten Unicode-Werten erstellen.

Beginnen Sie mit der Erstellung eines Fontreports für ein PDF, das den problematischen Font enthält (siehe »Analyse von PDF-Dokumenten mit dem Plugin PDFlib Font-Reporter«, Seite 124) und ermitteln Sie die falsch zugeordneten Glyphen. Abhängig vom Fontformat können Sie die fehlenden Unicode-Zuordnungen anhand der verfügbaren Konfigurationstabellen bereitstellen. Ein ausführliches Beispiel für die Codeliste zu einem Font für ein Logo finden Sie in »Codelist-Ressourcen für alle Fonttypen«, Seite 126.

**TeX-Dokumente.** Mit TeX erzeugte PDF-Dokumente enthalten oft numerische Glyphennamen, Type-3-Fonts und andere problematische Properties, wodurch sich der Text mit anderen Produkten oft nicht mehr erfolgreich extrahieren lässt. TET bietet für solche Dokumente viele Heuristiken und Workarounds. Für eine ganz bestimmte Variante von TeX-Dokumenten benötigt der Workaround von TET allerdings mehr Verarbeitungszeit und ist deshalb standardmäßig deaktiviert. Sie können CPU-intensivere Fontverarbeitung für diese Dokumente mit der folgenden Dokumentoption aktivieren:

```
checkglyphlists=true
```



# 6 Textextraktion

## 6.1 PDF-Dokumentdomänen

PDF-Dokumente können außer den Seiteninhalten noch an vielen anderen Stellen Text enthalten. Obwohl die meisten Anwendungen nur die Seiteninhalte verarbeiten, sind oft auch andere Domänen eines Dokuments relevant.

Während sich Seiteninhalte mit den Standardfunktionen `TET_get_text()` und `TET_get_image()` abfragen lassen, können mit Hilfe der integrierten pCOS-Schnittstelle auch Inhalte aus anderen Dokumentdomänen extrahiert werden.

Im Folgenden wird die Suche in anderen Dokumentdomänen mit der TET-Bibliothek und TETML näher erläutert. Um die Suchtreffer für diese Dokumentdomänen auch in Acrobat lokalisieren zu können, werden die dazu erforderlichen Schritte in Acrobat X/XI/DC aufgeführt.

**Text auf der Seite.** Seiteninhalte sind die Hauptquelle für Text in PDF. Der Text auf einer Seite wird mit Hilfe von Fonts dargestellt und mit einer der vielen Encoding-Techniken von PDF kodiert.

- ▶ Anzeige in Acrobat: Seiteninhalte sind immer sichtbar
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: *Bearbeiten, Suchen* oder *Bearbeiten, Erweiterte Suche*. TET ist in der Lage, auch Text in Dokumenten zu verarbeiten, für den Acrobat keine korrekte Glyphenzuordnung zu Unicode-Werten liefern kann. In diesem Fall können Sie das auf TET basierende TET Plugin verwenden (siehe Abschnitt 4.1, »Kostenloses TET Plugin für Adobe Acrobat«, Seite 49). Es verfügt über einen eigenen Suchdialog *Zusatzmodule, PDFlib TET Plugin...*, *TET Suchen*, ist jedoch nicht als vollwertige Suchfunktionalität konzipiert.
- ▶ Durchsuchen mehrerer PDF-Dateien mit Acrobat X/XI/DC: *Bearbeiten, Erweiterte Suche*. Klicken Sie auf *Mehr Optionen anzeigen*. Wählen Sie unter *Suchen in:* das gewünschte Verzeichnis mit PDF-Dokumenten.
- ▶ Beispielcode für die TET-Bibliothek: *extractor*-Minibeispiel
- ▶ TETML-Element: `/TET/Document/Pages/Page/Content`

**Vordefinierte Dokument-Infofelder.** Prinzipiell handelt es sich bei Dokument-Infefeldern um Schlüssel-/Wertpaare.

- ▶ Anzeige in Acrobat X/XI/DC: *Datei, Eigenschaften...*
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: nicht verfügbar
- ▶ Durchsuchen mehrerer PDF-Dateien mit Acrobat X/XI/DC: Klicken Sie auf *Bearbeiten, Erweiterte Suche* und dann im unteren Bereich des Fensters auf *Mehr Optionen anzeigen*. Wählen Sie unter *Suchen in:* das gewünschte Verzeichnis mit PDF-Dokumenten und unter *Folgende Zusatzkriterien verwenden:* aktivieren Sie eins der folgenden: *Erstellungsdatum, Änderungsdatum, Verfasser, Titel, Thema, Stichwörter*.
- ▶ Beispielcode für die TET-Bibliothek: *dumper*-Minibeispiel
- ▶ TETML-Element: `/TET/Document/DocInfo`

**Benutzerdefinierte Dokument-Infofelder.** Sie können zusätzliche Dokument-Infofelder nach Ihren eigenen Wünschen definieren.

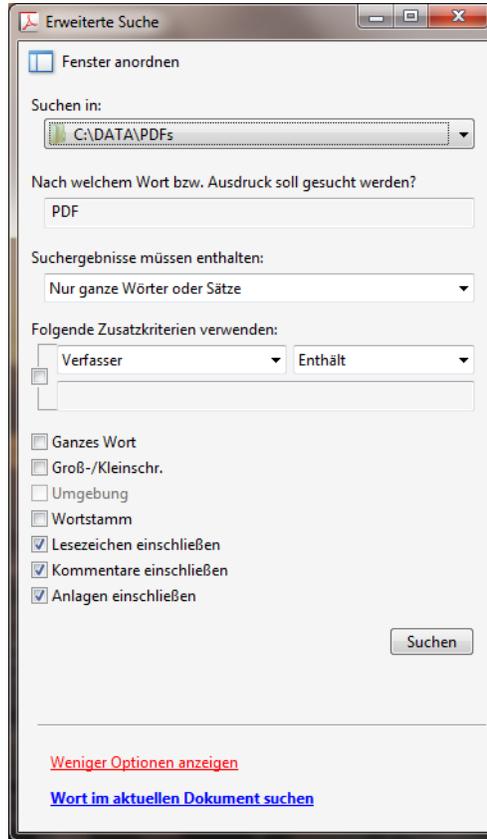


Abb. 6.1  
Die erweiterte Suche  
von Acrobat

- ▶ Anzeige in Acrobat X/XI/DC: *Datei, Eigenschaften...*, *Benutzerdefiniert* (im kostenlosen Adobe Reader nicht verfügbar)
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: nicht verfügbar
- ▶ Beispielcode für die TET-Bibliothek: *dumper*-Minibeispiel
- ▶ TETML-Element: `/TET/Document/DocInfo/Custom`

**XMP-Metadaten auf Dokumentebene.** XMP-Metadaten bestehen aus einem XML-Stream mit erweiterten Metadaten.

- ▶ Anzeige in Acrobat X/XI/DC: *Datei, Eigenschaften...* und in der Registerkarte *Beschreibung* klicken Sie auf *Zusätzliche Metadaten...* (im kostenlosen Adobe Reader nicht verfügbar)
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: nicht verfügbar
- ▶ Durchsuchen mehrerer PDF-Dateien mit Acrobat X/XI/DC: Klicken Sie auf *Bearbeiten, Erweiterte Suche* und dann im unteren Bereich des Fensters auf *Mehr Optionen anzeigen*. Wählen Sie unter *Suchen in:* das gewünschte Verzeichnis mit PDF-Dokumenten und unter *Folgende Zusatzkriterien verwenden:* aktivieren Sie *XMP-Metadaten* (im kostenlosen Adobe Reader nicht verfügbar).
- ▶ Beispielcode für die TET-Bibliothek: *dumper*-Minibeispiel
- ▶ TETML-Element: `/TET/Document/Metadata`

**XMP-Metadaten auf Bildebene.** XMP-Metadaten können an Dokumentbestandteile wie Bilder, Seiten, Fonts usw. angehängt werden. XMP ist jedoch außer auf Dokumentenebene meist nur noch auf Bildebene vorhanden.

- ▶ Anzeige in Acrobat X: *Werkzeuge, Inhalt, Objekt bearbeiten*, rechtsklicken Sie auf das Bild und wählen Sie *Metadaten anzeigen...* (im kostenlosen Adobe Reader nicht verfügbar)
- ▶ Anzeige in Acrobat XI/DC: *Anzeige, Ein-/Ausblenden, Navigationsfenster, Inhalt*, navigieren Sie zum gewünschten Bild, rechtsklicken Sie darauf und wählen Sie *Metadaten anzeigen...* (im kostenlosen Adobe Reader nicht verfügbar)
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: nicht verfügbar
- ▶ Beispielcode für die TET-Bibliothek: pCOS-Cookbook-Topic *image\_metadata*
- ▶ TETML-Element: */TET/Document/Pages/Page/Resources/Images/Image/Metadata*

**Text in Formularfeldern.** Formularfelder werden über der Seite liegend angezeigt. Technisch gesehen sind sie nicht Teil des Seiteninhalts, sondern werden durch separate Datenstrukturen dargestellt.

- ▶ Anzeige in Acrobat X/XI: *Werkzeuge, Formulare, Bearbeiten* (im kostenlosen Adobe Reader nicht verfügbar)
- ▶ Anzeige in Acrobat DC: *Werkzeuge, Formulare vorbereiten...* (im kostenlosen Adobe Reader nicht verfügbar)
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: nicht verfügbar
- ▶ Beispielcode für die TET-Bibliothek: pCOS-Cookbook-Topic *fields*
- ▶ TETML-Element: */TET/Document/Pages/Page/Fields/Field/Value*

**Text in Kommentaren (Anmerkungen).** Ähnlich wie Formularfelder werden Anmerkungen (Notizen, Kommentare usw.) über der Seite liegend und durch separate Datenstrukturen dargestellt. Die relevanten Textinhalte einer Anmerkung sind abhängig von ihrem Typ. Bei Weblinks kann der relevante Teil der URL sein, während für andere Anmerkungstypen die sichtbaren Textinhalte von Bedeutung sein können.

- ▶ Anzeige in Acrobat X/XI: *Kommentar, Kommentarliste*
- ▶ Anzeige in Acrobat DC: *Werkzeuge, Kommentar, Kommentarliste*
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: *Bearbeiten, Suche*, aktivieren Sie *Kommentare einfügen* oder verwenden Sie die Schaltfläche *Kommentare suchen* in der Kommentarleiste
- ▶ Durchsuchen mehrerer PDF-Dateien mit Acrobat X/XI/DC: Klicken Sie auf *Bearbeiten, Erweiterte Suche* und dann im unteren Bereich des Fensters auf *Mehr Optionen anzeigen*. Wählen Sie unter *Suchen in*: das gewünschte Verzeichnis mit PDF-Dokumenten und unter *Folgende Zusatzkriterien verwenden*: aktivieren Sie *Kommentare einschließen*.
- ▶ Beispielcode für die TET-Bibliothek: pCOS-Cookbook-Topic *annotations*
- ▶ TETML-Element: */TET/Document/Pages/Page/Annotations/Annotation*

**Text in Lesezeichen.** Lesezeichen sind nicht direkt seitenbezogen, auch wenn sie eine Aktion enthalten können, die auf eine bestimmte Seite springt. Lesezeichen können hierarchisch verschachtelt werden.

- ▶ Anzeige in Acrobat X/XI/DC: *Anzeige, Ein-/Ausblenden, Navigationsfenster, Lesezeichen*
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: *Bearbeiten, Erweiterte Suche*, aktivieren Sie *Lesezeichen einfügen*
- ▶ Durchsuchen mehrerer PDF-Dateien mit Acrobat X/XI/DC: Klicken Sie auf *Bearbeiten, Erweiterte Suche* und dann im unteren Bereich des Fensters auf *Mehr Optionen an-*

zeigen. Wählen Sie unter *Suchen in*: das gewünschte Verzeichnis mit PDF-Dokumenten und unter *Folgende Zusatzkriterien verwenden*: aktivieren Sie *Lesezeichen einschließen* (im kostenlosen Adobe Reader nicht verfügbar)

- ▶ Beispielcode für die TET-Bibliothek: *pCOS-Cookbook-Topic bookmarks*
- ▶ TETML-Element: */TET/Document/Bookmarks/Bookmark/Title*

**Dateianhänge.** PDF-Dokumente können Dateianhänge auf Seiten- oder Dokumentenebene enthalten, die selbst wiederum PDF-Dokumente sind.

- ▶ Anzeige in Acrobat X/XI/DC: *Anzeige, Ein-/Ausblenden, Navigationsfenster, Anlagen*
- ▶ Durchsuchen mit Acrobat X/XI/DC: Klicken Sie auf *Bearbeiten, Erweiterte Suche* und aktivieren Sie *Anlagen einschließen* (im kostenlosen Adobe Reader nicht verfügbar). Verschachtelte Dateianhänge werden nicht rekursiv durchsucht.
- ▶ Beispielcode für die TET-Bibliothek: *get\_attachments*-Minibeispiel
- ▶ TETML-Element: */TET/Document/Attachments/Attachment/Document*

**PDF-Pakete und -Portfolios.** Bei PDF-Paketen und PDF-Portfolios handelt es sich um Dateianhänge mit zusätzlichen Eigenschaften.

- ▶ Anzeige in Acrobat X/XI/DC: Acrobat zeigt das Deckblatt der Pakete/Portfolios und ihre einzelnen Bestandteile mit einer speziellen Benutzeroberfläche für PDF-Pakete an.
- ▶ Durchsuchen eines einzelnen PDF-Pakets mit Acrobat X/XI/DC: *Bearbeiten, Gesamtes Portfolio durchsuchen*
- ▶ Durchsuchen mehrerer PDF-Pakete mit Acrobat X/XI/DC: nicht verfügbar
- ▶ Beispielcode für die TET-Bibliothek: *get\_attachments*-Minibeispiel
- ▶ TETML-Element: */TET/Document/Attachments/Attachment/Document*

**PDF-Standards und andere PDF-Merkmale.** Diese Domäne enthält keinen expliziten Text, sondern wird als Container für verschiedene interne Eigenschaften eines PDF-Dokuments verwendet, z.B. Status von PDF/X und PDF/A oder Tagged PDF.

- ▶ Anzeige in Acrobat X/XI/DC: *Anzeige, Ein-/Ausblenden, Navigationsfenster, Standards* (nur bei standardkonformen PDF-Dokumente verfügbar)
- ▶ Durchsuchen einer einzelnen PDF-Datei mit Acrobat X/XI/DC: nicht verfügbar
- ▶ Beispielcode für die TET-Bibliothek: *dumper*-Minibeispiel
- ▶ TETML-Elemente und -Attribute: */TET/Document/@pdfa, /TET/Document/@pdfe, /TET/Document/@pdfua, /TET/Document/@pdfvt, /TET/Document/@pdfx*

**Tagged PDF.** TET rekonstruiert die Layout-Struktur und -Hierarchie direkt aus den Seiteninhalten, ohne den Strukturbaum zu verwenden, der in Tagged PDF vorhanden ist. Seiteninhalte, die zum Verständnis des Dokuments nicht erforderlich sind, sondern lediglich zu Layoutzwecken oder als Dekoration erzeugt werden, können in Tagged PDF als Artefakte ausgezeichnet werden. Die häufigste Verwendung von Artefakten findet sich bei laufenden Kopf- und Fußzeilen sowie Seitenzahlen und Kapitelüberschriften. Je nach Anwendungsfall kann es wünschenswert sein, als Seiteninhalte ausgezeichnete Artefakte zu verarbeiten oder auch nicht.

- ▶ Anzeige in Acrobat XI/DC: *Anzeige, Ein-/Ausblenden, Navigationsfenster, Tags*. Klicken Sie im Menü *Tags* auf *Suchen...* und wählen Sie *Artefakte*. Als Artefakte ausgezeichnete Text, Bilder und Vektorgrafiken werden markiert. Alternativ können Sie folgendes aktivieren: *Werkzeuge, Barrierefreiheit, TouchUp*

*Leserichtung.* Das Tool markiert die mit Tags versehenen Inhalte mit schattierten Rechtecken. Bei den nicht markierten Inhalten handelt es sich um Artefakte.

- ▶ Ignorieren von Artefakten bei der Suche in Acrobat X/XI/DC: nicht verfügbar
- ▶ Ignorieren von Artefakten bei TET: übergeben Sie die Seitenoption *ignoreartifacts*.
- ▶ TETML: Artefakte werden von TETML nicht identifiziert, können aber mit der Seitenoption *ignoreartifacts* ausgeblendet werden.

**Ebenen.** Mit Hilfe von Ebenen (der technische Begriff ist *optional content*) können Seiteninhalte sichtbar oder unsichtbar gemacht werden. Je nach Anwendungsfall kann es wünschenswert sein, Seiteninhalte auf unsichtbaren Ebenen zu verarbeiten oder auch nicht:

- ▶ Anzeige in Acrobat XI/DC: *Anzeige, Ein-/Ausblenden, Navigationsfenster, Ebenen.* Für sichtbare Ebenen wird vor dem Namen ein Augensymbol eingeblendet. Mit einem Klick auf dieses Symbol lässt sich die Sichtbarkeit steuern.
- ▶ Suche in Acrobat X/XI/DC: Acrobat sucht den Inhalt aller Ebenen. Wird ein Suchergebnis auf einer unsichtbaren Ebene gefunden, bietet Acrobat an, die Ebene sichtbar zu machen.
- ▶ Verarbeitung von Ebenen mit TET: mit der Seitenoption *layers* lässt sich die Inhaltsextraktion auf sichtbare oder unsichtbare Ebenen einschränken. Alternativ können alle Ebenen verarbeitet werden, was nur sinnvoll ist, wenn diese sich nicht überlappen.
- ▶ TETML: Ebenen-Inhalte werden gemäß der Seitenoption *layers* verarbeitet. Name sowie Sichtbarkeitsstatus und andere Eigenschaften von Ebenen werden im TETML-Element */TET/Document/Pages/Graphics/Layers/Layer* aufgeführt.

## 6.2 Geometrie von Seite und Text

**Standardkoordinatensystem.** TET zeigt per Voreinstellung Größenangaben zu Seite und Text im Standardkoordinatensystem von PDF an. Der Ursprung des Koordinatensystems (der sich auch außerhalb der Seite befinden kann) wird dabei jedoch auf die linke untere Ecke der sichtbaren Seite verlegt. Genauer gesagt befindet sich der Ursprung in der linken unteren Ecke der *CropBox* (sofern vorhanden) oder sonst der *Mediabox*. Die Seite wird gedreht, wenn sie über das Schlüsselwort *Rotate* verfügt. Das Koordinatensystem verwendet als Einheit den DTP-Punkt:

$$1 \text{ pt} = 1 \text{ inch} / 72 = 25.4 \text{ mm} / 72 = 0.3528 \text{ mm}$$

Die erste Koordinate wächst nach rechts, die zweite nach oben. Alle von TET benötigten oder zurückgegebenen Koordinaten werden per Voreinstellung gemäß diesem Koordinatensystem interpretiert, unabhängig von ihrer Darstellung im PDF-Dokument. Zur Bestimmung der Größe einer PDF-Seite siehe die pCOS-Pfadreferenz.

**Top-Down-Koordinatensystem.** Im Gegensatz zum Bottom-Up-Koordinatensystem von PDF arbeiten manche grafischen Umgebungen mit Top-Down-Koordinaten. Um die Verwendung von Top-Down-Koordinaten zu vereinfachen, unterstützt TET ein alternatives Koordinatensystem, dessen Ursprung nicht in der linken unteren, sondern in der linken oberen Ecke liegt, wobei die *y*-Koordinaten nach unten zunehmen. Die *topdown*-Einstellung ermöglicht dem TET-Benutzer, seine gewohnte Arbeitsweise auch im Top-Down-Koordinatensystem beizubehalten. Als zusätzlicher Vorteil sind Top-Down-Koordinaten identisch mit den in Acrobat angezeigten Koordinatenwerten (siehe unten). Das Top-Down-Koordinatensystem für eine Seite lässt sich mit der Option *topdown={output}* aktivieren.

**Koordinatenanzeige in Acrobat.** Seitenkoordinaten können Sie in Acrobat folgendermaßen anzeigen lassen (siehe Abbildung 6.2):

- ▶ Zum Anzeigen der Cursor-Koordinaten in Acrobat X/XI/DC wählen Sie den Menübefehl *Anzeige, Ein-/Ausblenden, Cursorkoordinaten*.
- ▶ Die Koordinaten werden in der Einheit angezeigt, die aktuell in Acrobat eingestellt ist. Zur Verwendung der in TET verwendeten Einheit Punkt in Acrobat X/XI/DC gehen Sie folgendermaßen vor: Wählen Sie den Menübefehl *Bearbeiten, Voreinstellungen, Einheiten und Hilfslinien* und aktivieren Sie unter *Einheiten* den Wert *Punkt*.

Beachten Sie dabei, dass sich die angezeigten Koordinaten auf einen Ursprung in der linken oberen Ecke der Seite beziehen und nicht, wie bei PDF und TET üblich, auf die linke untere Ecke. Für weitere Informationen zur Auswahl von Top-Down-Koordinatensystemen, die sich an die Koordinatenanzeige von Acrobat anpassen, siehe den vorigen Abschnitt.

**Bereich der Textextraktion.** Standardmäßig extrahiert TET jeden Text, der sich im sichtbaren Bereich der Seite befindet. Mit der Option *clippingarea* von *TET\_open\_page()* (siehe Tabelle 10.10, Seite 199) können Sie diesen Bereich durch eine der PDF-Seitengrößenangaben (z.B. *TrimBox*) ersetzen. Mit dem Schlüsselwort *unlimited* wird der gesamte

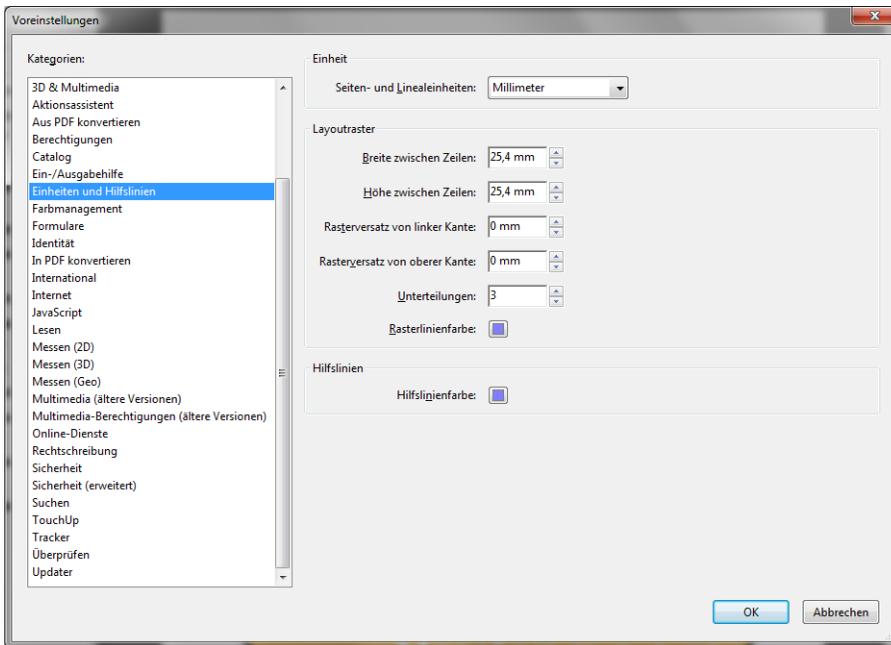


Abb. 6.2 Konfigurieren der Koordinatenanzeige in Acrobat; Zum Anzeigen der Cursor-Koordinaten wählen Sie den Menübefehl Anzeige, Ein-/Ausblenden, Cursorkoordinaten.

Text unabhängig von den Boxeinträgen der Seite extrahiert. Mit dem Standardwert *cropbox* extrahiert TET den Text innerhalb des in Acrobat sichtbaren Bereichs.

Mit den Optionen *includebox* und *excludebox* von *TET\_open\_page()* können beliebig viele Rechtecke übergeben werden, die den Bereich der Textextraktion genauer festlegen. Dies ist zum Beispiel sinnvoll, wenn Text nur aus bestimmten Seitenabschnitten (wie etwa einzelnen Spalten) extrahiert werden soll oder irrelevante Seiteninhalte auszusparsen sind (wie etwa Ränder und Kopf- oder Fußzeilen). Der letztendlich verwendete Clipping-Bereich besteht aus der Vereinigung aller mit der Option *includebox* festgelegten Rechtecke abzüglich aller mit der Option *excludebox* übergebenen Rechtecke. Eine Glyphe wird als innerhalb des Clipping-Bereichs angesehen, wenn sein Referenzpunkt im Clipping-Bereich liegt. Ein Zeichen kann damit auch dann zum Clipping-Bereich gehören, wenn es teilweise daraus hinausragt.

**Glyphenmetrik.** Mit *TET\_get\_char\_info()* lassen sich Font- und Größenangaben für alle Zeichen abfragen, die für eine Glyphe zurückgegeben werden. Die folgenden Werte sind für jedes ausgegebene Zeichen verfügbar (siehe Abbildung 6.3 und Tabelle 10.16):

- Das Feld *uv* enthält den UTF-32-Unicode-Wert des aktuellen Zeichens, d.h. desjenigen Zeichens, über das Informationen abgefragt werden. Dieses Feld enthält stets UTF-32, und zwar auch in Sprachbindungen, die Unicode-Unterstützung nur für UTF-16-Strings bieten. Durch den Zugriff auf das Feld *uv* können Anwendungen Zeichen außerhalb der BMP behandeln, ohne Surrogatpaare interpretieren zu müssen. Surrogatpaare werden als zwei getrennte Zeichen zurückgegeben. Das *uv*-Feld für den ers-

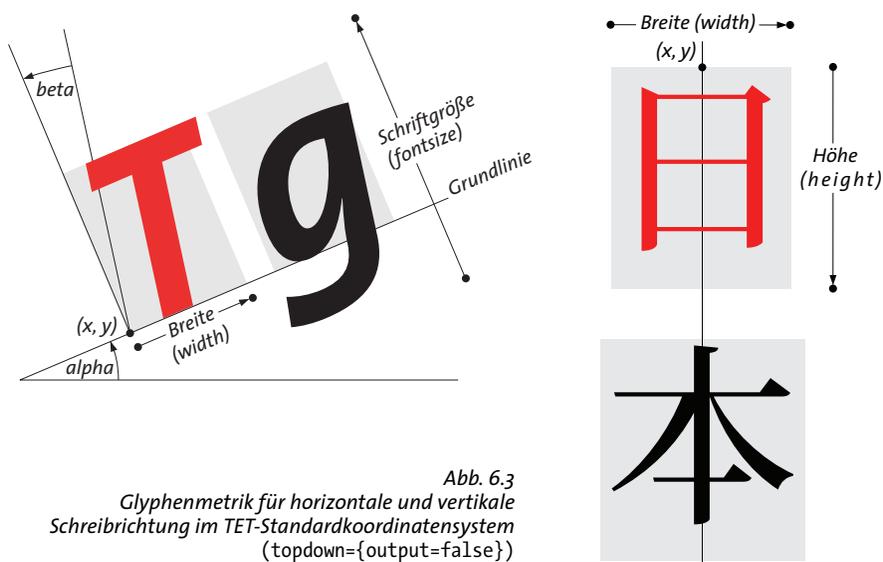


Abb. 6.3  
 Glyphenmetrik für horizontale und vertikale  
 Schreibrichtung im TET-Standardkoordinatensystem  
 (topdown={output=false})

ten Surrogatwert enthält den tatsächlichen Unicode-Wert (größer als U+FFFF). Der zweite Surrogatwert dagegen wird als künstliches Zeichen angesehen und erhält den *uv*-Wert 0.

- ▶ Das Feld *type* legt fest, auf welche Art das Zeichen erzeugt wurde. Dabei wird zwischen zwei Gruppen unterschieden, und zwar realen und künstlichen Zeichen. Die Gruppe der realen Zeichen besteht aus normalen Zeichen (d.h. der vollständigen Entsprechung einer einzelnen Glyphe) sowie aus Zeichen, die eine Sequenz aus mehreren Zeichen einleiten, die einer einzelnen Glyphe entspricht (z.B. das erste Zeichen einer Ligatur). Zur Gruppe der künstlichen Zeichen gehören die Fortsetzung einer Sequenz aus mehreren Zeichen (z.B. das zweite Zeichen einer Ligatur), der schließende Wert eines Surrogatpaars sowie eingefügte Trennzeichen. Bei künstlichen Zeichen gibt die Position  $(x, y)$  den Endpunkt des letzten realen Zeichens wieder, *width* und *height* sind gleich 0, und alle anderen Felder außer *uv* entsprechen denen des letzten realen Zeichens. Der Endpunkt berechnet sich aus dem Punkt  $(x, y)$  plus *width* in Richtung *alpha* (bei horizontaler Schreibrichtung) oder plus *height* in Richtung  $-90^\circ$  (bei vertikaler Schreibrichtung).
- ▶ Das Feld *unknown* ist gewöhnlich auf *false* gesetzt (in C und C++: 0), hat aber den Wert *true* (in C und C++: 1), wenn die Originalglyphe nicht in Unicode konvertiert werden konnte und deswegen durch das mit der Option *unknownchar* definierte Zeichen ersetzt wurde. Mit diesem Feld können Sie echten Dokumentinhalt von Ersatzzeichen unterscheiden, wenn Sie ein gängiges Zeichen als *unknownchar* festgelegt haben, zum Beispiel ein Frage- oder Leerzeichen.
- ▶ Das Feld *attributes* enthält den von der TET-Inhaltsanalyse bestimmten Status einer Glyphe: ob es sich z.B. um ein hoch- oder tiefgestelltes Zeichen, einen mehrzeiligen Anfangsbuchstaben oder um Schattentext handelt.
- ▶ Die Felder  $(x, y)$  legen die Position des Referenzpunkts der Glyphe fest, also die untere linke Ecke des Glyphenrechtecks bei horizontaler Schreibrichtung bzw. Mitte oben bei vertikaler Schreibrichtung (siehe Abschnitt 6.4, »Chinesischer, japanischer und koreanischer Text«, Seite 93). Bei künstlichen Zeichen, die keiner Glyphe auf der

Seite entsprechen, ist  $(x, y)$  der Endpunkt des letzten realen Zeichens. Der Wert von  $y$  hängt von der Seitenoption *topdown* ab.

- ▶ Das Feld *width* bestimmt die Breite einer Glyphe unter Berücksichtigung der Parameter für Fontgröße und Textausgabe, wie etwa dem Zeichenabstand oder der horizontalen Skalierung. Da diese Parameter die Position der nächsten Glyphe beeinflussen, weicht der Abstand zwischen den Referenzpunkten zweier benachbarter Glyphen unter Umständen von *width* ab. Die Breite *width* kann gleich 0 sein für Zeichen, die keinen Platz brauchen. Außerdem kann die Glyphe insgesamt breiter sein als der *width*-Wert der Glyphe, zum Beispiel bei schräggestelltem Text. Bei künstlichen Zeichen ist *width* gleich 0.
- ▶ Das Feld *height* in vertikaler Schreibrichtung bestimmt die Höhe der zugehörigen Glyphe unter Berücksichtigung der Parameter für Fontgröße und Textausgabe (z.B. Zeichenabstand). Die Höhe ist im Standardkoordinatensystem positiv, aber im Top-Down-Koordinatensystem negativ. Bei äquidistanten vertikalen Fonts haben alle Glyphen *fontsize* als Höhe, sofern kein zusätzlicher Zeichenabstand angewendet wurde. Künstliche Zeichen (z.B. Separatoren) haben eine Höhe von 0. Bei horizontaler Schreibrichtung wird ein angenäherter Wert für die Glyphenhöhe zurückgeben. Dieser Wert wird aus den Fonteigenschaften berechnet und ist deshalb für alle Glyphen in einem Font identisch. Es ist nicht garantiert, dass die sichtbare Glyphe exakt die gleiche Höhe hat wie der hier übergebene Wert.
- ▶ Der Winkel *alpha* liefert die Laufrichtung des Textes, die als Abweichung von der Standardrichtung definiert ist. Die Standardrichtung ist  $0^\circ$  bei horizontaler Schreibrichtung und  $-90^\circ$  bei vertikaler Schreibrichtung (Einzelheiten zu vertikaler Schreibrichtung finden Sie weiter unten). Der Winkel *alpha* ist deshalb für normalen horizontalen wie vertikalen Text gleich  $0^\circ$ . Die Werte von *alpha* und *beta* hängen von der Seitenoption *topdown* ab.
- ▶ Der Winkel *beta* legt die Neigung (Scherung) eines Textes fest, zum Beispiel für schräggestellten Text. Der Winkel wird gegen die Senkrechte von *alpha* gemessen. Für normalen aufrechten Text beträgt er  $0^\circ$  (sowohl für horizontale als auch vertikale Schreibrichtung). Ist der Absolutbetrag von *beta* größer als  $90^\circ$ , wird der Text an der Grundlinie gespiegelt.
- ▶ Das Feld *fontid* enthält die pCOS-ID des für die Glyphe verwendeten Fonts. Dieser kann zur Abfrage detaillierter Fontinformationen verwendet werden, wie Fontname, Einbettungsstatus, Schreibrichtung (horizontal/vertikal) usw. Die pCOS-Pfadreferenz enthält ein Codebeispiel zur Abfrage von Fontdetails.
- ▶ Das Feld *fontsize* liefert die Textgröße in Punkt. Sie ist normalisiert und deshalb immer positiv, selbst bei *topdown={output}*.
- ▶ Das Feld *colorid* enthält einen Index für die Textfarbe. Er stellt die eindeutige Kombination aus Füllfarbe, Linienfarbe und Textdarstellung (Parameter *textrendering*) dar. Gleiche Kombinationen in einem Dokument erhalten die gleiche Farb-ID. Unterschiedliche Kombinationen werden durch unterschiedliche Farb-IDs dargestellt, die Farben mehrerer Glyphen können also anhand ihrer Farb-IDs auf Gleichheit geprüft werden. Durch den Vergleich der *colorid*-Werte aufeinanderfolgender Glyphen lassen sich zum Beispiel Änderungen in der Textfarbe erkennen. Der genaue Farbraum und die genauen Farbkomponenten für das Füllen und/oder Zeichnen von Text lassen sich mit *TET\_get\_color\_info()* abfragen (siehe Abschnitt 6.3, »Textfarbe«, Seite 91).
- ▶ Das Feld *textrendering* legt die Art der Glyphendarstellung fest, zum Beispiel gefüllt, mit Umrisslinien oder unsichtbar, sowie die eventuelle Verwendung des Textes als Beschneidungspfad. Dieses Feld enthält den numerischen Wert für den Darstel-

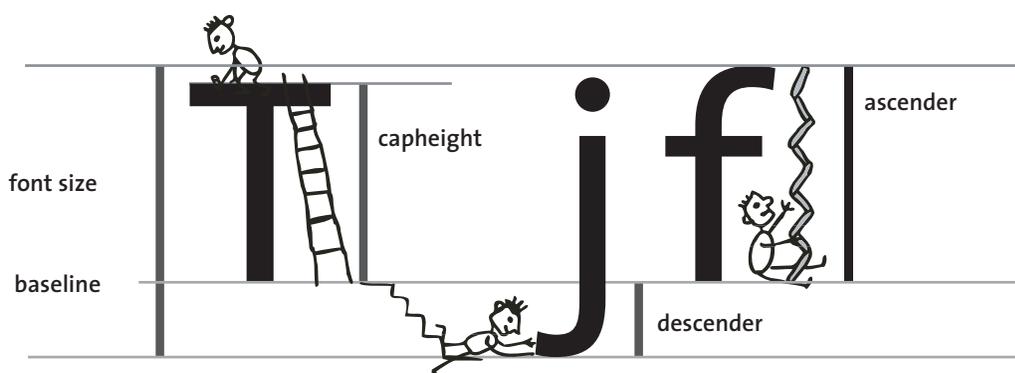


Abb. 6.4 Fontspezifische Metrikwerte

lungsmodus für Text (Parameter *textrendering*), wie er für PDF definiert ist (siehe Tabelle 10.16, Seite 212). Unsichtbarer Text (d.h. *textrendering=3*) wird standardmäßig extrahiert; dies lässt sich mit der Option *ignoreinvisibletext* von *TET\_open\_page()* umstellen.

Text in Type-3-Fonts: *textrendering=3* und *7* ergeben unsichtbaren Text; alle anderen Werte von *textrendering* sind irrelevant und werden ignoriert.

**Fontspezifische Metrikwerte.** TET verwendet dasselbe System für Glyphen- und Fontmetrik wie PostScript und PDF. Es soll hier kurz beschrieben werden.

Die Fontgröße ist der Abstand zwischen zwei aufeinander folgenden Textzeilen, der minimal erforderlich ist, damit Zeichen nicht überlappen. Die Fontgröße ist gewöhnlich höher als die einzelnen Zeichen des Fonts, da sie auch Ober- und Unterlängen und möglicherweise einen Zwischenraum zwischen den Zeilen umfasst.

Die Versalhöhe (*capheight*) bezeichnet die Höhe von Großbuchstaben wie *T* oder *H* in den meisten lateinischen Fonts. Die x-Höhe (*xheight*) bezeichnet die Höhe von Kleinbuchstaben wie *x* in den meisten lateinischen Fonts. Die Oberlänge (*ascender*) bezeichnet die Höhe von Kleinbuchstaben wie *t* oder *d* in den meisten lateinischen Fonts. Die Unterlänge (*descender*) ist der Abstand von der Grundlinie zum unteren Ende von Kleinbuchstaben wie *j* oder *p* in den meisten lateinischen Fonts. Sie ist in der Regel negativ. Die Werte *xheight*, *capheight*, *ascender* und *descender* werden in Tausendstel der Fontgröße gemessen.

Die Werte sind je nach Font unterschiedlich und können mit der pCOS-Schnittstelle abgefragt werden. Mit dem folgenden Beispielcode lassen sich die Werte für Ober- und Unterlänge abfragen:

```
/* Werte für Ober- und Unterlänge abfragen */
path = "fonts[" + i + "]/ascender";
System.out.println("Ascender=" + p.pcos_get_number(doc, path));

path = "fonts[" + i + "]/descender";
System.out.println("Descender=" + p.pcos_get_number(doc, path));
```

Beachten Sie, dass *ascender* und andere Werte für die Fontmetrik nur nach dem Aufruf von *TET\_get\_char\_info()* für eine Glyphen mit diesem Font abgefragt werden sollten. Font-IDs, die von *TET\_get\_char\_info()* zurückgegeben wurden, können also problemlos

verwendet werden, das Auflisten aller Fonts im Array `fonts[]` liefert jedoch nicht unbedingt Metrikwerte aus eingebetteten Fontdaten, sondern die eventuell ungenauen Werte aus dem PDF-Dictionary `FontDescriptor`. Für weitere Informationen siehe die pCOS-Pfadreferenz.

**Endpunkte von Glyphen und Wörtern.** Um den Endpunkt eines Wortes (z.B. zur Hervorhebung) zu berechnen, benötigen Sie den Endpunkt des letzten Zeichens im Wort. Mit den von `TET_get_char_info()` zurückgegebenen Werten `x`, `y`, `width` und `alpha` können Sie bei horizontaler Schreibrichtung den Endpunkt einer Glyphe, d.h. den Fortschaltungsvektor der Glyphe (die untere rechte Ecke der Glyphenbox) wie folgt bestimmen:

$$\begin{aligned}x_{\text{end}} &= lr_x = x + \text{width} * \cos(\alpha) \\y_{\text{end}} &= lr_y = y + \text{width} * \sin(\alpha)\end{aligned}$$

Bei Text in üblicher horizontaler Schreibrichtung (d.h.  $\alpha=0$ ) verkürzen sich die Terme zu

$$\begin{aligned}x_{\text{end}} &= lr_x = x + \text{width} \\y_{\text{end}} &= lr_y = y\end{aligned}$$

Generell lässt sich die Größe der Glyphenbox durch Bestimmung der Koordinaten der oberen rechten Ecke berechnen (wenn  $\beta=0$ , d.h. bei dieser Formel wird die Scherung einer Glyphe nicht berücksichtigt):

$$\begin{aligned}ur_x &= x + \text{width} * \cos(\alpha) - \text{dir} * \text{height} * \sin(\alpha) \\ur_y &= y + \text{width} * \sin(\alpha) + \text{dir} * \text{height} * \cos(\alpha)\end{aligned}$$

mit  $\text{dir}=1$  im Standardfall `topdown={output=false}` und  $\text{dir}=-1$  bei `topdown={output=true}` (siehe »Top-Down-Koordinatensystem«, Seite 84). Der Wert von `height` hängt von Fontgröße und -geometrie ab. Das folgende Beispiel ergibt für die meisten Fonts sinnvolle Werte (für die Abfrage von `ascender`-Werten siehe »Fontspezifische Metrikwerte«, Seite 88):

$$\text{height} = \text{fontsize} * \text{ascender} / 1000$$

In vielen grafischen Entwicklungsumgebungen können die Glyphen-Transformationen wie folgt ausgedrückt werden:

```
translate(x, y);
rotate(alpha);
skew(0, -beta);
if (abs(beta) > 90)
    scale(1, -1);
```

Nach Anwendung dieser Transformationen kann die rechte obere Ecke der Glyphenbox wie folgt ausgedrückt werden:

$$\begin{aligned}ur_x &= x + \text{width} \\ur_y &= y + \text{dir} * \text{height}\end{aligned}$$

**Glyphenberechnung für vertikale Schreibrichtung.** Für Text mit vertikaler Schreibrichtung können Sie den Endpunkt wie folgt berechnen:

$$\begin{aligned}x_{\text{end}} &= x \\y_{\text{end}} &= y - \text{height}\end{aligned}$$

Die rechte obere und linke untere Ecke der Glyphenbox lassen sich wie folgt berechnen (wenn  $\beta=0$ ):

$$ul_x = x - \text{width}/2 * \cos(\alpha)$$

$$ul_y = y - \text{width}/2 * \sin(\alpha)$$

$$lr_x = ul_x + \text{width} * \cos(\alpha) + \text{dir} * \text{height} * \sin(\alpha)$$

$$lr_y = ul_y + \text{width} * \sin(\alpha) - \text{dir} * \text{height} * \cos(\alpha)$$

mit  $\text{dir}=1$  im Standardfall  $\text{topdown}=\{\text{output}=\text{false}\}$  und  $\text{dir}=-1$  bei  $\text{topdown}=\{\text{output}=\text{true}\}$  (siehe »Top-Down-Koordinatensystem«, Seite 84).

## 6.3 Textfarbe

Mit Hilfe der von `TET_get_char_info()` zurückgegebenen Farb-ID lassen sich Füll- und/oder Linienfarbe der Glyphen abfragen, die zu einem ausgegebenen Zeichen gehört. Dies kann mit `TET_get_color_info()` erreicht werden, das die folgenden Werte für eine Farb-ID zurückgibt. Diese Werte können für die Füll- und Linienfarbe einer Glyphen separat abgefragt werden:

- ▶ Das Feld `colorspaceid` enthält den Index des Farbraums im Pseudo-Objekt `colorspaces[]` (siehe die pCOS-Pfadreferenz) oder -1, sofern auf die Glyphen keine Farbe angewendet wird.
- ▶ Das Feld `patternid` enthält den Index des Patterns (Füllmuster) im Pseudo-Objekt `pattern[]` (siehe die pCOS-Pfadreferenz) oder -1, sofern auf die Glyphen kein Pattern angewendet wird.
- ▶ Das Array `components` enthält die Farbwerte, die in dem Farbraum interpretiert werden, der mit `colorspaceid` ausgegeben wird.
- ▶ Das Feld `n` (nur in den Sprachbindungen C und C++ verfügbar) enthält die Anzahl der relevanten Einträge im Feld `components` (nur in den Sprachbindungen C und C++).

Im Minibeispiel `glyphinfo` wird gezeigt, wie die Farbwerte interpretiert werden, die von `TET_get_color_info()` zurückgegeben werden und wie diese Information mit allgemeinen, von pCOS zurückgegebenen Farbraum-Attributen erweitert werden kann. Die Topics `colorspaces` und `page_colors` im pCOS Cookbook zeigen, wie noch detailliertere Informationen extrahiert werden können, wie zum Beispiel `WhitePoint` für kalibrierte Farbräume oder der alternative Farbraum eines Farbraums vom Typ `Separation` oder `DeviceN`.

Das Zeichnen von Text, d.h. das Zeichnen der Umrisslinien einer Glyphen (im Gegensatz zum Füllen des Inneren) wird in PDF-Dokumenten nur selten verwendet. Die meisten Anwendungen können die Angaben zur Linienfarbe ignorieren. Ebenso werden Patterns nur selten im Text verwendet.

Die Bestimmung der Textfarbe lässt sich mit der folgenden Dokumentoption deaktivieren:

```
engines={notextcolor}
```

Ist die Bestimmung von Textfarbe deaktiviert, darf das Feld `colorid` von `TET_char_info` nicht verwendet werden, da es keinen sinnvollen Wert enthält.

Tabelle 6.1 gibt einen Überblick über die PDF-Farbräume. Sofern nicht anders angegeben, liegen die Farbwerte im Bereich 0...1.

Tabelle 6.1 Farbräume in PDF

Farbraum	Anzahl der Farbkomponenten	Hinweise
<b>Geräteabhängige Farbräume</b>		
DeviceGray	1	Die gerätespezifischen Farbräume sind weit verbreitet, sind aber geräteabhängig und stellen Farbinformationen daher nicht zuverlässig dar.
DeviceRGB	3	
DeviceCMYK	4	
<b>CIE-basierte (geräteunabhängige) Farbräume</b>		
ICCBased	1, 3 oder 4	Farbräume vom Typ ICCBased werden für Graustufen-, RGB- oder CMYK-Farben durch ein ICC-Profil definiert.

Tabelle 6.1 Farbräume in PDF

Farbraum	Anzahl der Farbkomponenten	Hinweise
Lab	3	Farbräume vom Typ Lab werden durch einen Farbraum vom Typ CIE 1976 L*a*b* definiert. Sie benötigen einen Helligkeitswert (Luminance) zwischen 0...100 und zwei Farbwerte, die oft im Bereich -128...127 liegen.
CalGray	1	Kalibrierte Farbräume definieren einen WhitePoint und optional einen BlackPoint. Sie werden heutzutage selten verwendet, da ICC-basierte Farbräume flexibler sind.
CalRGB	3	
<b>Spezielle Farbräume</b>		
Pattern	<ul style="list-style-type: none"> <li>o (PaintType=1)</li> <li>N (PaintType=2)</li> <li>o (PatternType=2)</li> </ul>	Mit Pattern-Farbräumen können statt flächiger Farbe grafische Muster angewendet werden. Kachel-Pattern (PatternType=1) tragen grafische Formen wiederholt über den zu füllenden Bereich auf, wobei die Form intrinsisch gefärbt (PaintType=1) oder ungefärbt sein kann und wie eine Stencil-Maske externe Farbe benötigt (PaintType=2). Pattern für einen Farbverlauf (PatternType=2) wenden statt einer Volltonfarbe einen Farbverlauf an.
Separation	1	Ein Separation-Farbraum beschreibt eine benannte Schmuckfarbe und benötigt einen Alternativ-Farbraum, falls die benannte Schmuckfarbe für die Ausgabe nicht verfügbar ist.
DeviceN	N	DeviceN ist eine Verallgemeinerung des Separation-Farbraums für mehr als eine benannte Schmuckfarbe. DeviceN wird auch verwendet, um eine Teilmenge von CMYK-Prozessfarben anzuwenden.
Indexed	1, aber N im Basis-Farbraum	Mit Indexed-Farbräumen lässt sich eine kleine Anzahl verschiedener Farbwerte (bis zu 256) effizient speichern. Sie benötigen einen zugrunde liegenden Basis-Farbraum.

## 6.4 Chinesischer, japanischer und koreanischer Text

### 6.4.1 CJK-Encodings und CMaps

TET unterstützt chinesischen, japanischen und koreanischen (CJK) Text und konvertiert horizontalen und vertikalen CJK-Text in beliebigen Encodings (CMaps) nach Unicode.

TET unterstützt alle CJK-Zeichenkollektionen von Adobe (*CJK character collections*):

- ▶ Vereinfachtes Chinesisch: *Adobe-GB1-5*
- ▶ Traditionelles Chinesisch: *Adobe-CNS1-6*
- ▶ Japanisch: *Adobe-Japan1-6*
- ▶ Koreanisch: *Adobe-Korea1-2*

Die PDF-CMaps wiederum berücksichtigen alle CJK-Encodings, die gegenwärtig im Einsatz sind, zum Beispiel Shift-JIS, EUC, Big-5, KSC und viele andere. CJK-Fontnamen in sprachabhängigen Encodings (zum Beispiel japanische Fontnamen in Shift-JIS) werden ebenfalls in Unicode dargestellt.

*Hinweis* Zur Extraktion von CJK-Text in nicht Unicode-Encodings müssen Sie den Zugriff auf die CMap-Dateien konfigurieren, die mit TET ausgeliefert werden (siehe Abschnitt 0.1, »Installation der Software«, Seite 7).

### 6.4.2 Wortgrenzen für CJK-Text

Ideografische Zeichen stellen keine Wortgrenzen dar, Satzzeichen und der Übergang zwischen einem ideografischen und nicht ideografischen Zeichen hingegen schon. Bei *granularity=word* stellen das ideografische Komma *U+3001* und der ideografische Punkt *U+3002* ebenfalls Wortgrenzen dar. Bei *granularity=page* wird am Zeilenende kein Zeilenseparator eingefügt.

*Hinweis* Dieses Verhalten hat sich mit TET 5 geändert. In TET 4 wurden ideografische Zeichen immer als Wortgrenzen behandelt.

### 6.4.3 Vertikale Schreibrichtung

TET unterstützt horizontale wie vertikale Schreibrichtung und führt alle Metrikberechnungen entsprechend der jeweiligen Schreibrichtung durch. Bei Text in vertikaler Schreibrichtung sollten Sie folgendes beachten:

- ▶ Bei vertikaler Schreibrichtung befindet sich der Referenzpunkt der Glyphe in der Mitte oben. Die Textposition bewegt sich gemäß Glyphenhöhe nach unten, wobei die Breite der Glyphe keine Rolle spielt (siehe Abbildung 6.3).
- ▶ Der Winkel *alpha* ist bei normalem vertikalen Text gleich  $0^\circ$ . Das heißt, dass in vertikaler Schreibrichtung Fonts mit *alpha=0°* nach unten, also in Richtung  $-90^\circ$  verlaufen.
- ▶ Wegen der oben erwähnten Unterschiede muss der Client-Code die Schreibrichtung berücksichtigen und sie mit dem folgenden pCOS-Code ermitteln. Beachten Sie dabei, dass nicht jeder Text, der vertikal verläuft, einen Font in vertikaler Schreibrichtung verwendet:

```
count = p.pcos_get_number(doc, "length:fonts");
for (i=0; i < count; i++)
{
```

```

if (p.pcos_get_number(doc, "fonts[" + id + "]/vertical"))
{
    /* Font verwendet vertikale Schreibrichtung */
    vertical = true;
}
}

```

- ▶ Vorrotierte Glyphen für vertikalen Text und Satzzeichen werden in entsprechende nicht gedrehte Unicode-Zeichen umgewandelt. Mit der folgenden Dokumentoption lassen sich vorrotierte Zeichen erhalten:

```
decompose={vertical=_none}
```

#### 6.4.4 CJK-Dekomposition: Narrow, wide, vertical usw.

Unicode und viele andere Encodings unterstützen Zeichen voller oder halber Breite (auch Double- oder Single-Byte-Zeichen genannt). Standardmäßig wendet TET die Unicode-Dekompositionen *wide* und *narrow* an, die Zeichen voller und halber Breite durch die entsprechenden Standardbreiten ersetzen.

Um die ursprünglichen Zeichen voller und halber Breite beizubehalten, können Sie mit der Dokumentoption *decompose* die entsprechenden Dekompositionen deaktivieren:

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

Ähnlich wirken sich die Dekompositionen *small*, *square* und *vertical* auch auf CJK-Zeichen aus. Da standardmäßig all diese Dekompositionen aktiviert sind, einschließlich *wide* und *narrow*, werden die Zeichen in ihre Normalform konvertiert. Um die ursprünglichen Zeichen beizubehalten, deaktivieren Sie die entsprechenden Dekompositionen. Mit der folgenden Dokumentoption werden alle Dekompositionen deaktiviert:

```
decompose={none}
```

Tabelle 6.2 führt die CJK-Dekompositionen mit Beispielen auf. Für weitere Informationen zu Dekompositionen siehe Abschnitt 7.3, »Unicode-Nachbearbeitung«, Seite 112.

Tabelle 6.2 Beispiel für CJK-Kompatibilitätsdekomposition (Unteroptionen für die Option decompose)

Dekomposition	Beschreibung	betroffene Unicode-Zeichen	bei aktivierter Dekomposition (Standard)	bei deaktivierter Dekomposition
<b>narrow</b>	Schmale (hankaku) Kompatibilitätsformen	U+FF61-U+FFDC, U+FFE8-U+FFEE	 U+30F2	 U+FF66
<b>small</b>	Kleine Formen für die Kompatibilität zu CNS 11643	U+FE50-U+FE6B	 U+002C	 U+FE50
<b>square</b>	eckige CJK-Fontvarianten	U+3250, U+32CC-U+32CF, U+3300-U+3357, U+3371-U+33DF, U+337B-U+337F, U+33FF, U+1F131-U+1F14E, U+1F190, U+1F200, U+1F210-U+1F231	  U+30AD U+30ED	  U+3314
<b>vertical</b>	Vertikale Layout-Präsentationsformen	U+309F, U+30FF, U+FE10-U+FE19 U+FE30-U+FE48	 U+FE37	 U+007B
<b>wide</b>	Breite (zenkaku) Kompatibilitätsformen	U+3000, U+FF01-U+FF60, U+FFE0-U+FFE6	 U+00A3	 U+FFE1

## 6.5 Bidirektionaler arabischer und hebräischer Text

Um Text aus Dokumenten mit linksläufigen Schriftsystemen wie Arabisch und Hebräisch zu extrahieren, wendet TET zusätzliche Verarbeitungsschritte an. Da solche Schriftsysteme häufig rechtsläufige Texteneinschübe (z.B. Zahlen) enthalten, nennt man solche Dokumente bidirektional. Das Extrahieren von bidirektionalem Text umfasst einen oder mehrere der unten aufgeführten Verarbeitungsschritte.

### 6.5.1 Allgemeine Bidi-Themen

**Umsortieren von linksläufigem und bidirektionalem Text.** Linksläufige und rechtsläufige Sequenzen müssen umsortiert werden, um die korrekte logische Reihenfolge des Textes auszudrücken. Bei Wort-Granularität oder höher liefert TET Text in logischer Reihenfolge mit der folgenden Seitenoption (Standardwert):

```
contentanalysis={bidi=logical}
```

Die Bidi-Verarbeitung lässt sich mit der folgenden Seitenoption deaktivieren:

```
contentanalysis={bidi=visual}
```

**Ermitteln der dominanten Schreibrichtung des Textes auf der Seite.** Außer den Zeichen innerhalb eines Worts und den Wörtern innerhalb einer Zeile sind auch andere Aspekte der Seitenlayout-Erkennung von der Bidi-Umsortierung betroffen. In einigen Fällen können gemischte Bidi-Zeilen nur korrekt sortiert werden, wenn berücksichtigt wird, dass die Seite hauptsächlich links- oder rechtsläufigen Text enthält. Dazu prüft TET die Haupt-Schreibrichtung des Textes auf der Seite und passt seine Algorithmen automatisch entsprechend an.

Dieses Verhalten lässt sich mit der Option *bidilevel* ändern. Die folgende Option erzwingt zum Beispiel linksläufige Verarbeitung, auch wenn der Großteil des Textes auf der Seite rechtsläufig ist:

```
contentanalysis={bidilevel=rtl}
```

**Glyphen-Reihenfolge.** In TETML werden die von *TET\_get\_char\_info()* und den Elementen *Glyph* zurückgegebenen Glyphen-Informationen immer in visueller Reihenfolge angeordnet, d.h. von links nach rechts für horizontale Grundlinien. Diese rechtsläufige Glyphen-Anordnung sorgt dafür, dass Client-Anwendungen die Glyphen-Koordinaten in deterministischer Reihenfolge erhalten, ohne den Bidi-Status des Textes überprüfen zu müssen. Dieses Verhalten spiegelt die Tatsache wider, dass die Glyphen in arabischen und hebräischen Fonts den Referenzpunkt in der Regel am linken Rand haben und nach rechts verlaufen, obwohl die Textrichtung eigentlich linksläufig ist.

### 6.5.2 Nachbearbeitung von arabischem Text

**Normalisierung arabischer Präsentationsformen und Dekomposition von Ligaturen.**

Arabische Zeichen nehmen bis zu vier verschiedene Zeichenformen an je nach Position innerhalb eines Worts: isoliert, am Anfang, in der Mitte oder am Ende. Auch wenn sie semantisch das selbe Zeichen repräsentieren, haben diese Zeichenformen unterschiedliche Unicode-Werte. TET konvertiert standardmäßig alle Präsentationsformen in die

entsprechenden kanonischen Formen. Wie Tabelle 6.3 zeigt, können die Präsentationsformen mit der Option *decompose* explizit erhalten werden (siehe Abschnitt 7.3.2, »Unicode-Dekomposition«, Seite 116).

Da das PDF-Dokument Präsentationsformen entweder dem isolierten Unicode-Zeichen oder einer der Präsentationsformen (z.B. in der ToUnicode-CMap des Dokuments) zuordnen kann, kann die Ausgabe unter Umständen selbst dann keine Präsentationsformen enthalten, wenn die Dekompositionen deaktiviert sind.

Tabelle 6.3 Verarbeitung arabischer Präsentationsformen mit der Option *decompose*

Beschreibung und Optionsliste	vor der Dekomposition	nach der Dekomposition (in logischer Reihenfolge)
Dekomposition von isolierten, am Anfang, in der Mitte oder am Ende eines Worts positionierten Präsentationsformen: Option <i>decompose</i> nicht angegeben (Standardwert) oder <i>decompose=</i> {final=_all medial=_all initial=_all isolated=_all}	س U+FEB2	س U+0633
	س U+FEB3	س U+0633
Beachten Sie, dass Ligaturen nur zerlegt werden, wenn sie tatsächlich durch eine Ligaturglyphe dargestellt werden. Wenn mehrere separate Glyphen verwendet werden, werden diese in der Ausgabe zurückgehalten.	سر U+FD0E	س ر U+0633 U+0631
	س U+FEB4	س U+0633
	لا U+FEFC	ل ا U+0644 U+0627
	ل ا U+0644 U+0627	ل ا U+0644 U+0627
Beibehalten von isolierten, am Anfang, in der Mitte oder am Ende eines Worts positionierten Präsentationsformen:	س U+FEB2	س U+FEB2
<i>decompose=</i> {final=_none medial=_none initial=_none isolated=_none} oder	س U+FEB3	س U+FEB3
<i>decompose=none</i>	سر U+FD0E	سر U+FD0E
	س U+FEB4	س U+FEB4
	لا U+FEFC	لا U+FEFC

**Entfernen arabischer Tatweel-Zeichen.** Das Tatweel-Zeichen U+0640 (auch *Kashida* genannt) wird im Arabischen häufig zur Streckung von Wörtern verwendet, damit sie die Zeile komplett ausfüllen. Da Tatweel selbst keine Textinformation transportiert, wird es im extrahierten Text nicht benötigt. Deshalb werden Tatweel-Zeichen in TET standardmäßig aus dem extrahierten Text entfernt. Wie Tabelle 6.4 zeigt, können Tatweel-Zeichen mit der Option *fold* explizit erhalten werden (siehe Abschnitt 7.3.1, »Unicode-Folding«, Seite 112).

Tabelle 6.4 Verarbeitung des Tatweel-Zeichens U+0640 mit der Option fold

Beschreibung und Optionsliste	vor dem Folding	nach dem Folding
Entfernen arabischer Tatweel-Zeichen: Option fold nicht angegeben (Standardwert) oder fold={{[U+0640] remove}} oder fold={default}	- U+0640	n/a
Erhalten von arabischen Tatweel-Zeichen (die standardmäßig entfernt werden): fold={{[U+0640] preserve}}	- U+0640	- U+0640

## 6.6 Inhaltsanalyse

PDF-Dokumente liefern die Semantik (Unicode-Zuordnung) einzelner Textzeichen sowie ihre Position auf der Seite. Sie enthalten aber normalerweise keine Angaben zu Wörtern, Zeilen, Spalten oder anderen Texteinheiten. Der Text auf der Seite setzt sich aus Fragmenten zusammen, die aus einzelnen Zeichen, Silben, Wörtern, Zeilen oder einer beliebigen Mischung daraus bestehen, ohne Anfang oder Ende eines Worts, einer Zeile oder einer Spalte explizit zu markieren.

Um die Sache weiter zu verkomplizieren, unterscheidet sich die Reihenfolge der Textfragmente häufig von der logischen Lesereihenfolge. Es gibt keine Regeln für die Reihenfolge der Platzierung von Textfragmenten auf der Seite. Beim Anlegen einer Seite mit zwei Textspalten könnte zuerst die erste Zeile der linken Spalte, dann die erste Zeile der rechten Spalte, dann die zweite Zeile der linken Spalte und dann die zweite Zeile der rechten Spalte etc. generiert werden. Zur Verarbeitung in logischer Reihenfolge benötigt man jedoch erst den kompletten Text der linken Spalte und dann den Text der rechten Spalte. Wenn man den Text aus solchen Dokumenten extrahiert und dabei einfach die Instruktionen auf der PDF-Seite nachvollzieht, so erhält man in der Regel unbrauchbare Ergebnisse, da die logische Struktur des Textes völlig verloren geht.

TET analysiert den Inhalt, die Position und die Beziehungen von Textfragmenten mit folgenden Zielen:

- ▶ Wörter aus Zeichen zu erzeugen und Leerzeichen zwischen Wörtern einzufügen, sofern erwünscht;
- ▶ Redundanten Text zu entfernen, zum Beispiel doppelten Text für Schatteneffekte
- ▶ Die Teile eines getrennten Worts, das sich über mehrere Zeilen erstreckt, zusammenzufügen;
- ▶ Textspalten (Zonen) zu identifizieren;
- ▶ Textfragmente innerhalb einer Zone sowie Zonen auf der Seite zu sortieren.

Diese Operationen sowie die Optionen zur Steuerung der Inhaltsanalyse werden im folgenden ausführlich beschrieben.

**Granularität des Textes.** Mit der Option *granularity* von `TET_open_page()` können Sie die Textmenge festlegen, die bei einem Aufruf von `TET_get_text()` zurückgegeben wird:

- ▶ Bei *granularity=glyph* enthält jedes Fragment die Unicode-Entsprechung einer einzelnen Glyphe. Dabei kann es sich um ein oder mehrere Zeichen handeln (z.B. bei Ligaturen). In diesem Modus findet keine Inhaltsanalyse statt. TET gibt die Textfragmente auf der Seite unverändert und in ihrer ursprünglichen Reihenfolge zurück. Dies ist zwar der schnellste Modus, aber nur dann sinnvoll, wenn der TET-Client über komplexe Nachbearbeitungsverfahren verfügt (oder sich nur für die Textposition, nicht aber für die logische Textstruktur interessiert), da der Text über die ganze Seite verstreut sein kann.
- ▶ Bei *granularity=word* fasst der Wordfinder-Algorithmus Zeichen zu logischen Wörtern zusammen. Jedes Fragment enthält dabei ein Wort. Einzelne Satzzeichen (wie Komma, Doppelpunkt, Frage- oder Anführungszeichen) werden als separate Fragmente zurückgegeben. Folgen von mehreren Satzzeichen werden zu einem einzelnen Wort gruppiert (zum Beispiel mehrere Punkte, die eine gepunktete Linie simulieren sollen). Die Verarbeitung von Satzzeichen lässt sich auch umstellen (siehe »Erkennen von Wortgrenzen in westlichem Text«, Seite 100).

- ▶ Bei *granularity=line* werden die vom Wordfinder ermittelten Wörter zu Zeilen zusammengefasst. Ist die Enttrennung (*dehyphenation*) aktiviert (was standardmäßig der Fall ist), werden die Bestandteile eines am Zeilenende getrennten Wortes zusammengefasst und das enttrennte Wort vollständig in die Zeile aufgenommen.
- ▶ Bei *granularity=page* werden alle Wörter auf der Seite in einem einzigen Fragment zurückgegeben.

Zwischen Wörtern, Zeilen oder Absätzen werden Trennzeichen eingefügt, wenn die gewählte Granularität größer als die jeweilige Einheit ist. Bei *granularity=word* zum Beispiel brauchen keine Worttrennzeichen eingefügt zu werden, da bei jedem Aufruf von *TET.get\_text()* genau ein Wort zurückgegeben wird.

Mit den Optionen *wordseparator*, *lineseparator* und *paraseparator* von *TET.open\_document()* lassen sich Trennzeichen festlegen (mit U+0000 deaktivieren Sie ein Trennzeichen), zum Beispiel:

```
lineseparator=U+000A
```

Alle Operationen zur Inhaltsanalyse sind bei *granularity=glyph* deaktiviert und für alle anderen Granularitätsmodi aktiviert. Über zusätzliche Optionen lässt sich die Verarbeitung weiter differenzieren (siehe unten).

**Erkennen von Wortgrenzen in westlichem Text.** Der Wordfinder, der in allen Granularitätsmodi außer *glyph* aktiv ist, generiert logische Wörter aus Glyphen, die in unbestimmter Anordnung über die ganze Seite verteilt sein können. Wortgrenzen werden nach zwei Kriterien festgelegt:

- ▶ Ein komplexer Algorithmus analysiert die geometrischen Beziehungen zwischen den Glyphen, um Zeichengruppen zu ermitteln, die zusammen ein Wort ergeben. Der Algorithmus berücksichtigt dabei eine Vielzahl von Eigenschaften und Sonderfällen, um Wörter selbst dann korrekt zu identifizieren, wenn das Layout unübersichtlich und der Text beliebig auf der Seite angeordnet ist. Mit der Unteroption *usemetrics* der Seitenoption *contentanalysis* kann dieser Algorithmus für besondere Fälle deaktiviert werden.
- ▶ Manche Zeichen wie Leer- oder Satzzeichen (z.B. Doppelpunkt, Komma, Punkt oder Klammern) werden immer als Wortgrenze angesehen, unabhängig von ihrer Breite oder Position. Mit der Unteroption *useclasses* der Seitenoption *contentanalysis* kann dieser Algorithmus für besondere Fälle deaktiviert werden.

Satzzeichen als Wortgrenze zu ignorieren kann zum Beispiel beim Erhalt von Web-URLs sinnvoll sein, bei denen Punkt und Schrägstrich in der Regel als Bestandteil eines Wortes gelten (siehe Abbildung 6.5). Ist die Seitenoption *punctuationbreaks* auf *false* gesetzt, behandelt der Wordfinder Satzzeichen nicht mehr als Wortgrenzen:

```
contentanalysis={punctuationbreaks=false}
```

*Hinweis* Das Erkennen von Wortgrenzen für Text mit ideografischen Zeichen funktioniert anders; für weitere Informationen siehe Abschnitt 6.4.2, »Wortgrenzen für CJK-Text«, Seite 93.

www.pdfli**b**.com

www.pdfli**lib**.com

Abb. 6.5  
Der Standardwert `punctuationbreaks = true` trennt die Bestandteile von URLs (oben), während `punctuationbreaks = false` sie zusammenhält (unten).

### Enttrennung (Dehyphenation).

Wörter, die am Zeilenende getrennt wurden, sind wenig sinnvoll für Anwendungen, die den extrahierten Text auf logischer Ebene weiterverarbeiten. Deshalb enttrennt TET die Bestandteile eines getrennten Wortes oder setzt sie wieder zusammen. Endet ein Wort am Zeilenende mit einem Trennzeichen und beginnt das erste Wort der nächsten Zeile mit einem Kleinbuchstaben, wird das Trennzeichen entfernt und der erste Teil des Wortes mit dem zwei-

ten Teil in der Folgezeile kombiniert. Dies setzt allerdings voraus, dass sich eine Folgezeile in derselben Zone befindet. Bindestriche bzw. Gedankenstriche werden im Gegensatz zu Trennzeichen unverändert gelassen. Die Bestandteile eines getrennten Wortes werden dabei nicht geändert, es wird lediglich das Trennzeichen entfernt. Die Enttrennung lässt sich mit der folgenden Optionsliste von `TET_open_page()` deaktivieren:

```
contentanalysis={dehyphenate=false}
```

**Entfernen von Schattentext und künstlicher Fettschrift.** PDF-Dokumente enthalten manchmal redundanten Text, der nicht zur Semantik der Seite beiträgt, sondern nur visuelle Effekte erzeugt. Schattentext erreicht man zum Beispiel meist dadurch, dass ein und derselbe Text leicht verschoben mehrfach übereinander gelegt wird. Färbt man den Text ein, bleiben die unteren Schichten größtenteils verborgen, aber die sichtbaren Bereiche erzeugen einen Schatteneffekt.

Textverarbeitungsprogramme bieten manchmal eine ähnliche Funktion für

künstliche Fettschrift. Um fett ausgezeichneten Text zu erstellen, wenn kein fetter Fontschnitt vorhanden ist, wird der Text mehrfach in derselben Farbe auf der Seite platziert. Durch eine leichte Verschiebung wird der Eindruck von Fettschrift erzeugt.

Redundante Textinhalte wie simulierte Schatteneffekte, künstliche Fettschrift und ähnliche visuelle Artefakte können bei der Weiterverarbeitung von extrahiertem Text

strategische Grundsätze – der  
t der Nutzung von Synergie-  
in Branchen sowie in Unter-  
lukterstellung. So verringert  
t bei der Produkterstellung –  
g – seit längerem nicht nur

# Introduction

ernste Probleme verursachen, da sie normal verarbeitet werden, obwohl sie nur zur visuellen Darstellung und nicht zum Seiteninhalt beitragen.

Ist der Wordfinder aktiviert, erkennt und entfernt TET redundante visuelle Artefakte automatisch. Dieses Verhalten lässt sich mit der folgenden Optionsliste von `TET_open_page()` deaktivieren:

```
contentanalysis={shadowdetect=false}
```

**Buchstaben mit Akzent.** In vielen Sprachen werden Akzente und andere diakritische Zeichen nah an anderen Zeichen platziert, um kombinierte Zeichen zu bilden. Einige Satzprogramme, vor allem TeX, geben zwei getrennte Zeichen aus (Basiszeichen und Akzent), um ein kombiniertes Zeichen zu erzeugen. Um zum Beispiel das Zeichen *ä* zu erzeugen, wird zuerst der Buchstabe *a* auf der Seite platziert und dann die Umlautpunkte " darüber gesetzt. TET erkennt dies und kombiniert die beiden Zeichen zu dem entsprechenden Buchstaben mit Akzent.

## 6.7 Layout-Analyse

TET analysiert das Textlayout auf der Seite und ermittelt die bestmögliche Reihenfolge für die Textextraktion. Dieser Vorgang kann mit verschiedenen Optionen unterstützt werden. Sie können die Ergebnisse der Textextraktion mit geeigneten Optionen verbessern, wenn Sie wissen, um welche Art von Dokumenten es sich handelt.

**Dokumentstile.** Zur Verarbeitung unterschiedlicher Layouts und Stile stehen mehrere interne Parameter zur Verfügung. Zeitungsseiten enthalten meist viel Text in mehreren Spalten, Business Reports enthalten dagegen oft Kommentare und Marginalien usw. TET bietet vordefinierte Einstellungen für verschiedene Arten von Dokumenten. Diese Einstellungen lassen sich mit einer Option von `TET_open_page()` aktivieren:

```
docstyle=papers
```

Ist der Dokumenttyp bekannt, empfehlen wir dringend, geeignete Werte für die Seitenoption `docstyle` und gegebenenfalls auch für die Seitenoption `layouthint` zu übergeben. Mit der Option `docstyle` wird ein erweiterter Algorithmus zur Layout-Erkennung aktiviert. Dabei kann ein ungeeigneter Wert für diese Option allerdings zu schlechteren Ergebnissen führen.

Folgende Dokumenttypen sind für die Option `docstyle` verfügbar (Tabelle 6.5 listet typische Beispiele für einige Dokumenttypen auf):

- ▶ *Book*: typisches Buch mit regulären Seiten
- ▶ *Business*: Unternehmensdokumente
- ▶ *Cad*: typischerweise stark fragmentierte technische oder architektonische Zeichnungen
- ▶ *Fancy*: Seiten mit komplexem und manchmal irregulärem Layout
- ▶ *Forms*: strukturierte Formulare
- ▶ *Generic*: allgemeinste Dokumentklasse ohne weitere Besonderheiten
- ▶ *Magazines*: Zeitschriftenartikel, meist mit drei oder mehr Spalten und vereinzelt Bildern und Grafiken
- ▶ *Papers*: Zeitungen mit vielen Spalten, großen Seiten und kleinen Fonts
- ▶ *Science*: wissenschaftliche Artikel, meist mit zwei oder mehr Spalten und vereinzelt Bildern, Formeln und Tabellen usw.
- ▶ *Search engine*: Diese Klasse bezieht sich nicht auf einen bestimmten Typ von Eingabedokumenten, sondern optimiert TET vielmehr für die typischen Anforderungen von Suchmaschinen-Indexern. Einige Funktionen zur Layout-Erkennung sind deaktiviert und liefern nur Rohtext, um die Verarbeitungsgeschwindigkeit zu erhöhen. Beispielsweise ist die Erkennung von Tabellen- und Seitenstruktur deaktiviert.
- ▶ *Space grid*: Diese Klasse ist für listenartige Berichte bestimmt, die oft auf Mainframe-Systemen generiert werden. Bei dieser Dokumentklasse wird das visuelle Layout mit Leerzeichen und nicht durch explizite Positionierung des Textes erzeugt. Da viele Verarbeitungsschritte wie Schatten-Erkennung für diese Klasse von Dokumenten übersprungen werden, kann die Textextraktion mit dieser Option beschleunigt werden.

Durch Angabe des am besten geeigneten Dokumentstils können die Verarbeitungsgeschwindigkeit und die Ergebnisse der Textextraktion erheblich verbessert werden.

Tabelle 6.5 Dokumentstile

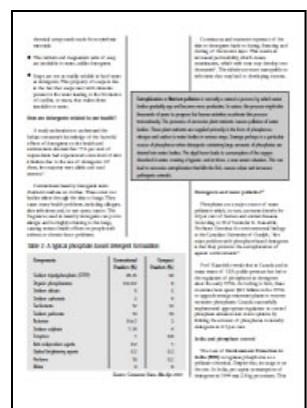
docstyle=book



docstyle=business



docstyle=fancy



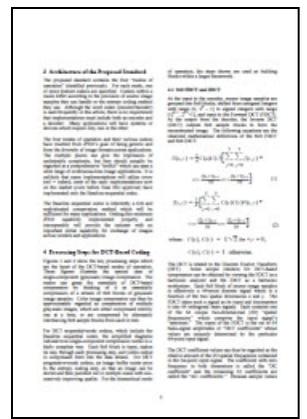
docstyle=magazines



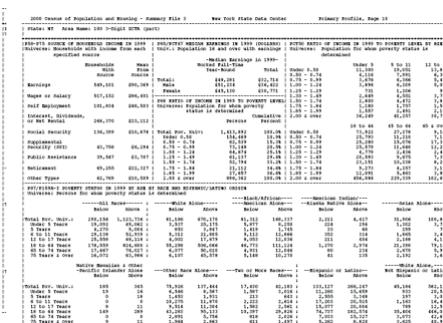
docstyle=papers



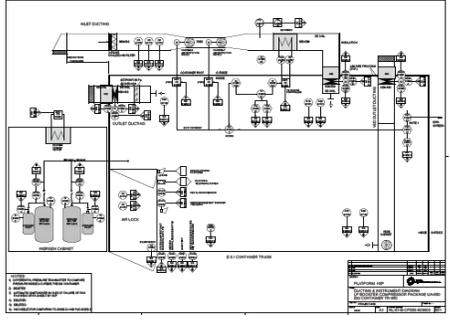
docstyle=science



docstyle=spacegrid



docstyle=cad



**Komplexe Layouts.** Manche Dokumentklassen verwenden oft ein recht komplexes Seitenlayout. Bei Zeitschriften und Illustrierten kann TET zum Beispiel die Beziehung zwischen den Spalten auf der Seite eventuell nicht richtig erkennen. In solchen Fällen ist es möglich, den extrahierten Text auf Kosten der Verarbeitungszeit zu verbessern. Dies lässt sich mit den Seitenoptionen *structureanalysis* und *layoutanalysis* steuern, zum Beispiel:

```
structureanalysis={list=true bullets={{fontname=ZapfDingbats}}}  
layoutanalysis = {layoutrowhint={full separation=preservecolumns}}  
layoutdetect=2  
layouteffort=high
```

**Tabellenerkennung.** TET erkennt Tabellenstrukturen auf der Seite und ordnet die Tabelleninhalte in Zeilen, Spalten und Zellen an. Informationen zu den auf der Seite erkannten Tabellen werden nicht direkt vom API bereitgestellt, sondern sind nur in der TETML-Ausgabe verfügbar, wie im folgenden Beispiel dargestellt:

```
<Table llx="302.14" lly="639.72" urx="525.50" ury="731.50">  
<Row>  
  <Cell colSpan="3" llx="306.14" lly="641.52" urx="516.67" ury="650.52">  
    <Para>  
      <Box llx="306.14" lly="641.52" urx="516.67" ury="650.52">  
        <Word>  
          <Text>TET</Text>  
          <Box llx="306.14" lly="641.52" urx="319.70" ury="650.52"/>  
        </Word>  
        <Word>  
          <Text>processes</Text>  
          <Box llx="321.67" lly="641.52" urx="356.89" ury="650.52"/>  
        </Word>  
        <Word>  
          <Text>all</Text>  
          <Box llx="358.85" lly="641.52" urx="368.15" ury="650.52"/>  
        </Word>  
        ...  
      </Box>  
    </Para>  
  </Cell>  
</Row>  
</Table>
```

TET kann optional horizontale und vertikale Linien oder Farbrechtecke analysieren, die oft zur Gestaltung von Tabellen verwendet werden. Diese Analyse von Vektorgrafik ist standardmäßig deaktiviert. Durch Layout-Analyse kann das Ergebnis der Tabellen- und Layout-Erkennung verbessert werden, wenn solche grafischen Elemente vorhanden sind. Die Analyse von Vektorgrafik lässt sich mit der Seitenoption *vectoranalysis* aktivieren, zum Beispiel:

```
vectoranalysis={structures=tables}
```

## 6.8 Überprüfen auf leere Bereiche

Mit TET kann auch überprüft werden, ob ein bestimmter Bereich auf einer Seite leer ist, d.h. weder Text-, Bild-, noch Vektorgrafik-Objekte enthält. Wenn Sie beispielsweise einen Stempel, eine Seitenzahl, einen Barcode oder ein anderes Element auf einer Seite platzieren müssen und die Seiteninhalte variieren, kann es schwierig sein, das Element so zu platzieren, dass bereits vorhandene Inhalte nicht verdeckt werden. Mit TET können Sie überprüfen, ob der Zielbereich tatsächlich leer ist. Dies funktioniert mit dem TET-API folgendermaßen:

- ▶ Die Seitenoption *emptycheck* aktiviert die Funktion und deaktiviert jegliche Extraktion von Seiteninhalten.
- ▶ Die Koordinaten des zu prüfenden rechteckigen Bereichs werden in der Seitenoption *includebox* übergeben. Hierzu sind doppelte Klammern erforderlich, da diese Funktion gewöhnlich die Angabe mehrerer Rechtecke akzeptiert (für *emptycheck* macht jedoch nur die Angabe eines einzigen Rechtecks Sinn):

```
includebox={{100 20 500 100}}
```

- ▶ Wird die Option *includebox* nicht übergeben, wird der gesamte Clipping-Bereich überprüft. Auf diese Weise lassen sich auch leere Seiten identifizieren.
- ▶ Anstatt der Extraktion von Seiteninhalten gibt *TET\_get\_text()* einen der Strings *empty* oder *notempty* als Prüfergebnis aus.

Mit dem TET-Kommandozeilen-Tool kann die Funktion *emptycheck* folgendermaßen ausgeführt werden:

```
tet --pageopt "emptycheck includebox={{300 760 450 820}}" input.pdf  
box on page 1: empty  
box on page 2: empty  
box on page 3: notempty
```

# 7 Fortgeschrittene Unicode-Verarbeitung

## 7.1 Wichtige Unicode-Konzepte

Dieses Kapitel enthält grundlegende Informationen zu Unicode, da sich die Textverarbeitung von TET stark auf den Unicode-Standard stützt. Die Unicode-Website bietet eine Fülle von zusätzlichen Informationen:

[www.unicode.org](http://www.unicode.org)

**Zeichen und Glyphen.** Beim Arbeiten mit Text ist es wichtig, klar zwischen folgenden Konzepten unterscheiden:

- ▶ *Zeichen* sind die kleinsten Einheiten, die in einer Sprache Informationen übermitteln, wie zum Beispiel die Buchstaben im lateinischen Alphabet, chinesische Bildzeichen oder japanische Silben. Zeichen haben eine bestimmte Bedeutung; Sie stellen semantische Einheiten dar.
- ▶ *Glyphen* sind grafische Formen, die ein oder mehrere Zeichen darstellen. Glyphen haben ein bestimmtes Aussehen: Sie stellen grafische Einheiten dar.

Es gibt keine Eins-zu-Eins-Beziehung zwischen Zeichen und Glyphen. So ist eine Ligatur zum Beispiel eine einzelne Glyphe, die mindestens zwei Zeichen zugeordnet ist. Genau so kann eine Glyphe je nach Kontext zur Darstellung verschiedener Zeichen dienen (manche Zeichen sehen gleich aus, siehe Abbildung 7.1).

Die Unicode-Nachbearbeitung in TET kann die Beziehung zwischen Glyphen und Zeichen noch stärker verändern. Beispielsweise kann durch Dekomposition ein einzelnes Zeichen in mehrere Zeichen zerlegt werden und durch Foldings können Zeichen ganz entfernt werden. Deshalb dürfen Sie nicht von einer bestimmten Beziehung zwischen Zeichen und Glyphen ausgehen.

### Zeichen

### Glyphen

U+0067 kleines lateinisches g

U+0066 kleines lateinisches f +  
U+0069 kleines lateinisches i

U+2126 Ohm-Zeichen oder  
U+03A9 großes griechisches Omega

U+2167 römische Zahl 8 oder  
U+0056 V U+0049 I U+0049 I U+0049 I

Abb. 7.1  
Beziehung zwischen  
Glyphen und Zeichen

**BMP und PUA.** Die folgenden Begriffe werden häufig in Unicode-basierten Umgebungen verwendet:

- ▶ Die *Basic Multilingual Plane (BMP)* umfasst die Codepunkte im Unicode-Bereich U+0000...U+FFFF. Der Unicode-Standard enthält viele weitere Codepunkte in den Supplementär-Ebenen, d.h. im Bereich U+10000...U+10FFFF.
- ▶ Die *Private Use Area (PUA)* bezeichnet einen von mehreren Unicode-Bereichen, die für den privaten Gebrauch reserviert sind. PUA-Codepunkte können nicht für den allgemeinen Austausch verwendet werden, da der Unicode-Standard keine Zeichen in diesem Bereich festlegt. Die Basic Multilingual Plane enthält den PUA-Bereich U+E000...U+F8FF. Ebene fünfzehn (U+F0000...U+FFFFD) und Ebene sechzehn (U+100000...U+10FFFFD) sind vollständig dem privaten Gebrauch vorbehalten.

**Unicode-Encoding-Formen (UTF-Formate).** Der Unicode-Standard ordnet jedem Zeichen eine Nummer (Codepunkt) zu. Um diese Nummern verwenden zu können, müssen sie sich in irgendeiner Form darstellen lassen. Im Unicode-Standard wird dazu eine Encoding-Form (früher: Transformationsformat) herangezogen, was nicht mit Font-Encodings zu verwechseln ist. Unicode definiert folgende Encoding-Formen:

- ▶ *UTF-8:* Format variabler Breite, in dem die Codepunkte durch 1-4 Bytes repräsentiert werden. ASCII-Zeichen im Bereich U+0000...U+007F werden durch ein einzelnes Byte im Bereich 00...7F dargestellt. Latin-1-Zeichen im Bereich U+00A0...U+00FF werden durch zwei Bytes dargestellt, wobei das erste Byte immer 0xC2 oder 0xC3 ist (diese Werte repräsentieren Â und Ã in Latin-1).
- ▶ *UTF-16:* Codepunkte in der Basic Multilingual Plane (BMP) werden durch einen einzelnen 16-Bit-Wert dargestellt. Codepunkte in den Supplementär-Ebenen, d.h. im Bereich U+10000...U+10FFFF, werden durch zwei 16-Bit-Werte dargestellt. Solche Paare werden Surrogatpaare genannt. Ein Surrogatpaar besteht aus einem höherwertigen Surrogatwert im Bereich D800...DBFF und einem niederwertigen Surrogatwert im Bereich DC00...DFFF. Beide können nur als Bestandteil eines Surrogatpaars und in keinem anderen Zusammenhang auftreten.
- ▶ *UTF-32:* Jeder Codepunkt wird durch einen 32-Bit-Wert dargestellt.

**Unicode-Encodings und der Byte Order Mark (BOM).** Computerarchitekturen unterscheiden sich in der Bytereihenfolge, d.h. ob die Bytes eines (16 oder 32 Bit großen) Werts mit dem höchstwertigen Byte (Big-Endian) oder niederstwertigen Byte (Little-Endian) zuerst gespeichert werden. PowerPC ist zum Beispiel eine Big-Endian-Architektur, während die x86-Architektur die Bytereihenfolge Little-Endian benutzt. Da UTF-8- und UTF-16-Werte größer als ein Byte sind, muss hier die Bytereihenfolge beachtet werden. Ein Encoding-Schema (im Unterschied zur Encoding-Form oben) legt Encoding-Form plus Bytereihenfolge fest. UTF-16BE steht beispielsweise für UTF-16 mit Bytereihenfolge Big-Endian. Steht die Bytereihenfolge nicht vorab fest, kann sie mit dem Codepunkt U+FEFF, dem so genannten Byte Order Mark (BOM), festgelegt werden. Der BOM ist in UTF-8 nicht unbedingt erforderlich, kann aber optional vorhanden sein, um eine Bytefolge als UTF-8 zu kennzeichnen. Tabelle 7.1 zeigt die Darstellung des BOM für verschiedene Encoding-Formen.

Tabelle 7.1 Byte Order Marks für verschiedene Unicode-Encoding-Formen

Encoding-Form	Byte Order Mark (Hex)	grafische Darstellung in WinAnsi <sup>1</sup>
UTF-8	EF BB BF	ï»¿
UTF-16 Big-Endian	FE FF	þÿ
UTF-16 Little-Endian	FF FE	ÿþ
UTF-32 Big-Endian	00 00 FE FF	■■þÿ
UTF-32 Little-Endian	FF FE 00 00	ÿþ■■

1. Das schwarze Quadrat ■ steht für ein Nullbyte.

**Zusammengesetzte Zeichen und Sequenzen.** Manchen Glyphen wird eine Sequenz mehrerer Zeichen zugeordnet. Dies ist zum Beispiel bei Ligaturen der Fall, die entsprechend ihrer Glyphen-Bestandteile auf mehrere Zeichen abgebildet werden. Bei zusammengesetzten Zeichen dagegen (wie der römischen Ziffer in Abbildung 7.1) hängt es von den im Font oder PDF vorliegenden Informationen sowie der Dokumentoption *decompose* ab, ob sie zerlegt werden oder nicht (siehe Abschnitt 7.3, »Unicode-Nachbearbeitung«, Seite 112).

Zusammengesetzte Zeichen werden von TET gegebenenfalls in eine Sequenz ihrer Glyphen-Bestandteile zerlegt. Diese Sequenz ist dann in dem Text enthalten, der von *TET\_get\_text()* zurückgegeben wird. Für jedes Zeichen können mit *TET\_get\_char\_info()* Informationen über die zugrunde liegenden Glyphen abgefragt werden; es kann damit auch ermittelt werden, ob das Zeichen eine Zeichensequenz einleitet oder abschließt. Die Position wird nur für das erste Zeichen einer Sequenz zurückgegeben. Nachfolgende Zeichen in der Sequenz haben keine eigenen Angaben zu Position oder Breite und müssen deshalb gemeinsam mit dem ersten Zeichen verarbeitet werden.

**Zeichen ohne zugehörige Glyphe.** Obwohl jede Glyphe auf der Seite einem oder mehreren Unicode-Zeichen zugeordnet ist, entsprechen nicht alle von TET zurückgegebenen Zeichen einer Glyphe. Zeichen, die einer Glyphe entsprechen, werden reale Zeichen genannt, die anderen dagegen künstliche Zeichen. Es gibt mehrere Klassen künstlicher Zeichen, die zurückgegeben werden, obwohl keine entsprechende Glyphe verfügbar ist:

- ▶ Ein zusammengesetztes Zeichen (siehe oben) wird auf eine Sequenz mehrerer Unicode-Zeichen abgebildet. Das erste Zeichen der Sequenz entspricht der Glyphe, die anderen Zeichen dagegen sind keiner Glyphe zugeordnet.
- ▶ Separator-Zeichen, die mit den Optionen *lineseparator*, *wordseparator* und *paragraphseparator* eingefügt wurden, sind Artefakte ohne entsprechende Glyphe.

## 7.2 Unicode-Vorbereitung (Filtern von Text)

TET wendet verschiedene Filter an, um unnützen Text zu entfernen. Mit diesen Filtern wird der Text modifiziert, bevor die Unicode-Nachbearbeitung beginnt. Während einige Filter immer aktiv sind, erfordern andere den Wordfinder und sind daher nur für *granularity=word* oder höher aktiv.

### 7.2.1 Filter für alle Granularitätsstufen

Folgende Filter können für alle Granularitätsstufen verwendet werden.

**Text in unhandlichen Fontgrößen.** Sehr kleiner oder sehr großer Text kann ignoriert werden, z.B. große Zeichen im Hintergrund der Seite. Die jeweiligen Grenzwerte werden mit der Seitenoption *fontsize* gesteuert. Standardmäßig wird Text in allen Fontgrößen extrahiert.

Mit der folgenden Seitenoption wird der Bereich der Fontgröße für den extrahierten Text auf 10 bis 50 Punkt begrenzt; Text in anderen Fontgrößen wird ignoriert:

```
fontsize={10 50}
```

**Unsichtbarer Text.** Unsichtbarer Text (d.h. Text mit *textrendering=3*) wird standardmäßig extrahiert. Beachten Sie, dass Text im PDF auch aus anderen Gründen als der *textrendering*-Eigenschaft unsichtbar sein kann, z.B. wenn Text- und Hintergrundfarbe identisch sind, wenn der Text von anderen Objekten auf der Seite verdeckt wird usw. Das hier beschriebene Verhalten bezieht sich nur auf *textrendering=3*. Diese PDF-Methode wird meist für OCR-Ergebnisse verwendet, wo der Text unsichtbar über dem Scan liegt.

Unsichtbarer Text lässt sich mit der Option *textrendering* der Struktur *TET\_char\_info* ermitteln, die von *TET\_get\_char\_info()* zurückgegeben wird (siehe Tabelle 10.16, Seite 212) sowie mit dem Attribut *Glyph/@textrendering* in TETML.

Mit der folgenden Seitenoption kann unsichtbarer Text ignoriert werden:

```
ignoreinvisibletext=true
```

**Text mit bestimmten Fontnamen und -typen ignorieren.** Manchmal kann es sinnvoll sein, Text in einem bestimmten angegebenen Font zu ignorieren, zum Beispiel einen Symbolfont, der keinen semantisch relevanten Inhalt transportiert. Statt des Fontnamens kann auch der Fonttyp angegeben werden. Dies empfiehlt sich vor allem bei Type-3-Fonts, die manchmal für Ornamente verwendet werden. Dieser Filter lässt sich mit der Unteroption *remove* der Dokumentoption *glyphmapping* steuern.

Im folgenden Codebeispiel wird jeglicher Text in Type-3-Fonts ignoriert:

```
glyphmapping={{fonttype={Type3} remove}}
```

Text in den Fonts Webdings, Wingdings, Wingdings 2 und Wingdings 3 wird ignoriert:

```
glyphmapping={{fontname=webdings remove} {fontname=wingdings* remove}}
```

Die Bedingungen für Fontname und -typ können auch kombiniert werden, um zum Beispiel Text in allen Type-3-Fonts zu ignorieren, deren Name mit A beginnt:

```
glyphmapping={{fonttype={Type3} fontname=A* remove}}
```

## 7.2.2 Filter für die Granularität Wort oder höher

Folgende Filter können nur für *granularity=word, line* und *page* verwendet werden.

**Enttrennung (Dehyphenation).** Bei der Enttrennung (dehyphenation) werden Trennzeichen entfernt und die getrennten Teile eines Worts wieder zusammengesetzt.

Trennzeichen für das Trennen von Wörtern am Zeilenende lassen sich mit der Option *attributes* der Struktur *TET\_char\_info* ermitteln (siehe Tabelle 10.16, Seite 212) oder mit dem Attribut *Glyph/@hyphenation* in TETML.

Die Enttrennung lässt sich mit folgender Seitenoption deaktivieren:

```
contentanalysis={dehyphenate=false}
```

**Auflisten von Trennzeichen.** Bei aktiver Enttrennung können Sie festlegen, ob Trennzeichen zwischen den Bestandteilen eines getrennten Worts in der erzeugten Glyphliste aufgeführt werden sollen, also der von *TET\_get\_char\_info()* und den *Glyph*-Elementen in TETML erzeugten Liste von Glyphen. Trennzeichen werden standardmäßig entfernt.

Für einige Anwendungen kann die genaue Position eines Trennzeichens auf der Seite allerdings wichtig sein. In den Topics *highlight\_search\_terms* und *search\_and\_replace\_text* im TET Cookbook werden Trennzeichen zum Beispiel bei der Platzierung einer Anmerkung oder für Ersatztext über dem ursprünglichen Wort berücksichtigt. Mit der folgenden Seitenoption lassen sich in solchen Fällen alle bei der Enttrennung entdeckten Trennzeichen erhalten:

```
contentanalysis={keeplyphenglyphs=true}
```

Die Trennzeichen lassen sich mit dem Flag *TET\_ATTR\_DEHYPHENATION\_ARTIFACT* der Option *attributes* in der Struktur *TET\_char\_info* ermitteln, die von *TET\_get\_char\_info()* zurückgegeben wird (siehe Tabelle 10.16, Seite 212) bzw. mit dem Attribut *Glyph/@dehyphenation* mit dem Wert *artifact* in TETML.

**Entfernen von Schattentext.** Redundanter Text, der nur visuelle Effekte erzeugt, wie zum Beispiel Schattentext und künstliche Fettschrift, wird entfernt.

Schattentext und künstliche Fettschrift lassen sich mit der Option *attributes* der Struktur *TET\_char\_info* ermitteln (siehe Tabelle 10.16, Seite 212) bzw. mit dem Attribut *Glyph/@shadow* in TETML.

Die Entfernung von Schattentext lässt sich mit folgender Seitenoption deaktivieren:

```
contentanalysis={shadowdetect=false}
```

## 7.3 Unicode-Nachbearbeitung

TET bietet verschiedene Verfahren für die Feinabstimmung der Unicode-Zeichen im extrahierten Text. Die in diesem Kapitel dargestellten Nachbearbeitungsschritte sind im Unicode-Standard definiert. Sie sind in TET verfügbar und werden in der folgenden Reihenfolge abgearbeitet:

- ▶ Foldings werden durch die Dokumentoption *fold* gesteuert und erhalten, entfernen oder ersetzen bestimmte Zeichen. Beispiele: Entfernen von Trennzeichen für die Silbentrennung, Entfernen von arabischen Tatweel-Zeichen.
- ▶ Dekompositionen werden durch die Dokumentoption *decompose* gesteuert und ersetzen ein Zeichen durch eine äquivalente Folge von einem oder mehreren anderen Zeichen. Beispiele: Zerlegen von Ligaturen, Abbilden von ASCII- und Symbolvarianten voller Breite auf die entsprechenden Zeichen ohne die volle Breite.
- ▶ Normalisierung wird durch die Dokumentoption *normalize* gesteuert und konvertiert Text in eine der Unicode-Normalformen. Beispiele: Kombination von Basiszeichen und diakritischem Zeichen zu einem gemeinsamen Zeichen; Abbilden des Zeichens Ohm auf das griechische Omega.

### 7.3.1 Unicode-Folding

Beim Folding werden ein oder mehrere Unicode-Zeichen verarbeitet und eine bestimmte Aktion für jedes der Zeichen ausgeführt. Folgende Aktionen sind verfügbar:

- ▶ Erhalten des Zeichens;
- ▶ Entfernen des Zeichens;
- ▶ Ersetzen des Zeichens durch ein anderes (festes) Zeichen.

Foldings sind nicht verkettet: die Ausgabe eines Foldings wird nicht weiter durch die verfügbaren Foldings verarbeitet. Foldings betreffen nur Unicode-Textausgabe, jedoch nicht die Glyphen, die von der Struktur *TET\_char\_info* oder den *<Glyph>*-Elementen in TETML zurückgegeben werden. Wenn z.B. bestimmte Unicode-Zeichen beim Folding entfernt werden, werden die zugeordneten Glyphen der ursprünglichen Zeichen trotzdem ausgegeben.

Um die Lesbarkeit zu verbessern, werden in den Beispielen der folgenden Tabellen isolierte Unteroptionen der Optionsliste *fold* aufgelistet. Beachten Sie, dass diese Unteroptionen bei Anwendung mehrerer Foldings in einer großen Optionsliste zusammengefasst werden müssen; die Option *fold* darf nicht mehrfach übergeben werden. Das folgende Beispiel ist falsch:

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

Die folgende Optionsliste zeigt die korrekte Syntax für mehrfache Foldings:

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

**Beispiele für Foldings.** Tabelle 7.2 zeigt Beispiele für die Option *fold* für verschiedene Anwendungen von Foldings. Die Optionen müssen in der Optionsliste von *TET\_open\_document()* übergeben werden. TET kann Foldings auf eine ausgewählte Teilmenge aller Unicode-Zeichen anwenden. Diese werden Unicode-Mengen genannt; für die entsprechende Syntax siehe »Unicode-Mengen«, Seite 177.

Tabelle 7.2 Beispiele für die Option *fold*

Beschreibung und Optionsliste	vor dem Folding	nach dem Folding
<b>Entfernen aller Zeichen in einer Unicode-Menge</b>		
In der Ausgabe bleiben nur die in ISO 8859-1 (Latin-1) definierten Zeichen erhalten, d.h. alle Zeichen außerhalb des Unicode-Blocks Latin-1 werden entfernt: fold={{[^U+0020-U+00FF] remove} default}	A U+0104	n/a
Entfernen aller nicht alphabetischen Zeichen (z.B. Satzzeichen, Zahlen): fold={{[:Alphabetic=No:] remove} default}	7 U+0037	n/a
	A U+0041	A U+0041
Entfernen aller Zeichen außer Zahlen: fold={{[^[:General_Category=Decimal_Number:]] remove} default}	7 U+0037	7 U+0037
	A U+0041	n/a
Entfernen aller Gedankenstriche und Satzzeichen: fold={{[:General_Category=Dash_Punctuation:] remove} default}	- U+002D	n/a
Entfernen aller Bidi-Steuerzeichen: fold={{[:Bidi_Control:] remove} default}		n/a
Entfernen aller Variantenselektoren für Standard oder Ideographic Variation Sequences (IVS): fold={{[[\uFE00-\uFE0F][\U000E0100-\U000E01EF]] remove} default}	≠ U+2268	≠ U+2268
	FE00 U+FE00	
		≠ U+2268
<b>Ersetzen aller Zeichen in einer Unicode-Menge durch ein anderes Zeichen</b>		
Leerzeichen-Folding: alle Varianten von Unicode-Leerzeichen werden auf U+0020 abgebildet: fold={{[:blank:] U+0020} default}	U+00A0	U+0020
Folding für Gedanken-/Bindestriche: alle Varianten von Unicode-Gedanken-/Bindestrichen werden auf U+002D abgebildet: fold={{[:Dash:] U+002D} default}	- U+2011	- U+002D
Ersetzen aller unbelegten Zeichen (d.h. Unicode-Codepunkten, denen kein Zeichen zugewiesen ist) durch U+FFFD: fold={{[:Unassigned:] U+FFFD} default}	U+03A2	U+FFFD
<b>Verarbeitung spezieller Zeichen</b>		
Erhalten aller Trennstriche am Zeilenende unter Beibehaltung der Standard-Foldings. Da diese Zeichen intern in TET identifiziert werden (denn sie haben keine bestimmte Unicode-Property) wird das Folding für den Bereich mit Hilfe des Schlüsselworts <i>_dehyphenation</i> identifiziert: fold={{_dehyphenation preserve} default}	- U+002D	- U+002D
Erhalten von arabischen Tatweel-Zeichen (die standardmäßig entfernt werden): fold={{[U+0640] preserve} default}	- U+0640	- U+0640

Tabelle 7.2 Beispiele für die Option `fold`

Beschreibung und Optionsliste	vor dem Folding	nach dem Folding
Ersetzen verschiedener Interpunktionszeichen durch die entsprechenden ASCII-Zeichen: <code>fold={ {[U+2018] U+0027} {[U+2019] U+0027} {[U+201C] U+0022} {[U+201D] U+0022} default }</code>	“ U+201C	” U+0022
<b>Verarbeitung fontspezifischer PUA-Zeichen, z.B. japanischer EUDC- oder Logo-Font</b>		
Standardverhalten: Ersetzen von PUA-Zeichen durch das Unicode-Ersatzzeichen U+FFFD: <code>fold={{[:Private_Use:] U+FFFD} default }</code>		 U+FFFD
Erhalten von PUA-Zeichen: <code>fold={{[:Private_Use:] preserve} default }</code>		
Entfernen von PUA-Zeichen: <code>fold={{[:Private_Use:] remove} default }</code>		n/a
Entfernen von TET-spezifischen PUA-Werten für Glyphen ohne Unicode-Wert, aber Erhalten von fontspezifischen PUA-Zeichen: <code>fold={{_tet_pua remove} {[:Private_Use:] preserve} default }</code>	 	

**Standard-Foldings.** Außer bei *granularity=glyph* wendet TET die folgenden, in Tabelle 7.3 beschriebenen Standard-Foldings an:

```
{[:blank:] U+0020}
{tet_pua unknownchar}
{[:Private_Use:] U+FFFD}
{ _dehyphenation remove}
{[[\u0640][:Control:][:Unassigned:]] remove}
```

Um benutzerdefinierte mit Standard-Foldings zu kombinieren, müssen Sie das Schlüsselwort *default* nach den Optionen für benutzerdefinierte Foldings angeben (wie in allen Beispielen in Tabelle 7.2 aufgeführt). Beispielsweise erhält die folgende Optionsliste *fold* Trennstriche in getrennten Wörtern und wendet dann die Standard-Foldings an:

```
fold= { _dehyphenation preserve} default }
```

Es wird generell empfohlen, der Optionsliste *fold* das Schlüsselwort *default* mitzugeben, es sei denn, Sie möchten alle Standard-Foldings explizit deaktivieren.

Tabelle 7.3 Standard-Foldings

Folding und Beschreibung	Beispiel-Eingabe	Ausgabe
<p>Leerzeichen-Folding: alle Varianten von Unicode-Leerzeichen werden auf U+0020 abgebildet:</p> <pre>{{[:blank:] U+0020}}</pre>	U+00A0	U+0020
<p>Zuordnen von TET-spezifischen PUA-Werten für Glyphen ohne Unicode-Wert auf das in der Option <code>unknowchar</code> angegebene Zeichen (oder Anwenden der spezifizierten Aktion <code>preserve/remove</code>):</p> <pre>{_tetpua unknowchar}</pre>		 U+FFFF
<p>Zuordnen von PUA-Zeichen auf das Unicode-Ersatzzeichen U+FFFF:</p> <pre>{{[:Private_Use:] U+FFFF}}</pre>		 U+FFFF
<p>Entfernen von Trennzeichen in enttrennten Wörtern:</p> <pre>{{_dehyphenation remove}}</pre>	- U+002D	n/a
<p>Entfernen von arabischen Tatweel-Zeichen, Steuerzeichen und in Unicode unbelegten Zeichen (diese Foldings werden bei der Erstellung von TETML immer nach allen anderen Foldings ausgeführt):</p> <pre>{{[\u0640][:Control:][:Unassigned:] remove}}</pre>	- U+0640  U+000C U+03A2	n/a

## 7.3.2 Unicode-Dekomposition

Dekompositionen ersetzen ein Zeichen durch eine äquivalente Folge von einem oder mehreren anderen Zeichen. Ein Unicode-Zeichen wird als (entweder kompatibel oder kanonisch) äquivalent zu einem anderen Zeichen oder einer Folge von Zeichen bezeichnet, wenn sie die gleiche Bedeutung haben, aber aus historischen Gründen (meist im Zusammenhang mit Round Tripping bei Legacy-Encodings) unterschiedlich in Unicode kodiert werden. Dekompositionen zerstören Information. Dies ist nützlich, wenn Sie nicht am Unterschied zwischen dem ursprünglichen Zeichen und seinem Äquivalent interessiert sind. Wenn Sie jedoch an diesem Unterschied interessiert sind, sollten Sie die entsprechende Dekomposition nicht anwenden. Für weitere Informationen zu Unicode-Dekompositionen siehe

[www.unicode.org/versions/Unicode8.0.0/cho2.pdf](http://www.unicode.org/versions/Unicode8.0.0/cho2.pdf) (Section 2.12) und  
[www.unicode.org/versions/Unicode8.0.0/cho3.pdf](http://www.unicode.org/versions/Unicode8.0.0/cho3.pdf) (Section 3.7)

*Hinweis* Der Begriff »Dekomposition« wird hier verwendet, wie im Unicode-Standard definiert, obwohl viele Dekompositionen ein Zeichen gar nicht in Einzelteile zerlegen, sondern es in ein anderes Zeichen konvertieren.

**Kanonische Dekomposition.** Wenn Zeichen oder Zeichenfolgen kanonisch äquivalent sind, dann stellen sie exakt das gleiche abstrakte Zeichen dar und sollten deshalb immer das gleiche Aussehen und Verhalten zeigen. Typische Beispiele sind vorkombinierte

Zeichen (z.B.  $\text{Ä}_{U+00C4}$ ) im Gegensatz zu kombinierenden Sequenzen (z.B.  $\text{A}_{U+0041} \text{¨}_{U+0308}$ ); beide Darstellungen sind kanonisch äquivalent. Durch den Wechsel von einer Darstellungsform zur anderen wird keine Information entfernt. Bei kanonischer Dekomposition wird eine Darstellung durch eine andere ersetzt, die als kanonische Darstellung bezeichnet wird.

In den Unicode-Codetabellen<sup>1</sup> (nicht jedoch in den Zeichentabellen) sind kanonische Zuordnungen mit dem Symbol IDENTICAL TO  $\equiv_{U+2261}$  versehen. Der Dekompositionsname *<canonical>* wird implizit angenommen. Tabelle 7.4 enthält verschiedene Beispiele.

**Kompatibilitätsdekomposition.** Wenn Zeichen kompatibel äquivalent sind, dann stellen sie zwar das gleiche abstrakte Zeichen dar, können sich aber in Aussehen und Verhalten unterscheiden. Typische Beispiele sind isolierte Formen arabischer Zeichen (z.B.  $\text{س}_{U+0633}$ ) im Gegensatz zu kontextspezifischen Formen (z.B.  $\text{س}_{U+FEB2}$ ,  $\text{س}_{U+FEB4}$ ,  $\text{س}_{U+FEB3}$ ). Kompatibel äquivalente Zeichen unterscheiden sich in der Formatierung. Das Entfernen dieser Formatierung bedeutet zwar einen Informationsverlust, kann aber die Verarbeitung bei bestimmten Anwendungen (z.B. Suchläufen) vereinfachen.

In den Unicode-Codetabellen sind kompatible Zuordnungen mit dem Symbol ALMOST EQUAL TO  $\approx_{U+2248}$  versehen, gefolgt vom Dekompositionsnamen (oder »Tag«) in spitzen Klammern, z.B. *<noBreak>*. Wird kein Tag-Name angegeben, wird *<compat>* angenommen. Die Tag-Namen sind identisch mit den Optionsnamen in Tabelle 7.5. Wie einige Beispiele zeigen, kann durch die Dekomposition ein einzelnes Zeichen in eine Folge mehrerer Zeichen konvertiert werden.

<sup>1</sup> Siehe [www.unicode.org/Public/8.0.0/charts/](http://www.unicode.org/Public/8.0.0/charts/)

Tabelle 7.4 Kanonische Dekomposition: Unteroption für die Option `decompose` (kanonisch äquivalente Zeichen sind in den Unicode-Codetabellen mit dem Symbol `IDENTICAL TO` <sup>U+2261</sup> gekennzeichnet)

Dekomposition	Beschreibung	vor der Dekomposition	nach der Dekomposition
<i>canonical</i> <sup>1</sup>	Kanonische Dekomposition	À U+00C0	A ` U+0041 U+0300
		林 U+F9F4	林 U+6797
		Ω U+2126	Ω U+03A9
		ば U+3070	は " U+306F U+3099
		ℵ U+FB2F	ℵ U+05D0 U+05B8

1. Damit bestimmte Zeichen erhalten bleiben, wird diese Dekomposition standardmäßig nicht auf alle Zeichen angewendet; für weitere Informationen siehe »Standard-Dekompositionen«, Seite 119.

**Hinweis** Beachten Sie, dass Glyphen in einem PDF-Dokument manchmal bereits auf eine zerlegte Sequenz statt auf die unzerlegten Unicode-Werte abgebildet werden. In diesem Fall hat die Option `decompose` keine Wirkung auf die Ausgabe.

**Beispiele für Dekompositionen.** Dekompositionen können in TET mit der Dokumentoption `decompose` gesteuert werden. Eine Dekomposition kann auf bestimmte Unicode-Zeichen beschränkt werden. Die Teilmenge der Zeichen, auf die die Dekomposition angewendet wird, heißt Domäne. Tabelle 7.5 führt die Unteroptionen für alle Unicode-Dekompositionen mit Beispielen auf.

Die folgenden Beispiele für die Option `decompose` müssen in der Optionsliste von `TET_open_document()` übergeben werden. Die Namen der Dekompositionen in der Optionsliste `decompose` sind Tabelle 7.5 entnommen.

Deaktivieren aller Dekompositionen:

```
decompose={none}
```

Erhalten breiter Zeichen (*double-byte* oder *zenkaku*) und schmaler (*hankaku*) Zeichen:

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

Zuordnen aller kanonischen Äquivalente auf ihre Entsprechungen:

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

Mit der folgenden Optionsliste lässt sich die Dekomposition `circle` aktivieren, alle anderen Dekompositionen werden deaktiviert:

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

Tabelle 7.5 Kanonische Dekomposition: Unteroptionen für die Option decompose (kanonisch äquivalente Zeichen werden in den Unicode-Codetabellen mit dem Symbol ALMOST EQUAL TO ≈ gekennzeichnet)  
U+2248

Dekomposition	Beschreibung	vor der Dekomposition	nach der Dekomposition (in logischer Reihenfolge)
<b>circle</b>	Eingekreiste Zeichen	Ⓢ U+3251	2 1 U+0032 U+0031
<b>compat</b>	Sonstige Kompatibilitätsdekompositionen, z.B. gängige Ligaturen	fi U+FB01	f i U+0066 U+0069
<b>final</b>	Finale Präsentationsformen, besonders Arabisch	س U+FEB2	س U+0633
<b>font</b>	Fontvarianten, z.B. mathematische Mengenzeichen, hebräische Ligaturen	Ⓒ U+2102	Ⓒ U+0043
<b>fraction</b>	Gemeine Brüche	¼ U+00BC	1 / 4 U+0031 U+2044 U+0034
<b>initial</b>	Initiale Präsentationsformen, besonders Arabisch	س U+FEB3	س U+0633
<b>isolated</b>	Isolierte Präsentationsformen, besonders Arabisch	س U+FD0E	س ر U+0633 U+0631
<b>medial</b>	Mittlere Präsentationsformen, besonders Arabisch	س U+FEB4	س U+0633
<b>narrow</b>	Schmale (hankaku) Kompatibilitätszeichen	ㇿ U+FF66	ㇾ U+30F2
<b>nobreak</b>	Umbruchgeschützte Leerzeichen	 U+00A0	 U+0020
<b>none</b>	Deaktivieren aller Dekompositionen, die nicht explizit in der Optionsliste decompose angegeben sind.	(alle Zeichen bleiben unverändert)	
<b>small</b>	Kleine Formen für die Kompatibilität zu CNS 11643	⸣ U+FE50	⸣ U+002C
<b>square</b>	eckige CJK-Fontvarianten	𠂇 U+3314	𠂇 𠂆 U+30AD U+30ED
<b>sub</b>	Tiefgestellte Zeichen	₁ U+2081	₁ U+0031
<b>super</b>	Hochgestellte Zeichen	ᵃ U+00AA ™ U+2122	ᵃ U+0061 ™ M U+0054 U+004D
<b>vertical</b>	Vertikale Layout-Präsentationsformen	⸮ U+FE37	{ U+007B
<b>wide</b>	Breite (zenkaku) Kompatibilitätsformen	£ U+FFE1	£ U+00A3

Im Gegensatz dazu lassen sich mit der folgenden Optionsliste alle Dekompositionen aktivieren (da das Weglassen aller anderen Optionen das Standardverhalten aktiviert):

decompose={circle=\_all}

**Standard-Dekompositionen.** Standardmäßig sind alle Dekompositionen außer *fraction* aktiviert. Während die Standard-Dekompositionen für die Domäne *\_all* gelten (d.h. sie werden auf alle Zeichen angewendet), gelten andere standardmäßig nur für kleinere Domänen gemäß Tabelle 7.6. Dekompositionen lassen sich am einfachsten über die Normalisierung steuern (siehe Abschnitt 7.3.3, »Unicode-Normalisierung«, Seite 120). Da die Unicode-Nachbearbeitung für *granularity=glyph* deaktiviert ist, sind in diesem Fall keine Dekompositionen aktiv.

Tabelle 7.6 Standarddomänen für Unicode-Dekompositionen (Unteroptionen für die Option decompose)

Dekomposition	TET-Standardwert
<b>canonical</b>	<p>canonical=[{U+0374 U+037E U+0387 U+1FBE U+1FEF U+1FFD U+2000 U+2001 U+2126 U+212A U+212B U+2329-U+232A}]</p> <p>Die Standarddomäne umfasst kanonische Duplikate (Singletons), nicht aber andere kanonisch äquivalente Zeichen. Um Zeichen wie <math>\text{Ä}</math> zu erhalten, ist der Standardwert nicht <i>_all</i>.  <small>U+00C4</small></p>
<b>compat</b>	<p>compat=[{U+FB00-U+FB17}]</p> <p>Die Standarddomäne umfasst lateinische und armenische Ligaturen, aber keine weiteren Kompatibilitätszeichen. Um Zeichen wie <math>\text{IJ}</math> zu erhalten, ist der Standardwert nicht <i>_all</i>.  <small>U+0132</small></p>
<b>fraction</b>	<p>fraction=_none</p> <p>Brüche werden standardmäßig nicht zerlegt, da dies zu unerwünschten Ziffernfolgen bei den Bestandteilen der Brüche führen würde, Client-Anwendungen könnten z.B. die Sequenz <math>\text{9 } \frac{1}{2}</math> (also den numerischen Wert 9,5) fälschlicherweise als <math>\text{9 } \frac{1}{2}</math> interpretieren, was den numerischen Wert <math>(91)/2=45,5</math> darstellt.  <small>U+0039 U+00BD U+0039 U+0031 U+2044 U+0032</small></p>
<b>sub</b> <b>super</b>	<p>sub=[{U+208A-U+208E}]</p> <p>super=[{U+207A-U+207E}]</p> <p>Die Standarddomäne umfasst nur mathematische Zeichen. Hoch- und tiefgestellte Zeichen werden standardmäßig nicht zerlegt, um Probleme mit der numerischen Interpretation zu vermeiden, ähnlich wie oben bei <i>fraction</i> beschrieben. Zeichen wie das Trademark-Symbol <math>\text{TM}</math> werden standardmäßig nicht in <math>\text{T M}</math> zerlegt.  <small>U+0054 U+004D U+2122</small></p>
<b>alle anderen</b>	<p>circle=_all final=_all ... vertical=_all wide=_all</p> <p>Alle sonstigen Dekompositionen sind standardmäßig für alle Zeichen aktiviert.</p>

### 7.3.3 Unicode-Normalisierung

Der Unicode-Standard definiert vier Normalformen, zwei für kanonische und zwei für kompatible Äquivalenz (siehe Abschnitt 7.3.2, »Unicode-Dekomposition«, Seite 116). Alle Normalformen platzieren kombinierende Zeichen in einer definierten Reihenfolge und wenden Zerlegung (Dekomposition) und Zusammensetzung (Komposition) auf unterschiedliche Weise an:

- ▶ Normalform C (NFC): kanonische Dekomposition gefolgt von kanonischer Komposition
- ▶ Normalform D (NFD): kanonische Dekomposition
- ▶ Normalform KC (NFKC): Kompatibilitätsdekomposition gefolgt von kanonischer Komposition
- ▶ Normalform KD (NFKD): Kompatibilitätsdekomposition

Die Normalformen sind im Unicode Standard Annex #15 »Unicode Normalization Forms« spezifiziert (siehe [www.unicode.org/versions/Unicode8.0.0/cho3.pdf#G21796](http://www.unicode.org/versions/Unicode8.0.0/cho3.pdf#G21796) und [www.unicode.org/reports/tr15/](http://www.unicode.org/reports/tr15/)).

TET unterstützt alle vier Normalformen. Die Unicode-Normalisierung lässt sich mit der Dokumentoption *normalize* steuern, z.B.:

```
normalize=nfc
```

TET wendet standardmäßig keine Normalisierung an. Wegen der möglichen Wechselwirkung zwischen den Optionen *decompose* und *normalize* werden die Standard-Dekompositionen deaktiviert, wenn die Option *normalize* auf einen anderen Wert als *none* gesetzt wird.

Die Wahl der Normalform hängt von den Anforderungen der Anwendung ab. Beispielsweise erwarten einige Datenbanken Text im Format NFC, was auch das gängige Format für Unicode-Text im Web ist. Tabelle 7.7 zeigt die Auswirkung der Normalisierung auf verschiedene Zeichen.

Tabelle 7.7 Beispiele für Unicode-Normalformen

vor der Normalisierung	NFC	NFD	NFKC	NFKD
Ä U+00C4	Ä U+00C4	À ¨ U+0041 U+0308	Ä U+00C4	À ¨ U+0041 U+0308
À ¨ U+0041 U+0308	Ä U+00C4	À ¨ U+0041 U+0308	Ä U+00C4	À ¨ U+0041 U+0308
¨ À U+0308 U+0041	¨ À U+0308 U+0041	¨ À U+0308 U+0041	¨ À U+0308 U+0041	¨ À 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	À ° U+0041 U+030A	Å U+00C5	À ° U+0041 U+030A

Tabelle 7.7 Beispiele für Unicode-Normalformen

vor der Normalisierung	NFC	NFD	NFKC	NFKD
TM U+2122	TM U+2122	TM U+2122	T M U+0054 U+004D	T M U+0054 U+004D
IV U+2163	IV U+2163	IV U+2163	I V U+0049 U+0056	I V U+0049 U+0056
ㄱ U+FB48	ㄱ U+05E8 U+05BC	ㄱ U+05E8 U+05BC	ㄱ U+05E8 U+05BC	ㄱ 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	ぢ U+3061 U+3099	ぢ U+3062	ぢ U+3061 U+3099
10月 U+32C9	10月 U+32C9	10月 U+32C9	1 0 月 U+0031 U+0030 U+6708	1 0 月 U+0031 U+0030 U+6708

## 7.4 Zeichen außerhalb der BMP und Surrogatpaare

Zeichen außerhalb der Basic Multilingual Plane (BMP) von Unicode, d.h. mit Unicode-Werten über  $U+FFFF$ , lassen sich nicht in einem einzelnen, sondern nur in einem Paar von UTF-16-Werten kodieren, dem sogenannten Surrogatpaar. Zu den Zeichen außerhalb der BMP gehören mathematische und musikalische Symbole ab  $U+1DXXX$  sowie Tausende von CJK-Erweiterungszeichen ab  $U+20000$ .

TET verwendet auch die *Supplementary Private Use Area*, um auch solchen Glyphen Unicode-Werte zuzuweisen, für die im PDF-Dokument keine Unicode-Zuordnung gefunden wurde. Standardmäßig werden diese Zeichen durch das Unicode-Ersatzzeichen  $U+FFFD$  ersetzt. Mit der Option *unknownchar=preserve* können sie jedoch in der Ausgabe als Unicode-Werte außerhalb des BMP, also als Werte oberhalb von  $U+FFFF$ , auftreten (siehe »Glyphen ohne Unicode-Werte und die TET-PUA«, Seite 123).

TET interpretiert und erhält Surrogatpaare und ermöglicht auch in Programmiersprachen, die nur UTF-16-Strings unterstützen, den Zugriff auf die zugehörigen UTF-32-Werte. Das von `TET_get_char_info()` für den ersten Surrogatwert zurückgegebene Feld *uv* enthält den zugehörigen UTF-32-Wert. Damit kann auch in einer UTF-16-Umgebung ohne UTF-32-Unterstützung direkt auf den UTF-32-Wert eines außerhalb der BMP liegenden Zeichens zugegriffen werden.

Die beiden Surrogatwerte bleiben erhalten. Der von `TET_get_text()` zurückgegebene String enthält zwei UTF-16-Werte.

## 7.5 Unicode-Zuordnung für Glyphen

Während der Text in PDF-Dokumenten durch eine Vielzahl von Font- und Encoding-Schemata darstellbar ist, abstrahiert TET vollkommen von den Glyphen und konvertiert Text unabhängig von der Originaldarstellung prinzipiell in Unicode-Zeichen. Die Umwandlung der im PDF vorliegenden Informationen in geeignete Unicode-Werte wird Unicode-Zuordnung (*Unicode mapping*) genannt. Dieser Prozess ist für das Textverständnis von entscheidender Bedeutung (im Gegensatz zur visuellen Darstellung des Textes auf dem Bildschirm oder Papier). Um eine passende Unicode-Zuordnung zu gewährleisten, untersucht TET verschiedene Datenstrukturen, die es aus dem PDF-Dokument, aus eingebetteten oder externen Fontdateien sowie integrierten oder anwendungsspezifischen Tabellen bezieht. Außerdem ermittelt TET mittels verschiedener Verfahren die Unicode-Abbildung standardmäßiger Glyphennamen.

Trotz aller Anstrengungen bleiben immer noch einige wenige PDF-Dokumente, deren Text sich nicht in Unicode konvertieren lässt. Um auch diese Fälle abzudecken, bietet TET eine Reihe von Konfigurationsmöglichkeiten, mit denen sich die Unicode-Zuordnung problematischer PDF-Dateien steuern lässt.

**Glyphen ohne Unicode-Werte und die TET-PUA.** Es gibt verschiedene Gründe, warum sich Text in einem PDF nicht in Unicode umsetzen lässt. Type-1-Fonts können zum Beispiel unbekannte Glyphennamen enthalten und TrueType-, OpenType oder CID-Fonts können Glyph-IDs ohne Unicode-Werte im Font oder im PDF enthalten. Wenn TET nach der Analyse der Informationen im PDF-Dokument, in eingebetteten und externen Fonts, konfigurierten und internen Tabellen keinen Unicode-Wert bestimmen kann, gilt die Glyphe als unbekannt.

Glyphen ohne Unicode-Wert können mit dem Feld *unknown* der Struktur *TET\_char\_info()* (siehe Tabelle 10.16, Seite 212) oder dem Attribut *Glyph/@unknown* in TETML identifiziert werden.

TET weist allen Glyphen ohne Unicode-Wert abnehmende Werte in der *TET Private Use Area* (TET-PUA) zu. Die TET-PUA liegt in der *Supplementary Private Use Area*, d.h. außerhalb der BMP, um Konflikte mit in Fonts zugewiesenen PUA-Werten zu vermeiden. Die TET-PUA kann mit dem Schlüsselwort *\_tetpua* in der Option *fold* adressiert werden.

Standardmäßig werden TET-PUA-Werte für nicht zuzuordnende Glyphen durch das Unicode-Ersatzzeichen U+FFFD ersetzt. Dieses Verhalten lässt sich mit der Dokumentoption *unknownchar* ändern, die auf ein beliebiges Unicode-Zeichen gesetzt werden kann oder mit der angegeben werden kann, dass TET-PUA-Werte für nicht zuzuordnende Glyphen erhalten oder entfernt werden sollen. In Tabelle 7.2 werden verschiedene Kombinationen der Optionen *fold* und *unknownchar* für verschiedene Anwendungsfälle aufgeführt.

**Zeichen in der Private Use Area (PUA).** Ein Font oder ein PDF-Dokument kann eine Glyphe auf ein Unicode-Zeichen in der Private Use Area zuweisen. Dies wird gewöhnlich für Symbole ohne global standardisierte Bedeutung verwendet, wie z.B. Fonts für japanische benutzerdefinierte Zeichen (EUDC) oder Logo-Fonts. Da PUA-Zeichen in generischen Unicode-Workflows nicht sinnvoll verwendet werden können, werden sie standardmäßig durch das Unicode-Ersatzzeichen U+FFFD ersetzt. Siehe Tabelle 7.2 für die Erhaltung von PUA-Werten in Fällen, in denen die Anwendung PUA-Werte verarbeiten kann.

Tabelle 7.8 Behandlung von TET-PUA-Werten für nicht zuzuordnende Glyphen mit der Dokumentoption `unknownchar`

Beschreibung und Optionsliste	reine Eingabe	Ergebnis
Standardverhalten: Ersetzen von TET-PUA-Werten für nicht zuzuordnende Glyphen durch das Unicode-Ersatzzeichen U+FFFD: <code>unknownchar=U+FFFD</code>	☒	◊ U+FFFD
Ersetzen von TET-PUA-Werten für nicht zuzuordnende Glyphen durch ein Fragezeichen (oder ein beliebiges anderes geeignetes Unicode-Zeichen); dies kann nützlich sein, um problematische Glyphen im Text visuell zu identifizieren: <code>unknownchar=?</code>	☒	? U+003F
Entfernen von TET-PUA-Werten für nicht zuzuordnende Glyphen: <code>unknownchar=remove</code>	☒	n/a
Erhalten von TET-PUA-Werten für nicht zuzuordnende Glyphen; dies kann für die Analyse und das Debuggen nützlich sein: <code>unknownchar=preserve</code>	☒	(TET-PUA-Wert)

**Steuerung der Unicode-Zuordnung.** TET implementiert zahlreiche Funktionen, um den Text über Umwege auch aus PDF-Dokumenten zu extrahieren, die keine Unicode-Werte enthalten. Es gibt jedoch immer Dokumente, deren Text nicht extrahierbar ist, weil die Informationen in den Datenstrukturen des PDFs und der Fonts nicht ausreichen. TET enthält verschiedene Konfigurationsmöglichkeiten, mit denen sich zusätzliche Angaben zur Unicode-Zuordnung übergeben lassen. Diese werden im folgenden beschrieben.

Mit der Option `glyphmapping` von `TET_open_document()` (siehe Abschnitt 10.3, »Dokumentfunktionen«, Seite 190) können Sie die Unicode-Zuordnung für Glyphen in verschiedener Weise beeinflussen. Die folgende Liste gibt eine Übersicht über alle verfügbaren Möglichkeiten (die auch kombinierbar sind). Sie lassen sich sowohl auf einzelne Fonts als auch global auf alle Fonts eines Dokuments anwenden:

- ▶ Mit der Unteroption `forceencoding` kann prinzipiell jedes Auftreten der vordefinierten PDF-Encodings `WinAnsiEncoding` oder `MacRomanEncoding` überschrieben werden.
- ▶ Mit den Unteroptionen `codelist` und `tounicodecmap` können Unicode-Werte in einfachem Textformat (einer `codelist`-Ressource) übergeben werden.
- ▶ Mit der Unteroption `glyphlist` können Unicode-Werte für Glyphnamen, die nicht dem Standard entsprechen, übergeben werden.
- ▶ Mit der Unteroption `glyphrule` kann eine Regel definiert werden, die zur algorithmischen Ableitung von Unicode-Werten aus numerischen Glyphnamen eingesetzt wird. Einige Regeln sind in TET bereits integriert. Mit der Option `encodinghint` lassen sich die internen Regeln steuern.
- ▶ Zusätzlich zu den vielen vordefinierten Encodings können eigene Encodings definiert und mit der Option `encodinghint` oder der Unteroption `encoding` der Option `glyphrule` eingesetzt werden.
- ▶ Um Informationen zur Unicode-Zuordnung verfügbar zu machen, können externe Fonts konfiguriert werden, für den Fall, dass das PDF nicht über ausreichende Angaben verfügt und der Font nicht im PDF eingebettet ist.

**Analyse von PDF-Dokumenten mit dem Plugin PDFlib FontReporter<sup>1</sup>.** Um alle Angaben zu erhalten, die zur Erstellung von geeigneten Unicode-Zuordnungstabellen erforderlich sind, müssen Sie die problematischen PDF-Dokumente analysieren.

1. Das Plugin PDFlib FontReporter können Sie hier kostenlos herunterladen: [www.pdfli.com/products/fontreporter](http://www.pdfli.com/products/fontreporter)

PDFlib FontReporter																	PDFlib FontReporter																
PIAPOC+ThesisAntiqua-Normal, Type1, Embedded, Subset Encoding: custom, symbol bit, ToUnicode table, CharSet table																	ShinGo-Medium, CIDFontType0 Encoding: Identity-H Character collection: Adobe-Japan1-2 CID x0300 (50 glyphs)																
x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF		
0x																0x																	
1x																1x																	
2x																2x																	
3x																3x																	
4x																4x																	
5x																5x																	
6x																6x																	
7x																7x																	
8x																8x																	
9x																9x																	
Ax																Ax																	
Bx																Bx																	
Cx																Cx																	
Dx																Dx																	
Ex																Ex																	
Fx																Fx																	

Abb. 7.2  
Beispiel für einen Fontreport, der mit dem Acrobat-Plugin PDFlib FontReporter erstellt wurde

PDFlib GmbH bietet ein frei verfügbares Begleitprodukt zu TET, das in dieser Situation hilfreich ist: PDFlib FontReporter ist ein Plugin für Adobe Acrobat, mit dem Sie auf einfache Weise Informationen zu Fonts, Encodings und Glyphen ermitteln können. Das Plugin generiert einen detaillierten Fontbericht, der die eigentlichen Glyphen mitsamt folgender Daten enthält:

- ▶ Zugehöriger Code: die erste Hex-Zahl wird in der linken Spalte und die zweite Hex-Zahl in der oberen Zeile angezeigt. Bei CID-Fonts muss der im Header angeführte Offset addiert werden, um den zur Glyphe gehörenden Code zu ermitteln.
- ▶ Glyphennamen, sofern vorhanden.
- ▶ Unicode-Wert(e), die der Glyphe entsprechen (sofern sie von Acrobat bestimmbar sind).

Diese Angaben spielen eine wichtige Rolle bei der Konfiguration der Glyphenzuordnung von TET. Abbildung 7.2 zeigt beispielhaft zwei Seiten aus einem Fontbericht. Anhand der mit dem FontReporter-Plugin erstellten Reports können PDF-Fonts analysiert und Zuordnungstabellen erstellt werden, um den Text mit TET erfolgreich zu extrahieren. Sie sollten unbedingt einen Blick auf den Fontreport werfen, bevor Sie Unicode-Zuordnungstabellen oder eine Glyphennamen-Heuristik erstellen, um die Textextraktion mit TET zu konfigurieren.

**Vorrangregeln für die Glyphenzuordnung.** TET wendet die Regeln zur Glyphenzuordnung in der folgenden Reihenfolge an:



Abb. 73

Der Fontreport für das Firmenlogo zeigt, dass der Font die falschen Unicode-Zuordnungen enthält. Mit einer benutzerdefinierten Codeliste lassen sich solche Zuordnungen korrigieren.

- ▶ Zuerst werden Codelist- und ToUnicode-CMap-Ressourcen durchgesehen.
- ▶ Verfügt der Font über eine interne ToUnicode-CMap, so wird diese untersucht.
- ▶ Bei Glyphnamen wendet TET eine externe oder interne Zuordnungsregel an, sofern eine zum Font und zum Glyphnamen passende Regel vorhanden ist.
- ▶ Zuletzt kommt eine vom Benutzer übergebene Glyphliste zur Anwendung.

**Codelist-Ressourcen für alle Fonttypen.** Codelisten ähneln Glyphlisten, definieren Unicode-Werte jedoch für einzelne Codes statt für Glyphnamen. Obwohl es vorkommt, dass verschiedene Fonts eines Herstellers identische Codezuordnungen besitzen, sind Codes (auch Glyph-IDs genannt) in der Regel fontspezifisch. Demzufolge sind für einzelne Fonts getrennte Codelisten erforderlich. Eine Codeliste ist eine Textdatei, in der jede Zeile die Unicode-Zuordnung für einen Code beschreibt, gemäß folgender Regeln:

- ▶ Text nach einem Prozentzeichen '%' wird ignoriert; dies ist für Kommentare gedacht.
- ▶ Die erste Spalte enthält den Glyphencode in dezimaler oder hexadezimaler Schreibweise. Für einfache Fonts muss der Wert im Bereich 0-255 und für CID-Fonts im Bereich 0-65535 liegen.
- ▶ Der Rest einer Zeile enthält bis zu 7 Unicode-Werte für den Code. Die Werte können in dezimaler oder (mit dem Präfix *x* oder *ox*) in hexadezimaler Schreibweise angegeben werden. UTF-32 wird unterstützt, d.h. Surrogatpaare können verwendet werden.

Per Konvention verwenden Codelisten die Dateinamenserweiterung *.cl*. Codelisten werden mit der Ressourcenkategorie *codelist* konfiguriert. Wird nicht explizit eine Codelist-Ressource festgelegt, sucht TET eine Datei namens *<mycodelist>.gl* (mit *<mycodelist>* als Ressourcenname) in der Hierarchie *searchpath* (siehe auch Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70). Anders ausgedrückt: Sind Ressourcen- und Dateiname (bis auf die Endung *.cl*) identisch, müssen Sie die Ressource nicht konfigurieren, da TET implizit in etwa folgenden Aufruf absetzt (wobei *name* ein beliebiger Ressourcenname ist):

```
set_option("codelist {name name.cl}");
```

Das folgende Beispiel zeigt den Einsatz von Codelisten. Beachten Sie die falsch zugeordneten Logo-Glyphen in Abbildung 7.3, wo eine einzelne Glyphe des Fonts mehrere Zeichen darstellt und alle Zeichen gemeinsam das Firmenlogo bilden. Die Glyphen werden jedoch fälschlicherweise den Zeichen *a*, *b*, *c*, *d* und *e* zugeordnet. Dieser Fehler lässt sich beheben, indem Sie zunächst folgende Codeliste erstellen:

```
% Unicode-Zuordnungen für Codes im Font GlobeLogosOne
```

```
x61    x0054 x0068 x0065 x0020    % The
x62    x0042 x006F                    % Bo
x63    x0073 x0074 x006F x006E x0020 % ston
x64    x0047 x006C x006F                    % Glo
x65    x0062 x0065                    % be
```

Diese Codeliste übergeben Sie dann mit der folgenden Option an `TET_open_document()` (vorausgesetzt die Codeliste befindet sich in der Datei `GlobeLogosOne.cl` und ist über den Suchpfad (`searchpath`) auffindbar:

```
glyphmapping {{fontname=GlobeLogosOne codelist=GlobeLogosOne}}
```

**ToUnicode-CMap-Ressourcen für alle Fonttypen.** PDF unterstützt eine Datenstruktur namens ToUnicode CMap, mit der Sie Unicode-Werte für die Glyphen eines Fonts definieren können. TET verwendet diese Datenstruktur, wenn sie im PDF verfügbar ist. Eine ToUnicode CMap kann zudem in einer externen Datei übergeben werden. Dies ist sinnvoll, wenn im PDF keine ToUnicode CMap existiert oder die vorhandene CMap Lücken oder falsche Einträge aufweist. Eine ToUnicode CMap hat Vorrang vor einer Codeliste. Da Codelisten aber ein einfacheres Format besitzen, sind sie den ToUnicode CMaps vorzuziehen.

Per Konvention besitzen CMaps keine Dateinamenendung. ToUnicode CMaps können mit der `cmap`-Ressourcenkategorie konfiguriert werden (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70). Der Inhalt einer `cmap`-Ressource muss der Standardsyntax für CMaps entsprechen.<sup>1</sup> Um eine ToUnicode CMap auf alle Fonts der Familie *Warnock* anzuwenden, verwenden Sie `TET_open_document()` mit folgender Option:

```
glyphmapping {{fontname=Warnock* tounicodecmap=warnock}}
```

**Glyphlisten-Ressourcen für einfache Fonts.** Glyphlisten (Abkürzung für: Glyphnamenlisten) kommen zum Einsatz, wenn benutzerdefinierte Unicode-Werte für Glyphnamen zu übergeben sind, die von den Standardnamen abweichen, oder Werte für Standard-Glyphnamen überschrieben werden sollen. Die Glyphliste ist eine Textdatei, in der jede Zeile die Unicode-Zuordnung für einen Glyphnamen beschreibt, gemäß dieser Regeln:

- ▶ Text nach einem Prozentzeichen '%' wird ignoriert; dies ist für Kommentare gedacht.
- ▶ Die erste Spalte enthält den Glyphnamen. Jeder in einem Font verwendete Name ist erlaubt (d.h. auch die Unicode-Werte der Standard-Glyphnamen können überschrieben werden). Das Prozentzeichen kann im Glyphnamen nur benutzt werden, wenn ihm ein Gegenschrägstrich vorangestellt wird \%, um es vom Kommentarzeichen zu unterscheiden.
- ▶ Pro Glyphe ist nur eine Zuordnung erlaubt; mehrere Zuordnungen pro Glyphe führen zu einem Fehler.
- ▶ Der Rest der Zeile enthält bis zu 7 Unicode-Werte für den Glyphnamen. Die Werte können in dezimaler oder (mit dem Präfix `x` oder `ox`) in hexadezimaler Schreibweise angegeben werden. UTF-32 wird unterstützt, d.h. Surrogatpaare können verwendet werden.
- ▶ Steuerzeichen können in Glyphnamen mit den Escape-Sequenzen für Textdateien eingefügt werden (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70).

Per Konvention verwenden Glyphlisten die Dateinamenserweiterung `.gl`. Glyphlisten werden mit der Ressource `glyphlist` konfiguriert. Wird nicht explizit eine Glyphlisten-Ressource festgelegt, sucht TET eine Datei namens `<myglyphlist>.gl` (mit `<myglyphlist>` als Ressourcenname) in der Hierarchie `searchpath` (siehe auch Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70). Anders ausgedrückt: Sind Ressourcen- und

<sup>1</sup>. Siehe [partners.adobe.com/public/developer/en/acrobat/5411.ToUnicode.pdf](http://partners.adobe.com/public/developer/en/acrobat/5411.ToUnicode.pdf)

Dateiname (bis auf die Endung *.gl*) identisch, müssen Sie die Ressource nicht konfigurieren, da TET implizit in etwa folgenden Aufruf absetzt (wobei *name* ein beliebiger Ressourcenname ist):

```
set_option("glyphlist {name name.gl}");
```

Aufgrund der Reihenfolge der Regeln zur Glyphenzuordnung werden Glyphlisten nicht herangezogen, wenn der Font eine ToUnicode CMap enthält. Das folgende Beispiel zeigt den Einsatz von Glyphlisten:

```
% Unicode-Werte für Glyphnamen in TeX-Dokumenten
```

```
precedesequal 0x227C
similarequal  0x2243
negationslash 0x2044
union         0x222A
prime        0x2032
```

Um eine Glyphliste namens *tarski.gl* auf alle Fontnamen anzuwenden, die mit *CMSY* beginnen, verwenden Sie *TET\_open\_document()* mit folgender Option:

```
glyphmapping {{fontname=CMSY* glyphlist=tarski}}
```

**Regeln zur Interpretation numerischer Glyphnamen in einfachen Fonts.** PDF-Dokumente enthalten gelegentlich Glyphen, deren Namen nicht einer vordefinierten Liste entstammen, sondern algorithmisch generiert wurden. Dies kann ein »Feature« der Anwendung sein, die das PDF generiert, oder bei der Konvertierung des Fonts durch einen Druckertreiber verursacht werden: Manchmal gehen dabei die Originalnamen der Glyphen verloren und werden schematisch durch Namen der Art *Go0*, *Go1*, *Go2* usw. ersetzt. TET enthält Regeln zur Verarbeitung von numerischen Glyphnamen, die von verschiedenen gängigen Anwendungen und Treibern erzeugt wurden. Da für verschiedene Encodings dieselben Glyphnamen generiert werden können, lässt sich in *TET\_open\_document()* mit der Option *encodinghint* das gewünschte Encoding für die schematisch generierten Glyphnamen eines Dokuments übergeben. Wenn Sie zum Beispiel wissen, dass das Dokument russischen Text enthält, der Text aber aufgrund fehlender Informationen im PDF nicht erfolgreich extrahiert werden kann, können Sie mit der Option *encodinghint=cp1250* eine kyrillische Codepage angeben.

Zusätzlich zu den in TET integrierten Regeln zur Interpretation numerischer Glyphnamen können Sie in *TET\_open\_document()* mit den Unteroptionen *fontname* und *glyphrule* der Option *glyphmapping* eigene Regeln definieren. Dazu müssen Sie folgende Daten übergeben:

- ▶ Den vollständigen Namen oder Kurznamen des Fonts, auf den die Regel angewendet wird (Option *fontname*)
- ▶ Ein Präfix für den Glyphnamen, d.h. die Zeichen vor dem numerischen Teil (Unteroption *prefix*)
- ▶ Die (dezimale oder hexadezimale) Basis, auf der die Zahlen interpretiert werden (Unteroption *base*)
- ▶ Das Encoding, in dem die numerischen Codes interpretiert werden sollen (Option *encoding*)

Wenn Sie zum Beispiel (vielleicht mit PDFlib FontReporter) herausgefunden haben, dass die Glyphen in den Fonts *T1*, *T2*, *T3* usw. mit *co0*, *co1*, *co2*, ..., *cFF* bezeichnet werden, wobei

jeder Glyphenname einem WinAnsi-Zeichen der gegebenen hexadezimalen Position (00, ..., FF) entspricht, verwenden Sie für `TET_open_document()` folgende Option:

```
glyphmapping {{fontname=T* glyphrule={prefix=c base=hex encoding=winansi} }}
```

**Externe Fontdateien und Systemfonts.** Wenn ein PDF nicht ausreichend Informationen für die Unicode-Zuordnung enthält und der Font nicht eingebettet ist, können Sie zusätzliche Fontdaten konfigurieren, aus denen TET die Unicode-Zuordnung ableitet. Die Fontdaten können aus einer TrueType- oder OpenType-Fontdatei auf der Festplatte bezogen werden, die mit der Ressourcenkategorie *fontoutline* konfigurierbar ist. Zudem kann TET unter OS X und Windows auf die im Betriebssystem installierten Fonts zugreifen. Mit der Option *usehostfonts* von `TET_open_document()` lässt sich der Zugriff auf die Systemfonts deaktivieren.

Mit folgendem Aufruf konfigurieren Sie eine Datei für den Font *WarnockPro*:

```
set_option("fontoutline {WarnockPro WarnockPro.otf}");
```

Für weitere Informationen zur Konfigurierung von externen Fontdateien siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70.



# 8 Extraktion von Rasterbildern

## 8.1 Grundlagen der Bildextraktion

**Rasterbildformate.** TET extrahiert Rasterbilder aus PDF-Dokumenten und speichert die extrahierten Bilder in einem der folgenden Formate:

- ▶ Im Allgemeinen werden Bilder im Format TIFF (*.tif*) erzeugt. Die meisten von TET erzeugten TIFF-Bilder können von der Mehrzahl der TIFF-Viewer und -Anwendungen verwendet werden. Einige fortgeschrittene TIFF-Features werden jedoch nicht von allen Viewern unterstützt, vor allem zusätzliche Schmuckfarbkanäle (siehe »Schmuckfarben«, Seite 142). Als Maßstab für die Gültigkeit von TIFF-Bildern nehmen wir Adobe Photoshop. Beachten Sie, dass der Image-Viewer von Windows XP keine TIFF-Bilder mit Flate-Kompression unterstützt. Diese Einschränkung bei Viewern können Sie durch Aktivieren der LZW-Kompression mit der Option *preferred-tiffcompression=lzw* von *TET\_write\_image\_file()* oder *TET\_get\_image\_data()* umgehen.
- ▶ Das Format JPEG (*.jpg*) wird für Bilder verwendet, die mit dem JPEG-Algorithmus (*DCTDecode*-Filter) in PDF komprimiert sind. JPEG-komprimierte Bilddaten im PDF-Dokument werden validiert, sofern die Validierung nicht mit der Option *validate-jpeg=false* von *TET\_write\_image\_file()* oder *TET\_get\_image\_data()* deaktiviert wurde, wodurch sich die Verarbeitung etwas beschleunigen lässt. In manchen Fällen müssen DCT-komprimierte Bilder als TIFF extrahiert werden, da nicht alle PDF-Farbräume in JPEG ausgedrückt werden können (z.B. Schmuckfarben).
- ▶ Das Format JPEG 2000 wird für Bilder verwendet, die mit dem JPEG-2000-Algorithmus (*JPXDecode*-Filter) in PDF komprimiert sind. JPEG-2000-Bilder gibt es in verschiedenen Varianten. Die Hauptvariante mit dem MIME-Typ *image/jp2* und der Dateinamenserweiterung *.jp2* ist gemäß ISO 15444-1 (Annex I) kodiert. Die erweiterte Variante mit dem MIME-Typ *image/jpx* ist gemäß ISO 15444-2 (Annex M) kodiert. Sie unterstützt zusätzliche Features wie CMYK- und Lab-Farbe und verwendet als Dateinamenserweiterung *.jpf* (beachten Sie den Unterschied zwischen dem MIME-Typ und dem empfohlenen Suffix). Schließlich enthalten reine JPEG-2000-Codestreams nur die bloßen Pixeldaten ohne zusätzliche Eigenschaften, wie Farbrauminformationen. Sie werden mit der Dateinamenserweiterung *.jzk* extrahiert. Anwendungen, die JPEG-2000-Ausgabe nicht verarbeiten können, können dieses Extraktionsformat mit der Dokumentoption *allowjpeg2000=false* vermeiden. In diesem Fall werden anstelle von JPEG-2000-Bildern 8-Bit- oder 16-Bit-TIFF-Bilder erzeugt, was zu einer größeren Ausgabe führen kann. TIFF-Bilder für JPX-komprimierte Daten werden auch erzeugt, wenn Sonderfarbinformationen erhalten werden müssen oder bei der Bildzusammensetzung. Wenn ein JPX-komprimiertes Bild als TIFF extrahiert wird, werden implizite interne ICC-Profile im JPX-Stream ignoriert. So werden beispielsweise sRGB-Bilder vom Typ JPEG 2000 als reines RGB-TIFF extrahiert.
- ▶ Das Format JBIG2 (*.jbig2*) wird für Bilder verwendet, die mit dem JBIG2-Algorithmus (*JBIG2Decode*-Filter) in PDF komprimiert sind. JBIG2-Dateien werden gemäß ISO 14492 mit »sequentieller Anordnung« erzeugt.

**Extrahieren von Bildern auf die Festplatte oder in den Arbeitsspeicher.** Mit dem TET-API können aus PDF-Dokumenten extrahierte Bilder auf zwei Arten gespeichert werden:

- ▶ Mit `TET_write_image_file()` wird eine Bilddatei auf der Festplatte erzeugt. Der Basisname dieser Bilddatei muss in der Option `filename` angegeben werden. Abhängig vom Bildformat wird das passende Dateinamenserweiterung automatisch angefügt.
- ▶ Mit `TET_get_image_data()` wird eine Bilddatei im Arbeitsspeicher erzeugt. Dies ist besonders nützlich, wenn die Datei gleich weiterverarbeitet werden soll.

Details hängen von Ihren Anforderungen an die Bildextraktion ab (siehe Abschnitt »Seitenbasierte und ressourcenbasierte Bildextraktion«, Seite 135). In beiden Fällen lässt sich der Typ des extrahierten Bildes ermitteln (siehe unten).

**Ermitteln von Dateiformat und -namen extrahierter Bilder.** Das Format einer Bilddatei wird mit dem Attribut `Image/@extractedAs` in TETML ausgegeben. Auf API-Ebene können Sie mit dem folgenden Code den Typ des extrahierten Bildes ermitteln:

```
int imageType = tet.write_image_file(doc, tet.imageid, "typeonly");
```

```
/* Numerischer Bildtyp wird einem Formatsuffix zugeordnet */
String imageSuffix;
switch (imageType) {
case 10:
    imageSuffix = ".tif";
    break;

case 20:
    imageSuffix = ".jpg";
    break;

case 31:
    imageSuffix = ".jp2";
    break;

case 32:
    imageSuffix = ".jpf";
    break;

case 33:
    imageSuffix = ".j2k";
    break;

case 50:
    imageSuffix = ".jbig2";
    break;

default:
    System.err.println("write_image_file() returned unknown value "
        + imageType + ", skipping image, error: "
        + tet.get_errmsg());
}
```

Der Name einer Bilddatei wird mit dem Attribut `Image/@filename` in TETML ausgegeben. Auf API-Ebene können Sie den Namen der Bilddatei an `TET_write_image_file()` übergeben.

Für die vom TET-Kommandozeilen-Tool erzeugte Struktur des Bilddateinamens siehe Abschnitt »Kommandozeilen-Optionen«, Seite 19.

**XMP-Metadaten für Bilder.** In PDF können dem gesamten Dokument oder Teilen davon mit Hilfe des XMP-Formats Metadaten hinzugefügt werden. Für weitere Informationen zu XMP und seiner Verwendung in PDF siehe [www.pdflib.com/knowledge-base/xmp-metadata/](http://www.pdflib.com/knowledge-base/xmp-metadata/).

In einem PDF-Dokument können auch einem Bildobjekt XMP-Metadaten zugeordnet sein. Sie können das Vorhandensein des Bild-XMP in Acrobat XI/DC folgendermaßen prüfen:

- ▶ *Anzeige, Ein-/Ausblenden, Navigationsfenster, Inhalt.*
- ▶ Navigieren Sie in der Baumstruktur zum gewünschten Bild, rechtsklicken Sie darauf und wählen Sie *Metadaten anzeigen...*
- ▶ Das Bild wird markiert und die XMP-Registerkarte mit den XMP-Metadaten für das ausgewählte Bild angezeigt.

Sind XMP-Metadaten vorhanden, werden sie von TET für die Ausgabeformate JPEG und TIFF standardmäßig in die extrahierten Bilder eingebettet. Dieses Verhalten lässt sich mit der Option *keepxmp* von *TET\_write\_image\_file()* und *TET\_get\_image\_data()* steuern. Wird diese Option auf *false* gesetzt, ignoriert TET die Bild-Metadaten bei der Erzeugung der Bildausgabedatei.

Sind Bild-Metadaten vorhanden, hängt TET in der TETML-Ausgabe ein *Metadata*-Element an das Bild an. Dieses Verhalten lässt sich mit der Bildoption *tetml={elements={metadata}}* steuern.

Das Topic *image\_metadata* im pCOS Cookbook zeigt die direkte Extraktion von Metadaten mit pCOS ohne Erzeugung einer Bilddatei.

TET implementiert eine spezielle Heuristik für XMP Bild-Metadaten, die das normale PDF-Verfahren zum Anfügen von XMP an ein Bildobjekt umgehen, sondern stattdessen eine alternative Methode verwenden, die auf *marked content properties* basiert. Dieses Konstrukt wird typischerweise von Adobe InDesign erzeugt. Beachten Sie, dass diese Art von XMP auf Bildebene in pCOS nicht verfügbar ist, sondern nur in TETML und den extrahierten Bilddateien.

**Einschränkungen.** Manchmal scheint die Form der extrahierten Bilder von der Form auf der PDF-Seite abzuweichen:

- ▶ Bilder können entweder horizontal oder vertikal gespiegelt erscheinen. Das liegt daran, dass TET die ursprünglichen Pixeldaten des Bildes extrahiert, ohne die Transformation zu berücksichtigen, die eventuell auf der PDF-Seite auf das Bild angewendet wurde.
- ▶ Maskierungseffekte durch die Anwendung einer Transparenzmaske auf ein anderes Bild sind im extrahierten Bild nicht sichtbar. Sie können die Bildmaske jedoch als separates Bild extrahieren.

Inline-Bilder können nicht extrahiert werden, d.h. *TET\_write\_image\_file()* gibt -1 zurück: Bei Inline-Bildern handelt es sich um einen seltenen PDF-Bildtyp, sie werden manchmal für kleine Rasterbilder oder Glyphen in Type-3-Fonts verwendet.

## 8.2 Bildextraktion

### 8.2.1 Platzierte Bilder und Bild-Ressourcen

TET unterscheidet zwischen platzierten Bildern und Bild-Ressourcen.

- ▶ Ein *platziertes Bild* entspricht einem Bild auf einer Seite. Ein platziertes Bild besitzt geometrische Eigenschaften: es ist an einer bestimmten Stelle platziert und hat eine bestimmte Größe (gemessen in Punkten, Millimetern oder einer anderen Einheit). Meistens ist das Bild auf der Seite sichtbar; in einigen Fällen kann es aber auch unsichtbar sein, weil es von anderen Objekten auf der Seite verdeckt wird, außerhalb des sichtbaren Seitenbereichs angeordnet oder ganz bzw. teilweise abgeschnitten ist usw. Platzierte Bilder werden durch das Element *PlacedImage* in TETML dargestellt. Die Verarbeitung platzierter Bilder ist abhängig von den Optionen *clippingarea*, *excludebox* und *includebox*.
- ▶ Eine *Bild-Ressource* ist eine Ressource, die die eigentlichen Pixeldaten, den Farbraum, die Anzahl der Komponenten und Anzahl von Bits pro Komponente usw. darstellt. Anders als platzierte Bilder haben Bild-Ressourcen keine eigene Geometrie, verfügen jedoch über Breiten- und Höhenangaben (gemessen in Pixeln). Jede Bild-Ressource besitzt eine eindeutige ID, mit deren Hilfe ihre Pixeldaten extrahiert werden können. Bild-Ressourcen werden durch das Element *Image* in TETML dargestellt. Die Verarbeitung platzierter Bild-Ressourcen ist nicht abhängig von den Optionen *clippingarea*, *excludebox* und *includebox*.

Eine Bild-Ressource kann als Grundlage für eine beliebige Anzahl von platzierten Bildern im Dokument verwendet werden. Im Allgemeinen wird jede Bild-Ressource genau einmal platziert, sie könnte jedoch auch mehrfach auf der selben Seite oder auf mehreren Seiten platziert werden. Denken Sie zum Beispiel an ein Firmenlogo, das immer wieder auf der Kopfzeile jeder Seite im Dokument verwendet wird. Jedes Logo auf einer Seite stellt ein platziertes Bild dar, aber in einer optimierten PDF-Datei können all diese platzierten Bilder von der selben Bild-Ressource erzeugt werden. In einer nicht optimierten PDF-Datei dagegen würde jedes platzierte Logo auf einer eigenen Kopie der gleichen Bild-Ressource basieren. Dies hätte zwar den gleichen visuellen Effekt, führt aber zu einem wesentlich größeren PDF-Dokument. Nicht optimierte PDF-Dokumente können sogar Bild-Ressourcen enthalten, die auf überhaupt keiner Seite referenziert werden (d.h. ungenutzte Ressourcen).

In Tabelle 8.1 werden verschiedene Aspekte von platzierten Bildern und Bild-Ressourcen verglichen.

Tabelle 8.1 Unterschiede zwischen platzierten Bildern und Bild-Ressourcen

Eigenschaft	platzierte Bilder	Bild-Ressourcen
TETML-Element	PlacedImage	Image
von der Bildzusammensetzung betroffen	ja	ja
einer Seite zugeordnet	ja	-
Breite und Höhe in Pixeln	ja	ja
Breite und Höhe in Punkten	ja	-

Tabelle 8.1 Unterschiede zwischen platzierten Bildern und Bild-Ressourcen

Eigenschaft	platzierte Bilder	Bild-Ressourcen
Bildauflösung kann bestimmt werden	ja	–
Position auf der Seite	ja	–
Anzahl des Auftretens	1	0, 1 oder mehr
Eindeutige ID	nein: die Option <code>imageid</code> , die von <code>TET_get_image_info()</code> zurückgegeben wird und das Attribut <code>PlacedImage/@image</code> in TETML identifizieren die zugrunde liegende Bild-Ressource	ja, nämlich: das Feld <code>imageid</code> , das von <code>TET_get_image_info()</code> zurückgegeben wird und das Attribut <code>Image/@id</code> in TETML
Dateinamen-Konventionen im TET-Kommandozeilen-Tool	<code>&lt;filename&gt;_p&lt;pagenumber&gt;_&lt;imagenumber&gt;.[tif jpg jp2 jpf j2k jbig2]</code>	<code>&lt;filename&gt;_I&lt;imageid&gt;.[tif jpg jp2 jpf j2k jbig2]</code>
Verarbeitung von Bildmasken im TET-Kommandozeilen-Tool	Masken werden extrahiert als <code>&lt;filename&gt;_p&lt;pagenumber&gt;_&lt;imagenumber&gt;_mask.[tif jpg jp2 jpf j2k jbig2]</code>	Masken werden gemäß ihrer eigenen Bild-ID ohne zusätzliche Labels im Dateinamen extrahiert

**Wie viele Bilder gibt es in einem Dokument?** Auf diese Frage gibt es erstaunlicherweise keine einfache Antwort. Die Antwort hängt von folgenden Entscheidungen ab:

- ▶ Möchten Sie die Anzahl von Bild-Ressourcen oder von platzierten Bildern abfragen?
- ▶ Sollen auch Bilder berücksichtigt werden, die nur als Bestandteil zusammengesetzter Bilder verwendet, aber nie einzeln platziert werden?
- ▶ Sollen auch Bilder berücksichtigt werden, die nur als Masken verwendet werden?

Mit den TET- und pCOS-Pseudo-Objekten können Sie alle Varianten dieser und anderer Fragen zur Anzahl von Bildern beantworten. Das Topic `image_count` im TET Cookbook stellt verschiedene Möglichkeiten der Bildermittlung dar. Es erzeugt Ausgabe ähnlich der folgenden Zeilen:

```
No of raw image resources before merging: 82
No of placed images: 12
No of images after merging (all types): 83
  normal images: 1
  artificial (merged) images: 1
  consumed images: 81
No of relevant (normal or artificial) image resources: 2
```

## 8.2.2 Seitenbasierte und ressourcenbasierte Bildextraktion

Aus den Unterschieden zwischen platzierten Bildern und Bild-Ressourcen ergeben sich zwei grundsätzlich verschiedene Ansätze zur Bildextraktion: seitenbasierte und ressourcenbasierte Schleifen zur Bildextraktion. Mit beiden Methoden lassen sich Bilder auf die Festplatte oder in den Arbeitsspeicher extrahieren.

**Seitenbasierte Schleife zur Bildextraktion.** Hierbei liegt der Fokus der Anwendung auf dem genauen Seitenlayout und den platzierten Bildern. Ob dabei Bilddaten mehrfach ausgegeben werden, spielt keine Rolle. Bildextraktion anhand von seitenbasierten Schleifen erzeugt für jedes platzierte Bild eine Bilddatei und kann zu identischen Bilddaten für mehrere platzierte Bilder führen. Die Anwendung könnte die Duplizierung von Bildern zwar durch Überprüfung auf doppelte Bild-IDs vermeiden. Eindeutige Bild-

Ressourcen lassen sich jedoch einfacher mit der ressourcenbasierten Schleife zur Bildextraktion extrahieren (siehe unten).

Die seitenbasierte Schleife zur Bildextraktion lässt sich im TET-Kommandozeilen-Tool mit der Option `--imageloop page` aktivieren. API-Codebeispiele für die seitenbasierte Bildextraktion finden Sie im Cookbook-Topic und Minibeispiel *images\_per\_page*. Diese Beispiele zeigen auch, wie die Bildgeometrie abgefragt werden kann.

Einzelheiten der seitenbasierte Schleife zur Bildextraktion (siehe die oben aufgeführten Codebeispiele): mit `TET_get_image_info()` lassen sich geometrische Angaben zum platzierten Bild sowie die pCOS-ID für das Bild (im Feld *imageid*) der zugrunde liegenden Bilddaten abfragen. Mit dieser ID lassen sich weitere Bildinformationen mit `TET_pcos_get_number()` abfragen, wie z.B. der Farbraum und die Breite und Höhe in Pixeln, sowie mit `TET_write_image_file()` oder `TET_get_image_data()` die eigentlichen Pixeldaten. Mit `TET_get_image_info()` bleiben die eigentlichen Pixeldaten des Bildes unverändert. Wird das selbe Bild auf einer oder mehreren Seiten mehrfach referenziert, sind die zugehörigen IDs identisch.

**Ressourcenbasierte Schleife zur Bildextraktion.** Hierbei liegt der Fokus der Anwendung auf den Bild-Ressourcen des Dokuments, nicht jedoch darauf, welches Bild auf welcher Seite verwendet wird. Mehrfach platzierte Bild-Ressourcen (auf einer oder mehreren Seiten) werden nur einmal extrahiert. Überhaupt nicht platzierte Bild-Ressourcen werden dabei ebenfalls extrahiert.

Die ressourcenbasierte Schleife zur Bildextraktion lässt sich im TET-Kommandozeilen-Tool mit der Option `--imageloop resource` aktivieren. API-Codebeispiele für die ressourcenbasierte Bildextraktion finden Sie im Minibeispiel und Cookbook-Topic *image\_resources*.

Einzelheiten der ressourcenbasierten Schleife zur Bildextraktion (siehe die oben aufgeführten Codebeispiele): alle Seiten werden vor der Bildextraktion geöffnet, um sicherzustellen, dass das Zusammensetzen von Bildern aktiviert ist; ist dies nicht relevant, kann dieser Schritt übersprungen werden. Zur Extraktion eines Bildes wird die entsprechende Bild-ID benötigt. Der Code listet alle Werte von 0 bis zur höchsten Bild-ID auf, die folgendermaßen abgefragt werden kann:

```
n_images = (int) tet.pcos_get_number(doc, "length:images");
```

Um verarbeitete Teile eines zusammengesetzten Bildes (z.B. die Streifen eines Bildes vom Typ *multi-strip*) zu überspringen, wird der Typ jeder Bild-Ressource mit dem pCOS-Pseudo-Objekt *mergetype* überprüft. Damit lassen sich bei der Bildzusammensetzung in größere Bilder überführte Bilder überspringen (da an dieser Stelle nur das resultierende zusammengesetzte Bild von Interesse ist). Sobald eine Bild-ID ermittelt wurde, kann eine der Funktionen `TET_write_image_file()` oder `TET_get_image_data()` aufgerufen werden, um die Bilddaten auf die Festplatte oder in den Arbeitsspeicher zu schreiben.

## 8.2.3 Geometrie von platzierten Bildern

Mit `TET_get_image_info()` lassen sich geometrische Angaben zu einem platzierten Bild abfragen. Die folgenden Werte sind für jedes Bild in der Struktur *image\_info* verfügbar (siehe Abbildung 8.1):

- ▶ Die Felder *x* und *y* bilden die Koordinaten des Referenzpunkts für das Bild. Der Referenzpunkt ist die untere linke Ecke des Bildes. Transformationen des Koordinatensystems auf der Seite können jedoch zu einem anderen Referenzpunkt führen. Das

Bild kann zum Beispiel horizontal gespiegelt sein, was dazu führt, dass die obere linke Ecke zum Referenzpunkt wird. Der Wert von  $y$  hängt von der Seitenoption *topdown* ab.

- ▶ Die Felder *width* und *height* entsprechen den physikalischen Dimensionen des platzierten Bildes auf der Seite. Sie werden in Punkt angegeben (d.h. 1/72 Zoll).
- ▶ Der Winkel *alpha* beschreibt die Richtung der Pixelreihen. Der Winkel liegt im Bereich  $-180^\circ < \alpha \leq +180^\circ$ . Der Winkel *alpha* dreht das Bild an seinem Referenzpunkt. Für aufrecht stehende Bilder ist *alpha* gleich  $0^\circ$ . Die Werte von *alpha* und *beta* hängen von der Seitenoption *topdown* ab.
- ▶ Der Winkel *beta* beschreibt die Richtung der Pixelspalten, relativ zur Senkrechten von *alpha*. Der Winkel liegt im Bereich  $-180^\circ < \beta \leq +180^\circ$ , ist aber nicht  $\pm 90^\circ$ . Der Winkel *beta* schert das Bild und *beta*= $180^\circ$  spiegelt das Bild an der  $x$ -Achse. Für aufrecht stehende Bilder liegt *beta* im Bereich  $-90^\circ < \beta < +90^\circ$ . Ist  $abs(\beta) > 90^\circ$ , wird das Bild an der Grundlinie gespiegelt.
- ▶ Das Feld *imageid* enthält die pCOS-ID für das Bild. Mit dieser ID lassen sich weitere Bildinformationen mit pCOS-Funktionen abfragen, sowie mit *TET\_write\_image\_file()* oder *TET\_get\_image\_data()* die Pixeldaten des Bildes.

Infolge von Bildtransformationen kann die Ausrichtung der extrahierten Bilder falsch erscheinen, da die extrahierten Bilddaten auf der Bild-Ressource im PDF-Dokument basieren. Jegliche auf das platzierte Bild angewendeten Drehungen oder Spiegelungen auf der PDF-Seite werden nicht auf die extrahierten Pixeldaten angewendet, vielmehr werden die ursprünglichen Pixeldaten extrahiert.

**Koordinatenberechnung aller Bildecken.** Die mit *TET\_get\_image\_info()* abgefragten Felder *x* und *y* geben die Koordinaten des Referenzpunkts des Bildes an, der sich oft in der unteren linken Ecke des Bildes befindet. Mit den Bildwerten *x/y*, *width/height* und *alpha/beta* lassen sich die Koordinaten aller Bildecken wie folgt berechnen:

$$ll_x = x$$

$$ll_y = y$$

$$lr_x = x + width * \cos(\alpha)$$

$$lr_y = y + width * \sin(\alpha)$$

$$ul_x = x + dir * height * (\tan(\beta) * \cos(\alpha) - \sin(\alpha))$$

$$ul_y = y + dir * height * (\tan(\beta) * \sin(\alpha) + \cos(\alpha))$$

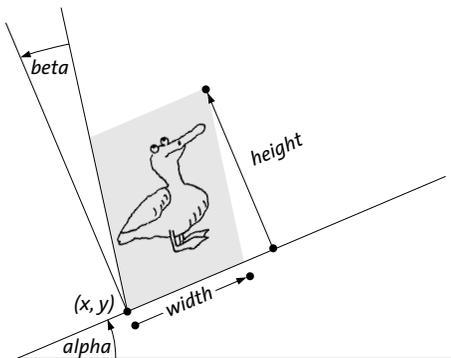


Abb. 8.1  
Bild-Geometrie

```
ur_x = x + width * cos(alpha) + dir * height * (tan(beta)*cos(alpha) - sin(alpha))
ur_y = y + width * sin(alpha) + dir * height * (tan(beta)*sin(alpha) + cos(alpha))
```

mit  $dir=1$  im Standardfall  $topdown=\{output=false\}$ . In Topdown-Koordinaten, d.h., wenn  $topdown=\{output=true\}$  (siehe »Top-Down-Koordinatensystem«, Seite 84), müssen Sie  $dir=-1$  setzen, womit die Ecken vertauscht werden, d.h.  $ll$  muss mit  $ul$  getauscht werden und  $lr$  mit  $ur$ .

**Bildauflösung.** Zur Berechnung der Bildauflösung in dpi (dots per inch) müssen Sie die Bildbreite in Pixeln durch die Bildbreite in Punkt teilen und mit 72 multiplizieren:

```
while (tet.get_image_info(page) == 1) {
    String imagePath = "images[" + tet.imageid + "]";
    int width = (int) tet.pcos_get_number(doc, imagePath + "/Width");
    int height = (int) tet.pcos_get_number(doc, imagePath + "/Height");

    double xDpi = 72 * width / tet.width;
    double yDpi = 72 * height / tet.height;
    ...
}
```

Beachten Sie, dass dpi-Werte für gedrehte oder gescherte Bilder sinnlos sein können. Den vollständigen Code zur dpi-Berechnung für ein Bild finden Sie im Topic *determine\_image\_resolution* im TET Cookbook.

TET speichert in erzeugten TIFF-Bildern generell eine Dummy-Auflösung von 72 dpi, um der TIFF-Spezifikation zu genügen. Mit der Option *dpi* von *TET\_write\_image\_file()* lassen sich Werte für die Auflösung in den generierten TIFF-Bildern speichern. Diese können von TET nicht automatisch eingebettet werden, da ein bestimmtes Bild mehrfach platziert worden sein kann, jedes Mal in unterschiedlicher Größe und damit anderer Auflösung. Mit dem Wert  $dpi=0$  lassen sich die Dummy-Werte für die Auflösung unterdrücken.

Das TET-Kommandozeilen-Tool speichert berechnete Auflösungs-werte bei der seitenbasierten Bildextraktion.

## 8.3 Zusammensetzen fragmentierter Bilder

In manchen Fällen ist es nicht sinnvoll, Bilder exakt so wie im PDF-Dokument dargestellt zu extrahieren: denn was als einzelnes Bild erscheint, ist in Wirklichkeit oft eine Vielzahl von nebeneinander liegenden Bildfragmenten. Bildfragmentierung im PDF kann häufig aus folgenden Gründen passieren:

- ▶ Einige Anwendungen und Treiber konvertieren TIFF-Bilder mit mehreren Streifen (multi-strip) in fragmentierte PDF-Bilder. Die Anzahl der Streifen kann Dutzende bis Hunderte betragen.
- ▶ Scan-Software unterteilt gescannte Dokumente in kleinere Fragmente (Streifen oder Kacheln). Die Anzahl der Streifen beträgt meist nicht mehr als ein paar Dutzend.
- ▶ Manche Anwendungen zerlegen Bilder beim Druck oder bei der PDF-Ausgabe in kleine Einheiten. Im Extremfall, vor allem bei Dokumenten, die mit Microsoft Office erzeugt wurden, kann eine Seite aus Tausenden kleiner Bildfragmente bestehen.
- ▶ Manche Seitenlayout-Programme wie Adobe InDesign zerschneiden Bilder für die PDF-Ausgabe in kleinere und manchmal unregelmäßige Fragmente (siehe Abbildung 8.2).

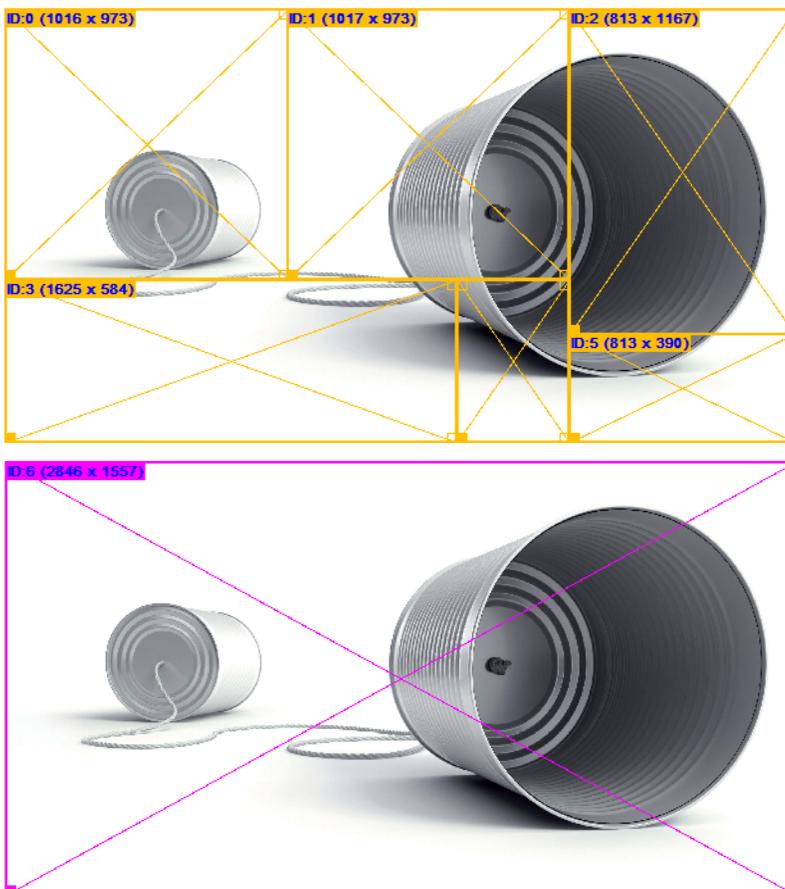


Abb. 8.2  
Obwohl dieses Bild in kleinere Teile unterteilt ist (oben), extrahiert TET es als ganzes wiederverwendbares Bild (unten).

Beim Zusammensetzen von Bildern erkennt TET diese Fälle und setzt die Einzelbilder wieder zu einem brauchbaren Ganzen zusammen. Können die Komponenten zu einem größeren Bild kombiniert werden, werden sie zusammengesetzt.

Mit der folgenden Seitenoption lässt sich das Zusammensetzen von Bildern deaktivieren:

```
imageanalysis={merge={disable}}
```

**Zusammengesetzte Bilder in pCOS.** Zusammengesetzte Bilder können mit dem pCOS-Pseudo-Objekt *images[ ]/mergetype* identifiziert werden: der Wert 1 (künstlich) steht für zusammengesetzte Bilder und der Wert 2 (verarbeitet) steht für Bilder, die in ein größeres Bild überführt wurden. Verarbeitete Bilder sollten von der empfangenden Anwendung ignoriert werden.

Um Ungenauigkeiten bei den Bildpositionen zu kompensieren, sind etwas Abstand oder Überlappungen zwischen benachbarten Bildern erlaubt. Standardmäßig werden Bilder zusammengesetzt, wenn die Lücke oder Überlappung kleiner als ein Punkt ist. Der Wert kann mit folgender Seitenoption geändert werden:

```
imageanalysis={merge={gap=2}}
```

Bei der Extraktion von Bildern aus Zeitungen oder Zeitschriften sind oft höhere Werte für Lücken/Überlappungen erforderlich

**Wann werden Bilder zusammengesetzt?** Analyse und Zusammensetzen von Bildern auf einer Seite werden durch den entsprechenden Aufruf von *TET\_open\_page()* gestartet. Dies hat folgende Auswirkungen:

- ▶ Die Anzahl der Einträge im pCOS-Array *images[ ]*, d.h. der Wert des Pseudo-Objekts *length:images* kann sich erhöhen: je mehr Seiten verarbeitet werden, desto mehr aus dem Zusammensetzen resultierende künstliche Bilder werden dem Array hinzugefügt. Um alle zusammengesetzten Bilder zu extrahieren, müssen Sie zuerst alle Seiten im Dokument öffnen, bevor Sie *length:images* abfragen und die Bilddaten extrahieren. Künstliche (zusammengesetzte) Bilder werden als *artificial* (numerischer Wert 1) im Pseudo-Objekt *images[ ]/mergetype* gekennzeichnet.
- ▶ Auf der anderen Seite können manche Elemente im Array *images[ ]* nur als Teile zusammengesetzter Bilder verarbeitet worden sein und nicht als ganze Bilder. Solche Einträge werden aus dem Array *images[ ]* nie entfernt, die Einträge für verarbeitete Bilder werden jedoch als *consumed* (numerischer Wert 2) im Pseudo-Objekt *images[ ]/mergetype* entsprechend gekennzeichnet.

## 8.4 Filtern von Bildfragmenten

TET ignoriert sehr kleine Bilder, da sie oft irrelevant oder unbrauchbar sind. Da beim Zusammensetzen von Bildern oft viele Einzelbilder zu einem größeren Bild zusammengefasst werden, werden die Bildfragmente erst danach entfernt. Dabei kommen nur Bildfragmente in Betracht, die nach dem Zusammensetzen des Bildes noch übrig geblieben sind. Diese müssen außerdem die Bedingungen für Höhe, Breite und Fläche erfüllen, die in den Unteroptionen *maxheight*/*maxwidth*/*maxarea* der Unteroption *smallimages* der Seitenoption *imageanalysis* angegeben werden. Mit der folgenden Seitenoption lässt sich die Entfernung von kleinen Bildfragmenten vollständig deaktivieren:

```
imageanalysis={smallimages={disable}}
```

**Kleine Bildfragmente in pCOS.** Bilder, die gemäß der Option *smallimages* als klein eingestuft werden, werden von *TET write\_image\_file()* und *TET get\_image\_data()* ignoriert, sind aber im pCOS-Array *images[ ]* immer noch vorhanden. Sie können mit dem pCOS-Pseudo-Objekt *images[ ]/small* identifiziert werden.

## 8.5 Bildfarben und Masken

### 8.5.1 Farbräume

**Farbgetreue Wiedergabe von Bildern.** Tabelle 6.1 gibt einen Überblick über PDF-Farbräume. Für Bilder werden alle Farbräume unterstützt. Bei der Bildextraktion setzt TET die Bildqualität nicht herab:

- ▶ Die Auflösung von Rasterbildern wird nie reduziert (kein Downsampling).
- ▶ Der Farbraum eines Bildes bleibt in der Ausgabe erhalten. Außerdem wird auf Farbkonvertierung verzichtet, z.B. von CMYK nach RGB o.ä.

**ICC-Profil.** Zu einem Bild in einem PDF-Dokument kann ein ICC-Profil gehören, das die präzise Farbwiedergabe ermöglicht. Standardmäßig verarbeitet TET zugehörige ICC-Profile und bettet sie in die erzeugte TIFF- oder JPEG-Bilddatei ein. Die Einbettung eines ICC-Profiles lässt sich mit der Option `keepiccprofile=false` von `TET_write_image_file()` und `TET_get_image_data()` deaktivieren. Dadurch wird die Größe der Bilddatei auf Kosten der Farbtreue reduziert. Wenn es bei Ihrer Anwendung also auf die farbgetreue Wiedergabe ankommt, sollten Sie das Einbetten von ICC-Profilen aktiviert lassen.

**Schmuckfarben.** In PDF können Bilder mit einer benannten Farbe eingefärbt werden. Normalerweise werden benannte Farben verwendet, um benutzerdefinierte Schmuckfarben auszugeben, aber der gleiche Mechanismus kann auch verwendet werden, um eine Teilmenge von CMYK-Prozessfarben auf ein Bild anzuwenden (z.B. nur die Kanäle Cyan und Magenta). Der *Separation*-Farbraum in PDF umfasst eine einzige benannte Farbe, während sich mit dem *DeviceN*-Farbraum mehrere benannte Farben zuweisen lassen. Separation-Farben werden durch eine so genannte Alternativfarbe ergänzt, mit der die Farbe auch dann dargestellt werden kann, wenn die Schmuckfarbe nicht verfügbar ist (z.B. auf einem Bildschirm). Bei einer Separation-Farbe namens *Company Red* ist es zum Beispiel sinnvoll, eine Alternativfarbe in einem gängigen Farbraum wie RGB oder CMYK zu haben, um die Schmuckfarbe auch auf Geräten ausgeben zu können, auf denen *Company Red* nicht als benannte Farbe verfügbar ist

TET extrahiert Bilder mit *Separation*- oder *DeviceN*-Farben folgendermaßen: Namen von CMYK-Prozessfarben werden identifiziert: gibt es eine benannte Farbe *Black*, wird sie als Prozessfarbe behandelt und das Bild als Graustufen-Bild extrahiert. Die Farbnamen *Cyan*, *Magenta* und *Yellow* werden ebenfalls identifiziert und das Bild wird als CMYK-Bild extrahiert. Benutzerdefinierte Schmuckfarbnamen, d.h. Namen verschieden von *Cyan*, *Magenta*, *Yellow* und *Black* können auf verschiedene Arten verarbeitet werden, abhängig von der Dokumentoption *spotcolor*:

- ▶ Mit `spotcolor=convert` (Standardeinstellung) werden Schmuckfarben, wenn möglich, in den zugehörigen Alternativfarbraum konvertiert. Ist eine solche Konvertierung nicht möglich, verhält sich diese Methode wie `spotcolor=ignore` (bei einer einzelnen benutzerdefinierten Schmuckfarbe) oder wie `spotcolor=preserve` (bei zwei oder mehr benutzerdefinierten Schmuckfarben).
- ▶ Die Option `spotcolor=ignore` ist ähnlich wie `spotcolor=convert`, außer, dass Bilder mit genau einer benutzerdefinierten Schmuckfarbe als Graustufenbild extrahiert werden und der Schmuckfarbname verlorengeht.
- ▶ Mit `spotcolor=preserve` bleiben Schmuckfarbnamen erhalten und das Bild wird als Graustufen- oder CMYK-Bild mit einem oder mehreren zusätzlichen Schmuckfarb-

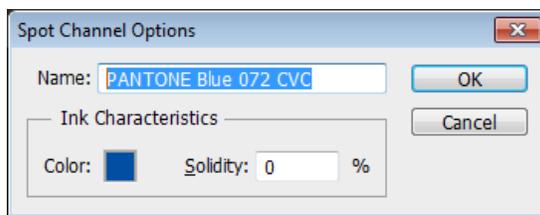
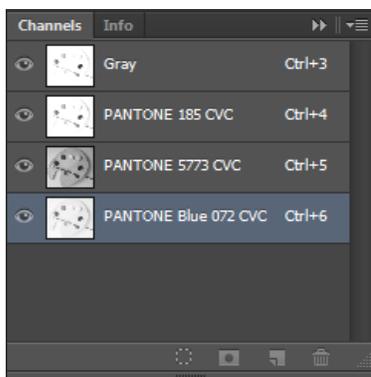


Abb. 8.3  
 Adobe Photoshop zeigt Schmuckfarbkanäle von TIFF-Bildern im Optionsdialog für Farbkanäle an, die mit `spotcolor=preserve` extrahiert wurden (links). Mit einem Doppelklick auf eins der Symbole wird die Alternativfarbe angezeigt (oben).

kanälen ausgegeben. Dies erfordert TIFF-Ausgabe; die erzeugte TIFF-Variante kann mit Adobe Photoshop und anderen kompatiblen Programmen angezeigt werden (siehe Abbildung 8.3). Einfache TIFF-Viewer ignorieren die zusätzlichen Schmuckfarbkanäle.

Tabelle 8.2 fasst die Ausgabeformate für unterschiedliche Kombinationen von Schmuckfarbnamen und Einstellungen der Dokumentoption `spotcolor` zusammen.

Tabelle 8.2 Ausgabeformate für Bilder mit Separation- und DeviceN-Farben

Farbnamen für Separation oder DeviceN	<code>spotcolor=ignore</code>	<code>spotcolor=convert</code>	<code>spotcolor=preserve</code>
nur Black	Graustufen		
eins oder mehr von Cyan, Magenta, Yellow, Black	CMYK (unbenutzte Kanäle sind leer)		
genau eine benutzerdefinierte Schmuckfarbe (d.h. verschieden von Cyan, Magenta, Yellow, Black)	Graustufen	Alternativfarbraum (wenn möglich) <sup>1</sup>	leerer Graustufen-Kanal plus ein benannter zusätzlicher Kanal
zwei oder mehr Farbnamen, alle verschieden von Cyan, Magenta, Yellow	Alternativfarbraum (wenn möglich) <sup>1</sup>		Graustufen-Kanal plus ein oder mehrere benannte zusätzliche Kanäle
zwei oder mehr Farbnamen, einschließlich einem oder mehreren von Cyan, Magenta, Yellow	Alternativfarbraum (wenn möglich) <sup>1</sup>		CMYK plus ein oder mehrere benannte zusätzliche Kanäle

1. Verhält sich wie `spotcolor=ignore` (bei einer einzelnen benutzerdefinierten Schmuckfarbe) oder wie `spotcolor=preserve` (bei zwei oder mehr benutzerdefinierten Schmuckfarben), falls die Konvertierung in den Alternativfarbraum nicht möglich ist.

## 8.5.2 Bildmasken und Transparenzmasken

Mit TET lassen sich Maskierungsinformationen und die zur Maskierung eines anderen Bildes verwendeten Bilddaten abfragen. PDF unterstützt die folgenden Typen von Bildmasken:

- ▶ Eine Stencil-Maske ist ein 1-Bit-Bild mit dem PDF-Schlüsselwort *ImageMask*. Das Bild wird als Schablone (Stencil) verwendet, halb transparent und halb undurchsichtig: Farbe wird eingesetzt, wenn das Bild den Pixelwert 0 hat, und der Hintergrund scheint unverändert durch, wenn das Bild den Pixelwert 1 hat.
- ▶ Eine Maske ist ein Graustufenbild mit einer Tiefe von 1 Bit, das auf ein anderes Bild angewendet wird (PDF-Schlüsselwort *Mask*). Die Maske legt fest, welcher Bildbereich eingefärbt werden und welcher maskiert werden soll (also unverändert bleibt).
- ▶ Eine Transparenzmaske ist ein Graustufenbild mit beliebiger Bit-Tiefe, das auf ein anderes Bild angewendet wird (PDF-Schlüsselwort *SMask*). Sie schafft einen weichen Übergang zwischen dem maskierten Bild und seinem Hintergrund, wodurch ein echter Transparenzeffekt entsteht.

Da Bild- und Transparenzmasken sich nur in der Bit-Tiefe unterscheiden, werden sie in TET gleich behandelt.

**Bildmasken in TETML.** In TETML wird die Maskierung von Bildern folgendermaßen verarbeitet:

- ▶ Stencil-Masken: das TETML-Attribut *Image/@stencilmask* signalisiert, dass ein Bild mit einer Bittiefe von 1 selbst als Stencil-Maske verwendet wird.
- ▶ Bildmasken: das TETML-Attribut *Image/@maskid* referenziert eine Bildmaske (*Mask* oder *SMask*), die an ein Bild angehängt sein kann. Weitere Angaben zum maskierten Bild lassen sich im Eintrag zum maskierten Bild im Array *images[ ]* abfragen.

**Bildmasken im TET-Kommandozeilen-Tool.** Bildmasken werden im TET-Kommandozeilen-Tool folgendermaßen verarbeitet (Informationen zu Stencil-Masken stehen nicht zur Verfügung):

- ▶ Bildextraktion mit *--imagemap page* extrahiert alle normalen Bilder wie üblich. Als Masken in einem der extrahierten normalen Bilder verwendete Bilder werden ebenfalls extrahiert, wobei die Dateinamenserweiterung *\_mask* angehängt wird.
- ▶ Bei der Bildextraktion mit *--imagemap page* werden alle normalen Bilder wie üblich extrahiert. Die generierten Dateinamen enthalten das TETML-Attribut *image/@id* des maskierten Bildes (das identisch ist mit dem Attribut *image/@maskid* des maskierten Bildes) so dass Anwendungen die entsprechenden Dateien für die in TETML referenzierten Bilder finden können.

**Bildmasken in pCOS.** Die Bildmaskierung wird im pCOS-Pseudo-Objekt *images[ ]* und *TET\_pcos\_get\_number()* wie folgt verarbeitet:

- ▶ Als Stencil-Maske verwendete Bilder können mit dem Pseudo-Objekt *images[ ]/stencilmask* identifiziert werden.
- ▶ Bei Bildern mit einer Transparenzmaske hat das zugehörige Pseudo-Objekt *images[ ]/maskid* einen Wert verschieden von -1. Mit dem Wert kann die Bild-ID der Maske bestimmt und weitere Angaben zur Maske mit Hilfe des entsprechenden Eintrags im Array *images[ ]* abgefragt werden.

**Bildmasken im API.** Im TET-API wird die Maskierung von Bildern folgendermaßen verarbeitet:

- ▶ Mit `TET_get_image_info()` werden nur normale, auf der Seite platzierte Bilder aufgelistet und Bildmasken werden übersprungen. Mit dem Feld `imageid` der Struktur `image_info` kann die pCOS-ID des Bildes abgefragt werden, mit der wiederum Informationen zur Bildmaske und Stencil-Maske wie oben beschrieben von pCOS abgefragt werden können.
- ▶ Mit `TET_write_image_file()` und `TET_get_image_data()` können die Pixeldaten der Maske mittels der im pCOS-Objekt `maskid` des maskierten Bildes ermittelten Bild-ID abgefragt werden. Dies wird im Beispiel `images_per_page` dargestellt. Alternativ können Sie alle Einträge im pCOS-Array `images[]` durchlaufen, um Bilddateien für alle normalen und maskierten Bilder zu erstellen. Dies wird im Beispiel `image_resources` dargestellt.





# 9 TET Markup Language (TETML)

## 9.1 Erzeugen von TETML

Mit TET kann der Inhalt eines PDF-Dokuments nicht nur über eine Programmierschnittstelle abgefragt werden, sondern auch als XML-Ausgabe erzeugt werden. Die von TET erzeugte XML-Ausgabe wird als TET Markup Language (TETML) bezeichnet. TETML enthält die Textinhalte der PDF-Seiten sowie optionale Informationen wie Textposition, Font, Fontgröße usw. Wenn TET tabellenartige Strukturen auf der Seite erkennt, werden die Tabellen in TETML als eine Hierarchie von Tabellen-, Zeilen- und Zellenelementen ausgedrückt. Beachten Sie, dass Tabelleninformationen nicht über die TET-Programmierschnittstelle, sondern nur über TETML verfügbar sind. TETML enthält auch Informationen zu Bildern und Farbräumen, sowie Angaben zu Anmerkungen, Formularfeldern, Lesezeichen und anderen interaktiven Elementen.

Sie können PDF-Dokumente mit dem TET-Kommandozeilen-Tool oder der TET-Bibliothek nach TETML konvertieren. In beiden Fällen gibt es verschiedene Optionen für die genaue Steuerung der TETML-Erzeugung.

**Erzeugen von TETML mit dem TET-Kommandozeilen-Tool.** Mit dem TET-Kommandozeilen-Tool können Sie TETML-Ausgabe mit der Option `--tetml` erzeugen. Mit dem folgenden Kommando erzeugen Sie ein TET-Ausgabedokument *file.tetml*:

```
tet --tetml word file.pdf
```

Sie können verschiedene Optionen verwenden, um zum Beispiel nur einige Seiten eines Dokuments zu konvertieren, Verarbeitungsoptionen zu übergeben usw. Für weitere Informationen siehe Abschnitt 2.1, »Kommandozeilen-Optionen«, Seite 19.

**Erzeugen von TETML mit der TET-Bibliothek.** Mit einer einfachen Sequenz von API-Funktionsaufrufen können Sie TETML-Ausgabe mit der TET-Bibliothek erzeugen. Im Mini-Beispiel *tetml* wird die kanonische Sequenz für die Erzeugung von TETML dargestellt. Das Programmbeispiel ist in allen unterstützten Sprachbindungen verfügbar.

TETML wird seitenweise erzeugt, das heißt der Client kann auch lediglich eine Untermenge an Seiten verarbeiten. Der TETML-Trailer muss nach der Verarbeitung der letzten Seite erzeugt werden:

```
final int n_pages = (int) tet.pcos_get_number(doc, "length:pages");
```

```
/* Über alle Seiten im Dokument laufen */
for (int pageno = 1; pageno <= n_pages; ++pageno)
{
    tet.process_page(doc, pageno, pageoptlist);
}
```

```
/* Dies kann mit dem letzten seitenbezogenen Aufruf kombiniert werden */
tet.process_page(doc, 0, "tetml={trailer}");
```

Wurde die Option *filename* an `open_document()` übergeben, wird die TETML-Ausgabe in die angegebene Datei auf der Festplatte geschrieben.

Andernfalls wird TETML im Arbeitsspeicher aufgebaut und kann mit `get_tetml()` geholt werden.

Für den vollständigen TETML-Stream kann dies in einem einzigen Aufruf (nur bei kleinen Dokumenten empfohlen) oder in mehreren Aufrufen erfolgen, wobei jeder Aufruf ein kleineres Fragment des vollständigen TETML-Stream extrahiert. Für den vollständigen TETML-Stream kann dies in einem einzigen Aufruf (nur bei kleinen Dokumenten empfohlen) oder in mehreren Aufrufen erfolgen, wobei jeder Aufruf ein kleineres Fragment des vollständigen TETML-Stream extrahiert.

Der erzeugte TETML-Stream kann mit Hilfe von XML-Funktionen, die die meisten modernen Programmiersprachen bieten, als XML-Baum verarbeitet werden. Die Verarbeitung des TETML-Baums wird in den *tetml*-Programmbeispielen ebenfalls für Programmiersprachen mit integrierter XML-Unterstützung dargestellt..

**Was ist in TETML enthalten?** Die von TET generierte XML-Ausgabe ist in UTF-8 (auf zSeries-Systemen mit USS oder MVS: EBCDIC-UTF-8) kodiert und enthält die folgenden Informationen (einige davon sind optional):

- ▶ allgemeine Dokument-Informationen, Verschlüsselungsstatus, PDF-Standards, Tagged PDF usw.
- ▶ Dokument-Infelder und XMP-Metadaten
- ▶ die Textinhalte jeder Seite (Wörter oder Absätze; optional Zeilen)
- ▶ Font, Geometrie und Farbe der Glyphen
- ▶ Layout-Attribute für die Glyphen (tief-/hochgestellt, mehrzeilige Anfangsbuchstaben (dropcaps), Schattierungen)
- ▶ Attribute zur Beschreibung von Worttrennungen
- ▶ Strukturinformationen, z.B. Tabellen
- ▶ Informationen über platzierte Bilder auf der Seite
- ▶ Ressourcen-Informationen, d.h. Fonts, Farbräume, Rasterbilder, ICC-Profile
- ▶ interaktive Elemente: Lesezeichen, benannte Ziele, Anmerkungen, Formularfelder, Aktionen und JavaScript
- ▶ Anker im Text ermöglichen bequemes Referenzieren von Links, Formularfeldern und Lesezeichenzielen
- ▶ digitale Signaturen
- ▶ Fehlermeldungen, wenn während der PDF-Verarbeitung eine Exception ausgelöst wird

Manche Elemente und Attribute in TETML sind optional. Für weitere Informationen siehe Abschnitt 9.3, »Steuerung von TETML-Informationen«, Seite 153.

## 9.2 TETML-Beispiele

Die TETML-Beispiele unten zeigen einige wichtige Funktionen. Für eine vollständige Liste und Beschreibung aller TETML-Elemente siehe Abschnitt 9.4, »TETML-Elemente und das TETML-Schema«, Seite 157.

**Dokument-Header und Textausgabe.** Das folgende Codefragment zeigt die wichtigsten Teile eines TETML-Dokuments:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Created by the PDFlib Text and Image Extraction Toolkit TET (www.pdflib.com) -->
<TET xmlns="http://www.pdflib.com/XML/TET5/TET-5.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.pdflib.com/XML/TET5/TET-5.0
  http://www.pdflib.com/XML/TET5/TET-5.0.xsd"
  version="5.0">
<Creation platform="Win64" tetVersion="5.0" date="2015-08-05T18:26:02+02:00" />
<Document filename="TET-datasheet.pdf" pageCount="6" filesize="508093" linearized="true"
pdfVersion="1.7">
<DocInfo>
<Author>PDFlib GmbH</Author>
<CreationDate>2015-08-05T17:43:14+02:00</CreationDate>
<Creator>Adobe InDesign CS6 (Windows)</Creator>
<ModDate>2015-08-05T17:43:15+02:00</ModDate>
<Producer>Adobe PDF Library 10.0.1</Producer>
<Subject>PDFlib TET: Text and Image Extraction Toolkit (TET)</Subject>
<Title>PDFlib TET datasheet</Title>
</DocInfo>
<Metadata>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Adobe XMP Core 5.3-c011 66.145661, 2012/02/
06-14:56:27
">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    ...XMP metadata...
  </rdf:RDF>
</x:xmpmeta>
</Metadata>
<Options> tetml={filename={TET-datasheet.word.tetml}}</Options>
<Pages>
<Page number="1" width="595.28" height="841.89">
<Options> granularity=word tetml={}</Options>
<Content granularity="word" dehyphenation="false" dropcap="false" font="false"
geometry="false" shadow="false" sub="false" sup="false">
<Para>
<Box llx="235.80" lly="796.02" urx="397.67" ury="816.72">
  <Word>
    <Text>PDFlib</Text>
    <Box llx="235.80" lly="796.02" urx="291.91" ury="814.02"/>
  </Word>
  <Word>
    <Text>datasheet</Text>
    <Box llx="306.14" lly="796.22" urx="397.67" ury="816.72"/>
  </Word>
</Box>
...weitere Seiteninhalte...
</Content>
```

```

</Page>
...weitere Seiten...
<Resources>
<Fonts>
<Font id="F0" name="TheSans-Plain" fullname="FXLUMY+TheSans-Plain" type="Type 1 CFF"
embedded="true" ascender="1170" capheight="675" italicangle="0" descender="-433"
weight="400" xheight="497"/>
<Font id="F1" name="PDFlibLogo2-Regular" fullname="DUMIKC+PDFlibLogo2-Regular"
type="Type 1 CFF" embedded="true" ascender="800" capheight="700" italicangle="0"
descender="-9" weight="400" xheight="537"/>
...more fonts...
</Fonts>
<Images>
<Image id="I0" filename="TET-datasheet_I0.tif" extractedAs=".tif" width="885"
height="565" colorspace="CS3" bitsPerComponent="8"/>
<Image id="I1" filename="TET-datasheet_I1.tif" extractedAs=".tif" width="1253"
height="379" colorspace="CS4" bitsPerComponent="8"/>
...more images...
</Images>
<ColorSpaces>
<ColorSpace id="CS0" name="DeviceCMYK" components="4"/>
<ColorSpace id="CS1" name="DeviceGray" components="1"/>
...more colorspaces...
</ColorSpaces>
</Resources>
...
</Pages>
</Document>
</TET>

```

**Glyphenkoordinaten und Farbe.** Abhängig vom gewählten TETML-Modus können weitere Glypheninformationen in TETML ausgedrückt werden. Für weitere Informationen zu TETML-Modi siehe »Auswahl des TETML-Modus«, Seite 153. Im folgenden finden Sie eine Variation des obigen Beispiels mit zusätzlichen Glypheninformationen. Das Element *Glyph* enthält Angaben zu Font, Position und Farbe (für weitere Details zur Farbdarstellung siehe unten):

```

<Word>
<Text>datasheet</Text>
<Box llx="306.14" lly="796.22" urx="397.67" ury="816.72">
<Glyph font="F0" size="20.5000" x="306.14" y="796.22" width="11.42" fill="Co">d</Glyph>
<Glyph font="F0" size="20.5000" x="317.87" y="796.22" width="10.68" fill="Co">a</Glyph>
<Glyph font="F0" size="20.5000" x="328.61" y="796.22" width="7.61" fill="Co">t</Glyph>
<Glyph font="F0" size="20.5000" x="336.52" y="796.22" width="10.68" fill="Co">a</Glyph>
<Glyph font="F0" size="20.5000" x="347.51" y="796.22" width="8.71" fill="Co">s</Glyph>
<Glyph font="F0" size="20.5000" x="356.53" y="796.22" width="11.79" fill="Co">h</Glyph>
<Glyph font="F0" size="20.5000" x="368.63" y="796.22" width="10.41" fill="Co">e</Glyph>
<Glyph font="F0" size="20.5000" x="379.35" y="796.22" width="10.41" fill="Co">e</Glyph>
<Glyph font="F0" size="20.5000" x="390.07" y="796.22" width="7.61" fill="Co">t</Glyph>
</Box>
</Word>

```

**Farbwerte und Farbräume.** Farben werden durch einen Farbraum (z.B. DeviceRGB) sowie durch einen Farbwert dargestellt. Farbwerte für Text sind für Füll- und Linienfarben verfügbar. Da Glyphen mit Umrisslinien in PDF eher selten vorkommen, begegnet einem das Füllattribut häufiger. Die Farbwerte für Bilder kommen von den tatsächlichen

Pixeldaten. Farbräume für Text, Vektorgrafiken und Bilder werden im Element *ColorSpaces* im Abschnitt *Resources* aufgeführt. Abhängig vom Typ des Farbraums enthält jedes *ColorSpace*-Element bestimmte Informationen. Manche Farbräume beziehen sich auf andere, z.B. erfordert ein *Indexed*-Farbraum einen zugrunde liegenden Basis-Farbraum bzw. *Separation* und *DeviceN* einen Alternativfarbraum:

```
<Resources>
<ColorSpaces>
<ColorSpace id="CS0" name="DeviceCMYK" components="4"/>
<ColorSpace id="CS1" name="DeviceGray" components="1"/>
<ColorSpace id="CS2" name="Indexed" components="1" base="CS0" hival="255">
  <Lookup>000000000705000349340029745300416F50003E775600...</Lookup>
</ColorSpace>
<ColorSpace id="CS0" name="Separation" components="1" alternate="CS0">
  <Colorant name="PANTONE 294 CVC"/>
  <Function type="interpolate">
    ...
  <C1>
    <Value>0.93</Value>
    <Value>0.62</Value>
    <Value>0.00</Value>
    <Value>0.00</Value>
  </C1>
  <Exponent>1</Exponent>
</Function>
</ColorSpace>
...
</ColorSpaces>
</Resources>
```

**Tabellen in TETML.** Von TET identifizierte Tabellen werden in TETML als Tabellen-, Reihen- und Zellenstruktur ausgedrückt. Über mehrere Spalten verbundene Zellen werden mit einem *colSpan*-Attribut versehen:

```
<Table>
<Box llx="302.14" lly="639.72" urx="525.50" ury="731.50">
  <Row>
    <Box llx="311.64" lly="721.10" urx="521.50" ury="730.70"/>
    <Cell>
      <Box llx="311.64" lly="721.10" urx="375.22" ury="730.70"/>
      <Para>
        <Word>
          <Text>Device-dependent</Text>
          <Box llx="311.64" lly="721.90" urx="375.22" ury="729.90"/>
        </Word>
      </Para>
    </Cell>
    <Cell>
      <Box llx="397.91" lly="721.10" urx="431.99" ury="730.70"/>
      <Para>
        <Word>
          <Text>CIE-based</Text>
          <Box llx="397.91" lly="721.90" urx="431.99" ury="729.90"/>
        </Word>
      </Para>
    </Cell>
  </Row>
</Table>
```

```

</Cell>
...
<Row>
  <Box llx="306.14" lly="641.52" urx="516.67" ury="650.52"/>
  <Cell colSpan="3">
    <Box llx="306.14" lly="706.42" urx="516.67" ury="716.02"/>
  <Para>
    <Word>
      <Text>TET</Text>
      <Box llx="306.14" lly="641.52" urx="319.70" ury="650.52"/>
    </Word>
    <Word>
      <Text>.</Text>
      <Box llx="514.83" lly="641.52" urx="516.67" ury="650.52"/>
    </Word>
  </Para>
</Cell>
</Row>
</Box>
</Table>

```

**Interaktive Elemente.** Links, Lesezeichen, Formularfelder usw. sind in TETML, wie im Beispiel unten gezeigt, ebenfalls verfügbar:

```

<Page number="6" width="595.27600" height="841.89000">
<Annotations>
<Annotation id="ANNO" type="Link" anchor="A0">
<Box llx="327.14" lly="64.89" urx="395.08" ury="79.18"/>
<Action type="URI" trigger="activate" URI="mailto:sales%40pdflib.com"/>
</Annotation>
<Annotation id="ANN1" type="Link" anchor="A1">
<Box llx="327.14" lly="52.89" urx="391.05" ury="67.18"/>
<Action type="URI" trigger="activate" URI="http://www.pdflib.com"/>
</Annotation>
</Annotations>

```

Text in einem Link wird von A-Elementen (Anker) eingefasst, welche die Beziehung zwischen der geometrisch definierten PDF-Anmerkung und den entsprechenden Seiteninhalten liefert, d.h. dem Text, der den Link aktiviert. Beachten Sie, dass der aktive Inhalt keinen vollständigen semantischen Einheiten entsprechen muss. Zum Beispiel kann sich ein Link über ein Wort- oder Absatzfragment erstrecken. Da Anker sich nicht zwangsläufig über komplette TETML-Elemente erstrecken müssen, sind separate *start/stop*-Ankerelemente erforderlich, anstatt den Linkinhalt mit nur einem A-Element einzufassen:

```

<A id="A1" type="start"/>
<Word>
  <Text>www.pdflib.com</Text>
  <Box llx="327.14" lly="56.71" urx="391.05" ury="65.71"/>
</Word>
<A id="A1" type="stop"/>

```

## 9.3 Steuerung von TETML-Informationen

**TETML-Modi.** TETML kann in verschiedenen Modi erzeugt werden, die sich in der Menge an Font- und Geometrie-Informationen und in Bezug auf die Gruppierung von Text in größere Einheiten (Granularität) unterscheiden. Der TETML-Modus kann für jede Seite einzeln festgelegt werden. Im Allgemeinen enthalten TETML-Dateien die Daten für alle Seiten im selben Modus. Die folgenden TETML-Modi liefern Angaben zu Text und Bild sowie zu interaktiven Elementen:

- ▶ Der Modus *Glyph* ist eine einfache Variante, die den Text, den Font, die Koordinaten und die Farbe für jede Glyphen enthält, jedoch weder Gruppierung von Glyphen zu Wörtern noch Strukturinformationen. Er eignet sich für die Fehlersuche und -analyse, da die ursprüngliche Textinformation auf der Seite dargestellt wird.
- ▶ Der Modus *Word* gruppiert Text zu Wörtern und fügt *Box*-Elemente hinzu sowie die Koordinaten für jedes Wort. Fontinformationen sind nicht verfügbar. Dieser Modus eignet sich für Anwendungen, die auf Wortebene arbeiten. Satzzeichen werden standardmäßig als eigene Wörter behandelt, dies lässt sich jedoch mit einer Seitenoption ändern (siehe »Erkennen von Wortgrenzen in westlichem Text«, Seite 100). Textzeilen können optional mit dem *Line*-Element identifiziert werden; dies lässt sich mit der Seitenoption *tetml* steuern.
- ▶ Der Modus *Wordplus* ähnelt dem Modus *word*, fügt aber für alle Glyphen in einem Wort Angaben zu Font, Koordinaten und Farbe hinzu. Die Koordinaten werden abhängig von der Seitenoption *topdown* relativ zur linken unteren oder linken oberen Ecke ausgedrückt. Im Modus *Wordplus* kann die Verwendung eines Fonts analysiert und Änderungen am Font, der Fontgröße usw. innerhalb eines Wortes nachverfolgt werden. Da *wordplus* als einziger TETML-Modus alle relevanten TETML-Elemente enthält, eignet er sich für alle möglichen Verarbeitungsaufgaben. Durch die Fülle von Informationen vergrößert sich die TETML-Ausgabe allerdings beträchtlich.
- ▶ Der Modus *Line* enthält alle Texte, die in einer Zeile stehen, in einem separaten *Line*-Element. Außerdem können mehrere Zeilen zu einem Element *Para* gruppiert werden. Der Modus *Line* ist nur zu empfehlen, wenn die empfangende Anwendung nur zeilenbasierten Text verarbeiten kann.
- ▶ Der Modus *Page* enthält, auf Absatzebene beginnend, Strukturinformationen, jedoch keine Angaben zu Font oder Koordinaten. Beachten Sie, dass die Ergebnisse der Layout-Erkennung im Seitenmodus leicht von denen im Wortmodus abweichen können, da Anker für Bilder und Ziele unterschiedlich behandelt werden.

Wenn Sie nur an Bildinformationen interessiert sind, können Sie die Textausgabe in TETML auch überspringen:

- ▶ Der Modus *Image* umfasst Informationen zu platzierten Bildern und Bild-Ressourcen, jedoch weder text- oder fontbezogene Elemente noch Informationen zu interaktiven Elementen.

In Tabelle 9.1 sind die in den verschiedenen TETML-Modi verfügbaren TETML-Elemente aufgeführt.

**Auswahl des TETML-Modus.** Mit dem TET-Kommandozeilen-Tool (siehe Abschnitt 2.1, »Kommandozeilen-Optionen«, Seite 19) können Sie den gewünschten Modus als Parameter für die Option *--tetml* angeben. Mit folgendem Kommando lässt sich TETML-Ausgabe im Modus *wordplus* erzeugen:

Tabelle 9.1 Textbezogene Elemente in verschiedenen TETML-Modi; PlacedImage und Image sind immer vorhanden.

TETML-Modus	Struktur	Tabellen	Textposition	Glyphendetails
<i>glyph</i>	–	–	–	Glyph, Color
<i>word</i>	Para, Word <i>optional</i> : Line	Table, Row, Cell	Box <i>innerhalb von</i> Word <i>optional</i> : Box <i>innerhalb von</i> Para	–
<i>wordplus</i>	Para, Word <i>optional</i> : Line	Table, Row, Cell	Box <i>innerhalb von</i> Word <i>optional</i> : Box <i>innerhalb von</i> Para	Glyph, Color
<i>line</i>	Para, Line	–	<i>optional</i> : Box <i>innerhalb von</i> Para	–
<i>page</i>	Para	Table, Row, Cell	<i>optional</i> : Box <i>innerhalb von</i> Para	–
<i>image</i>	–	–	–	–

```
tet --tetml wordplus file.pdf
```

Mit der TET-Bibliothek kann der TETML-Modus nicht direkt, sondern nur als Kombination von Optionen angegeben werden:

- ▶ Mit der Option *granularity* von *TET\_process\_page()* lässt sich die Textmenge im kleinsten Element festlegen.
- ▶ Für *granularity=glyph* oder *word* können Sie zusätzlich die Menge an Glypheninformationen festlegen. Mit der Unteroption *glyphdetails* der Option *tetml* können nicht benötigte Glypheninformationen ausgespart werden.
- ▶ Um die gesamte Textausgabe zu unterdrücken (d.h. im Modus *image*), können Sie die Text-Engine mit der folgenden Dokumentoption deaktivieren:

```
engines={notext}
```

Mit der folgenden Seitenoption lässt sich TETML-Ausgabe im Modus *wordplus* mit allen Glypheninformationen ausgeben:

```
granularity=word tetml={ glyphdetails={all} }
```

In Tabelle 9.2 sind die Optionen zur Erzeugung von TETML-Modi zusammengefasst.

**Dokumentoptionen zur Steuerung der TETML-Ausgabe.** In diesem Abschnitt wird die Auswirkung verschiedener Optionen zusammengefasst, die die erzeugte TETML-Ausgabe direkt steuern. Mit allen anderen Dokumentoptionen können Details der Verarbeitung gesteuert werden. Eine vollständige Beschreibung aller Dokumentoptionen finden Sie in Tabelle 10.8.

Dokumentoptionen müssen an die Kommandozeilen-Option *--docopt* oder an die Funktion *TET\_open\_document()* übergeben werden.

Die Option *tetml*<sup>1</sup> steuert allgemeine Aspekte von TETML. Mit der Unteroption *elements* können bestimmte, nicht benötigte TETML-Elemente unterdrückt werden. Mit der folgenden Dokument-Optionsliste werden XMP-Metadaten auf Dokumentebene in der TETML-Ausgabe unterdrückt:

```
tetml={ elements={nometadata} }
```

1. Beachten Sie, dass es zwei verschiedene *tetml*-Optionen gibt: eine auf Dokumentebene und eine auf Seitenebene.

Tabelle 9.2 Erzeugen von TETML-Modi mit der TET-Bibliothek

TETML-Modus	Dokumentoption	Optionen von <i>TET_process_page()</i>
<i>glyph</i>	-	granularity=glyph tetml={glyphdetails={all}}
<i>word</i>	-	granularity=word
<i>wordplus</i>	-	granularity=word tetml={glyphdetails={all}}
<i>word mit Line-Elementen</i>	-	granularity=word tetml={elements={line}}
<i>wordplus mit Line-Elementen</i>	-	granularity=word tetml={glyphdetails={all} elements={line}}
<i>line</i>	-	granularity=line
<i>page</i>	-	granularity=page
<i>image</i>	skipengines={text vector}	tetml={elements={annotations=false bookmarks=false destinations=false docinfo=true fields=false javascripts=false metadata=true options=true}}

Mit der Option *engines* lassen sich einige der Verarbeitungsprozesse im TET-Kern deaktivieren. Mit der folgenden Optionsliste wird die Verarbeitung von Textinhalten in TET aktiviert, aber die Extraktion von Textfarbe und Bildverarbeitung deaktiviert:

```
engines={notextcolor noimage}
```

Die folgende Dokumentoption ist nur für *granularity=page* von Bedeutung. Damit wird das Standardzeichen für den Zeilenseparator von Zeilenvorschub (linefeed) auf Leerzeichen umgestellt:

```
lineseparator=U+0020
```

Alle Dokumentoptionen, die bei der Erzeugung von TETML übergeben wurden, werden im Element */TET/Document/Options* aufgeführt, sofern dies nicht mit der folgenden Dokumentoption deaktiviert wurde:

```
tetml={ elements={nooptions} }
```

### Dokumentoptionen zur Steuerung der TETML-Ausgabe von interaktiven Elementen.

TETML kann auch Informationen zu interaktiven Elementen in das PDF-Dokument einbinden. Mit der Unteroption *elements* der Dokumentoption *tetml* lässt sich TETML-Ausgabe für verschiedene Aspekte aktivieren oder deaktivieren, z.B.:

```
elements={annotations=true bookmarks=true destinations=true fields=true javascripts=true}
```

**Seitenoptionen zur Steuerung der TETML-Ausgabe.** Eine vollständige Beschreibung aller Seitenoptionen finden Sie in Tabelle 10.10. Seitenoptionen müssen an die Kommandozeilen-Option *--pageopt* oder an *TET\_process\_page()* übergeben werden.

Mit der Seitenoption *tetml* lassen sich die Angaben zu Font und Koordinaten im Element *Glyph* aktivieren oder deaktivieren. Mit der folgenden Optionsliste lassen sich Fontinformationen im Element *Glyph* aktivieren, aber andere Glyphenattribute deaktivieren:

```
tetml={ glyphdetails={font} }
```

Mit der folgenden Seitenoptionsliste lassen sich der TETML-Ausgabe *Line*-Elemente hinzufügen:

```
tetml={ glyphdetails={font} elements={line} }
```

Mit der folgenden Seitenoption lassen sich dem Element *Glyph* die Attribute *sub* und *sup* hinzufügen, um tief- und hochgestellte Zeichen zu identifizieren:

```
tetml={ glyphdetails={sub sup} }
```

Die folgende Seitenoption verwendet *all*, um dem Element *Glyph* alle verfügbaren Attribute hinzuzufügen:

```
tetml={ glyphdetails={all} }
```

Mit der folgenden Seitenoption werden statt des Standardkoordinatensystems Top-Down-Koordinaten verwendet:

```
topdown={output}
```

Mit der folgenden Seitenoptionsliste werden Satzzeichen mit dem benachbarten Wort zusammengefasst und nicht mehr als separate Wörter behandelt:

```
contentanalysis={nopunctuationbreaks}
```

Alle Seitenoptionen, die bei der Erzeugung von TETML übergeben wurden, werden im Element */TET/Document/Pages/Page/Options* für jede Seite einzeln aufgeführt, sofern dies nicht mit der folgenden Dokumentoption deaktiviert wird:

```
tetml={ elements={nooptions} }
```

**Verarbeitung von Exceptions.** Wenn während der PDF-Analyse ein Fehler auftritt, versucht TET diesen zunächst zu beheben oder zu ignorieren, wenn möglich, ansonsten wird eine Exception ausgelöst. Bei der Erzeugung von TETML-Ausgabe mit TET werden Probleme bei der PDF-Analyse jedoch als Element *Exception* in TETML ausgegeben:

```
<Exception errnum="4506">Object 'objects[49]/Subtype' does not exist</Exception>
```

Bei der Verarbeitung von TETML sollten Anwendungen darauf vorbereitet sein, dass statt des erwarteten Elements ein Element vom Typ *Exception* auftritt.

Probleme, die die Erzeugung einer TETML-Ausgabedatei verhindern, wie ungültige Optionen oder die fehlende Schreibberechtigung für die Ausgabedatei lösen eine Exception zur Laufzeit aus und es wird keine gültige TETML-Ausgabe geschrieben.

## 9.4 TETML-Elemente und das TETML-Schema

Im TET-Paket ist eine formale XML-Schemabeschreibung (XSD) für alle TETML-Elemente und -Attribute sowie deren Beziehungen enthalten. Der TETML-Namensraum lautet wie folgt:

<http://www.pdflib.com/XML/TET5/TET-5.0>

Das Schema können Sie unter folgender Adresse herunterladen:

<http://www.pdflib.com/XML/TET5/TET-5.0.xsd>

Sowohl der TETML-Namensraum als auch der Speicherort des Schemas sind im Wurzelement jedes TETML-Dokuments vorhanden.

Tabelle 9.3 beschreibt die Bedeutung aller TETML-Elemente. Ab TET 4.0 eingeführte Elemente und Attribute sind entsprechend gekennzeichnet. Abbildung 9.1 und Abbildung 9.2 visualisieren die XML-Hierarchie von TET-Elementen.

Table 9.3 TETML-Elemente und Attribute

<b>TETML-Element</b>	<b>Beschreibung und Attribute</b>
<b>A</b>	(TET 5.0; nur für granularity=glyph und word) Anker für eine Anmerkung, ein Ziel oder ein Feld innerhalb des Seiteninhalts Attribute: id, type (der Typ start/stop umfasst Text, der Typ rect kürzt Anker ohne Inhalt ab)
<b>Action</b>	(TET 5.0) Beschreibt eine PDF-Aktion. Attribute: filename, name, javascript, URI, trigger, type
<b>Annotation</b>	(TET 5.0) Beschreibt eine PDF-Anmerkung (außer Formularfeldern). Verfügt die Anmerkung über eine zugehörige Popup-Anmerkung, wird das Popup als verschachteltes Annotation-Element ausgegeben. Attribute: alignment, anchor, color, creationdate, destination, hidden, icon, id, intent, interiorcolor, invisible, moddate, name, onscreen, opacity, open, print, readonly, rotate, subject, symbol, type Untergeordnete Elemente: Action, Box, Annotation, Contents, Title
<b>Annotations</b>	(TET 5.0) Container für Annotation-Elemente Attribut: xml:space
<b>Attachment</b>	Für PDF-Anhänge: beschreibt die Inhalte in einem verschachtelten Element Document. Für Nicht-PDF-Anhänge: nur der Name, aber keine Inhalte werden aufgelistet. Attribute: name, level, pagenumber
<b>Attachments</b>	Container für Attachment-Elemente
<b>BitPerSample</b>	(TET 5.0) Anzahl der Bits pro Sample für Funktionen mit Function/@type="sampled"
<b>BlackPoint</b>	(TET 5.0) Tristimulus-Wert des Schwarzpunkts für die Farbräume CalGray, CalRGB und Lab Attribute: x, y, z
<b>Bookmark</b>	(TET 5.0) Enthält Elemente vom Typ Bookmark und Title zur Beschreibung von Text, Eigenschaften und verschachtelten Lesezeichen eines PDF-Lesezeichens Attribute: color, destination, fontstyle, open Untergeordnete Elemente: Action, Bookmark, Title
<b>Bookmarks</b>	(TET 5.0) Container für Bookmark-Elemente Attribut: xml:space

Tabelle 9.3 TETML-Elemente und Attribute

<b>TETML-Element</b>	<b>Beschreibung und Attribute</b>
<b>Bounds</b>	(TET 5.0) Intervalle für Funktionen mit Function/@type="stitching" Untergeordnetes Element: Value
<b>Box</b>	Beschreibt die Koordinaten eines Wortes, Absatzes, einer Anmerkung oder eines Formularfelds. Die Attribute llx und lly beschreiben die linke untere Ecke, urx und ury beschreiben die rechte obere Ecke der Box. Wenn die Box ein Rechteck mit Kanten parallel zu den Seitenrändern darstellt, beschreiben die vier Werte llx, lly, urx, ury die linke untere und rechte obere Ecke; anderenfalls sind die Koordinaten aller vier Ecken vorhanden. Ein Wort oder Absatz kann mehrere Box-Elemente enthalten, z.B. ein getrenntes Wort, das sich über mehrere Textzeilen erstreckt oder ein Wort, das mit einem großen Zeichen (dropcap) beginnt. Attribute: llx, lly <sup>1</sup> , urx, ury <sup>1</sup> , ulx, uly <sup>1</sup> , lrx, lry <sup>1</sup> Untergeordnete Elemente: A (TET 5.0), Glyph, Line (TET 5.0), Para (TET 5.0), PlacedImage (TET 5.0), Table (TET 5.0), Text (TET 5.0), Word (TET 5.0) Übergeordnete Elemente: Para, Word
<b>Co</b>	(TET 5.0) Anfangsfarbwert für Funktionen mit Function/@type="interpolate" Untergeordnetes Element: Value
<b>C1</b>	(TET 5.0) End-Farbwert für Funktionen mit Function/@type="interpolate". Als Convenience-Feature wird dieses Element auch für Funktionen mit Function/@type="sampled" erzeugt, obwohl es für solche Funktionen in PDF nicht vorhanden ist. Dieses Element beschreibt die Alternativfarbe für eine Schmuckfarbe. Untergeordnetes Element: Value
<b>Calculator</b>	(TET 5.0) Operatoren für PostScript-Funktionen, d.h. Function/@type="PostScript"
<b>Cell</b>	Beschreibt den Inhalt einer einzelnen Tabellenzelle. Attribute: colSpan, llx, lly <sup>1</sup> , urx, ury <sup>1</sup> , ulx, uly <sup>1</sup> , lrx, lry <sup>1</sup>
<b>Color</b>	(TET 5.0) Beschreibt eine PDF-Farbe. Attribute: colorspace, id, svgname, pattern
<b>Colorant</b>	(TET 5.0) Farbe eines Separation- oder DeviceN-Farbraums Attribute: name, colorspace
<b>Colors</b>	(TET 5.0) Container für Color-Elemente
<b>ColorSpace</b>	Beschreibt einen PDF-Farbraum. Attribute: alternate, base, components, hival (TET 5.0), iccprofile (TET 5.0), id, name, pattern (TET 5.0), subtype (TET 5.0) Untergeordnete Elemente (TET 5.0): BlackPoint, Colorant, Exception, Function, Gamma, Lookup, Matrix, Process, Range, WhitePoint
<b>ColorSpaces</b>	Container für ColorSpace-Elemente
<b>Content</b>	Beschreibt die Seiteninhalte als hierarchische Struktur. Attribute: granularity, dehyphenation (TET 4.0), dropcap (TET 4.0), font, geometry, shadow (TET 4.0), sub (TET 4.0), sup (TET 4.0)
<b>Contents</b>	(TET 5.0) Als untergeordnetes Element von Annotation: Inhalt einer Anmerkung (TET 5.0) Als untergeordnetes Element von Field: Inhalt eines Formularfelds
<b>Creation</b>	Beschreibt das Datum und die Betriebssystem-Plattform für die TET-Verarbeitung sowie die Versionsnummer von TET. Attribute: date, platform, tetVersion
<b>Decode</b>	(TET 5.0) Zuordnung von Beispielwerten für Sample-Funktionen, d.h. Function/@type="sampled" Untergeordnetes Element: Value

Tabelle 9.3 TETML-Elemente und Attribute

<b>TETML-Element</b>	<b>Beschreibung und Attribute</b>
<b>Destination</b>	(TET 5.0) Beschreibt ein PDF-Ziel im Dokument. Attribute: anchor, bottom <sup>1</sup> , id, left, name, page, right, top <sup>1</sup> , type, zoom
<b>Destinations</b>	(TET 5.0) Container für Destination-Elemente
<b>DefaultValue</b>	(TET 5.0) Standardwert eines Formularfelds
<b>DocInfo</b>	Standard- und benutzerdefinierte Dokument-Infofelder Untergeordnete Elemente: Author, CreationDate, Creator, GTS_PDFXConformance, GTS_PDFX-Version, GTS_PPMLVDXConformance, GTS_PPMLVDXVersion, ISO_PDFFVersion, Keywords, Mod-Date, Producer, Subject, Title, Trapped, Custom (Attribut: key), CustomBinary (Attribut: key)
<b>Document</b>	Beschreibt allgemeine Dokument-Informationen, einschließlich PDF-Dateiname und -größe sowie die PDF-Versionsnummer. Attribute: filename, destination (TET 5.0), pageCount, filesize, linearized, pdfVersion, pdfa (TET 4.0: neue Werte für PDF/A-2; TET 4.1: neue Werte für PDF/A-3), pdfe (TET 4.0; TET 4.1: neue Werte für PDF/E-2), pdfua (TET 4.1), pdfvt (TET 4.1), pdfx (TET 4.1: aufgelistete Werte), revisions (TET 5.0), tagged, usagerights (TET 5.0) Untergeordnete Elemente: Action (TET 5.0), Attachments, Bookmarks (TET 5.0), Destinations (TET 5.0), DocInfo, Encryption, Exception, JavaScripts (TET 5.0), Metadata, Options, Output-Intents, Pages, SignatureFields (TET 5.0), XFA (TET 5.0)
<b>Domain</b>	(TET 5.0) Intervall(e) von Eingabewerten für Funktionen Untergeordnetes Element: Value
<b>Encode</b>	(TET 5.0) Zuordnung von Eingabewerten für Funktionen mit Function/@type="stitching" Untergeordnetes Element: Value
<b>Encryption</b>	Beschreibt verschiedene Sicherheitseinstellungen. Attribute: keylength, algorithm (TET 4.1: neue Werte 8-11), attachment (TET 4.1), description (TET 4.1: neue Werte für Algorithmen 8-11), masterpassword, userpassword, noprint, nomodify, nocopy, noannots, noassemble, noforms, noaccessible, nohiresprint, plainmetadata
<b>Exception</b>	Fehlermeldung und -nummer der zugehörigen Exception, die von TET ausgelöst und in TETML übersetzt wurde. Das Element Exception kann andere Elemente ersetzen, wenn nicht genügend Informationen aus der Eingabe extrahiert werden können, da die PDF-Datenstrukturen fehlerhaft sind. Folgende Elemente können ein Exception-Element als untergeordnetes Element haben: Annotation, Annotations, Attachment, Attachments, Bookmark, Bookmarks, Color, ColorSpace, ColorSpaces, Document, Field, Fields, Font, Fonts, ICCProfile, Image, Images, Metadata, Page, Pattern, Patterns, SignatureField, SignatureFields Attribut: errnum
<b>Exponent</b>	(TET 5.0) Interpolationsexponent für Funktionen mit Function/@type="interpolate"
<b>Field</b>	(TET 5.0) Beschreibt ein PDF-Formularfeld. Attribute: alignment, anchor, backgroundcolor, bordercolor, caption, captiondown, caption-rollover, destination, export, exportvalue (nur für type=radiobutton und checkbox), hidden, id, mappingname, name, onscreen, print, readonly, required, rotate, sort, state, type, visible Untergeordnete Elemente: Action, Box, Contents, Field (für die Schaltflächen mit einem Feld type=radiogroup), DefaultValue, Optionalvalue, Tooltip, Value
<b>Fields</b>	(TET 5.0) Container für Field-Elemente Attribut: xml:space

Tabelle 9.3 TETML-Elemente und Attribute

<b>TETML-Element</b>	<b>Beschreibung und Attribute</b>
<b>Font</b>	<p>Beschreibt eine Font-Ressource. Das erforderliche Attribut <code>name</code> enthält den kanonischen Fontnamen, während das optionale Attribut <code>fullname</code> den Fontnamen einschließlich Präfix für eine Untergruppe enthält.</p> <p>Attribute: <code>ascender</code> (TET 4.1), <code>capheight</code> (TET 4.1), <code>descender</code> (TET 4.1), <code>embedded</code>, <code>fullname</code> (TET 4.0), <code>id</code>, <code>italicangle</code> (TET 4.1), <code>type</code>, <code>name</code>, <code>vertical</code>, <code>weight</code> (TET 4.1), <code>xheight</code> (TET 4.1)</p>
<b>Fonts</b>	Container für Font-Elemente
<b>Function</b>	<p>(TET 5.0) Funktion für die Umrechnung der Farbwerte für einen Separation- oder DeviceN-Farbraum</p> <p>Attribut: <code>type</code></p> <p>Untergeordnete Elemente: <code>BitsPerSample</code>, <code>Bounds</code>, <code>Calculator</code>, <code>C0</code>, <code>C1</code>, <code>Decode</code>, <code>Domain</code>, <code>Encode</code>, <code>Functions</code>, <code>Exponent</code>, <code>Order</code>, <code>Range</code>, <code>Samples</code>, <code>Size</code></p>
<b>Functions</b>	<p>(TET 5.0) Container für die Unterfunktionen von Funktionen mit <code>Function/@type="stitching"</code></p> <p>Untergeordnetes Element: <code>Function</code></p>
<b>Gamma</b>	<p>(TET 5.0) Gamma-Werte für die Farbräume <code>CalGray</code> oder <code>CalRGB</code></p> <p>Untergeordnetes Element: <code>Value</code></p>
<b>Glyph</b>	<p>Beschreibt Font- und Geometrie-Informationen für eine einzelne Glyphe. Das Element enthält die von dieser Glyphe produzierten Unicode-Zeichen. Eine einzelne Glyphe kann mehr als ein Zeichen liefern, z.B. bei Ligaturen. Die Glyphenelemente für ein Wort werden innerhalb eines oder mehrerer Box-Elemente gruppiert.</p> <p>Attribute: <code>x</code>, <code>y</code>, <code>width</code>, <code>height</code> (TET 5.0; nur für vertikale Schreibrichtung und wenn die Glyphenhöhe von der Fontgröße verschieden ist), <code>alpha</code>, <code>beta</code>, <code>shadow</code> (TET 4.0), <code>dropcap</code> (TET 4.0), <code>fill</code> (TET 5.0), <code>font</code>, <code>size</code>, <code>stroke</code> (TET 5.0), <code>sub</code> (TET 4.0), <code>sup</code> (TET 4.0), <code>textrendering</code>, <code>unknown</code>, <code>dehyphenation</code> (TET 4.0)</p>
<b>Graphics</b>	(TET 5.0) Container für die Elemente <code>Colors</code> , <code>ICCProfiles</code> und <code>Layers</code>
<b>ICCProfiles</b>	(TET 5.0) Container für ICCProfile-Elemente
<b>ICCProfile</b>	<p>(TET 5.0) Beschreibt ein ICC-Farbprofil.</p> <p>Attribute: <code>checksum</code>, <code>iccversion</code>, <code>id</code>, <code>deviceclass</code>, <code>embedded</code>, <code>fromCIE</code>, <code>profilecls</code>, <code>filename</code>, <code>toCIE</code></p>
<b>Image</b>	<p>Beschreibt eine Bild-Ressource, d.h. das Pixel-Array, aus dem das Bild besteht.</p> <p>Attribute: <code>bitsPerComponent</code>, <code>colorspace</code>, <code>extractedAs</code> (TET 4.0, zusätzlicher Wert seit TET 4.2), <code>filename</code> (TET 5.0), <code>height</code>, <code>id</code>, <code>maskid</code> (TET 5.0), <code>mergetype</code>, <code>stencilmask</code> (TET 5.0), <code>width</code></p>
<b>Images</b>	Container für Image-Elemente
<b>JavaScript</b>	<p>(TET 5.0) Beschreibt eine Sequenz von JavaScript-Code</p> <p>Attribute: <code>id</code>, <code>name</code></p>
<b>JavaScripts</b>	(TET 5.0) Container für JavaScript-Elemente
<b>Layer</b>	<p>(TET 5.0) Beschreibt eine optionale Inhaltsgruppe (OCG, optional content group), auch Layer (Ebene) genannt</p> <p>Attribute: <code>name</code>, <code>visible</code>, <code>label</code>, <code>locked</code></p> <p>Untergeordnetes Element: <code>Layer</code></p>
<b>Layers</b>	(TET 5.0) Container für Layer-Elemente
<b>Line</b>	Einzelne Textzeile. TET 4.0: Line kann auch Word-Elemente enthalten.
<b>Lookup</b>	(TET 5.0) Lookup-Tabelle für indizierte Farbräume, d.h. <code>ColorSpace/@name="Indexed"</code> . Sie enthält eine hexadezimale Sequenz von Werten, die im Basis-Farbraum des indizierten Farbraums interpretiert werden muss.

Tabelle 9.3 TETML-Elemente und Attribute

<b>TETML-Element</b>	<b>Beschreibung und Attribute</b>
<b>Matrix</b>	(TET 5.0) Transformationsmatrix eines CalRGB-Farbraums Untergeordnetes Element: Value
<b>Metadata</b>	XMP-Metadaten für ein Dokument, einen Font oder ein Bild
<b>OptionalValue</b>	(TET 5.0) Optionaler Wert eines Formularfelds
<b>Options</b>	Dokument- oder Seitenoptionen für die Erzeugung von TETML
<b>Order</b>	(TET 5.0) Reihenfolge der Sample-Interpolation für Funktionen mit Function/@type="sampled"
<b>OutputIntent</b>	(TET 5.0) Beschreibt die Druckausgabebedingung eines Dokuments oder einer Seite Attribute: iccprofile, subtype Untergeordnete Elemente: OutputCondition, OutputConditionIdentifizier, RegistryName, Info
<b>OutputIntents</b>	(TET 5.0) Container für OutputIntent-Elemente
<b>Page</b>	Inhalt einer einzelnen Seite. Attribute: hasdefaultcmyk, hasdefaultgray, hasdefaulttrgb, height, label (TET 5.0), number, topdown (TET 4.0), width Untergeordnete Elemente: Action (TET 5.0), Annotations (TET 5.0), Content, Exception, Fields (TET 5.0), Options, OutputIntents
<b>Pages</b>	Container für Page-Elemente
<b>Para</b>	Text, der einen einzelnen Absatz umfasst. Untergeordnete Elemente: A, Box, Para
<b>Pattern</b>	(TET 5.0) Beschreibt ein PDF-Pattern. Attribute: id, patterntype, painttype, tilingtype
<b>Patterns</b>	(TET 5.0) Container für Pattern-Elemente
<b>PlacedImage</b>	Beschreibt ein auf der Seite platziertes Bild. Attribute: alpha <sup>1</sup> , beta <sup>1</sup> , height, image, width, x, y <sup>1</sup>
<b>Process</b>	(TET 5.0) Beschreibung des Prozessfarbraums eines DeviceN-Farbraums mit dem Untertyp NChannel Attribut: colorspace Untergeordnetes Element: Component
<b>Range</b>	(TET 5.0) Als untergeordnetes Element von ColorSpace: Wertebereich eines Lab-Farbraums. Als untergeordnetes Element von Function: Bereich der Ausgabewerte für Funktionen Untergeordnetes Element: Value
<b>Resources</b>	Container der Ressource-Container ColorSpaces, Fonts, Images und Patternx
<b>Row</b>	Container für eine oder mehrere Tabellenzellen
<b>Samples</b>	(TET 5.0) Hexadezimale Sequenz von Samples für Funktionen mit Function/@type="sampled"
<b>SignatureField</b>	(TET 5.0) Beschreibt ein signiertes oder unsigniertes Feld Attribute: cades, field, fillablefields, permissions, preventchanges, sigtype, visible
<b>SignatureFields</b>	(TET 5.0) Container für SignatureField-Elemente
<b>Size</b>	(TET 5.0) Anzahl der Samples in jeder Eingabedimension für Funktionen mit Function/@type="sampled" Untergeordnetes Element: Value
<b>Table</b>	Container für eine oder mehrere Tabellenzeilen Attribute: llx, lly, urx, ury, ulx, uly, lrx, lry Untergeordnetes Element: Row

Tabelle 9.3 TETML-Elemente und Attribute

<b>TETML-Element</b>	<b>Beschreibung und Attribute</b>
<b>TET</b>	TETML-Stammelement. Attribut: version
<b>Text</b>	Textinhalt eines Wortes oder anderen Elements
<b>Title</b>	(TET 5.0) Als untergeordnetes Element von Annotation: Titel einer Anmerkung (TET 5.0) Als untergeordnetes Element von Bookmark: Titel eines Lesezeichens Als untergeordnetes Element von DocInfo: Dokument-Infofeld Title
<b>Tooltip</b>	(TET 5.0) Tooltip eines Formularfelds
<b>Value</b>	(TET 5.0) Wert eines Formularfelds
<b>WhitePoint</b>	(TET 5.0) Tristimulus-Wert des Weißpunkts für die Farbräume CalGray, CalRGB und Lab Attribute: x, y, z
<b>Word</b>	Einzelnes Wort
<b>XFA</b>	(TET 5.0) Das Dokument enthält XFA-Formulardaten Attribut: type (immer static, da TET keine dynamischen XFA-Formulare verarbeitet)

1. Alle vertikalen Koordinaten und Winkel werden abhängig von der Seitenoption topdown relativ zur linken unteren oder linken oberen Ecke ausgedrückt.



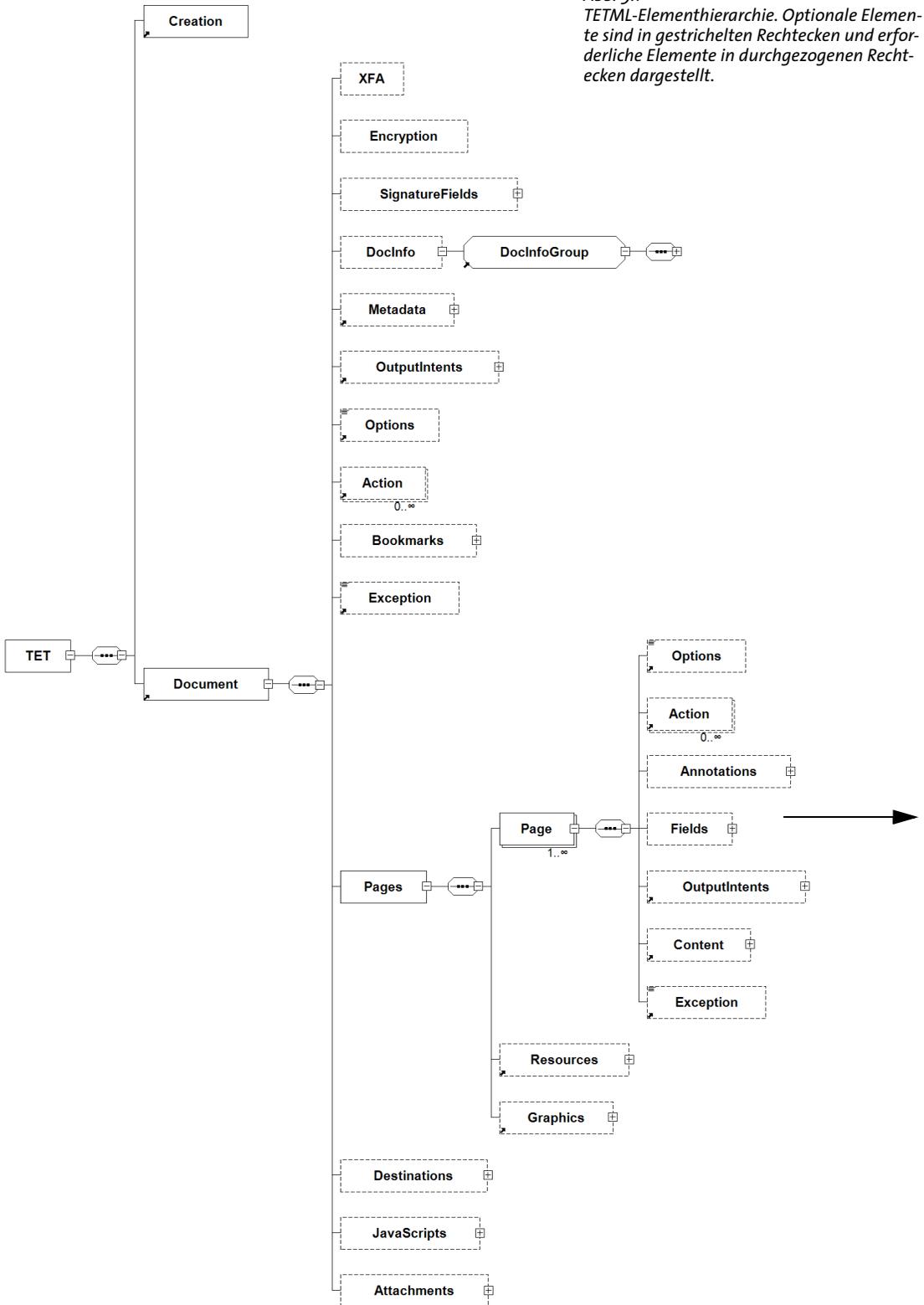


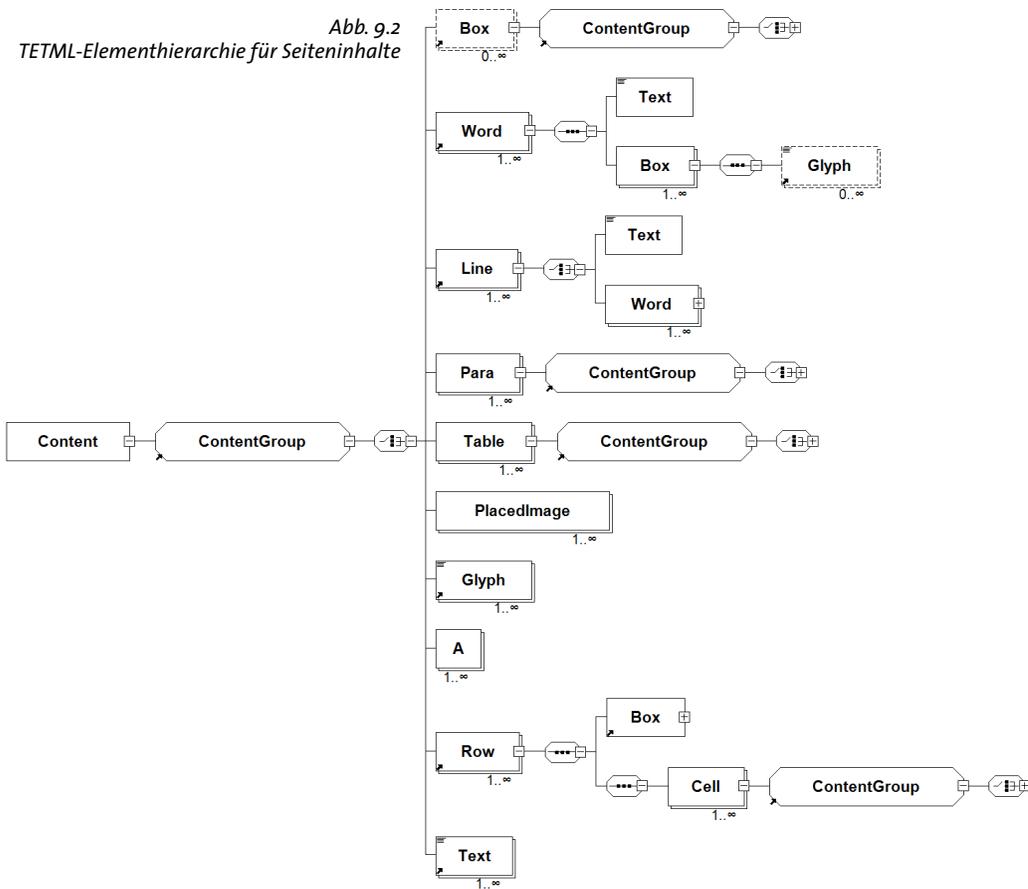
Abb. 9.1  
TETML-Elementhierarchie. Optionale Elemente sind in gestrichelten Rechtecken und erforderliche Elemente in durchgezogenen Rechtecken dargestellt.

## 9.5 TETML-Transformationen mit XSLT

**Kurzer Überblick über XSLT.** XSLT (*eXtensible Stylesheet Language Transformations*) ist eine Sprache zur Umwandlung von XML-Dokumenten in andere Dokumente. Während es sich bei der Eingabe immer um ein XML-Dokument (in unserem Fall TETML) handelt, muss die Ausgabe nicht zwingend XML sein. XSLT kann auch beliebige Berechnungen durchführen und Text- oder HTML-Ausgabe erzeugen. Wir werden XSLT-Stylesheets zur Verarbeitung von TETML-Eingabe verwenden, um einen neuen Datensatz zu erzeugen (in Text-, XML-, CSV- oder HTML-Format), basierend auf der Eingabe, die wiederum den Inhalt eines PDF-Dokuments spiegelt. Das TETML-Dokument wurde mit dem TET-Kommandozeilen-Tool oder der TET-Bibliothek erzeugt, wie in Abschnitt 9.1, »Erzeugen von TETML«, Seite 147 beschrieben.

XSLT ist sehr leistungsstark, unterscheidet sich aber erheblich von anderen Programmiersprachen. In diesem Abschnitt soll keine Einführung in die XSLT-Programmierung

Abb. 9.2  
TETML-Elementhierarchie für Seiteninhalte



gegeben werden; wählen Sie hierfür aus der Vielzahl an gedrucktem und im Web verfügbarem Material. Unsere Beispiele beziehen sich nur auf XSLT 1.0. Die XSLT-1.0-Spezifikation finden Sie unter [www.w3.org/TR/xslt](http://www.w3.org/TR/xslt).

Jedoch wollen wir Sie dabei unterstützen, die XSLT-Verarbeitung von TETML-Dokumenten schnell in Betrieb zu nehmen. Dieser Abschnitt beschreibt die wichtigsten Umgebungen für die Ausführung von XSLT-Stylesheets und listet gängige Software für diese Zwecke auf. Um XSLT-Stylesheets auf XML-Dokumente anzuwenden, benötigen Sie einen XSLT-Prozessor. Es sind verschiedene kostenlose und kommerzielle XSLT-Prozessoren erhältlich, die Sie entweder im Stand-alone-Betrieb oder mit Hilfe einer Programmiersprache in Ihren eigenen Programmen einsetzen können.

XSLT-Stylesheets können mit Hilfe von Parametern, die von der Umgebung an das Stylesheet übergeben werden, Details der Verarbeitung steuern. Da in einigen unserer XSLT-Beispiele Stylesheet-Parameter verwendet werden, liefern wir auch Informationen zur Übergabe von Parametern an Stylesheets in verschiedenen Umgebungen.

Zu den gängigen XSLT-Prozessoren, die in verschiedenen Paketen verwendet werden können, gehören die folgenden:

- ▶ XML-Implementierung von Microsoft namens MSXML
- ▶ XSLT-Implementierung von Microsoft im .NET Framework 2.0
- ▶ Saxon, erhältlich in kostenlosen und kommerziellen Ausführungen
- ▶ Xalan, ein Open-Source-Projekt (verfügbar in C++- und Java-Implementierungen), das von der Apache Foundation bereitgestellt wird
- ▶ Die Open-Source-Bibliothek *libxslt* des Projekts GNOME
- ▶ Sablotron, ein Open-Source XSLT-Toolkit

**XSLT über die Kommandozeile.** Die Anwendung von XSLT-Stylesheets über die Kommandozeile bietet eine komfortable Entwicklungs- und Testumgebung. Die Beispiele unten zeigen, wie XSLT-Stylesheets auf der Kommandozeile verarbeitet werden können. Alle Beispiele unten verarbeiten die Eingabedatei *TET-datasheet.tetml* mit dem Stylesheet *tetml2html.xsl*, wobei der XSLT-Parameter *toc-generate* (der im Stylesheet verwendet wird) auf den Wert *o* gesetzt wird, und schreiben die erzeugte Ausgabe in *TET-datasheet.html*:

- ▶ Der Java-basierte Saxon-Prozessor (siehe [www.saxonica.com](http://www.saxonica.com)) kann folgendermaßen aufgerufen werden:

```
java -jar saxon9.jar -o TET-datasheet.html TET-datasheet.tetml tetml2html.xsl
```

- ▶ Sie können XSLT-Skripte auch mit dem Build-Tool *ant* anwenden. Eine minimale Builddatei für die Anwendung von XSLT sieht folgendermaßen aus:

```
<project name="tetml2html" default="tetml2html">
  <target name="tetml2html">
    <xslt in="TET-datasheet.tetml" style="tetml2html.xsl" out="TET-datasheet.html"/>
  </target>
</project>
```

Die Datei *build.xml* in der TET-Distribution enthält XSLT-Tasks für alle Beispiele. Mit dem Kommando *ant* werden alle XSLT-Beispiele angewendet und das Eingabedokument *TET-datasheet.pdf* in TETML konvertiert. Mit folgendem Kommando wird ein weiteres PDF-Dokument verarbeitet:

```
ant -Dinput.pdf=myfile.pdf
```

- ▶ Das Tool *xsltproc* ist in den meisten Linux-Distributionen enthalten, siehe [xmlsoft.org/XSLT](http://xmlsoft.org/XSLT). Mit dem folgenden Kommando können Sie ein Stylesheet auf ein TETML-Dokument anwenden:

```
xsltproc --output TET-datasheet.html --param toc-generate 0 tetml2html.xml ←  
TET-datasheet.tetml
```

Mit dem Shell-Skript *runxslt.sh* in der TET-Distribution können alle XSLT-Beispiele mit *xsltproc* ausgeführt werden (führen Sie *ant* einmal aus, um die TETML-Eingabedateien zu erzeugen).

- ▶ Xalan C++ bietet ein Kommandozeilen-Tool, das Sie wie folgt aufrufen können:

```
Xalan -o TET-datasheet.html -p toc-generate 0 TET-datasheet.tetml tetml2html.xml
```

- ▶ Auf Windows-Systemen mit dem MSXML-Parser können Sie das von Microsoft kostenlos zur Verfügung gestellte Programm *msxsl.exe* verwenden. Das Programm (einschließlich Sourcecode) ist unter folgender Adresse verfügbar:

[www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=21714](http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=21714)

Führen Sie das Programm folgendermaßen aus:

```
msxsl.exe TET-datasheet.tetml tetml2html.xml -o TET-datasheet.html toc-generate=0
```

Mit den Skripten *runxslt.ps1* und *runxslt.vbs* in der TET-Distribution können alle XSLT-Beispiele mit *msxml* ausgeführt werden (führen Sie *ant* einmal aus, um die TETML-Eingabedateien zu erzeugen).

**XSLT in Ihrer eigenen Anwendung.** Wenn Sie die XSLT-Verarbeitung in Ihre Anwendung integrieren möchten, hängt die Wahl des XSLT-Prozessors natürlich von Ihrer Programmiersprache und -umgebung ab. Die TET-Distribution enthält Beispielcode für verschiedene wichtige Umgebungen. Die *runxslt*-Beispiele zeigen, wie Sie ein TETML-Dokument laden, ein XSLT-Stylesheet mit Parametern anwenden und die erzeugte Ausgabe in eine Datei schreiben. Wenn die Programme ohne Argumente ausgeführt werden, werden alle mit der TET-Distribution ausgelieferten XSLT-Beispiele durchlaufen. Alternativ können Sie Parameter für die Namen der TETML-Eingabedatei, des XSLT-Stylesheets, der Ausgabedatei sowie weitere Parameter/Wertpaare übergeben. Als Ausgangspunkt für die Integration der XSLT-Verarbeitung in Ihre Anwendung können Sie die *runxslt*-Beispiele verwenden:

- ▶ Java-Entwickler können die Methoden aus dem Paket *javax.xml.transform* verwenden. Dies wird im Beispiel *runxslt.java* dargestellt.
- ▶ .NET-Entwickler können die Methoden im Namensraum *System.Xml.Xsl.XslTransform* verwenden. Dies wird im PowerShell-Skript *runxslt.ps1* dargestellt. Ähnlicher Code kann in C# und anderen .NET-Sprachen verwendet werden.
- ▶ Alle Windows-basierten Programmiersprachen, die COM-Automatisierung unterstützen, können die Methoden der Automatisierungsklasse *MSXML2.DOMDocument* verwenden, die durch den MSXML-Parser übergeben wird. Dies wird im Beispiel *runxslt.vbs* dargestellt. Ähnlicher Code kann in anderen COM-fähigen Sprachen verwendet werden.

XSLT-Erweiterungen sind für viele weitere moderne Programmiersprachen erhältlich, wie z.B. Perl.

**XSLT auf dem Webserver.** Da die Konvertierung von XML nach HTML sehr häufig vorkommt, werden XSLT-Stylesheets oft auf Webservern eingesetzt. Einige wichtige Szenarien:

- ▶ Windows-basierte Webserver mit ASP oder ASP.NET können die oben erwähnten COM- oder .NET -Schnittstellen nutzen.
- ▶ Java-basierte Webserver können das Paket *javax.xml.transform* nutzen.
- ▶ PHP-basierte Webserver können den Sablotron-Prozessor nutzen, siehe [www.php.net/manual/en/intro.xsl.php](http://www.php.net/manual/en/intro.xsl.php).

**XSLT im Webbrowser.** XSLT-Transformationen werden auch von den meisten modernen Browsern unterstützt. Damit der Browser ein XSLT-Stylesheet auf ein TETML-Dokument anwenden kann, fügen Sie nach der ersten Zeile des TETML-Dokuments und vor dem Wurzelement eine Zeile mit einer geeigneten *xm/*-Verarbeitungsanweisung ein. Sie können es dann in den Browser laden, wo das Stylesheet angewendet und die resultierende Ausgabe angezeigt wird (beachten Sie, dass der Internet Explorer die Dateinamenserweiterung *.xml/* benötigt, um Dateien von der Festplatte zu verarbeiten):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="tetml2html.xsl" version="1.0"?>
<TET xmlns="http://www.pdflib.com/XML/TET5/TET-5.0"
...

```

Der Browser wendet das XSLT-Stylesheet auf das TETML-Dokument an und zeigt dann die resultierende Text-, HTML- oder XML-Ausgabe an. Alternativ dazu kann die XSLT-Verarbeitung im Browser auch durch JavaScript-Code ausgelöst werden.

Bei Firefox können Sie die Parameter mit der *xslt-param*-Verarbeitungsanweisung an das XSLT-Stylesheet übergeben:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="tetml2html.xsl" version="1.0"?>
<?xslt-param name="toc-generate" value="0"?>
<TET xmlns="http://www.pdflib.com/XML/TET5/TET-5.0"
...

```

## 9.6 XSLT-Beispiele

Die TET-Distribution enthält mehrere XSLT-Stylesheets, die die leistungsstarke Funktionalität von XSLT im Zusammenspiel mit TETML zeigen. Sie können als Ausgangspunkt für TETML-Anwendungen verwendet werden. Dieser Abschnitt gibt einen Überblick über die XSLT-Beispiele und zeigt Beispielausgabe. In Abschnitt 9.5, »TETML-Transformationen mit XSLT«, Seite 165, werden zahlreiche Optionen für die Anwendung von XSLT-Stylesheets vorgestellt. Weitere Informationen zu Funktionalität und Implementierung der Stylesheets finden Sie in den Kommentaren im XSLT-Code. Einige allgemeine Aspekte der Stylesheet-Beispiele:

- ▶ Die meisten XSLT-Beispiele unterstützen Parameter für die Steuerung verschiedener Verarbeitungsdetails. Diese Parameter können im XSLT-Code gesetzt oder von der Umgebung überschrieben werden (z.B. *ant*).
- ▶ Die meisten XSLT-Beispiele erfordern TETML-Eingabe in einem bestimmten TETML-Modus (z.B. im Modus *word*, für weitere Informationen siehe »TETML-Modi«, Seite 153). Um sich vor falscher Eingabe zu schützen, prüfen die Stylesheets, ob die übergebene TETML-Eingabe der Anforderung entspricht und geben ansonsten eine Fehlermeldung aus.
- ▶ Einige XSLT-Beispiele verarbeiten PDF-Anhänge im Dokument rekursiv (dies wird in den folgenden Beschreibungen entsprechend erwähnt). In den meisten Beispielen werden PDF-Anhänge jedoch ignoriert. Die Beispiele sind so geschrieben, dass sie leicht für die Verarbeitung von PDF-Anhängen erweitert werden können. Es genügt, die entsprechenden Elemente innerhalb des Elements *Attachments* auszuwählen; die relevanten *xsl:template*-Elemente selbst müssen nicht verändert werden.
- ▶ Alle XSLT-Beispiele arbeiten mit XSLT 1. Um die Nutzung zu erleichtern, werden nur XSLT-1-Funktionen verwendet, auch wenn einige Beispiele mit Hilfe von XSLT-2-Funktionen vereinfacht werden könnten.

**Erzeugen einer Konkordanz.** Das Stylesheet *concordance.xsl* erwartet TETML-Eingabe im Modus *word* oder *wordplus*. Es erzeugt eine Konkordanz, d.h. eine Liste aller eindeutigen Wörter in einem Dokument, sortiert nach absteigender Häufigkeit. Dies kann nützlich sein, um eine Konkordanz für die sprachliche Analyse, Querverweise für Übersetzer, Konsistenzprüfungen usw. zu erstellen.

Liste von Wörtern im Dokument mit der Anzahl der Fundstellen:

```
the 138
and 91
TET 87
to 63
of 59
for 57
PDF 53
text 51
in 50
a 44
is 37
be 36
as 34
are 34
PDFlib 32
...
```

**Filtern von Fonts.** Das Stylesheet *fontfilter.xsl* erwartet TETML-Eingabe im Modus *glyph* oder *wordplus*. Es listet alle Wörter in einem Dokument auf, die einen bestimmten Font verwenden, der größer ist als der angegebene Wert. Dies kann nützlich sein, um bestimmte Kombinationen von Font und Größe für die Qualitätssicherung zu identifizieren. Das gleiche Konzept kann verwendet werden, um ein Inhaltsverzeichnis aus solchen Textteilen zu erzeugen, die eine große Fontgröße verwenden.

Text im Font 'TheSansBold-Plain' mit Fontgröße > 10:

```
[ThesisAntiqua-Bold/32.0000] PDFlib
[ThesisAntiqua-Bold/32.0000] TET
[ThesisAntiqua-Bold/32.0000] 5
[ThesisAntiqua-Bold/14.0000] What
[ThesisAntiqua-Bold/14.0000] is
[ThesisAntiqua-Bold/14.0000] PDFlib
[ThesisAntiqua-Bold/14.0000] TET
[ThesisAntiqua-Bold/14.0000] ?
[ThesisAntiqua-Bold/14.0000] PDFlib
[ThesisAntiqua-Bold/14.0000] TET
[ThesisAntiqua-Bold/14.0000] Features
[ThesisAntiqua-Bold/14.0000] Challenges
[ThesisAntiqua-Bold/14.0000] with
[ThesisAntiqua-Bold/14.0000] PDF
[ThesisAntiqua-Bold/14.0000] Text
[ThesisAntiqua-Bold/14.0000] Extraction
[ThesisAntiqua-Bold/14.0000] Challenges
...
```

**Suche nach verwendeten Fonts.** Das Stylesheet *fontfinder.xsl* erwartet TETML-Eingabe im Modus *glyph* oder *wordplus*. Es listet alle Textfundstellen in einem bestimmten Font für alle Fonts im Dokument auf und gibt sie zusammen mit der Position auf der Seite und der Seitenzahl aus. Dies kann nützlich sein, um unerwünschte Fonts oder Fontgrößen zu identifizieren und um die Konsistenz zu prüfen usw.

TheSans-Plain used on:

page 1:  
(306, 796)

ThesisAntiqua-Bold used on:

page 1:  
(306, 757), (412, 757), (474, 757), (28, 514), (67, 514), (81, 514), (128, 514), (152, 514),  
...

**Font-Statistiken.** Das Stylesheet *fontstat.xsl* erwartet TETML-Eingabe im Modus *glyph* oder *wordplus*. Es erzeugt eine Statistik zu Fonts und Glyphen. Dies kann für die Qualitätskontrolle und sogar für die Überprüfung auf Zugänglichkeit nützlich sein, da für jeden Font auch die nicht zugeordneten Glyphen (d.h. Glyphen, die keinem Unicode-Zeichen zugeordnet werden können) aufgeführt werden.

17048 total glyphs in the document; breakdown by font:

```
85.21% TheSansLight-Plain: 14527 glyphs
5.19% TheSansLight-Italic: 885 glyphs
4.83% ThesisAntiqua-Bold: 823 glyphs, 3 uses of ligatures: fi
2.87% TheSansMonoCondensed-Plain: 489 glyphs
```

0.33% TheSansSemilight-Caps: 57 glyphs  
0.33% TheSansLight-Plain: 56 glyphs  
0.25% TheSansLight-Italic: 42 glyphs  
0.17% TheSansExtraLight-Italic: 29 glyphs  
0.16% TheSansLight-Plain: 28 glyphs  
0.16% TheSansLight-Plain: 28 glyphs  
0.16% TheSansLight-Italic: 28 glyphs  
0.16% TheSansLight-Plain: 28 glyphs  
0.06% TheSansBold-Plain: 10 glyphs  
0.05% TheSans-Plain: 9 glyphs  
0.04% WarnockPro-It: 7 glyphs, 7 uses of ligatures: fi fl ffi Th sp ct st  
0.01% PDFlibLogo2-Regular: 1 glyphs, 1 uses of ligatures: PDFlib  
0.01% WarnockPro-Regular: 1 glyphs

**Erzeugen eines Index.** Das Stylesheet *index.xsl* erwartet TETML-Eingabe im Modus *word* oder *wordplus*. Es erzeugt einen Index wie am Ende eines Buches, d.h. eine alphabetisch sortierte Liste der Wörter im Dokument mit Angabe der Seitenzahlen. Zahlen und Satzzeichen werden ignoriert.

Alphabetische Liste der Wörter im Dokument mit Seitenangaben:

A  
able 5  
about 2  
About 6  
accent 3  
Accented 3  
accents 3  
accept 5  
Accepted 1  
access 6  
accessible 6  
achieved 3  
Acrobat 1 2 4 6  
actual 2  
actually 5  
added 5  
adding 6  
addition 1 2 5  
additional 2 4 5  
Adobe 2 5 6  
advanced 1  
algorithm 3 4  
...

**Extraktion von XMP-Metadaten.** Das Stylesheet *metadata.xsl* erwartet TETML-Eingabe in einem beliebigen Modus. Es erkennt XMP-Metadaten auf Dokumentebene und extrahiert einige Metadaten-Eigenschaften aus XMP. PDF-Anhänge im Dokument (einschließlich PDF-Paketen und -Portfolios) werden rekursiv verarbeitet:

dc:creator = PDFlib GmbH  
xmp:CreatorTool = Adobe InDesign CS6 (Windows)

**Extraktion von Inhalten im CSV-Format.** Das Stylesheet *table.xsl* erwartet TETML-Eingabe im Modus *word*, *wordplus* oder *page*. Es extrahiert die Inhalte einer ausgewählten Tabelle und erzeugt eine CSV-Datei (komma-separierte Werte) mit den Tabelleninhalten.

ten. CSV-Dateien können mit allen Tabellenkalkulationsprogrammen geöffnet werden. Dies kann nützlich sein, um die Inhalte von Tabellen in PDF-Dokumenten weiterzuverarbeiten.

**Konvertierung von TETML nach HTML.** Das Stylesheet *tetml2html.xsl* erwartet TETML-Eingabe im Modus *wordplus*. Es konvertiert TETML nach HTML zur Anzeige im Browser. Der Konverter versucht nicht, das PDF-Dokument visuell identisch nachzubilden, sondern zeigt die folgenden Aspekte:

- ▶ Erzeugen eines verlinkten Inhaltsverzeichnisses am Anfang einer HTML-Seite, wobei die Einträge auf PDF-Lesezeichen oder Überschriften im Dokument basieren.
- ▶ Erzeugen von Überschriften-Elementen (*H1*, *H2* usw.) basierend auf konfigurierbaren Fontgrößen und -namen.
- ▶ Konvertieren von Link-Anmerkungen vom Typ URI in HTML-Links.
- ▶ Zuordnen von TETML-Tabellenelementen auf die entsprechenden Tabellenkonstrukte in HTML zur Anzeige der Tabellen im Browser.
- ▶ Erstellen einer Liste von Bildern für jede Seite, wobei die Bilder mit der zugehörigen Bilddatei verknüpft sind.
- ▶ Anlegen von Links basierend auf PDF-Anmerkungen.

**Extraktion von Rohtext aus TETML.** Das Stylesheet *textonly.xsl* erwartet TETML-Eingabe in einem beliebigen Modus. Es extrahiert Rohtext, indem es nur die *Text*-Elemente ausliest und alle anderen Elemente ignoriert. PDF-Anhänge im Dokument (einschließlich PDF-Paketen und -Portfolios) werden rekursiv verarbeitet.

# 10 API-Referenz für die TET-Bibliothek

## 10.1 Optionslisten

Optionslisten bieten eine ebenso leistungsstarke wie einfache Methode zur Steuerung von PDFlib-API-Funktionsaufrufen. Statt eine Vielzahl von einzelnen Funktionsparametern zu verlangen, unterstützen viele API-Methoden Optionslisten (*optlists*). Dabei handelt es sich um Strings, die beliebig viele Optionen enthalten können. Sie unterstützen verschiedene Datentypen und zusammengesetzte Datenstrukturen wie Listen. In den meisten Sprachbindungen lassen sich Optionslisten problemlos durch Konkatenieren der erforderlichen Schlüsselwörter und Werte bilden.

**Bindungen** C-Sprachbindung: Sie können zur Erstellung von Optionslisten die Funktion *sprintf()* nutzen.

**Bindungen** .NET-Sprachbindung: C#-Programmierer sollten beachten, dass die *StringBuilder*-Methode *AppendFormat()* die geschweiften Klammern `{` und `}` zur Darstellung von Formatelementen verwendet, die durch die String-Darstellung der Argumente ersetzt werden. Im Unterschied dazu haben die Klammerzeichen in der Methode *Append()* keine Sonderbedeutung. Da in der Syntax von Optionslisten geschweifte Klammern verwendet werden, sollten Sie darauf achten, die Methoden *AppendFormat()* oder *Append()* korrekt anzuwenden.

### 10.1.1 Syntax von Optionslisten

**Formale Syntaxdefinition von Optionslisten.** Optionslisten müssen nach folgenden Regeln konstruiert werden:

- ▶ Alle Elemente (Schlüssel und Werte) in einer Optionsliste müssen durch eines oder mehrere der folgenden Zeichen getrennt werden: Leerzeichen, Tabulator, Carriage Return, Newline oder durch ein Gleichheitszeichen '='.
- ▶ Das äußerste Paar von geschweiften Klammern ist nicht Teil des Elements. Aufeinanderfolgende geschweifte Klammern `{ }` kennzeichnen ein leeres Element.
- ▶ Trennzeichen innerhalb des äußersten Paares von geschweiften Klammern teilen das Element nicht, sondern sind Teil des Elements. Deshalb muss ein Element, das Trennzeichen enthält, von geschweiften Klammern eingeschlossen werden.
- ▶ Ein Element, das am Anfang oder Ende Klammern enthält, muss von geschweiften Klammern eingeschlossen werden.
- ▶ Wenn ein Element unausgeglichene Klammerzeichen enthält, müssen diese mit einem vorangestellten Backslash geschützt werden. Einem Backslash vor der schließenden Klammer eines Elements muss ein zusätzlicher Backslash vorangestellt werden.
- ▶ Optionslisten dürfen keine binären Null-Werte enthalten.

Eine Option kann einen Listenwert wie in dieser Referenz angegeben haben. Listenwerte enthalten ein oder mehrere Elemente (die selbst Listen sein können). Sie werden nach

den oben genannten Regeln getrennt, mit dem einzigen Unterschied, dass das Gleichheitszeichen nicht mehr als Trennzeichen behandelt wird.

**Einfache Optionslisten.** In vielen Fällen enthalten Optionslisten ein oder mehrere Schlüssel-/Wertpaare. Schlüssel und Wert sowie die Paare selbst müssen durch Leerzeichen, Tabulator, Carriage Return oder Newline getrennt werden. Sie können zwischen Schlüssel und Wert auch ein Gleichheitszeichen '=' setzen:

```
schlüssel=wert
schlüssel = wert
schlüssel wert
schlüssel1 = wert1  schlüssel2 = wert2
```

Zur Verbesserung der Lesbarkeit empfehlen wir, zwischen Schlüssel und Wert ein Gleichheitszeichen zu setzen und zwischen aufeinanderfolgende Schlüssel-/Wertpaare ein Leerzeichen.

Da Optionslisten von links nach rechts ausgewertet werden, kann eine Option in der Liste mehrfach übergeben werden. In diesem Fall hat das letzte Auftreten der Option Vorrang vor früheren Auftritten. Im folgenden Beispiel wird die erste Wertzuweisung an die Option *schlüssel* durch die zweite überschrieben, und *schlüssel* besitzt nach Verarbeitung der Optionsliste den Wert 2:

```
schlüssel=wert1 schlüssel=wert2
```

**Listenwerte.** Listenwerte bestehen aus mehreren Werten, die einfache Werte oder wiederum Listenwerte sein können. Listen werden mit { und } geklammert und müssen durch Leerzeichen, Tabulator, Carriage Return oder Newline getrennt werden, zum Beispiel:

```
searchpath={/usr/lib/tet d:\tet}          (Liste mit zwei Verzeichnisnamen)
```

Eine Liste kann auch aus verschachtelten Listen bestehen. Die Listen müssen in diesem Fall durch Leerzeichen voneinander getrennt werden. Zwischen den aufeinanderfolgenden Klammerzeichen } und { muss ein Leerzeichen gesetzt werden; dieses kann bei Klammern des gleichen Typs weggelassen werden:

```
fold={ {[:Private_Use:] remove} {[U+FFFD] remove} }  (Liste bestehend aus zwei Listen)
```

Wenn die Liste aus genau einer verschachtelten Liste besteht, dürfen die Klammern für die untergeordnete Liste nicht weggelassen werden:

```
fold={ {[:Private_Use:] remove} }          (Liste mit einer verschachtelten Liste)
```

**Verschachtelte Optionslisten und Listenwerte.** Einige Optionen arbeiten mit dem Typ *Optionsliste* oder *Liste von Optionslisten*. Optionen des Typs *Optionsliste* enthalten eine oder mehrere verschachtelte Optionen. Optionen des Typs *Liste von Optionslisten* enthalten eine oder mehrere verschachtelte Optionslisten. Achten Sie bei verschachtelten Optionslisten auf die richtige Anzahl schließender Klammern. Im Folgenden finden Sie hierzu einige Beispiele.

Der Wert der Option *contentanalysis* ist eine Optionsliste, die die einzige Option *punctuationbreaks* enthält:

```
contentanalysis={punctuationbreaks=false}
```

Der Wert der Option *glyphmapping* im folgenden Beispiel ist eine Liste von Optionslisten, die eine einzige Optionsliste enthält:

```
glyphmapping={ {fontname=GlobeLogosOne codelist=GlobeLogosOne} }
```

Der Wert der Option *glyphmapping* im folgenden Beispiel ist eine Liste von Optionslisten, die zwei Optionslisten enthält:

```
glyphmapping { {fontname=CMSY* glyphlist=tarski} {fontname=ZEH* glyphlist=zeh}}
```

Liste bestehend aus einer Optionsliste mit einem Wert für *fontname*, der Leerzeichen enthält und deshalb von einem zusätzlichen Klammernpaar eingeschlossen werden muss:

```
glyphmapping={ {fontname={Globe Logos One} codelist=GlobeLogosOne} }
```

Liste bestehend aus zwei Schlüsselwörtern:

```
fonttype={Type1 TrueType}
```

Liste bestehend aus unterschiedlichen Typen – die inneren Listen enthalten jeweils eine Unicode-Menge und ein Schlüsselwort, die äußere Liste enthält zwei Optionslisten und das Schlüsselwort *default*:

```
fold={ {[:Private_Use:] remove} {[U+FFFD] remove} default }
```

Liste bestehend aus einem Rechteck:

```
includebox={{10 20 30 40}}
```

**Häufige Fehler und Stolpersteine.** In diesem Abschnitt werden einige häufige Fehler in Bezug auf die Syntax von Optionslisten aufgeführt.

Geschweifte Klammern sind keine Trennzeichen; das folgende Beispiel ist falsch:

```
schlüssel1 {wert1}schlüssel2 {wert2} FALSCH!
```

Dies löst folgende Fehlermeldung aus: *Unknown option 'wert2'*. Ebenso ist folgendes Beispiel falsch, da die Trennzeichen fehlen:

```
schlüssel{wert} FALSCH!  
schlüssel={{wert1}{wert2}} FALSCH!
```

Klammern müssen ausgeglichen sein; das folgende Beispiel ist falsch:

```
schlüssel={offene klammer {} FALSCH!
```

Dies löst folgende Fehlermeldung aus: *Braces aren't balanced in option list 'key={offene klammer }'*. Einem einzelnen Klammerzeichen als Teil eines Strings muss ein zusätzlicher Backslash vorangestellt werden:

```
schlüssel={geschlossene klammer \} und offene klammer \{ } RICHTIG!
```

Einem Backslash am Ende eines String-Werts muss ein zusätzlicher Backslash vorangestellt werden, wenn darauf ein schließendes Klammerzeichen folgt:

schlüssel{\wert}  
schlüssel={\wert\}

FALSCH!  
RICHTIG!

## 10.1.2 Einfache Datentypen

**String.** Strings sind einfache ASCII-Strings (bzw. EBCDIC-Strings auf EBCDIC-Plattformen) wie sie im Allgemeinen für nicht lokalisierbare Schlüsselwörter verwendet werden. Strings, die Leerzeichen oder Gleichheitszeichen '=' enthalten, müssen mit { und } geklammert werden:

kennwort={ secret string } (Stringwert mit drei Leerzeichen)  
inhalt={length=3mm} (Stringwert mit einem Gleichheitszeichen)

Vor den Zeichen { und } muss ein zusätzlicher Backslash \ stehen, wenn sie zum String gehören sollen.

kennwort={weird\}string} (Stringwert mit einer rechten Klammer)

Einem Backslash vor der schließenden Klammer eines Elements muss ein zusätzlicher Backslash vorangestellt werden:

dateiname={C:\pfad\name\\} (String endet mit einem einzelnen Backslash)

Ein leerer String wird durch ein Klammernpaar dargestellt:

{}

Nicht Unicode-fähige Sprachbindungen: wenn die Optionsliste mit einem [EBCDIC]-UTF-8-BOM beginnt, wird jeder Content-, Hypertext- oder Name-String der Optionsliste als [EBCDIC]-UTF-8-String interpretiert.

**Unichar.** Ein Unichar ist ein einzelner Unicode-Wert, wobei mehrere syntaktische Varianten unterstützt werden: Dezimalwerte  $\geq 10$  (zum Beispiel 173), Hexadezimalwerte, die mit x, X, 0x, 0X oder U+ beginnen (xAD, 0xAD, U+00AD), numerische, Character- oder Glyphnamen-Referenzen, aber ohne Verzierung durch '&' oder ';' (shy, #xAD, #173). Beispiele:

unknownchar=? (Buchstabe)  
unknownchar=63 (dezimal)  
unknownchar=x3F (hexadezimal)  
unknownchar=0x3F (hexadezimal)  
unknownchar=U+003F (Unicode-Notation)  
lineseparator={CRLF} (Standard-Glyphnamen-Referenz)

Einzelne Zeichen, die gleichzeitig eine Ziffer darstellen, werden als literale Zeichen interpretiert und nicht als dezimale Unicode-Werte:

replacementchar=3 (U+0033 DREI, nicht U+0003!)

Unichars müssen im hexadezimalen Bereich 0-0x10FFFF (dezimal 0-1114111) liegen.

**Unicode-Mengen.** Unicode-Mengen können aus folgenden Bausteinen zusammengesetzt werden:

- ▶ Bei Patterns handelt es sich um eine Reihe von Zeichen, die von eckigen Klammern eingeschlossen sind und Listen von Unicode-Zeichen und -Mengen von Unicode-Properties enthalten.
- ▶ Bei Listen handelt es sich um eine Sequenz von Unicode-Zeichen mit Bereichen, die durch '-' zwischen zwei Zeichen dargestellt werden, wie in *U+FB00-U+FB17*. Die Sequenz gibt den Bereich aller Zeichen von links nach rechts in Unicode-Reihenfolge an. Mehrfache Unicode-Zeichen dürfen nicht durch Leerzeichen getrennt werden, sondern müssen direkt aufeinander folgen, zum Beispiel *U+0048U+006C*.
- ▶ Unicode-Zeichen in Listen können folgendermaßen angegeben werden:  
ASCII-Zeichen können als literale Buchstaben angegeben werden  
Genau 4-stellige hexadezimale Zahl: `\uhhhh` oder `U+hhhh`  
Genau 5-stellige hexadezimale Zahl: `U+hhhhh`  
1-6-stellige hexadezimale Zahl: `\x{hhhhhh}`  
Genau 8-stellige hexadezimale Zahl: `\Uhhhhhhhh`  
Geschützter Backslash: `\\`
- ▶ Unicode-Mengen von Properties werden als Unicode-Property ausgedrückt. Die Syntax zur Angabe der Property-Namen ist eine Erweiterung der POSIX- und Perl-Syntax, wobei *type* den Namen einer Unicode-Property bezeichnet (siehe [www.unicode.org/Public/UNIDATA/PropertyAliases.txt](http://www.unicode.org/Public/UNIDATA/PropertyAliases.txt)) und *value* den entsprechenden Wert (siehe [www.unicode.org/Public/UNIDATA/PropertyValueAliases.txt](http://www.unicode.org/Public/UNIDATA/PropertyValueAliases.txt)):  
Syntax im POSIX-Stil: `[:type=value:]`  
Syntax im POSIX-Stil mit Negation: `[:^type=value:]`  
Syntax im Perl-Stil: `\p{type=value}`  
Syntax im Perl-Stil mit Negation: `\P{type=value}`  
*type=* kann bei den Properties *Category* und *Script* entfallen, ist aber für andere Properties erforderlich.
- ▶ Mengenoperationen können auf Patterns angewendet werden:  
Um zwei Mengen zu vereinigen, hängen Sie sie aneinander: `[:letter:] [:number:]`  
Um die Schnittmenge zweier Mengen zu erhalten, verwenden Sie den Operator '&':  
`[:letter:] & [U+0061-U+007A]`  
Um die Differenz zweier Mengen zu erhalten, verwenden Sie den Operator '-':  
`[:letter:]-[U+0061-U+007A]`  
Um eine Menge zu invertieren, platzieren Sie ein '^' direkt hinter die öffnende Klammer '[':  
`^[U+0061-U+007A]`  
In allen anderen Positionen hat das '^' keine spezielle Bedeutung.

Für Beispiele zu Unicode-Mengen: siehe Tabelle 10.1. Auf der folgenden Website können Sie Ausdrücke von Unicode-Mengen interaktiv testen:

[unicode.org/cldr/utility/list-unicodeset.jsp](http://unicode.org/cldr/utility/list-unicodeset.jsp)

**Boolean.** Boolesche Optionen haben die Werte *true* oder *false*; wird bei einer Option Booleschen Typs kein Wert angegeben, wird von *true* ausgegangen. Als abkürzende Schreibweise kann *noname* statt *name=false* verwendet werden:

`usehostfonts` (äquivalent zu `usehostfonts=true`)  
`nousehostfonts` (äquivalent zu `usehostfonts=false`)

Tabelle 10.1 Beispiele für Unicode-Mengen

Unicode-Menge	Zeichen in der Unicode-Menge
[U+0061-U+007A]	Kleinbuchstaben a bis z
[U+0640]	einzelnes Zeichen Arabic Tatweel
[\x{0640}]	einzelnes Zeichen Arabic Tatweel
[U+FB00-U+FB17]	Lateinische und armenische Ligaturen
[^U+0061-U+007A]	alle Zeichen außer a bis z
[ :Lu: ] [ :UppercaseLetter: ]	alle Großbuchstaben (Kurz- und Langformen der Unicode-Menge)
[ :L: ] [ :Letter: ]	alle Unicode-Kategorien beginnend mit L (Kurz- und Langformen der Unicode-Menge)
[ :General_Category=Dash_Punctuation: ]	alle Zeichen in der allgemeinen Kategorie Dash_Punctuation
[ :Alphabetic=No: ]	alle nicht alphabetischen Zeichen
[ :Private_Use: ]	alle Zeichen in der Private Use Area (PUA)

**Schlüsselwort.** Eine Option vom Typ Schlüsselwort kann eine vordefinierte Liste von festen Schlüsselwörtern enthalten. Beispiel:

```
clippingarea=cropbox
```

Bei einigen Optionen kann der Wert entweder eine Zahl oder ein Schlüsselwort sein.

**Zahl.** Optionslisten unterstützen verschiedene numerische Typen.

Ganzzahlen (Integers) können dezimal oder hexadezimal sein. Integers können mit x, X, ox oder oX hexadezimale Werte angeben:

```
-12345  
0  
0xFF
```

Floats können dezimale Gleitkomma- oder Ganzzahlen enthalten; zur Trennung von Vor- und Nachkommastellen sind Punkt und Komma zulässig. Exponentialdarstellung wird ebenfalls unterstützt. Die folgenden Werte sind alle gleichbedeutend:

```
size = -123.45  
size = -123,45  
size = -1.2345E2  
size = -1.2345e+2
```

### 10.1.3 Geometrische Typen

**Rechteck.** Ein Rechteck besteht aus einer Liste von vier Float-Werten, die die x- und y-Koordinaten der linken unteren und der rechten oberen Ecke des Rechtecks festlegen. Das Koordinatensystem zur Interpretation der Rechteckkoordinaten (Standard- oder Benutzerkoordinatensystem) kann je nach Option unterschiedlich sein und wird deswegen bei der jeweiligen Option beschrieben, zum Beispiel:

```
includebox = {{0 0 500 100} {0 500 500 600}}
```

## 10.1.4 Encoding-Namen

Encoding-Namen werden von verschiedenen Optionen und Parametern unterstützt, z.B. von der Option *filenamehandling* von *TET\_set\_option()*, der Option *forceencoding* von *TET\_open\_document()* und dem Parameter *inputformat* von *TET\_convert\_to\_unicode()*. Die folgenden Schlüsselwörter können als Encoding-Namen übergeben werden:

- ▶ Mit dem Schlüsselwort *auto* wird das gängigste Encoding für eine bestimmte Umgebung ausgewählt:
  - ▶ Windows: die aktuelle System-Codepage
  - ▶ Unix und OS X: *iso8859-1*
  - ▶ i5/iSeries: Encoding des aktuellen Jobs (*IBMCCSIDoooooooooooo*)
  - ▶ zSeries: *ebcdic*
- ▶ *winansi* (=cp1252)
- ▶ *iso8859-1 - iso8859-10, iso8859-13 - iso8859-14*
- ▶ *cp1250 - cp1258*
- ▶ *macroman, macroman\_euro* (ersetzt das Währungssymbol durch Euro), *macroman\_apple* (ersetzt die Währung durch Euro und fügt zusätzliche mathematische/griechische Symbole hinzu)
- ▶ *adobesymbol* bezeichnet das Encoding Adobe Symbol
- ▶ *U+XXXX* (256 Zeichen beginnend beim angegebenen Wert)
- ▶ *ebcdic* (=Codepage 1047), *ebcdic\_37* (=Codepage 037)
- ▶ CJK-Encodings *cp932, cp936, cp949, cp950*
- ▶ auf den folgenden Systemen können alle auf dem Host-System verfügbaren Encodings verwendet werden:
  - ▶ Windows: *cpXXXX*
  - ▶ Linux: alle Codesets, die in *iconv* bekannt sind
  - ▶ i5/iSeries: jeder *Coded Character Set Identifier* ohne das Präfix *CCSID*
  - ▶ zSeries: jeder *Coded Character Set Identifier* (*CCSID*)
- ▶ benutzerdefinierte Encodings können als Ressourcen definiert und über ihren Ressourcennamen referenziert werden

# 10.2 Allgemeine Funktionen

## 10.2.1 Umgang mit Optionen

---

```
C++ Java void set_option(String optlist)
Perl PHP set_option(string optlist)
C void TET_set_option(TET *tet, const char *optlist)
```

---

Setzt eine oder mehrere globale Optionen für TET.

**optlist** Optionsliste mit globalen Optionen gemäß Tabelle 10.2. Wird eine Option mehrfach übergeben, überschreibt der letzte Wert alle früheren. Um einer Option mehrere Werte mitzugeben (z.B. *searchpath*), übergeben Sie alle Werte in einem Listenargument.

Folgende Optionen können verwendet werden: *asciifile*, *cmap*, *codelist*, *encoding*, *filenamehandling*, *fontoutline*, *glyphlist*, *license*, *licensefile*, *logging*, *userlog*, *outputformat*, *resourcefile*, *searchpath*

**Details** Mit mehrfachen Aufrufen dieser Funktion können Werte für die in Tabelle 10.2 markierten Optionen akkumuliert werden. Bei nicht markierten Optionen überschreibt der neue Wert den alten.

Tabelle 10.2 Globale Optionen für *TET\_set\_option()*

Option	Beschreibung
<i>asciifile</i>	(Boolean; nur unterstützt bei <i>i5/iSeries</i> und <i>zSeries</i> ). Erwartet Textdateien (z.B. UPR-Konfigurationsdateien, Glyphlisten, Codelisten) im ASCII-Encoding. Standardwert: <i>true</i> bei <i>i5/iSeries</i> ; <i>false</i> bei <i>zSeries</i>
<i>cmap</i> <sup>1, 2</sup>	(Liste von Name-Strings) Liste mit String-Paaren, wobei jedes Paar Name und Wert einer CMap-Ressource enthält (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70).
<i>codelist</i> <sup>1, 2</sup>	(Liste von Name-Strings) Liste mit String-Paaren, wobei jedes Paar Name und Wert einer Codelisten-Ressource enthält (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70).
<i>encoding</i> <sup>1, 2</sup>	(Liste von Name-Strings) Liste mit String-Paaren, wobei jedes Paar Name und Wert einer Encoding-Ressource enthält (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70).

Tabelle 10.2 Globale Optionen für `TET_set_option()`

Option	Beschreibung
<b>filename-handling</b>	<p>(Schlüsselwort) Encoding von Dateinamen. Als Funktionsparameter ohne UTF-8-BOM in nicht Unicode-fähigen Sprachbindungen übergebene Dateinamen werden gemäß dieser Option interpretiert, um im Dateisystem nicht zulässige Zeichen zu vermeiden und um eine Unicode-Version der Dateinamen zu erzeugen. Enthält der Dateiname Zeichen außerhalb des angegebenen Encodings, wird eine Exception ausgelöst. Standardwert: unicode für Windows und OS X, auto für i5/iSeries, sonst honorlang:</p> <p><b>ascii</b> 7-Bit-ASCII</p> <p><b>basicebdcic</b> EBCDIC gemäß Codepage 1047, aber nur mit Unicode-Werten <math>\leq U+007E</math></p> <p><b>basicebdcic_37</b> EBCDIC gemäß Code-Page 0037, aber nur Unicode-Werte <math>\leq U+007E</math></p> <p><b>honorlang</b> (Für i5/iSeries nicht unterstützt) Die Umgebungsvariablen <code>LC_ALL</code>, <code>LC_CTYPE</code> und <code>LANG</code> werden interpretiert. Sofern vorhanden, wird der in <code>LANG</code> angegebene Codeset auf die Dateinamen angewendet.</p> <p><b>legacy</b> Verwendet das Encoding <code>auto</code> (d.h. das aktuelle System-Encoding) zur Interpretation des Dateinamens und interpretiert die Variable <code>LANG</code>, sofern der Parameter <code>honorlang</code> gesetzt ist.</p> <p><b>unicode</b> Unicode-Encoding im (EBCDIC-)UTF-8-Format</p> <p><b>alle Namen eines 8-Bit- und CJK-Encodings</b> Encoding-Name gemäß Abschnitt 10.1.4, »Encoding-Namen«, Seite 179</p>
<b>fontoutline</b> <sup>1, 2</sup>	(Liste von Name-Strings) Liste mit String-Paaren, wobei jedes Paar Name und Wert einer FontOutline-Resource enthält (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70).
<b>glyphlist</b> <sup>1, 2</sup>	(Liste von Name-Strings) Liste mit String-Paaren, wobei jedes Paar Name und Wert einer Glyphlisten-Resource enthält (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70).
<b>hostfont</b> <sup>1, 2</sup>	(Liste von Name-Strings) Liste mit String-Paaren, wobei jedes Paar einen PDF-Fontnamen und den UTF-8-kodierten Namen eines Host-Fonts enthält, der als nicht eingebetteter Font verwendet werden soll.
<b>license</b>	(String) Setzt den Lizenzschlüssel. Diese Option muss vor dem ersten Aufruf von <code>TET_open_document*()</code> gesetzt werden.
<b>licensefile</b>	(String) Setzt den Namen einer Datei mit Lizenzschlüsseln. Die Lizenzdatei kann nur einmal vor dem ersten Aufruf von <code>TET_open_document*()</code> festgelegt werden. Alternativ kann der Name der Lizenzdatei in einer Umgebungsvariablen namens <code>PDFLICENSEFILE</code> oder unter Windows in der Registry übergeben werden.
<b>logging</b> <sup>1</sup>	(Optionsliste; nicht unterstützt) Optionsliste zur Steuerung der Logging-Ausgabe gemäß Tabelle 10.7. Alternativ können Logging-Optionen in einer Umgebungsvariablen namens <code>TETLOGGING</code> oder unter Windows in der Registry übergeben werden. Mit einer leeren Optionsliste kann die Protokollierung mit den in vorigen Aufrufen gesetzten Optionen aktiviert werden. Ist die Umgebungsvariable gesetzt, startet die Protokollierung direkt nach dem ersten Aufruf von <code>TET_new()</code> .
<b>userlog</b>	(Name-String; nicht unterstützt) Beliebiger String, der in die Logdatei geschrieben wird, wenn die Protokollierung aktiviert ist.
<b>output-format</b>	<p>(Schlüsselwort; nur für die Sprachbindungen C, Ruby, Perl, Python und PHP) Legt das Format des Textes fest, der von <code>TET_get_text()</code> zurückgegeben wird:</p> <p><b>utf8</b> Strings werden in (bei C: null-terminiertem) UTF-8-Format zurückgegeben.</p> <p><b>utf16</b> Strings werden im UTF-16-Format mit der Bytereihenfolge des Systems zurückgegeben.</p> <p><b>utf32</b> Strings werden im UTF-32-Format mit der Bytereihenfolge des Systems zurückgegeben.</p> <p><b>ebcdicutf8</b> (Nur auf EBCDIC-Systemen verfügbar) Strings werden in null-terminiertem UTF-8-Format in EBCDIC-Kodierung zurückgegeben. Auf i5/iSeries-Systemen wird Codepage 00037 und auf zSeries-Systemen Codepage 01047 verwendet.</p> <p>Standardwert: <code>utf8</code> für C, Ruby, Perl, Python, PHP und <code>ebcdicutf8</code> für C bei i5/iSeries und zSeries</p>

Tabelle 10.2 Globale Optionen für `TET_set_option()`

Option	Beschreibung
<b>resourcefile</b>	<p>(Name-String) Relativer oder absoluter Dateiname der UPR-Ressourcendatei. Die Ressourcendatei wird sofort geladen. Vorhandene Ressourcen bleiben erhalten; ihre Werte werden ggf. durch neue Werte überschrieben. Explizit gesetzte Ressourcen-Optionen werden nach den Einträgen in der Ressource-Datei ausgewertet.</p> <p>Der Ressource-Dateiname kann auch in der Umgebungsvariablen <code>TETRESOURCEFILE</code> oder mit einem Schlüssel in der Windows-Registry übergeben werden (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70). Standardwert: <code>tet.upr</code> (auf MVS: <code>upr</code>)</p>
<b>searchpath<sup>1</sup></b>	<p>(Liste von Name-Strings) Relative oder absolute Pfadname(n) eines Verzeichnisses mit zu lesenden Dateien. Der Suchpfad kann mehrfach gesetzt werden; die Einträge werden aneinandergelinkt und in umgekehrter Reihenfolge ihrer Definition abgearbeitet (für weitere Informationen siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70). Es wird empfohlen, selbst für einen einzigen Eintrag doppelt geschweifte Klammern zu verwenden, um Probleme mit Leerzeichen in Verzeichnisnamen zu vermeiden. Eine leere String-Liste (d.h. <code>{}</code>) löscht alle bestehenden <code>Searchpath</code>-Einträge einschließlich der Standardeinträge. Bei Windows kann der <code>Searchpath</code> auch über die Registry gesetzt werden. Standardwert: plattformabhängig, siehe »Dateisuche und Ressourcenkategorie <code>searchpath</code>«, Seite 71.</p>
<b>shutdown-strategy</b>	<p>(Integer) Methode für die Freigabe von globalen Ressourcen, die einmal für alle TET-Objekte alloziert werden. Jede globale Ressource wird initialisiert, sobald sie zum ersten Mal benötigt wird. Diese Option muss in einem Prozess für alle TET-Objekte auf denselben Wert eingestellt werden, da sonst das Verhalten undefiniert ist (Standardwert: 0):</p> <ul style="list-style-type: none"> <li>0 Ein Verweiszähler verfolgt, wie viele TET-Objekte die Ressource nutzen. Wenn das letzte TET-Objekt gelöscht wird und der Verweiszähler auf Null fällt, wird die Ressource freigegeben.</li> <li>1 Die Ressource wird bis zum Ende des Prozesses gehalten. Damit lässt sich die Leistung leicht steigern, es wird aber immer noch Speicher belegt, nachdem das letzte TET-Objekt gelöscht wurde.</li> </ul>

1. Optionswerte können mit Mehrfach-Aufrufen akkumuliert werden.

2. Anders als bei der UPR-Syntax ist kein Gleichheitszeichen '=' zwischen Name und Wert erforderlich bzw. erlaubt.

## 10.2.2 Setup

### C `TET *TET_new(void)`

Erzeugt ein neues TET-Objekt.

**Rückgabe** Handle für ein TET-Objekt, das in nachfolgenden Aufrufen verwendet werden kann. Schlägt diese Funktion aufgrund von mangelndem Speicher fehl, wird NULL zurückgegeben.

**Bindungen** Diese Funktion wird von objektorientierten Sprachbindungen nicht unterstützt, da sie im TET-Konstruktor verborgen ist.

---

**Java** `void delete()`  
**C#** `void Dispose()`  
**C** `void TET_delete(TET *tet)`

---

Löscht ein TET-Objekt und gibt alle zugehörigen internen Ressourcen frei.

**Details** Mit dem Löschen eines TET-Objekts werden automatisch alle seine geöffneten Dokumente geschlossen. Das TET-Objekt darf nach dem Löschen in keiner Funktion mehr benutzt werden.

**Bindungen** Diese Funktion ist in objektorientierten Sprachbindungen im allgemeinen nicht erforderlich, da sie im TET-Destruktor verborgen ist. Allerdings steht sie in Java zur expliziten Bereinigung zusätzlich zur automatischen Speicherbereinigung trotzdem zur Verfügung. In .NET sollte `Dispose()` am Ende der Verarbeitung aufgerufen werden, um »unmanaged« Ressourcen zu bereinigen.

## 10.2.3 PDFlib Virtual Filesystem (PVF)

---

**C++** `void create_pvf(wstring filename, const void *data, size_t size, wstring optlist)`  
**C# Java** `void create_pvf(String filename, byte[] data, String optlist)`  
**Perl PHP** `create_pvf(string filename, string data, string optlist)`  
**C** `void TET_create_pvf(TET *tet,  
const char *filename, int len, const void *data, size_t size, const char *optlist)`

---

Erzeugt eine benannte virtuelle, schreibgeschützte Datei aus Daten im Speicher.

**filename** (Name-String) Der Name der virtuellen Datei. Dies ist ein beliebiger String, mit dem in weiteren TET-Aufrufen die virtuelle Datei referenziert werden kann.

**len** (Nur C-Sprachbindung) Länge von *filename* (in Bytes) für UTF-16-Strings. Ist *len=0*, muss ein null-terminierter String übergeben werden.

**data** Verweis auf die Daten, die den Inhalt der virtuellen Datei bilden sollen. In COM ist es ein Variantentyp mit Bytes, der den Inhalt der virtuellen Datei enthält. In C und C++ handelt es sich um einen Zeiger auf einen Speicherbereich. In Java ist es ein Byte-Array. In Perl und PHP ist es ein String.

**size** (Nur C- und C++-Sprachbindung) Länge des Speicherblocks mit den Daten in Bytes.

**optlist** Optionsliste gemäß Tabelle 10.3. Die folgende Option kann verwendet werden:  
*copy*

**Details** Der Name der virtuellen Datei kann an alle API-Funktionen übergeben werden, die Eingabedateien verarbeiten. Manche Funktionen sperren die virtuelle Datei, solange die Daten verwendet werden. Virtuelle Dateien werden solange im Speicher gehalten, bis sie mit `TET_delete_pvf()` explizit oder mit `TET_delete()` automatisch gelöscht werden.

PVF-Dateien werden für jedes TET-Objekt getrennt gespeichert. Virtuelle Dateien lassen sich nicht von verschiedenen TET-Objekten gemeinsam nutzen. Arbeiten Threads mit verschiedenen TET-Objekten, müssen sie den PVF-Gebrauch nicht synchronisieren. Verweist *filename* auf eine bereits existierende virtuelle Datei, wird eine Exception aus-

gelöst. Diese Funktion überprüft nicht, ob *filename* bereits für eine auf der Festplatte liegende Datei verwendet wird.

Wird die Option *copy* nicht angegeben, darf der Aufrufer die übergebenen Daten erst nach dem erfolgreichen Aufruf von *TET\_delete\_pvf()* ändern oder freigeben (löschen). Bei Nichtbeachtung dieser Regel droht ein Absturz.

Tabelle 10.3 Option für *TET\_create\_pvf()*

Option	Beschreibung
<i>copy</i>	(Boolean) TET erzeugt sofort eine interne Kopie der übergebenen Daten. Damit kann der Aufrufer die übergebenen Daten unmittelbar nach dem Aufruf löschen. Die Option <i>copy</i> ist in den Sprachbindungen für COM, .NET und Java automatisch auf <i>true</i> gesetzt (Standardwert für andere Sprachbindungen: <i>false</i> ). In anderen Sprachbindungen werden die Daten nur kopiert, wenn die Option <i>copy</i> explizit übergeben wird.

---

**C++ Java** *int delete\_pvf(String filename)*

**Perl PHP** *int delete\_pvf(string filename)*

**C** *int TET\_delete\_pvf(TET \*tet, const char \*filename, int len)*

---

Löscht eine benannte virtuelle Datei und gibt die zugehörigen Datenstrukturen frei.

**filename** (Name-String) Der Name der virtuellen Datei wie an *TET\_create\_pvf()* übergeben.

**len** (Nur C-Sprachbindung) Länge von *filename* (in Bytes) für UTF-16-Strings. Ist *len=0*, muss ein null-terminierter String übergeben werden.

**Rückgabe** -1, falls die zugehörige virtuelle Datei existiert und gesperrt ist, sonst 1.

**Details** Ist die Datei nicht gesperrt, werden die zu *filename* gehörigen Datenstrukturen sofort von TET gelöscht. Verweist *filename* nicht auf eine virtuelle Datei, kehrt die Funktion sofort zurück. Nach einem erfolgreichen Aufruf kann *filename* wieder verwendet werden. Virtuelle Dateien werden durch *TET\_delete()* automatisch gelöscht.

Das genaue Verhalten hängt davon ab, ob das zugehörige *TET\_create\_pvf()* mit der Option *copy* aufgerufen wurde: Ist dies der Fall, werden sowohl die administrativen Datenstrukturen der Datei als auch die eigentlichen Daten freigegeben; andernfalls obliegt die Freigabe des Inhalts dem Client.

---

**C++ Java** *int info\_pvf(String filename, String keyword)*

**Perl PHP** *int info\_pvf(string filename, string keyword)*

**C** *int TET\_info\_pvf(TET \*tet, const char \*filename, int len, const char \*keyword)*

---

Liefert Eigenschaften einer virtuellen Datei oder des PDFlib Virtual File System (PVF).

**filename** (Name-String) Der Name der virtuellen Datei. Bei *keyword=filecount* kann der Dateiname leer sein.

**len** (Nur C-Sprachbindung) Länge von *filename* (in Bytes) für UTF-16-Strings. Ist *len=0*, muss ein null-terminierter String übergeben werden.

**keyword** Schlüsselwort gemäß Tabelle 10.4.

**Details** Liefert Eigenschaften einer virtuellen Datei oder des PDFlib Virtual File System (PVF). Die gewünschte Eigenschaft wird mit *keyword* angegeben.

Tabelle 10.4 Schlüsselwörter für `TET_info_pvf()`

Option	Beschreibung
<b>filecount</b>	Gesamtzahl der Dateien im PDFlib Virtual File System, die für das aktuelle TET-Objekt verwaltet werden. Die Option <i>filename</i> wird ignoriert.
<b>exists</b>	Gibt 1 aus, wenn die Datei im PDFlib Virtual File System existiert (und nicht gelöscht wurde), sonst 0.
<b>size</b>	(Nur für bestehende virtuelle Dateien) Gibt die Größe der entsprechenden virtuellen Datei in Bytes aus.
<b>iscopy</b>	(Nur für bestehende virtuelle Dateien) Gibt 1 aus, wenn die Option <i>copy</i> bei Erstellung der angegebenen virtuellen Datei übergeben wurde, sonst 0.
<b>lockcount</b>	(Nur für bestehende virtuelle Dateien) Anzahl der Sperren für die angegebene virtuelle Datei, die von TET-Funktionen intern gesetzt wurden. Die Datei kann nur gelöscht werden, wenn der Zähler für die Sperren auf 0 steht.

## 10.2.4 Funktion zur Unicode-Konvertierung

---

**C++** `string convert_to_unicode(wstring inputformat, string input, wstring optlist)`

**C# Java** `String convert_to_unicode(String inputformat, byte[] input, String optlist)`

**Perl PHP** `string convert_to_unicode(string inputformat, string input, string optlist)`

**C** `const char *TET_convert_to_unicode(TET *tet, const char *inputformat, const char *input, int inputlen, int *outputlen, const char *optlist)`

---

Konvertiert einen String mit beliebigem Encoding in einen Unicode-String mit wählbarem Format.

**inputformat** Unicode-Textformat oder Name des Encodings des zu konvertierenden Strings:

- ▶ Unicode-Textformate: *utf8, ebcdicutf8, utf16, utf16le, utf16be, utf32*
- ▶ Encoding-Name gemäß Abschnitt 10.1.4, »Encoding-Namen«, Seite 179
- ▶ Das Schlüsselwort *auto* führt zu folgendem Verhalten: wenn der Eingabe-String einen UTF-8-BOM oder UTF-16-BOM enthält, wird er zur Bestimmung des korrekten Formats verwendet, ansonsten wird die aktuelle Codepage des Systems herangezogen.

**input** Nach Unicode zu konvertierender String.

**inputlen** (Nur C-Sprachbindung) Länge des zu konvertierenden Strings in Bytes. Ist *inputlen=0*, muss ein null-terminierter String übergeben werden.

**outputlen** (Nur C-Sprachbindung) C-Zeiger auf einen Speicherplatz, an dem die Länge des zurückgegebenen Strings in Bytes abgelegt wird.

**optlist** Optionsliste mit Optionen gemäß Tabelle 10.5.

- ▶ Optionen für Eingabefilter: *charref, escapesequence*
- ▶ Optionen für Unicode-Konvertierung: *bom, errorpolicy, inflate, outputformat*

**Rückgabe** Ein aus dem zu konvertierenden String gemäß den angegebenen Parametern und Optionen erzeugter Unicode-String. Wenn der zu konvertierende String nicht dem angege-

benen Eingabeformat entspricht (z.B. bei einem ungültigen UTF-8-String), wird bei *errorpolicy=return* ein leerer String zurückgegeben und bei *errorpolicy=exception* wird eine Exception ausgelöst.

**Details** Diese Funktion ist für die Unicode-Konvertierung von Strings nützlich, besonders wenn Sie in einer Umgebung ohne geeignete Unicode-Konverter arbeiten.

**Bindungen** C-Sprachbindung: Bis zu 10 zurückgegebene String-Einträge werden in einem Ring-Puffer abgelegt. Werden mehr als 10 Strings konvertiert, werden die Puffer wiederverwendet. Clients müssen die Strings deshalb kopieren, wenn sie auf mehr als 10 Strings gleichzeitig zugreifen möchten. Bis zu 10 Aufrufe dieser Funktion können zum Beispiel als Parameter für eine *printf()*-Anweisung verwendet werden, da die Rückgabe-Strings unabhängig voneinander sind, sofern nicht mehr als 10 Strings gleichzeitig verwendet werden.

C++-Sprachbindung: Die Parameter *inputformat* und *optlist* müssen als *wstrings* übergeben werden, *input* und die zurückgegebenen Daten müssen jedoch vom Typ *string* sein.

Python-Sprachbindung: UTF-8-Ergebnisse werden als String zurückgegeben; Python 3: Rückgabewerte, die nicht in UTF-8 kodiert sind, werden als Bytes zurückgegeben.

Tabelle 10.5 Optionen für `TET_convert_to_unicode()`

Option	Beschreibung
<b>charref</b>	(Boolean) Bei true werden numerische Referenzen, Character-Referenzen und Glyphnamen-Referenzen ersetzt. Standardwert: false
<b>bom</b>	(Schlüsselwort; wird bei <code>outputformat=utf32</code> nicht ausgewertet; bei Unicode-fähigen Sprachbindungen ist nur none erlaubt) Steuert das Hinzufügen eines Byte Order Mark (BOM) zum Ausgabe-String. Unterstützte Schlüsselwörter (Standardwert: none): <b>add</b> Hinzufügen eines BOM <b>keep</b> Hinzufügen eines BOM, wenn der Eingabe-String einen BOM hat <b>none</b> Kein Hinzufügen eines BOM <b>optimize</b> Hinzufügen eines BOM, außer wenn <code>outputformat=utf8</code> oder <code>ebcdicutf8</code> und der Ausgabe-String nur Zeichen kleiner U+007F enthält
<b>errorpolicy</b>	(Schlüsselwort) Verhalten bei einem Konvertierungsfehler (Standardwert: exception): <b>return</b> Das Ersatzzeichen U+FFFD wird verwendet, wenn eine Character-Referenz nicht aufgelöst werden kann. Bei einem Konvertierungsfehler wird ein leerer String ausgegeben. <b>exception</b> Bei einem Konvertierungsfehler wird eine Exception ausgelöst.
<b>escape-sequence</b>	(Boolean) Bei true werden Escape-Sequenzen in Strings ersetzt. Standardwert: false
<b>inflate</b>	(Boolean; nur bei <code>inputformat=utf8</code> ; wird bei <code>outputformat=utf8</code> nicht ausgewertet) Bei true löst ein ungültiger UTF-8-Eingabe-String keine Exception aus, vielmehr wird ein String erzeugt, der die Bytes des Eingabe-Strings als Unicode-Zeichen enthält. Dies kann bei der Fehlersuche hilfreich sein. Standardwert: false
<b>output-format</b>	(Schlüsselwort) Unicode-Textformat des generierten Strings: <code>utf8</code> , <code>ebcdicutf8</code> , <code>utf16</code> , <code>utf16le</code> , <code>utf16be</code> , <code>utf32</code> . Ein leerer String ist äquivalent zu <code>utf16</code> . Standardwert: <code>utf16</code> Unicode-fähige Sprachbindungen: das Ausgabeformat wird immer auf <code>utf16</code> gesetzt. C++-Sprachbindung: nur die folgenden Ausgabeformate sind erlaubt: <code>ebcdicutf8</code> , <code>utf8</code> , <code>utf16</code> , <code>utf32</code> .

## 10.2.5 Verarbeitung von Exceptions

---

**C++ Java** *String get\_apiname()*

**Perl PHP** *string get\_apiname()*

**C** *const char \*TET\_get\_apiname(TET \*tet)*

---

Ermittelt den Namen der API-Funktion, die eine Exception ausgelöst hat oder scheiterte.

**Rückgabe** Name der Funktion, die eine Exception ausgelöst hat oder zuletzt mit einem Fehlercode gescheitert ist. Liegt kein Fehler vor, wird ein leerer String ausgegeben.

---

**C++ Java** *String get\_errmsg()*

**Perl PHP** *string get\_errmsg()*

**C** *const char \*TET\_get\_errmsg(TET \*tet)*

---

Ermittelt den beschreibenden Text der zuletzt ausgelösten Exception oder die Ursache eines gescheiterten Funktionsaufrufs.

**Rückgabe** Text mit der Beschreibung der zuletzt ausgelösten Exception oder mit der Ursache für einen gescheiterten Funktionsaufruf einschließlich Fehlercode. Liegt kein Fehler vor, wird ein leerer String ausgegeben.

---

**C++ Java** *int get\_errnum()*

**Perl PHP** *long get\_errnum()*

**C** *int TET\_get\_errnum(TET \*tet)*

---

Ermittelt die Nummer der zuletzt ausgelösten Exception oder die Ursache für einen gescheiterten Funktionsaufruf.

**Rückgabe** Nummer der Exception oder Fehlercode der zuletzt aufgerufenen Funktion, die mit einem Fehlercode gescheitert ist. Diese Funktion gibt 0 zurück, wenn kein Fehler vorliegt.

---

**C** *TET\_TRY(tet)*

**C** *TET\_CATCH(tet)*

**C** *TET\_RETHROW(tet)*

**C** *TET\_EXIT\_TRY(tet)*

---

Richtet einen Block zur Ausnahmebehandlung ein; fängt eine Exception ab oder löst sie erneut aus; oder benachrichtigt die Ausnahmebehandlung darüber, dass ein *TET\_TRY()*-Block verlassen wird, ohne dass in den entsprechenden *TET\_CATCH()*-Block gesprungen wird. Mit *TET\_RETHROW()* kann eine Exception nach dem Abfangen erneut ausgelöst werden, damit sie auf höherer Ebene abgefangen wird.

**Details** (Nur C-Sprachbindung) Siehe Abschnitt 3.2, »C-Sprachbindung«, Seite 29.

## 10.2.6 Logging

Mit der Logging-Funktion lassen sich API-Aufrufe protokollieren. Der Inhalt der Logdatei kann bei der Fehlersuche hilfreich sein oder vom Support der PDFlib GmbH benötigt werden. Tabelle 10.6 führt Optionen zur Aktivierung von Logging-Funktionen mit `TET_set_option()` auf (siehe Abschnitt 10.2.1, »Umgang mit Optionen«, Seite 180).

Tabelle 10.6 Logging-spezifische Optionen für `TET_set_option()`

Option	Beschreibung
<code>logging</code>	Optionsliste mit Logging-Optionen gemäß Tabelle 10.7
<code>userlog</code>	String, der in die Logdatei geschrieben wird

Die Logging-Optionen können auf folgende Arten übergeben werden:

- ▶ Als Optionsliste für die Option `logging` von `TET_set_option()`, zum Beispiel:

```
tet.set_option("logging={filename={debug.log} remove}")
```

- ▶ In der Umgebungsvariablen `TETLOGGING`. Dies aktiviert die Protokollierung ab dem ersten Aufruf einer API-Funktion

Tabelle 10.7 Unteroptionen für die logging-Option von `TET_set_option()`

Option	Beschreibung
<i>(leere Liste)</i>	Aktiviert die Protokollierung, nachdem sie mit <code>disable</code> deaktiviert wurde.
<code>disable</code>	(Boolean) Protokollierung deaktivieren. Standardwert: <code>false</code>
<code>enable</code>	(Boolean) Protokollierung aktivieren
<code>filename</code>	(String) Name der Logdatei ( <code>stdout</code> und <code>stderr</code> sind ebenfalls erlaubt). Die Ausgabe wird an bereits vorhandene Inhalte angehängt. Der Name der Logdatei kann auch in einer Umgebungsvariable namens <code>TETLOGFILENAME</code> übergeben werden (in diesem Fall wird die Option <code>filename</code> immer ignoriert). Standardwert: <code>tet.log</code> (unter Windows und OS X im Verzeichnis <code>/</code> , unter Unix in <code>/tmp</code> )
<code>flush</code>	(Boolean) Bei <code>true</code> wird die Logdatei nach jeder Ausgabe geschlossen und bei der nächsten Ausgabe erneut geöffnet. Damit ist gewährleistet, dass die Ausgabe sicher in die Datei geschrieben wird. Dies kann bei der Verfolgung von Programmabstürzen mit unvollständiger Logdatei nützlich sein. Die Verarbeitungsgeschwindigkeit verringert sich jedoch erheblich. Bei <code>false</code> wird die Logdatei nur einmal geöffnet. Standardwert: <code>false</code>
<code>includepid</code>	(Boolean; nicht für MVS) Angabe der Prozess-ID im Logdateinamen. Dies sollte aktiviert werden, wenn mehrere Prozesse den selben Logdateinamen verwenden. Standardwert: <code>false</code>
<code>includetid</code>	(Boolean; nicht für MVS) Angabe der Thread-ID im Logdateinamen. Dies sollte aktiviert werden, wenn mehrere Threads im selben Prozess den selben Logdateinamen verwenden. Standardwert: <code>false</code>
<code>includeoid</code>	(Boolean; nicht für MVS) Angabe der Objekt-ID im Logdateinamen. Dies sollte aktiviert werden, wenn mehrere TET-Objekte im selben Thread den selben Logdateinamen verwenden. Standardwert: <code>false</code>
<code>remove</code>	(Boolean) Bei <code>true</code> wird eine gegebenenfalls bereits vorhandene Logdatei gelöscht, bevor die neue Ausgabe geschrieben wird. Standardwert: <code>false</code>
<code>removeon-success</code>	(Boolean) Entfernt die generierte Logdatei in <code>TET_delete()</code> , sofern keine Exception ausgelöst wurde. Dies kann für die Analyse gelegentlicher Probleme in Multi-Threaded-Anwendungen oder bei sporadisch auftretenden Problemen nützlich sein. Wir empfehlen, diese Option nach Bedarf mit <code>includepid/includeid/includeoid</code> zu kombinieren.

Tabelle 10.7 Unteroptionen für die logging-Option von `TET_set_option()`

Option	Beschreibung
<b>stringlimit</b>	(Integer) Limit für die Anzahl der Zeichen in Text-Strings oder 0 für kein Limit. Standardwert: 0
<b>classes</b>	(Optionsliste) Liste mit Optionen des Typs Integer, wobei jede Option eine Logging-Klasse und der zugehörige Wert die Granularität beschreibt. Level 0 deaktiviert eine Logging-Klasse; positive Zahlen aktivieren eine Klasse. Je höher die Granularität ist, desto detaillierter ist die Ausgabe. Folgende Optionen werden unterstützt (Standardwert: {api=1 warning=1}):
<b>api</b>	Protokolliert alle API-Funktionsaufrufe mit Parametern und Rückgabewerten. Bei api=2 wird ein Zeitstempel vor jeden API-Funktionsaufruf gestellt, und veraltete Funktionen und Optionen werden markiert.
<b>filesearch</b>	Protokolliert alle Versuche, Dateien via SearchPath oder PVF zu finden.
<b>resource</b>	Protokolliert alle Versuche, Ressourcen über die Windows-Registry, UPR-Definitionen oder Ergebnisse der Ressourcensuche zu finden.
<b>user</b>	Benutzerdefinierte Logging-Ausgabe, die mit der Option userlog übergeben wurde.
<b>warning</b>	Protokolliert alle Warnungen, d.h. Fehlerbedingungen, die ignoriert oder intern behandelt werden können. Ist warning=2, werden auch Meldungen von Funktionen protokolliert, die zwar keine Exception auslösen, aber einen Meldungstext zur Abfrage mit <code>TET_get_errmsg()</code> liefern, sowie die Ursache für alle gescheiterten Versuche, eine Datei zu öffnen (Suche einer Datei via searchpath).

## 10.3 Dokumentfunktionen

---

**C++ Java** `int open_document(String filename, String optlist)`

**Perl PHP** `long open_document(string filename, string optlist)`

**C** `int TET_open_document(TET *tet, const char *filename, int len, const char *optlist)`

---

Öffnet ein auf der Festplatte liegendes oder virtuelles PDF-Dokument zur Extraktion von Inhalten.

**filename** Vollständiger Pfadname der zu verarbeitenden PDF-Datei. Die Datei wird mit Hilfe der *SearchPath*-Ressource gesucht.

In nicht Unicode-fähigen Sprachbindungen wird der Dateiname gemäß der Option *filenamehandling* nach UTF-8 konvertiert (sofern nicht *filenamehandling=unicode* oder der übergebene Dateiname mit einem UTF-8-BOM beginnt). Ist *len* von 0 verschieden (nur C-Sprachbindung), wird der Dateiname ungeachtet der Option *filenamehandling* von UTF-16 nach UTF-8 konvertiert. Wenn der Dateiname nicht konvertiert werden kann oder kein gültiges UTF-8 oder UTF-16 darstellt, wird ein Fehler zurückgegeben.

Unter Windows können UNC-Pfade oder Netzwerkfreigaben verwendet werden, sofern Sie die dazu erforderlichen Berechtigungen haben (was bei ASP nicht unbedingt der Fall ist).

**len** (Nur C-Sprachbindung) Länge von *filename* (in Bytes) für UTF-16-Strings. Ist *len=0*, muss ein null-terminierter String übergeben werden.

**optlist** Optionsliste mit globalen Optionen gemäß Tabelle 10.8. Folgende Optionen können verwendet werden:

*allowjpeg2000, checkglyphlists, decompose, encodinghint, engines, fold, glyphmapping, lineseparator, normalize, inmemory, paraseparator, password, repair, requiredmode, shrug, spotcolor, tetml, usehostfonts, wordseparator*

**Rückgabe** -1 im Fehlerfall, sonst ein Dokument-Handle. Kann beispielsweise das Eingabedokument oder das TETML-Ausgabedokument nicht geöffnet werden, handelt es sich um einen Fehler. Gibt die Funktion einen Fehler (-1) zurück, sollten Sie genauere Informationen zur Fehlerursache mit *TET\_get\_errmsg()* abfragen.

**Details** Innerhalb eines TET-Objekts können beliebig viele Dokumente gleichzeitig geöffnet sein. Ein TET-Objekt darf jedoch nicht parallel in mehreren Threads verwendet werden, ohne dass ein Sperrmechanismus den Zugriff synchronisiert.

Verschlüsselung: Ist das Dokument verschlüsselt, muss mit der Option *password* das Benutzerkennwort übergeben werden, sofern die Berechtigungseinstellungen eine Textextraktion erlauben. Erlauben die Berechtigungseinstellungen keine Textextraktion, muss das Master-Kennwort des Dokuments übergeben werden. Wurde die Option *requiredmode* angegeben, lassen sich Dokumente auch ohne das richtige Kennwort öffnen, die möglichen Operationen sind jedoch eingeschränkt. Mit der Option *shrug* kann unter bestimmten Bedingungen die Extraktion von Inhalten aus geschützten Dokumenten aktiviert werden (siehe Abschnitt 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67).

Unterstützte Dateisysteme auf i5/iSeries-Systemen: TET wurde nur auf PC-Dateisystemen getestet. Deswegen sollten Ein- und Ausgabedateien im IFS (Integrated File System) als PC-Dateien vorliegen. Das Dateisystem *QSYS.lib* wurde für Eingabedateien nicht

getestet und wird nicht unterstützt. *QSYS.lib*-Dateien werden hauptsächlich für blockorientierte oder Datenbank-Objekte verwendet. Beim Einsatz von TET mit *QSYS.lib*-Objekten können deshalb unvorhersehbare Effekte auftreten. In TET basiert das Lesen und Schreiben von Dateien nicht auf Datensätzen, sondern auf Datenströmen.

Tabelle 10.8 Dokumentoptionen für `TET_open_document()` und `TET_open_document_callback()`

Option	Beschreibung
<b>accept-dynamicxfa</b>	(Boolean) Bei <code>true</code> lassen sich dynamische XFA-Formulare öffnen. Das Abfragen von pCOS-Pfaden ist die einzige sinnvolle Aktion. Der Aufruf von <code>TET_open_page()</code> schlägt fehl, da kein sinnvoller Text oder Bilder extrahiert werden können. Standardwert: <code>false</code>
<b>allowjpeg-2000</b>	(Boolean) Bei <code>true</code> wird JPEG 2000 (*.jpx, *.jpf oder *.j2k) als Ausgabeformat für <code>TET_write_image_file()</code> und <code>TET_get_image_data()</code> erlaubt. Ansonsten wird statt JPEG 2000 TIFF ausgegeben, was zu größeren Bilddateien führen kann. Standardwert: <code>true</code>
<b>checkglyph-lists</b>	(Boolean) Bei <code>true</code> wendet TET vor Beginn der Textextraktion alle internen Regeln zum Glyphen-Mapping mit <code>condition=allfonts</code> an. Andernfalls werden die globalen Regeln nicht angewendet. Diese Option verlangsamt die Verarbeitung, ist aber für bestimmte Arten von TeX-Dokumenten mit Glyphnamen sinnvoll, die standardmäßig nicht nach Unicode konvertiert werden können. Standardwert: <code>false</code>
<b>decompose</b>	<p>(Schlüsselwort oder Optionsliste) Auf alle Zeichen mit einem entsprechenden Unicode-Dekompositionstag, die Teil der angegebenen Unicode-Menge sind, werden die Unicode-Dekompositionen angewendet. Diese Bedingungen werden im Namen und Wert der Unteroption übergeben. Mit Dekompositionen kann der Unterschied zwischen äquivalenten Unicode-Zeichen entweder erhalten bleiben oder entfernt werden (siehe Abschnitt 7.3, »Unicode-Nachbearbeitung«, Seite 112).</p> <p>Standardwert: siehe »Standard-Dekompositionen«, Seite 119. Wenn die Option <code>normalize</code> einen Wert ungleich <code>none</code> hat, werden alle Standard-Dekompositionen deaktiviert, d.h. mit der Option <code>normalize</code> wird der Standardwert auf <code>decompose=none</code> gesetzt. Benutzerdefinierte Dekomposition kann jedoch immer noch angewendet werden.</p> <p>Folgende Schlüsselwörter können anstelle einer Liste übergeben werden:</p> <p><b>none</b> Keine Dekomposition</p> <p><b>default</b> Es werden zuerst die Standard-Dekompositionen und anschließend die anderen angegebenen Dekompositionen angewendet (siehe »Standard-Dekompositionen«, Seite 119).</p> <p>Für Dekompositionen werden folgende Unteroptionen unterstützt:</p> <p><b>canonical, circle, compat, final, font, fraction, initial, isolated, medial, narrow, nobreak, small, square, sub, super, vertical, wide</b></p> <p>Jede dieser Unteroptionen akzeptiert einen String oder ein Schlüsselwort, die die Domäne der Dekomposition angeben, also die Menge von Unicode-Zeichen, auf die die Dekomposition angewendet wird. Ein String gibt eine Unicode-Menge für die Domäne an. Damit lässt sich die Dekomposition auf eine Untermenge der Zeichen mit dem jeweiligen Dekompositionstag beschränken. Zeichen außerhalb der Domäne werden nicht verändert.</p> <p>Alternativ zu einem String für eine Unicode-Menge können die folgenden Schlüsselwörter übergeben werden:</p> <p><b>_all</b> Die Menge aller Unicode-Zeichen, d.h. die Dekomposition wird auf alle Zeichen mit dem angegebenen Dekompositionstag angewendet.</p> <p><b>_none</b> Leere Menge, d.h. die Dekomposition wird nicht angewendet.</p>
<b>encodinghint</b>	(String <sup>1</sup> ) Name eines Encodings zur Festlegung von Unicode-Zuordnungen für Glyphnamen, die nur mit einer vordefinierten internen Regel zum Glyphen-Mapping und nicht mit Standardregeln zugeordnet werden können. Mit dem Schlüsselwort <code>none</code> lassen sich alle vordefinierten Regeln deaktivieren. Standardwert: <code>w1nansi</code>

Tabelle 10.8 Dokumentoptionen für `TET_open_document()` und `TET_open_document_callback()`

Option	Beschreibung
<b>engines</b>	(Optionsliste) Aktiviert oder deaktiviert TET-Engines für die Seitenanalyse. Deaktivierte Engines liefern keinerlei Informationen. Das Deaktivieren von nicht erforderlichen Engines verbessert die Leistung (Standardwert: alle Engines sind aktiv): <b>image</b> (Boolean) Aktiviert die Bildextraktion. <b>text</b> (Boolean) Aktiviert die Textextraktion. <b>textcolor</b> (Boolean) Aktiviert die Engine für Textfarbe. <b>vector</b> (Boolean) Aktiviert die Engine für Vektorgrafik, relevant für das Clipping sowie für verbesserte Tabellenerkennung.
<b>fold</b>	(Schlüsselwort oder Liste mit Listen; das erste Element jeder inneren Liste ist eine Unicode-Menge oder ein Schlüsselwort, das zweite Element ist ein Unichar oder ein Schlüsselwort) Wendet ein Post-Folding (Äquivalenz-Zuordnung) auf alle Zeichen in einer als Unicode-Menge oder Schlüsselwort angegebenen Folding-Domäne an. Die Foldings werden auf den gesamten Text angewendet, außer auf Separator-Zeichen, die mit den Optionen <code>lineseparator</code> , <code>wordseparator</code> oder <code>paraseparator</code> hinzugefügt wurden (siehe Abschnitt 7.3, »Unicode-Nachbearbeitung«, Seite 112). Standardwert: siehe Tabelle 7.3, Seite 115. Folgende Schlüsselwörter können anstelle einer Liste übergeben werden: <b>none</b> Kein Folding Folgende Schlüsselwörter können anstelle einer untergeordneten Liste übergeben werden: <b>default</b> Standard-Foldings werden angewendet. Das erste Element jeder Liste gibt eine Folding-Domäne an, d.h. die Menge von Unicode-Zeichen, auf die das Folding angewendet wird. Ein String gibt eine Unicode-Menge für die Domäne an. Ist ein Zeichen in mehreren in der Option <code>fold</code> angegebenen Mengen enthalten, hat die erste passende Definition Vorrang vor allen anderen. Um unerwartete Ergebnisse zu vermeiden, raten wir von überlappenden Mengen ab. Alternativ zur Angabe der Domäne als Unicode-Menge können die folgenden Schlüsselwörter übergeben werden: <b>_dehyphenation</b> Folding wird auf Trennzeichen angewendet, die innerhalb von getrennten Wörtern am Zeilenende auftreten. Diese Zeichen werden im Feld <code>attributes</code> markiert, das von <code>TET_get_char_info()</code> und dem Attribut <code>Glyph/@dehyphenation</code> in TETML zurückgegeben wird. Das zweite Element in jeder Liste enthält das Zielzeichen oder die Aktion für das Folding. Es wird mit einer der folgenden Varianten angegeben: <b>_tetpua</b> Foldings werden auf die TET-PUA-Werte angewendet, die nicht zuzuordnenden Glyphen zugewiesen sind. Diese Zeichen werden im Member <code>unknown</code> markiert, das von <code>TET_get_char_info()</code> und dem Attribut <code>Glyph/@unknown</code> in TETML zurückgegeben wird. <b>(Unichar)</b> Ersetzt alle Zeichen in der Domäne durch das angegebene Unicode-Zeichen. <b>preserve</b> Die Zeichen in der Domäne werden nicht verändert. <b>remove</b> Alle Zeichen in der Domäne werden entfernt. <b>shift</b> Verschiebt alle Zeichen in der Domäne um den angegebenen Wert (welcher negativ sein kann). <b>unknownchar</b> Ersetzt alle Zeichen in der Domäne durch das in der Option <code>unknownchar</code> angegebene Zeichen oder wendet die in der Option <code>unknownchar</code> angegebene Aktion an.
<b>glyphmapping</b>	(Liste von Optionslisten) Liste von Optionslisten, wobei jede eine Regel zum Glyphen-Mapping für eine oder mehrere Font/Encoding-Kombinationen beschreibt, die sich mit Standardmethoden nicht zuverlässig abbilden lassen. Die Zuordnungen werden in umgekehrter Reihenfolge ihrer Definition abgearbeitet. Enthält die letzte Optionsliste den Fontnamen »*«, werden die vorangehenden Zuordnungen nicht mehr berücksichtigt. Jede Regel besteht aus einer Optionsliste gemäß Tabelle 10.9. Alle Glyphen-Mappings, die auf einen bestimmten Fontnamen passen, werden auf diesen Font angewendet (Standardwert: vordefinierte interne Glyphen-Mappings werden angewendet). Beachten Sie, dass Regeln zum Glyphen-Mapping auch als externe Ressource in der UPR-Datei angegeben werden können (siehe Abschnitt 5.2, »Ressourcenkonfiguration und Dateisuche«, Seite 70).

Tabelle 10.8 Dokumentoptionen für `TET_open_document()` und `TET_open_document_callback()`

Option	Beschreibung
<b>ignore-actualtext</b>	(Boolean) Bei true werden alle ActualText-Zuordnungen im Dokument ignoriert. Standardwert: false
<b>inmemory</b>	(Boolean; nur für <code>TET_open_document()</code> ) Bei true lädt TET die Datei vollständig in den Speicher und verarbeitet sie dort. Auf manchen Systemen (insbesondere z/OS) lässt sich die Leistung damit erheblich steigern, es ist jedoch mehr Speicher erforderlich. Bei false wird das Dokument stückweise von der Festplatte gelesen. Standardwert: false
<b>linseparator</b>	(Unichar; nur bei granularity=page) Zeichen, das zwischen Zeilen eingefügt werden soll <sup>2</sup> . Standardwert: U+000A
<b>normalize</b>	(Schlüsselwort) Normalisiert die Textausgabe auf eine der Unicode-Normalisierungsformen (Standardwert: none): <b>none</b> Keine Normalisierung. <b>nfc</b> Normalisierungsform C (NFC): kanonische Dekomposition gefolgt von kanonischer Komposition <b>nfd</b> Normalisierungsform D (NFD): kanonische Dekomposition <b>nfkc</b> Normalisierungsform KC (NFKC): kanonische Dekomposition gefolgt von kanonischer Komposition <b>nfkd</b> Normalisierungsform KD (NFKD): Kompatibilitätsdekomposition Da die Unicode-Normalisierungsformen kanonische und Kompatibilitätsdekomposition durchführen, müssen die Optionen decompose und normalize vorsichtig kombiniert werden. Ist die Option normalize ungleich none, wird die Dekomposition auf den Standardwert decompose=none gesetzt.
<b>paraseparator</b>	(Unichar; nur für granularity=page) Zeichen, das zwischen Absätzen eingefügt werden soll <sup>2</sup> . Standardwert: U+000A
<b>password</b>	(String) Benutzer-, Master oder Dateianlagen-Kennwort für verschlüsselte Dokumente. Erlauben die Berechtigungseinstellungen des Dokuments das Kopieren von Text, ist das Benutzerkennwort ausreichend, andernfalls muss das Master-Kennwort übergeben werden. Für die Abfrage des Verschlüsselungsstatus eines Dokuments sowie pCOS-Operationen ohne Kenntnis des Benutzer- oder Master-Kennworts siehe die pCOS-Pfadreferenz. Mit der Option shrug kann unter bestimmten Bedingungen die Extraktion von Inhalten aus geschützten Dokumenten aktiviert werden (siehe Abschnitt 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67).
<b>repair</b>	(Schlüsselwort) Legt fest, wie beschädigte PDF-Eingabedokumente behandelt werden. Die Reparatur eines Dokuments benötigt zwar mehr Zeit als das normale Parsen, ermöglicht aber die Verarbeitung bestimmter beschädigter PDF-Dokumente. Beachten Sie, dass sich manche Dokumente trotz Reparatur nicht verarbeiten lassen (Standardwert: auto): <b>force</b> Es wird in jedem Fall versucht, das Dokument zu reparieren. <b>auto</b> Es wird nur versucht, das Dokument zu reparieren, wenn beim Öffnen Probleme auftreten. <b>none</b> Es wird nicht versucht, das Dokument zu reparieren. Bei Problemen mit dem PDF-Dokument, schlägt die Funktion fehl.
<b>requiredmode</b>	(Schlüsselwort) Minimaler pCOS-Modus (minimum/restricted/full), der beim Öffnen akzeptiert wird. Der Funktionsaufruf scheitert, wenn der resultierende pCOS-Modus niedriger als der erforderliche Modus ist (siehe die pCOS-Pfadreferenz). Ist der Aufruf erfolgreich, so ist sichergestellt, dass der resultierende Modus mindestens dem in dieser Option festgelegten Modus entspricht. Er kann jedoch auch höher sein; so ergibt sich zum Beispiel aus requiredmode=minimum für ein unverschlüsseltes Dokument der Modus full. Standardwert: full
<b>shrug</b>	(Boolean) Bei true wird das Feature »shrug« für die Extraktion von Inhalten aus geschützten Dokumenten unter bestimmten Bedingungen aktiviert (siehe Kapitel 5.1, »Extrahieren von Inhalten aus geschützten PDF-Dokumenten«, Seite 67). Mit der Option shrug versichern Sie, dass Sie die Urheberrechte des Dokuments beachten. Standardwert: false

Tabelle 10.8 Dokumentoptionen für `TET_open_document()` und `TET_open_document_callback()`

Option	Beschreibung
<b>spotcolor</b>	(Schlüsselwort) Steuert die Verarbeitung von Bildern mit Schmuckfarben in <code>TET_write_image_file()</code> und <code>TET_get_image_data()</code> . Bilder mit einem Separation- oder DeviceN-Farbraum, d.h. Bilder mit einer oder mehreren benannten Prozess- oder Schmuckfarben werden wie folgt extrahiert (Standardwert: ignore):
<b>convert</b>	Ausgabe eines Graustufen- oder CMYK-Bildes, falls keine benutzerdefinierten Schmuckfarben verwendet werden. Andernfalls Konvertierung von Schmuckfarben in den zugehörigen Alternativfarbraum. Für manche Bilder ist die Konvertierung in den Alternativfarbraum nicht möglich. In diesem Fall verhält sich diese Methode wie <code>spotcolor=ignore</code> (bei einer einzelnen benutzerdefinierten Schmuckfarbe) bzw. wie <code>spotcolor=preserve</code> (bei zwei oder mehreren benutzerdefinierten Schmuckfarben).
<b>ignore</b>	Wie <code>convert</code> , außer, dass Bilder mit genau einer benutzerdefinierten Schmuckfarbe als Graustufen-Bilder extrahiert werden und die Schmuckfarbe verlorengeht.
<b>preserve</b>	(Erzwingt TIFF-Ausgabe) Ausgabe eines Graustufen- oder CMYK-Bildes mit einem oder mehreren zusätzlichen Schmuckfarbkkanälen, falls dies für benutzerdefinierte Schmuckfarbnamen erforderlich ist. TIFF-Bilder mit beibehaltenen Schmuckfarben in zusätzlichen Kanälen funktionieren nur in Adobe Photoshop und kompatiblen Programmen, jedoch nicht in allen einfachen TIFF-Viewern.
<b>tetml</b>	(Optionsliste) Aktiviert die TETML-Ausgabe; sie kann mit <code>TET_process_page()</code> seitenweise erzeugt werden. Die folgenden Unteroptionen werden unterstützt:
<b>elements</b>	(Optionsliste) Legt fest, ob bestimmte TETML-Elemente in der Ausgabe enthalten sind.
<b>annotations</b>	(Boolean) Ausgabe von <code>/TET/Document/Pages[]/Page/Annotations</code> , wenn das Dokument Anmerkungen enthält. Standardwert: true
<b>bookmarks</b>	(Boolean) Ausgabe von <code>/TET/Document/Bookmarks</code> , wenn das Dokument Lesezeichen enthält. Standardwert: true
<b>destinations</b>	(Boolean) Ausgabe von <code>/TET/Document/Destinations</code> , wenn das Dokument benannte Ziele enthält. Standardwert: true
<b>docinfo</b>	(Boolean) Ausgabe des Elements <code>/TET/Document/DocInfo</code> , wenn das Dokument Dokument-Infofelder enthält. Standardwert: true
<b>fields</b>	(Boolean) Ausgabe von <code>/TET/Document/Pages[]/Page/Fields</code> und <code>TET/Document/SignatureFields</code> , wenn das Dokument AcroForm-Felder oder digitale Signaturen enthält. Standardwert: true
<b>javascripts</b>	(Boolean) Ausgabe von <code>/TET/Document/JavaScripts</code> , wenn das Dokument JavaScript enthält. Standardwert: true
<b>metadata</b>	(Boolean) Ausgabe von <code>/TET/Document/Metadata</code> und/oder <code>/TET/Document/Images[]/Image/Metadata</code> , wenn das Dokument XMP-Metadaten auf Dokument- oder Bildebene enthält. Standardwert: true
<b>options</b>	(Boolean) Die Elemente <code>/TET/Document/Options</code> und <code>/TET/Document/Pages[]/Page/Options</code> . Standardwert: true
<b>encodingname</b>	(Schlüsselwort) Name, der in der XML-Encoding-Deklaration der Textdeklaration des generierten TETML verwendet werden soll. Die Ausgabe wird immer in UTF-8 erzeugt (Standardwert: UTF-8):
<b>_none</b>	Es wird keine Encoding-Deklaration erzeugt; die Ausgabe wird im UTF-8-Format erzeugt.
<b>UTF-8</b>	Die Deklaration <code>encoding="UTF-8"</code> wird erzeugt. Jeder andere Encoding-Name wird wörtlich in der Encoding-Deklaration verwendet. Der Client muss für einen geeigneten Encoding-Namen sorgen, sowie für die Konvertierung des generierten TETML (also UTF-8) in das angegebene Encoding, nachdem TET die TETML-Ausgabe beendet hat.
<b>filename</b>	(String) Name der TETML-Datei. Wird kein Dateiname übergeben, wird die Ausgabe im Arbeitsspeicher erzeugt und kann mit <code>TET_get_tetml()</code> abgefragt werden. Schlägt der Funktionsaufruf fehl (d.h. das PDF-Eingabedokument konnte nicht geöffnet werden), wird keine TETML-Ausgabe erzeugt.

Tabelle 10.8 Dokumentoptionen für `TET_open_document()` und `TET_open_document_callback()`

Option	Beschreibung
<b>unknown-char</b>	(Unichar oder Schlüsselwort) Zeichen oder Aktion, die auf TET-PUA-Zeichen für nicht zuzuordnende Glyphen angewendet werden soll (siehe »Glyphen ohne Unicode-Werte und die TET-PUA«, Seite 123). Folgende Schlüsselwörter werden unterstützt (Standardwert: Unicode-Ersatzzeichen U+FFFD): <b>remove</b> Glyphen ohne Unicode-Wert werden entfernt. Der Wert U+0000 entspricht <code>remove</code> . <b>preserve</b> (Glyphen ohne Unicode-Wert werden durch einen TET-PUA-Wert dargestellt.
<b>usehostfonts</b>	(Boolean) Bei <code>true</code> werden Daten für nicht eingebettete Fonts, die für die Bestimmung der Unicode-Mappings erforderlich sind, auf dem Host-System von OS X oder Windows gesucht. Standardwert: <code>true</code>
<b>wordseparator</b>	(Unichar; nur bei <code>granularity=line</code> und <code>page</code> ) Zeichen, das zwischen Wörtern eingefügt werden soll <sup>2</sup> . Standardwert: U+0020

1. Siehe Fußnote 1 in Tabelle 10.9

2. Verwenden Sie zur Deaktivierung des Separators U+0000.

Tabelle 10.9 Unteroptionen für die Option `glyphmapping` von `TET_open_document()` und `TET_open_document_callback()`

Option	Beschreibung
<b><code>codelist</code></b>	(String) Name einer auf den Font anzuwendenden Codelist-Ressource. Sie hat höhere Priorität als die eingebettete ToUnicode-CMap oder der Encoding-Eintrag.
<b><code>fold</code></b>	Wendet Pre-Folding (Äquivalenz-Mapping) auf alle Zeichen in einer als Unicode-Satz angegebenen Folding-Domäne an; siehe die Beschreibung für die Option <code>fold</code> in Tabelle 10.8. Die Schlüsselwörter <code>remove</code> , <code>preserve</code> und <code>unknownchar</code> können nicht verwendet werden. Fontspezifische Foldings mit dem Schlüsselwort <code>shift</code> können zur Korrektur systematischer Fehler in einer ToUnicode-CMap eines Fonts verwendet werden.
<b><code>fontname</code></b>	(Name-String) Präfix oder vollständiger Name des oder der Font(s), auf den oder die die ausgewählten Regeln angewendet werden. Wenn ein Untergruppen-Präfix übergeben wurde, wird nur die angegebene Untergruppe ausgewählt. Wenn kein Untergruppen-Präfix übergeben wurde, werden alle Fonts mit passendem Namen (ohne Untergruppen-Präfix) ausgewählt. Mit dem Wildcard-Zeichen »*« können mehrere ähnliche Fontnamen angegeben werden. Standardwert: *
<b><code>fonttype</code></b>	(Liste von Schlüsselwörtern) Das Glyphen-Mapping wird nur auf die angegebenen Fonttypen angewendet: * (für alle Fonttypen), Type1, MType1, TrueType, CIDFontType2, CIDFontType0, Type3. Standardwert: *
<b><code>forceencoding</code></b>	(Liste mit ein oder zwei Strings <sup>1</sup> , bei zwei Namen muss der erste <code>winansi</code> , <code>macroman</code> oder <code>Custom</code> sein) Fonts mit einem 8-Bit-Encoding: Ersetzt das erste Encoding durch die mit dem zweiten Namen festgelegte Encoding-Ressource. Wird nur ein Eintrag übergeben, werden die Encodings <code>MacRoman</code> , <code>WinAnsi</code> und <code>MacExpert</code> an allen betroffenen Stellen durch das übergebene Encoding ersetzt. Wenn diese Option auf einen Font passt, werden keine anderen Glyphen-Mappings auf diesen Font mehr angewendet. CID-Fonts: Als einziger Wert wird nur <code>unicode</code> unterstützt. CID-Werte werden dabei als Unicode-Werte interpretiert.
<b><code>forcetsymbol-encoding</code></b>	(Schlüsselwort oder String <sup>1</sup> ) Name eines Encodings, mit dem Unicode-Zuordnungen für eingebettete Pseudo-Symbolfonts im TrueType-Format bestimmt werden, bei denen es sich eigentlich um Textfonts handelt, oder eins der folgenden Schlüsselwörter (Standardwert: <code>none</code> ): <ul style="list-style-type: none"> <li><b><code>auto</code></b> Wenn das Builtin-Encoding des Fonts (siehe unten) mindestens ein Unicode-Zeichen im symbolischen Bereich U+FOOO-U+FOFF enthält, wird das in der Option <code>encodinghint</code> angegebene Encoding verwendet, um die Pseudo-Symbolzeichen den tatsächlichen Textzeichen zuzuordnen. Andernfalls wird <code>encodinghint</code> nicht verwendet und die Zeichen werden gemäß dem Schlüsselwort <code>builtin</code> zugeordnet.</li> <li><b><code>builtin</code></b> Verwendet das Builtin-Encoding des Fonts, das aus den Unicode-Zuordnungen der Glyphennamen in der Fonttabelle <code>post</code> resultiert.</li> <li><b><code>none</code></b> Es wird kein Encoding berücksichtigt. Die bekannten TrueType-Fonts <code>Wingdings*</code> und <code>Webdings*</code> werden immer als Symbolfonts behandelt.</li> </ul>
<b><code>globalglyphlist</code></b>	(Boolean) Bei <code>true</code> wird die angegebene Glyphliste bis zum Abschluss des TET-Objekts im Arbeitsspeicher gehalten, d.h., sie kann auf mehrere Dokumente angewendet werden. Standardwert: <code>false</code>
<b><code>glyphlist</code></b>	(String) Name einer anzuwendenden Glyphlisten-Ressource

Tabelle 10.9 Unteroptionen für die Option `glyphmapping` von `TET_open_document()` und `TET_open_document_callback()`

Option	Beschreibung
<b>glyphrule</b>	(Optionsliste) Zuordnungsregel für numerische Glyphnamen (zusätzlich zu den vordefinierten Regeln). Die Optionsliste muss folgende Unteroptionen enthalten:
<b>prefix</b>	(String; kann leer sein) Präfix der Glyphnamen, auf die die Regel angewendet wird. Das Wildcard-Zeichen »?»« kann verwendet werden. Es steht für genau ein Zeichen, außer 0-9.
<b>base</b>	(Schlüsselwort) Bestimmt die Interpretation der Glyphnamen:
<b>ascii</b>	Ein-Byte-Glyphnamen werden als zugehörige ASCII-Literalzeichen interpretiert (1 wird z.B. nach U+0031 konvertiert).
<b>auto</b>	Erkennt automatisch, ob Glyphnamen dezimale oder hexadezimale Werte darstellen. Ist das Ergebnis nicht eindeutig, wird von dezimalen Werten ausgegangen.
<b>dec</b>	Die Glyphnamen werden als Dezimaldarstellung eines Codes interpretiert.
<b>hex</b>	Die Glyphnamen werden als Hexadezimaldarstellung eines Codes interpretiert.
<b>encoding</b>	(String) Name einer Encoding-Ressource für diese Regel oder das Schlüsselwort none zur Deaktivierung der Regel.
<b>ignoreto-unicodemap</b>	(Boolean) Bei <code>true</code> wird eine ToUnicode-CMap für den Font ignoriert. Standardwert: <code>false</code>
<b>override</b>	(Boolean; nur sinnvoll zusammen mit der Option <code>glyphlist</code> oder <code>glyphrule</code> ) Bei <code>true</code> wird die Regel zum Glyphen-Mapping vor den Standard-Glyphen-Mappings (Builtin) angewendet (d.h. die neuen Zuordnungen haben Priorität vor den Builtin-Regeln); andernfalls wird die Regel nach den Builtin-Zuordnungen angewendet. Standardwert: <code>true</code>
<b>remove</b>	(Boolean) Bei <code>true</code> wird der gesamte Text, der die angegebenen Fontnamen und/oder Fonttypen verwendet, aus dem extrahierten Text entfernt.
<b>tounicode-cmap</b>	(String) Name einer ToUnicode-CMap-Ressource, die auf den Font angewendet werden soll; sie hat höhere Priorität als eine eingebettete ToUnicode-CMap oder ein Encoding-Eintrag.

1. Encoding-Name gemäß Abschnitt 10.1.4, »Encoding-Namen«, Seite 179

```

C++ int open_document_callback(void *opaque, size_t filesize,
                               size_t (*readproc)(void *opaque, void *buffer, size_t size),
                               int (*seekproc)(void *opaque, long offset),
                               wstring optlist)

C   int TET_open_document_callback(TET *tet, void *opaque, size_t filesize,
                                   size_t (*readproc)(void *opaque, void *buffer, size_t size),
                                   int (*seekproc)(void *opaque, long offset),
                                   const char *optlist)

```

Öffnet ein PDF-Dokument aus einer benutzerdefinierten Datenquelle und bereitet es zur Extraktion von Inhalten vor.

**opaque** Zeiger auf Benutzerdaten, die dem zu öffnenden PDF-Dokument zugeordnet sind. Er wird als erster Parameter an die Callback-Funktionen übergeben und kann beliebig genutzt werden. TET verwendet diesen Zeiger selbst nicht.

**filesize** Größe des PDF-Dokuments in Bytes.

**readproc** C-Callback-Funktion, die `size` Bytes in den als `buffer` angegebenen Speicher kopiert. Ist das Ende des Dokuments früher erreicht, kann die Funktion entsprechend weniger Bytes kopieren. Die Funktion muss die Anzahl der tatsächlich kopierten Bytes zurückgeben.

**seekproc** C-Callback-Funktion, die die aktuelle Leseposition im Dokument setzt. *offset* bezeichnet die Position ausgehend vom Dokumentanfang (0 bezeichnet dabei das erste Byte). Bei Erfolg muss die Funktion 0 zurückgeben, sonst -1.

**optlist** Optionsliste mit globalen Optionen gemäß Tabelle 10.8.

*Rückgabe* Siehe [TET\\_open\\_document\(\)](#).

*Details* Siehe [TET\\_open\\_document\(\)](#).

*Bindungen* Nur in den C- und C++-Sprachbindungen verfügbar.

---

**C++ Java** `void close_document(int doc)`

**Perl PHP** `close_document(long doc)`

**C** `void TET_close_document(TET *tet, int doc)`

---

Schließt ein Dokument-Handle und gibt alle Ressourcen frei, die sich auf das Dokument beziehen.

**doc** Gültiges Dokument-Handle, das mit [TET\\_open\\_document\\*\(\)](#) erzeugt wurde.

*Details* Beim Schließen eines Dokuments werden automatisch auch all seine offenen Seiten geschlossen. Alle offenen Dokumente und deren Seiten werden automatisch beim Aufruf von [TET\\_delete\(\)](#) geschlossen. Es zählt jedoch zu guter Programmier-Praxis, Dokumente explizit zu schließen, wenn sie nicht mehr gebraucht werden. Geschlossene Dokument-Handles dürfen in Funktionsaufrufen nicht mehr verwendet werden.

## 10.4 Seitenfunktionen

---

**C++ Java** *int open\_page(int doc, int pagenumber, String optlist)*  
**Perl PHP** *long open\_page(long pagenumber, string optlist)*  
**C** *int TET\_open\_page(TET \*tet, int doc, int pagenumber, const char \*optlist)*

---

Öffnet eine Seite für die Extraktion von Inhalten.

**doc** Gültiges Dokument-Handle, das mit *TET\_open\_document\**() erzeugt wurde.

**pagenumber** Physikalische Nummer der zu öffnenden Seite. Die erste Seite hat die Seitennummer 1. Die Gesamtzahl der Seiten kann mit *TET\_pcos\_get\_number()* und dem pCOS-Pfad *length:pages* abgefragt werden.

**optlist** Optionsliste mit Seitenoptionen gemäß Tabelle 10.10. Folgende Optionen können verwendet werden:

*clippingarea, contentanalysis, docstyle, emptycheck, excludebox, fontsize, granularity, ignoreartifacts, ignoreinvisibletext, imageanalysis, includebox, layers, layoutanalysis, layouteffort, structureanalysis, topdown, vectoranalysis.*

**Rückgabe** Seiten-Handle oder -1 im Fehlerfall. Gibt die Funktion einen Fehler (-1) zurück, sollten Sie genauere Informationen zur Fehlerursache mit *TET\_get\_errmsg()* abfragen.

**Details** Innerhalb eines einzelnen Dokuments können beliebig viele Seiten gleichzeitig geöffnet sein. Eine Seite kann dabei mehrfach mit verschiedenen Optionen geöffnet werden. Die Optionen sind aber während der Verarbeitung der Seite nicht mehr änderbar.

Ebenendefinitionen (optionale Inhaltsgruppen): die Inhalte aller sichtbaren Ebenen auf der Seite werden extrahiert. Dieses Verhalten kann mit der Option *layers* geändert werden.

Tabelle 10.10 Seitenoptionen für *TET\_open\_page()* und *TET\_process\_page()*

Option	Beschreibung
<b>clippingarea</b>	(Schlüsselwort, wird bei Angabe von <i>includebox</i> ignoriert) Bestimmt den Bereich, aus dem Text und Bilder extrahiert werden (Standardwert: <i>cropbox</i> ): <b>mediabox</b> <i>MediaBox</i> wird verwendet (immer vorhanden) <b>cropbox</b> <i>CropBox</i> (sichtbarer Bereich in Acrobat) wird verwendet, falls vorhanden, sonst <i>MediaBox</i> <b>bleedbox</b> <i>BleedBox</i> wird verwendet, falls vorhanden, sonst <i>CropBox</i> <b>trimbox</b> <i>TrimBox</i> wird verwendet, falls vorhanden, sonst <i>CropBox</i> <b>artbox</b> <i>ArtBox</i> wird verwendet, falls vorhanden, sonst <i>CropBox</i> <b>unlimited</b> Unabhängig von seiner Position wird der gesamte Text berücksichtigt
<b>content-analysis</b>	(Optionsliste; nicht bei <i>granularity=glyph</i> ) Liste von Unteroptionen gemäß Tabelle 10.11 zur Steuerung von inhaltlicher Analyse und Textverarbeitung.

Tabelle 10.10 Seitenoptionen für `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>docstyle</b>	<p>(Schlüsselwort) Hinweis für die Layout-Erkennung, verschiedene Parameter auszuwählen. Diese Parameter optimieren die Layout-Erkennung, wenn das Dokument zu einer der unten aufgeführten Klassen gehört. Gehört das Dokument zu einer der Klassen, können die Ergebnisse der Layout-Erkennung durch Angabe eines geeigneten Werts für diese Option erheblich verbessert werden. Diese Option aktiviert die erweiterte Layout-Erkennung (Standardwert: none):</p> <ul style="list-style-type: none"> <li><b>book</b> Typisches Buch</li> <li><b>business</b> Arbeitsdokumente</li> <li><b>cad</b> Typischerweise stark fragmentierte technische oder architektonische Zeichnungen</li> <li><b>fancy</b> Seiten mit komplexem Layout</li> <li><b>forms</b> Strukturierte Formulare</li> <li><b>generic</b> Allgemeinste Dokumentklasse ohne weitere Besonderheiten</li> <li><b>magazines</b> Zeitschriftenartikel</li> <li><b>none</b> Kein bestimmter Dokumentstil ist bekannt, die erweiterte Layout-Erkennung wird deaktiviert.</li> <li><b>native</b> Deaktiviert die Layout-Erkennung und gibt die Inhalte in der ursprünglichen Reihenfolge der Seiteninhalte zurück. Dies kann bei Layouts wie Formularen nützlich sein, in denen Text überall auf der Seite platziert ist und die zeilenorientierte Textabfrage der Spalten-Erkennung vorgezogen wird.</li> <li><b>papers</b> Zeitung</li> <li><b>science</b> Wissenschaftlicher Artikel</li> <li><b>searchengine</b> Bei der Anwendung handelt es sich um einen Suchmaschinen-Indexer oder eine ähnliche Anwendung, bei der es auf die schnellstmögliche Abfrage einer Wortliste für die Seite ankommt. Erkennung der Tabellen- und Seitenstruktur ist deaktiviert.</li> <li><b>spacegrid</b> Listenartiger Report (oft auf Mainframe-Systemen generiert), bei dem das visuelle Layout mit Leerzeichen erzeugt wird. Da viele Heuristiken wie Schatten-Erkennung und fortgeschrittene Wortgrenzen-Erkennung für diese Klasse von Dokumenten nicht erforderlich sind, kann die Textextraktion mit dieser Option beschleunigt werden.</li> </ul>
<b>emptycheck</b>	<p>(Boolean) Bei true wird die normale Textextraktion deaktiviert. Stattdessen wird die in der Option <code>includebox</code> angegebene Box verwendet, um zu überprüfen, ob die Box Text, Bilder oder Vektorgrafik enthält (nur eine einzige <code>includebox</code> wird unterstützt). Wenn die Option <code>includebox</code> nicht übergeben wird, wird der gesamte Clipping-Bereich überprüft. Damit können leere Seiten identifiziert werden. Folgende Optionen werden ignoriert: <code>granularity</code>, <code>engines</code>, <code>fontsize-range</code>. Clipping-Operatoren werden ignoriert.</p> <p>Das Ergebnis der Überprüfung kann mit <code>TET_get_text()</code> abgerufen werden, das statt der Seiteninhalte einen der Strings <code>empty</code> oder <code>notempty</code> zurückgibt. Standardwert: <code>false</code></p>
<b>excludebox</b>	<p>(Liste von Rechtecken) Schließt den durch die übergebenen Rechtecke definierten Bereich von der Text- und Bildextraktion aus. Standardwert: leer</p>
<b>fontsize-range</b>	<p>(Liste mit zwei Floats) Zwei Zahlen geben die minimale und maximale Fontgröße des Textes an. Text mit einer Größe außerhalb dieses Bereichs wird ignoriert. Die Maximalgröße kann mit dem Schlüsselwort <code>unlimited</code> übergeben werden, das keine obere Grenze vorsieht. Standardwert: <code>{ 0 unlimited }</code></p>

Tabelle 10.10 Seitenoptionen für `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>granularity</b>	(Schlüsselwort) Granularität der von <code>TET_get_text()</code> zurückgegebenen Textfragmente; in allen Modi außer <code>glyph</code> wird der Wordfinder aktiviert. Für weitere Informationen siehe »Granularität des Textes«, Seite 99 (Standardwert: <code>word</code> ).
<b>glyph</b>	Ein Fragment enthält eine einzelne Glyphe, zu der aber mehrere Zeichen gehören können (z.B. für Ligaturen).
<b>word</b>	Ein Fragment enthält ein Wort, das vom Wordfinder ermittelt wurde.
<b>line</b>	Ein Fragment enthält eine Textzeile, oder die beste Annäherung einer solchen. Zwischen aufeinanderfolgenden Wörtern werden Wortseparatoren eingefügt.
<b>page</b>	Ein Fragment enthält den Inhalt einer einzelnen Seite. Wort-, Zeilen- und Absatzseparatoren werden entsprechend eingefügt.
<b>ignore-artifacts</b>	(Boolean; nur relevant für Tagged PDF) Ignoriert Text und Bilder, die als Artefakte ausgezeichnet sind. Damit lassen sich irrelevante Seiteninhalte überspringen, die als Artefakte ausgezeichnet sind. Standardwert: <code>false</code>
<b>ignore-invisibletext</b>	(Boolean) Bei <code>true</code> wird Text mit <code>Textrendering = 3</code> (unsichtbar) ignoriert. Standardwert: <code>false</code> (da unsichtbarer Text meist in Bild+Text-PDFs verwendet wird, die gescannte Seiten und den zugehörigen OCR-Text enthalten)
<b>image-analysis</b>	(Optionsliste) Liste von Unteroptionen gemäß Tabelle 10.13 zur Steuerung weiterführender Bildverarbeitung.
<b>includebox</b>	(Liste von Rechtecken) Beschränkt die Text- und Bildextraktion auf den durch die übergebenen Rechtecke definierten Bereich. Standardwert: der vollständige Clipping-Bereich
<b>layers</b>	(Schlüsselwort) Verarbeitung der Seiteninhalte innerhalb von Ebenen (auch als optionaler Inhalt bezeichnet). Unterstützte Schlüsselwörter (Standardwert: <code>visible</code> ):
<b>all</b>	Extrahiert alle Seiteninhalte unabhängig von Ebenen. Der Text kann durcheinander sein und das Zusammensetzen von Bildern kann misslingen, wenn die Inhalte mehrerer Ebenen sich auf der Seite überlappen.
<b>invisible</b>	Extrahiert Inhalte aller standardmäßig unsichtbaren Ebenen und ignoriert alle anderen Ebenen.
<b>visible</b>	Extrahiert Inhalte aller standardmäßig sichtbaren Ebenen und ignoriert alle anderen Ebenen.
<b>layout-analysis</b>	(Optionsliste; nicht bei <code>granularity=glyph</code> ) Liste von Unteroptionen gemäß Tabelle 10.12 zur Steuerung von Funktionen zur Layout-Erkennung.
<b>layouteffort</b>	(Schlüsselwort) Steuert das Verhältnis zwischen Qualität und Aufwand bei der Layout-Erkennung. Die Layout-Erkennung kann durch erhöhten Aufwand verbessert werden, was die Geschwindigkeit der Anwendung jedoch beeinträchtigt. Die Layout-Erkennung kann mit den Schlüsselwörtern <code>none</code> , <code>low</code> , <code>medium</code> , <code>high</code> und <code>extra</code> gesteuert werden. Standardwert: <code>low</code>
<b>layouthint</b>	(Optionsliste) Informiert die Layout-Erkennung über das Vorhandensein bestimmter Seitenlayout-Elemente:
<b>subsummary</b>	(Schlüsselwort) Informiert die Layout-Erkennung über das Vorhandensein und möglicherweise auch die Position bestimmter Gruppierungen (Subsummaries, Marginalien). Unterstützte Schlüsselwörter (Standardwert: <code>none</code> ):
<b>auto</b>	Keine Marginalien-Erkennung
<b>left</b>	Versucht, Marginalien links auf der Seite zu erkennen.
<b>none</b>	Versucht, Marginalien automatisch zu erkennen.
<b>right</b>	Versucht, Marginalien rechts auf der Seite zu erkennen.
<b>header</b>	(Boolean) Bei <code>true</code> wird versucht, Kopfzeilen auf der Seite zu erkennen (Standardwert: <code>false</code> ).
<b>footer</b>	(Boolean) Bei <code>true</code> wird versucht, Fußzeilen auf der Seite zu erkennen (Standardwert: <code>false</code> ).

Tabelle 10.10 Seitenoptionen für `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>maxvector-count</b>	(Float) Maximale Anzahl an Vektorobjekten, die von der Engine für Vektorgrafik berücksichtigt werden sollen. Standardwert: 500
<b>minvectorsize</b>	(Float) Mindestgröße eines Vektorobjekts, das von der Engine für Vektorgrafik berücksichtigt werden soll. Die Größe einer Vektorobjekts ist die Länge der Diagonale der Boundingbox in Punkt. Standardwert: 5
<b>skipengines</b>	Veraltet; verwenden Sie stattdessen die Option engines von <code>TET_open_document()</code>
<b>structure-analysis</b>	(Optionsliste; nicht bei granularity=glyph) Liste von Unteroptionen gemäß Tabelle 10.14 zur Steuerung der Analyse der Seitenstruktur.
<b>topdown</b>	(Optionsliste) Legt ein Koordinatensystem fest mit dem Ursprung in der oberen linken Ecke der sichtbaren Seite, wobei die y-Koordinaten nach unten hin größer werden; andernfalls wird das Standardkoordinatensystem mit dem Ursprung in der unteren linken Ecke verwendet. Mit der Aktivierung des Top-Down-Koordinatensystems wird das selbe Koordinatensystem aktiviert wie in Acrobat. Unterstützte Unteroptionen: <b>input</b> (Boolean) Bei true werden die Topdown-Koordinaten für die folgenden Elemente aktiviert (Standardwert: false): Seitenoptionen includebox, excludebox <b>output</b> (Boolean) Bei true werden die Topdown-Koordinaten für die folgenden Elemente aktiviert (Standardwert: false): TET_char_info: y, alpha, beta TET_image_info: y, alpha, beta TETML: Destination/@bottom, Destination/@top, Box/@lly, Box/@ury, Box/@uly, Box/@lry, Cell/@ury, Cell/@uly, Cell/@lry, Glyph/@y, Glyph/@alpha, Glyph/@beta, PlacedImage/@y, PlacedImage/@alpha, PlacedImage/@beta, Table/@lly, Table/uly, Table/@ury, Table/lry.
<b>vector-analysis</b>	(Optionsliste, nicht für granularity=glyph) Unteroptionen gemäß Tabelle 10.15 zur Steuerung der Analyse von Vektorgrafiken für Tabellen- und Layout-Erkennung. Wenn diese Option vorhanden ist, werden Vektorgrafik bei der Tabellen- und Layout-Erkennung berücksichtigt.

Tabelle 10.11 Unteroptionen für die Option contentanalysis von `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b> bidi </b>	<p>(Schlüsselwort; wird bei <code>granularity=glyph</code> ignoriert; nur wirksam, wenn linksläufige Zeichen auf der Seite vorhanden sind) Steuert den inversen Bidi-Algorithmus, der linksläufigen und rechtsläufigen Text in einem Textfragment umsortiert (Standardwert: <code>logical</code>):</p> <p><b>visual</b> linksläufige und rechtsläufige Zeichen bleiben in einem Textfragment in visueller Reihenfolge erhalten, d.h., der inverse Bidi-Algorithmus wird nicht angewendet</p> <p><b>logical</b> Inverser Bidi-Algorithmus wird angewendet, um Zeichen in einem Textfragment in logische Reihenfolge zu bringen.</p>
<b> bidilevel </b>	<p>(Schlüsselwort) Bestimmt die dominante Schreibrichtung des Textes auf der Seite für den inversen Bidi-Algorithmus (Standardwert: <code>auto</code>):</p> <p><b>auto</b> Bestimmt die dominante Schreibrichtung des Textes heuristisch basierend auf dem Inhalt.</p> <p><b>ltr</b> Geht von rechtsläufigem Text als dominanter Schreibrichtung aus (z.B. Dokumente in Lateinischer Schrift)</p> <p><b>rtl</b> Geht von linksläufigem Text als dominanter Schreibrichtung aus (z.B. Dokumente in Hebräischer oder Arabischer Schrift)</p>
<b> dehyphenate </b>	<p>(Boolean) Bei <code>true</code> werden getrennte Wörter identifiziert und die sie umgebenden Textfragmente zusammengefasst. Die Trennzeichen selbst werden gemäß der Option <code>keeplyphens</code> verarbeitet. Standardwert: <code>true</code></p>
<b> dropcapsize </b>	<p>(Float) Kleinste Größe, ab der große Glyphen als mehrzeilige Anfangsbuchstaben erkannt werden. Mehrzeilige Anfangsbuchstaben am Beginn einer Zone werden auch Drop Caps genannt. Sie werden mit der restlichen Zone zusammengeführt und bilden einen Teil des ersten Worts in der Zone. Standardwert: 35</p>
<b> dropcapratio </b>	<p>(Float) Mindestgröße des Fonts eines mehrzeiligen Anfangsbuchstabens im Verhältnis zum umgebenden Text. Große Zeichen werden als mehrzeilige Anfangsbuchstaben erkannt, wenn sie größer sind als <code>dropcapsize</code> und der Fontgrößen-Quotient größer ist als <code>dropcapratio</code>. Der Wert gibt also die Anzahl der Textzeilen an, die der Anfangsbuchstabe umfasst. Standardwert: 4 (dreizeilige Anfangsbuchstaben sind zwar weit verbreitet, aber zusätzlicher Zeilenabstand muss berücksichtigt werden)</p>
<b> ideographic </b>	<p>(Schlüsselwort; veraltet) In TET 4 wurde empfohlen, diese Option auf <code>keep</code> zu setzen, um das Standardverhalten von <code>split</code> zu vermeiden. Ab TET 5 werden ideografische Zeichen bei <code>granularity=word</code> nicht mehr als Wortgrenzen betrachtet, daher ist diese Option nicht mehr erforderlich.</p>
<b> include-boxorder </b>	<p>(Integer) Wenn mehrere Include-Rechtecke übergeben wurden (siehe Option <code>includebox</code>), steuert diese Option, wie sich die Reihenfolge der Rechtecke auf den Wordfinder auswirkt (Standardwert: 0):</p> <ul style="list-style-type: none"> <li>0 Ignoriert die Reihenfolge der Include-Rechtecke bei der Analyse der Seiteninhalte. Das Ergebnis ist das gleiche, als wenn der gesamte Text außerhalb der Include-Rechtecke gelöscht worden wäre. Dies ist zur Beseitigung von unerwünschtem Text (z. B. Kopf- und Fußzeilen) nützlich, wobei der Wordfinder in keiner Weise betroffen ist.</li> <li>1 Berücksichtigt die Reihenfolge der Include-Rechtecke beim Erzeugen von Wörtern und Zonen, nicht jedoch bei der Sortierung der Zonen. Ein Wort gehört nie zu mehr als einem Rechteck. Die resultierenden Zonen werden in logischer Reihenfolge sortiert. Bei überlappenden Rechtecken gehört der Text zum ersten in der Liste auftauchenden Rechteck. Ansonsten spielt die Reihenfolge der Include-Rechtecke in der Optionsliste keine Rolle. Diese Einstellung ist nützlich, wenn Text aus Formularen oder Tabellen extrahiert werden soll oder wenn sich Include-Rechtecke bei komplexen Layouts überlappen.</li> <li>2 Die Reihenfolge der Include-Rechtecke wird bei allen Operationen berücksichtigt. Die Inhalte jedes Include-Rechtecks werden unabhängig von anderen Rechtecken verarbeitet und der resultierende Text wird gemäß der Reihenfolge der Include-Rechtecke aneinander gehängt. Dies ist nützlich, wenn Text in einer bestimmten Reihenfolge aus Formularen oder in einer vordefinierten Reihenfolge aus Artikelspalten in einem Zeitschriften-Layout extrahiert werden soll. In diesen Fällen muss das Seitenlayout im Voraus bekannt sein, damit die Include-Rechtecke in der richtigen Reihenfolge angegeben werden können.</li> </ul>

Tabelle 10.11 Unteroptionen für die Option contentanalysis von `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>keeplyphen-glyphs</b>	(Boolean) Bei <code>true</code> und <code>dehyphenate=true</code> wird das Trennzeichen zwischen den enttrennten Wortteilen in der Liste der Glyphen belassen, die von <code>TET_get_char_info()</code> und dem Element <code>Glyph</code> in TETML zurückgegeben werden. Dies ist nützlich für Anwendungen, die detaillierte Informationen über die Position der Trennstriche benötigen, um z.B. Text auf der Seite exakt auszutauschen. Dies unterscheidet sich von <code>fold={_dehyphenation_remove}</code> , das nur Trennzeichen aus dem logischen Text entfernt, der von <code>TET_get_text()</code> zurückgegeben wurde, aber keine Auswirkungen auf Glyphen hat. Standardwert: <code>false</code>
<b>linespacing</b>	(Schlüsselwort) Gibt den typischen vertikalen Abstand zwischen Textzeilen innerhalb eines Absatzes an: <code>small</code> , <code>medium</code> oder <code>large</code> (Standardwert: <code>medium</code> )
<b>maxwords</b>	Wenn die Anzahl der Wörter auf der Seite nicht größer als die angegebene Anzahl ist (das Schlüsselwort <code>unlimited</code> bedeutet, dass keine Begrenzung aktiv ist), werden die auf der Seite erkannten Zonen korrekt sortiert und zusammengeführt. Wenn die Anzahl der Wörter auf der Seite größer als die angegebene Anzahl ist, werden keine Zonen gebildet und Wörter werden in der Lesereihenfolge des Seiteninhalts extrahiert. Die Verarbeitung ist im letzten Fall zwar schneller, die Reihenfolge der abgefragten Wörter ist aber nicht unbedingt optimal. Bei großen Seiten mit vielen Wörtern, wie z.B. Zeitungen, empfehlen wir, diese Option auf <code>unlimited</code> zu setzen. Standardwert: <code>5000</code>
<b>merge</b>	(Integer) Steuert das Kombinieren von Streifen und Zonen (Standardwert: <code>2</code> ): <ul style="list-style-type: none"> <li><b>0</b> Nach dem Erzeugen von Streifen werden diese nicht zu Zonen zusammengefasst. Dies kann die Verarbeitungsgeschwindigkeit erheblich erhöhen, aber gleichzeitig auch die Ausgabequalität vermindern und das Erkennen von Schatteneffekten verhindern.</li> <li><b>1</b> Einfaches Zusammensetzen von Streifen zu Zonen: Streifen werden zu einer Zone zusammengefasst, wenn sie diese Zone überlappen und höchstens direkt angrenzende Streifen überlappen (damit soll die Überlappung von Zonen verhindert werden, die sich nicht aus Schatteneffekten ergeben).</li> <li><b>2</b> Aufwändige Kombination von Zonen für Text ohne klare Lesereihenfolge: Zusätzlich zu <code>merge=1</code> werden mehrere überlappende Zonen zu einer einzigen Zone zusammengefasst, sofern sich die Textinhalte beider Zonen nicht überlappen.</li> </ul>
<b>numeric-entities</b>	(Schlüsselwort) Steuert die Erkennung von Wortgrenzen für numerische Elemente wie Zahlen, Brüche und Zeitangaben (Standardwert: <code>keep</code> ): <ul style="list-style-type: none"> <li><b>split</b> Teilt das Element gemäß der Unteroption <code>punctuationbreaks</code>.</li> <li><b>keep</b> Erhält das Element als ganzes Wort.</li> </ul>
<b>shadow-detect</b>	(Boolean) Bei <code>true</code> werden redundante Stellen überlappender Textfragmente, die zur Erzeugung von Schatteneffekten oder künstlicher Fettschrift verwendet werden, erkannt und entfernt. Standardwert: <code>true</code>
<b>punctuation-breaks</b>	(Boolean; nur bei <code>granularity=word</code> ) Bei <code>true</code> werden Satzzeichen, die sich nahe bei einem Buchstaben befinden, als Wortgrenzen interpretiert. Andernfalls werden sie in das benachbarte Wort integriert. Diese Option ist zum Beispiel bei der Verarbeitung von URLs und Mailadressen nützlich. Standardwert: <code>true</code>
<b>superscript</b>	(Integer) Steuert die Erkennung von tief- und hochgestellten Zeichen (Standardwert: <code>2</code> ): <ul style="list-style-type: none"> <li><b>0</b> Keine Erkennung von tief- und hochgestellten Zeichen</li> <li><b>1</b> Einfache Erkennung von tief- und hochgestellten Zeichen</li> <li><b>2</b> Erweiterter Algorithmus für die Erkennung von tief- und hochgestellten Zeichen</li> </ul>
<b>useclasses</b>	(Boolean) Bei <code>true</code> wird die Unicode-Klassifizierung bei der Bestimmung von Wortgrenzen berücksichtigt. Standardwert: <code>true</code>
<b>usemetrics</b>	(Boolean) Bei <code>true</code> wird die Entfernung zwischen Glyphen mit der Breite der Leerraum-Glyphe verglichen, um die Wortgrenzen zu bestimmen. Standardwert: <code>true</code>

Tabelle 10.12 Unteroptionen für die Option `layoutanalysis` von `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>layout-astable</b>	(Boolean) Bei <code>true</code> behandelt die Layout-Erkennung die Zonen auf der Seite als eine oder mehrere Tabellen. Die minimale Anzahl an Spalten, ab der eine Sequenz als Tabelle angesehen wird, hängt vom Dokumentstil ab. Bei <code>false</code> wird die Supertabellen-Erkennung deaktiviert (Standardwert: <code>true</code> ).
<b>layout-columnhint</b>	(Schlüsselwort) Diese Option kann das Lesen und Erkennen von Zonen bei komplexen Layouts verbessern. Unterstützte Schlüsselwörter (Standardwert: <code>multicolumn</code> ): <b>multicolumn</b> Die Seite enthält mehrspaltigen Text; Zonen werden spaltenweise sortiert. <b>none</b> Kein Hinweis verfügbar; die Reihenfolge der Zonen wird von der Reihenfolge des Seiteninhalts bestimmt. <b>singlecolumn</b> Die Seite enthält einspaltigen Text; Zonen werden zeilenweise sortiert. Dieses Schlüsselwort sollte zusammen mit <code>layouteffort=low</code> verwendet werden.
<b>layoutdetect</b>	(Integer) Legt die Rekursionstiefe der Layout-Erkennung fest (Standardwert: 1): <b>0</b> Keine Layout-Erkennung. <b>1</b> Layout-Erkennung für die gesamte Seite. Dies ist für die meisten Dokumente ausreichend. <b>2</b> Layout-Erkennung für die Ergebnisse aus Level 1. Dies ist erforderlich für Layouts mit unterschiedlichen mehrspaltigen Unter-Layouts und Überschriften in verschiedenen Bereichen der Seite sowie für Tabellen mit mehreren Absätzen. <b>3</b> Layout-Erkennung für die Ergebnisse aus Level 2. Dies ist nur für sehr komplexe Layouts erforderlich.
<b>layout-rowhint</b>	(Optionsliste) Steuert die Layout-Verarbeitung von Zeilen. Unterstützte Optionen (Standardwert: <code>none</code> ): <b>full</b> Aktiviert die Layout-Verarbeitung von Zeilen. <b>none</b> Deaktiviert die Layout-Verarbeitung von Zeilen. <b>separation</b> (Schlüsselwort) Aktiviert die Layout-Verarbeitung von Zeilen, deaktiviert sie aber wieder, wenn die Layout-Erkennung eine Supertabelle vermutet. Folgende Unteroptionen können übergeben werden: <b>preservercolumns</b> Versucht, basierend auf der geometrischen Beziehung zwischen Zonen, vertikale Spalten zu erhalten. Dies ist empfehlenswert, wenn Zonen innerhalb von Spalten durch große Abstände voneinander getrennt sind (z.B. durch Bilder). <b>thick</b> Versucht, benachbarte Zonen zusammenzufassen und sie in der selben Layout-Zeile zu platzieren. Dies führt zu einer kleineren Anzahl von größeren Layout-Zeilen. Dies ist empfehlenswert bei komplexen Layouts wie Zeitschriften und Zeitungen, wo Absätze innerhalb von Spalten durch mehr als die Fontgröße voneinander getrennt sind, sowie bei Layouts mit mehreren mehrspaltigen Artikeln untereinander. <b>thin</b> Versucht, benachbarte Zonen zu trennen und sie in unterschiedlichen Layout-Zeilen zu platzieren. Dies führt zu einer größeren Anzahl von kleineren Layout-Zeilen. Beispiel: <code>layoutanalysis = {layoutrowhint={full separation=thick}}</code>
<b>mergetables</b>	(Integer) Einzeilige Tabellen werden während der Tabellen-Erkennung übersprungen und als reguläre Zonen behandelt. Wenn es sich bei zwei aufeinander folgenden Zonen um Tabellen handelt (selbst einzeilige Tabellen), können sie zusammengefasst werden. (Standardwert: <code>none</code> ): <b>down</b> Nur nach unten zusammenfassen. <b>none</b> Nicht zusammenfassen. <b>up</b> Nur nach oben zusammenfassen. <b>updown</b> In beide Richtungen zusammenfassen.

Tabelle 10.12 Unteroptionen für die Option `layoutanalysis` von `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>splithint</b>	(Schlüsselwort oder Optionsliste) Aktiviert Sonderbehandlung von Doppelseiten (oder selbst Seiten mit mehr als zwei Bögen). Die Seite kann vertikal oder horizontal in zwei oder mehr Bereiche unterteilt werden. Das Schlüsselwort <code>includebox</code> bedeutet, dass die geteilten Bereiche durch die Option <code>includebox</code> definiert werden. Alternativ können folgende Optionen übergeben werden: <b>x</b> (Float) Trenner für die x-Achse, z.B. 0.5 für eine Doppelseite, 0.33 für eine Seite mit drei Bögen. <b>y</b> (Float) Trenner für die y-Achse.
<b>standalone-fontsize</b>	(Float) Minimale Fontgröße für sehr große Glyphen. Sehr große Glyphen bilden Ein-Glyph-Streifen und werden nicht mit anderen Zonen zusammengefasst (Standardwert: 70).
<b>supertable-columns</b>	(Integer; nur bei <code>layoutastable=true</code> ) Minimale Anzahl von Spalten in einer Layout-Zeile, um die Sequenz von Zonen als eine Supertabelle zu definieren. Wenn eine Tabelle aus Absätzen erzeugt wird, werden diese Spalten als separate Zonen betrachtet und nicht zusammengefasst. Daher kann die Layout-Erkennung diese Zonen-Sequenzen als Tabelle identifizieren (Standardwert: 4).
<b>tabledetect</b>	(Integer) Bestimmt die Tiefe der rekursiven Tabellen-Erkennung (Standardwert: 1): <b>0</b> Keine Tabellen-Erkennung. <b>1</b> Tabellen-Erkennung für jede Zone. <b>2</b> Tabellen-Erkennung für jede in Level 1 entdeckte Tabellenzelle. Dies ist erforderlich für verschachtelte Tabellen und für die Auflösung von Zellen, die sich über mehrere Zeilen erstrecken.

Tabelle 10.13 Unteroptionen für die Option `imageanalysis` von `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>merge</b>	(Optionsliste) Steuert das Zusammensetzen von Bildern. Dieser Vorgang fasst nebeneinander liegende Bilder zu einem einzigen größeren Bild zusammen. Dies ist bei mehrstreifigen (»multi-strip«) Bildern nützlich, bei denen die einzelnen Streifen im PDF erhalten bleiben, sowie bei Hintergrundbildern, die aus vielen sehr kleinen Bildern bestehen. Unterstützte Optionen: <b>disable</b> (Boolean) Bei <code>true</code> wird das Zusammensetzen von Bildern deaktiviert. Standardwert: <code>false</code> <b>gap</b> (Float) Maximaler Abstand zwischen zwei Bildern, die zusammengesetzt werden sollen. Der Wert wird als absolute Entfernung in Punkt sowie als Anzahl von Pixeln interpretiert. Standardwert: 1.0
<b>smallimages</b>	(Optionsliste) Steuert die Entfernung kleiner Bilder. Kleine Bilder müssen oft ignoriert werden, da sie für die Verarbeitung nicht sinnvoll sind. Die Verarbeitung kleiner Bildfragmente betrifft keine Bilder die als Masken verwendet werden. <b>disable</b> (Boolean) Bei <code>true</code> wird das Entfernen von kleinen Bildern deaktiviert. Standardwert: <code>false</code> <b>maxarea</b> (Integer) Maximaler Bereich (= Breite x Höhe) in Pixeln, bis zu dem ein Bild als kleines Bild gilt. Standardwert: 500 <b>maxcount</b> Veraltet, nicht zu verwenden. <b>maxheight</b> (Integer) Maximale Höhe in Pixeln, bis zu dem ein Bild als kleines Bild gilt. Standardwert: 20 <b>maxwidth</b> (Integer) Maximale Breite in Pixeln, bis zu dem ein Bild als kleines Bild gilt. Standardwert: 20

Tabelle 10.14 Unteroptionen für die Option structureanalysis von `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>bullets</b>	<p>(Liste von Optionslisten; nur bei <code>list=true</code>) Gibt die Kombinationen aus Unicode-Zeichen und Fontnamen an, die als Aufzählungszeichen in Listen verwendet werden. Unterstützte Unteroptionen:</p> <p><b>bulletchars</b> (Liste von Unicode-Werten) Ein oder mehrere Unicode-Werte für die Aufzählungszeichen. Wird diese Unteroption nicht übergeben, werden alle Zeichen, die den angegebenen fontname verwenden, als Aufzählungszeichen behandelt.</p> <p><b>fontname</b> (String) Name des Fonts, aus dem die Aufzählungszeichen entnommen werden. Wird diese Unteroption nicht übergeben, werden die in der Unteroption <code>bulletchars</code> angegebenen Zeichen immer als Aufzählungszeichen behandelt.</p> <p>Beispiele:  <code>bullets={{fontname=ZapfDingbats}}</code>  <code>bullets={{bulletchars={U+2022}}}</code>  <code>bullets={{fontname=KozGoPro-Medium bulletchars={U+2460 U+2461 U+2462 U+2463 U+2464}}}</code></p>
<b>list</b>	<p>(Boolean) Aktiviert die Listenerkennung (Standardwert: <code>false</code>). Bei <code>false</code> wird keine Information über die Listenstruktur ermittelt.</p>

Tabelle 10.15 Unteroptionen für die Option vectoranalysis von `TET_open_page()` und `TET_process_page()`

Option	Beschreibung
<b>closetable-area</b>	<p>(Boolean) Bei <code>true</code> wird ein Tabellenrahmen zur Überprüfung angelegt, selbst wenn keiner vorhanden ist. Standardwert: <code>false</code></p>
<b>ignorelines</b>	<p>(Schlüsselwort) Gibt die Linien an, die von der Überprüfung ausgeschlossen werden sollen. Unterstützte Schlüsselwörter (Standardwert: <code>none</code>):</p> <p><b>horizontal</b> Ignoriert horizontale Linien.</p> <p><b>none</b> Berücksichtigt alle Linien.</p> <p><b>vertical</b> Ignoriert vertikale Linien.</p>
<b>pagesizelines</b>	<p>(Boolean) Bei <code>true</code> werden lange Linien berücksichtigt, die fast so lang wie die Größe der Seite sind. Standardwert: <code>false</code></p>
<b>split-sequence</b>	<p>(Boolean) Bei <code>true</code> dürfen vertikale Linien Textsequenzen trennen (oder horizontale Linien bei gedrehtem Text).</p>
<b>structures</b>	<p>(Schlüsselwort) Gibt die Art von Vektorgrafik und die Art ihrer Verarbeitung an. Unterstützte Schlüsselwörter (Standardwert: <code>unions</code>):</p> <p><b>tables</b> Erweiterter <code>unions</code>-Modus: die Engine prüft zusätzlich, ob Unions ein Tabellennetz formen. Wenn ja, wird das Ergebnis als eine einzige Tabellenzone interpretiert.</p> <p><b>unions</b> Versucht, aus Linien ein untergeordnetes Unions-Layout zu erzeugen. Wird ein solches Unions-Layout erzeugt, wird es als vollständige untergeordnete Layout-Einheit behandelt, d.h. alle eingeschlossenen Textzonen gehören dazu.</p>

---

**C++ Java** `void close_page(int page)`

**Perl PHP** `close_page(long page)`

**C** `void TET_close_page(TET *tet, int page)`

---

Gibt ein Seiten-Handle und alle damit verbundenen Ressourcen frei.

**page** Gültiges Seiten-Handle, das mit `TET_open_page()` erzeugt wurde.

*Details* Alle offenen Seiten des Dokuments werden automatisch beim Aufruf von `TET_close_document()` geschlossen. Es zählt jedoch zu guter Programmier-Praxis, Seiten explizit zu schließen, wenn sie nicht mehr gebraucht werden. Geschlossene Seiten-Handle dürfen in Funktionsaufrufen nicht mehr verwendet werden.

## 10.5 Funktionen zur Abfrage von Text- und Glyphendetails

---

**C++ Java** *String get\_text(int page)*  
**Perl PHP** *string get\_text(long page)*  
**C** *const char \*TET\_get\_text(TET \*tet, int page, int \*len)*

---

Liefert das nächste Textfragment des Seiteninhalts.

**page** Gültiges Seiten-Handle, das mit *TET\_open\_page()* erzeugt wurde.

**len** (Nur C-Sprachbindung) Zeiger auf eine Variable, in der die Länge des zurückgegebenen Strings abhängig von der Option *outputformat* von *TET\_set\_option()* gespeichert wird:

Ist *outputformat=utf8*, wird die Stringlänge als Anzahl der Unicode-Zeichen zurückgegeben. Die Anzahl von Bytes in dem null-terminierten String (die mit der Anzahl von 8-Bit-Codeeinheiten identisch ist) kann mit der Funktion *strlen()* bestimmt werden.

Ist *outputformat=utf16*, wird die Stringlänge als Anzahl der 16-Bit-Codeeinheiten zurückgegeben; Surrogatpaare werden als zwei Codeeinheiten gezählt. Die Anzahl der Bytes im String ist  $2 * len$ .

Ist *outputformat=utf32*, wird die Stringlänge als Anzahl der 32-Bit-Codeeinheiten zurückgegeben (die mit der Anzahl von Unicode-Zeichen identisch ist). Die Anzahl der Bytes im String ist  $4 * len$ .

**Rückgabe** String mit dem nächsten Textfragment auf der Seite. Die Länge des Fragments wird mit der Option *granularity* von *TET\_open\_page()* bestimmt. Selbst bei *granularity=glyph* kann der String mehrere Zeichen enthalten (siehe Abschnitt 7.1, »Wichtige Unicode-Konzepte«, Seite 107).

Wurde bereits der komplette Text der Seite abgerufen, wird ein leerer String oder ein Null-Objekt zurückgegeben (siehe unten). In diesem Fall sollte mit *TET\_get\_errnum()* ermittelt werden, ob das Seitenende erreicht ist oder wegen eines Fehlers auf der Seite kein Text zurückgegeben wurde.

**Bindungen** C-Sprachbindung: das Ergebnis wird als null-terminierter UTF-8- (Standard) oder UTF-16-String gemäß der Option *outputformat* von *TET\_set\_option()* geliefert. Auf i5/iSeries und zSeries-Systemen kann auch UTF-8 in EBCDIC-Encoding gewählt werden. Dies ist standardmäßig aktiviert. Der zurückgegebene Datenpuffer kann bis zum nächsten Aufruf dieser Funktion verwendet werden. Ist kein Text mehr verfügbar, wird ein NULL-Zeiger und *\*len=0* zurückgegeben.

C++- und COM-Sprachbindungen: das Ergebnis wird als Unicode-String im Format UTF-16 geliefert (*wstring* in C++). Ist kein Text mehr verfügbar, wird ein leerer String zurückgegeben.

Java-, .NET- und Objective-C-Sprachbindungen: das Ergebnis wird als Unicode-String zurückgegeben. Ist kein Text mehr verfügbar, wird ein Null-Objekt zurückgegeben (*nil* in Objective-C).

Perl und PHP: das Ergebnis wird als UTF-8- (Standard) oder UTF-16-/UTF-32-String gemäß der Option `outputformat` von `TET_set_option()` geliefert. Ist kein Text mehr verfügbar, wird ein leerer String zurückgegeben.

Python: das Ergebnis wird als UTF-8- (Standard) oder UTF-16-/UTF-32-String gemäß der Option `outputformat` von `TET_set_option()` geliefert. In Python 3 werden UTF-16-/UTF-32-Ergebnisse als Bytes zurückgegeben. Ist kein Text mehr verfügbar, wird `None` zurückgegeben.

Ruby: das Ergebnis wird als UTF-8- (Standard) oder UTF-16-/UTF-32-String gemäß der Option `outputformat` von `TET_set_option()` geliefert. Ist kein Text mehr verfügbar, wird ein Null-Objekt zurückgegeben.

REALbasic/Xojo-Sprachbindung: das Ergebnis wird als Unicode-String zurückgegeben. Ist kein Text mehr verfügbar, wird ein leerer String zurückgegeben.

RPG-Sprachbindung: das Ergebnis wird als Unicode-String zurückgegeben. Ist kein Text mehr verfügbar, wird NULL zurückgegeben.

---

**C++** `const TET_char_info *get_char_info(int page)`

**C# Java** `int get_char_info(int page)`

**Perl PHP** `object get_char_info(long page)`

**C** `const TET_char_info *TET_get_char_info(TET *tet, int page)`

---

Fragt detaillierte Informationen zur nächsten Glyphe des aktuellen Textfragments ab.

**page** Gültiges Seiten-Handle, das mit `TET_open_page()` erzeugt wurde.

*Hinweis* Der Name dieser Funktion ist eine Fehlbezeichnung. Sie sollte richtiger mit `TET_get_glyph_info()` bezeichnet werden, da sie Informationen zu den sichtbaren Glyphen auf der Seite und nicht zu den entsprechenden Unicode-Zeichen liefert.

*Rückgabe* Sind im letzten von `TET_get_text()` zurückgegebenen Textfragment keine Glyphen mehr vorhanden, wird ein sprachspezifischer Wert zurückgegeben. Für weitere Informationen siehe Abschnitt *Sprachbindungen* unten.

*Details* Diese Funktion kann auch mehrfach nach `TET_get_text()` aufgerufen werden. Sie betrachtet die nächste Glyphe im aktuellen Textfragment, das zum übergebenen Seiten-Handle gehört und liefert detaillierte Informationen zu diesem Zeichen. Wenn keine Glyphen mehr vorhanden sind, gibt sie nichts zurück. Für einen Textblock mit  $N$  Glyphen und  $M$  logischen Zeichen gibt es einen oder mehrere erfolgreiche Aufrufe dieser Funktion. Das Verhältnis zwischen  $N$  und  $M$  hängt von der Granularität ab:

- ▶ Bei `granularity=glyph` bezieht sich jedes Textfragment auf eine einzige Glyphe, also  $N = 1$ . In vielen Fällen liefert eine Glyphe genau ein Zeichen, also  $M = 1$ . Bei Ligatur-Glyphen liefert eine Glyphe mehrere Zeichen, also  $M > 1$ , und `TET_get_char_info()` muss mehrfach aufgerufen werden.
- ▶ Bei einer von `glyph` verschiedenen Granularität erzeugt eine Glyphensequenz eine Sequenz von Zeichen, wobei jede Glyphe 0, 1 oder mehrere Zeichen liefern kann. Diese Glyphensequenz dient als Rohmaterial für die Sequenz von Unicode-Zeichen. Es gibt also keine bestimmtes Verhältnis zwischen  $N$  und  $M$ . Das Verhältnis zwischen  $N$  und  $M$  kann durch Inhaltsanalyse beeinflusst werden (Trennzeichen werden z.B. bei

der Enttrennung entfernt) oder durch die Unicode-Nachbearbeitung (Zeichen werden z.B. wegen eines Foldings hinzugefügt oder entfernt).

Bei von *glyph* verschiedenen Granularitäten springt diese Funktion zur nächsten Glyph, die zu dem Textfragment beiträgt, das vom letzten Aufruf von *TET\_get\_text()* zurückgegeben wurde. Auf diese Weise lassen sich Glyphendetails abfragen, wenn der Wordfinder aktiv ist und ein Textfragment mehr als ein Zeichen enthalten kann. Um detaillierte Informationen zu allen Glyphen des aktuellen Textfragments zu erhalten, muss diese Funktion mehrfach aufgerufen werden, bis sie keine Informationen mehr zurückgibt.

Die Glyphen-Informationen in der Struktur oder den Eigenschaften/Feldern sind gültig, bis *TET\_get\_char\_info()* oder *TET\_close\_page()* das nächste Mal mit dem selben Seiten-Handle aufgerufen werden. Da für jedes TET-Objekt nur ein Satz von Eigenschaften/Feldern für Glyphen-Informationen vorliegt, müssen diese vom Client komplett abgerufen werden, bevor *TET\_get\_char\_info()* erneut für die selbe oder eine andere Seite oder ein anderes Dokument aufgerufen wird.

**Bindungen** C- und C++-Sprachbindungen: Sind im letzten von *TET\_get\_text()* zurückgegebenen Textfragment keine Glyphen mehr vorhanden, wird ein NULL-Zeiger zurückgegeben. Andernfalls wird ein Zeiger auf eine Struktur *TET\_char\_info* mit Informationen über eine einzelne Glyph zurückgegeben. Die Elemente dieser Datenstruktur werden in Tabelle 10.16 dargestellt.

COM-, Java-, .NET- und Objective-C-Sprachbindungen: Sind im letzten von *TET\_get\_text()* zurückgegebenen Textfragment keine Glyphen mehr vorhanden, wird -1 zurückgegeben, sonst 1. Einzelne Glyphen-Informationen können aus den TET-Eigenschaften/Feldern gemäß Tabelle 10.16 abgefragt werden. Alle Eigenschaften/Felder enthalten den Wert -1 (das Feld *unknown* enthält *false*), wenn auf sie zugegriffen wird, obwohl die Funktion -1 zurückgegeben hat.

Perl- und Python-Sprachbindungen: Sind im letzten von *TET\_get\_text()* zurückgegebenen Textfragment keine Glyphen mehr vorhanden, wird 0 zurückgegeben, sonst ein Hash mit den in Tabelle 10.16 aufgeführten Schlüsselwörtern. Einzelne Glyphen-Informationen können mit den Schlüsselwörtern in diesem Hash abgefragt werden.

PHP-Sprachbindung: Sind im letzten von *TET\_get\_text()* zurückgegebenen Textfragment keine Glyphen mehr vorhanden, wird ein leeres Objekt (Null) zurückgegeben, sonst ein Objekt mit den in Tabelle 10.16 aufgeführten Feldern. Einzelne Glyphen-Informationen können aus den Member-Feldern dieses Objekts abgefragt werden. Integer-Felder im Info-Objekt der Glyph sind in der PHP-Sprachbindung als *long* implementiert.

REALbasic/Xojo-Sprachbindung: Sind im letzten von *TET\_get\_text()* zurückgegebenen Textfragment keine Glyphen mehr vorhanden, wird *nil* zurückgegeben, sonst ein Objekt vom Typ *TET\_char\_info* mit den in Tabelle 10.16 aufgeführten Member-Feldern. Einzelne Glyphen-Informationen können mit den Schlüsselwörtern in diesem Objekt abgefragt werden. Das Feld *attributes* wird in REALbasic/Xojo *attrs* genannt, um Schnittstellenprobleme zu umgehen.

Ruby-Sprachbindung: *nil* (Null-Objekt) wird zurückgegeben, wenn keine Glyphen mehr verfügbar sind, und andernfalls ein Objekt vom Typ *TET\_char\_info*.

Tabelle 10.16 Elemente der Struktur `TET_char_info` (C, C++, Ruby), äquivalente öffentliche Felder (Java, PHP, Objective-C), Schlüsselwörter (Perl) oder Eigenschaften (COM und .NET) mit Typ und Bedeutung. Für weitere Informationen siehe Abschnitt 6.2, »Geometrie von Seite und Text«, Seite 84 und Abschnitt 6.3, »Textfarbe«, Seite 91.

Eigenschaft/ Feldname	Beschreibung
<b>uv</b>	(Integer) UTF-32-Unicode-Wert für die aktuellen Glyphen. Bei von <code>glyph</code> verschiedenen Granularitäten kann dies ein künstlicher Wert oder ein eingefügtes Separatorzeichen sein, die keinen Bezug zum tatsächlichen Textfragment haben. Bei <code>granularity=glyph</code> ist die Sequenz von Unicode-Werten für die Glyphen mit dem logischen Text identisch, bei anderen Granularitäten kann sie aber durch verschiedene Verarbeitungsschritte verändert werden.
<b>type</b>	(Integer) Typ des Zeichens. Die folgenden Typen beschreiben reale Zeichen, die einer Glyphen auf der Seite entsprechen. Die Werte aller anderen Eigenschaften/Felder werden durch die zugehörige Glyphen bestimmt: <ul style="list-style-type: none"> <li><b>0</b> Normales Zeichen, das genau einer Glyphen entspricht</li> <li><b>1</b> Start einer Sequenz (z.B. Ligatur)</li> </ul> <p>Die folgenden Typen beschreiben künstliche Zeichen, die keiner Glyphen auf der Seite entsprechen. Die Felder <code>x</code> und <code>y</code> bezeichnen dann den Endpunkt des letzten realen Zeichens, das Feld <code>width</code> ist <code>0</code> und alle anderen Felder mit Ausnahme von <code>uv</code> enthalten die Werte für das letzte reale Zeichen:</p> <ul style="list-style-type: none"> <li><b>10</b> Fortsetzung einer Sequenz (z.B. Ligatur)</li> <li><b>11</b> (Veraltet, wird nicht mehr verwendet)</li> <li><b>12</b> Eingefügter Wort-, Zeilen- oder Absatzseparator</li> </ul>
<b>attributes<sup>1</sup></b>	(Integer) Glyphenattribute ausgedrückt als Bits, die kombiniert sein können: <ul style="list-style-type: none"> <li><b>bit 0</b> Geometrisch oder semantisch tiefgestellt</li> <li><b>bit 1</b> Geometrisch oder semantisch hochgestellt</li> <li><b>bit 2</b> Mehrzeiliger Anfangsbuchstabe (am Anfang eines Absatzes)</li> <li><b>bit 3</b> Glyph- oder Wort-basiertes Schattenduplikat dieser Glyphen wurde entfernt</li> <li><b>bit 4</b> Glyphen stellt das letzte Zeichen vor der Trennstelle dar</li> <li><b>bit 5</b> Trennungs-Artefakt (d.h. das Trennzeichen), das entfernt wurde, sofern nicht <code>contentanalysis={keeplyphenglyphs=true}</code> angegeben wurde.</li> <li><b>bit 6</b> Glyphen stellt das Zeichen hinter der Trennstelle dar</li> </ul>
<b>unknown</b>	(Boolean; in C, C++ und Perl: Integer) Normalerweise <code>false</code> ( <code>0</code> ), ist aber <code>true</code> ( <code>1</code> ), wenn die Originalglyphen nicht nach Unicode konvertiert werden konnte und durch das mit <code>unknownchar</code> definierte Zeichen ersetzt wurde.
<b>x, y</b>	(Double) Position des Referenzpunkts der Glyphen. Der Referenzpunkt ist die untere linke Ecke der Glyphenbox bei horizontaler Schreibrichtung und der Punkt in der Mitte oben bei vertikaler Schreibrichtung. Bei künstlichen Zeichen entsprechen die <code>x</code> - und <code>y</code> -Koordinaten dem Endpunkt des letzten realen Zeichens.
<b>width</b>	(Double) Breite der entsprechenden Glyphen (bei horizontaler wie vertikaler Schreibrichtung). Für künstliche Zeichen (d.h. eingefügte Separatoren mit <code>type=12</code> und Trennungs-Artefakten mit gesetztem Attribut <code>bit 5</code> ) ist die Breite gleich <code>0</code> .
<b>height</b>	(Double) In vertikaler Schreibrichtung: Höhe der zugehörigen Glyphen unter Berücksichtigung der Parameter zu Fontgröße und Textausgabe (z.B. Zeichenabstand). Die Höhe ist im Standardkoordinatensystem positiv, aber im Top-Down-Koordinatensystem negativ. In äquidistanten vertikalen Fonts haben alle Glyphen <code>fontsize</code> als Höhe, sofern kein Zeichenabstand angewendet wurde. Künstliche Zeichen (z.B. Separatoren) haben eine Höhe von <code>0</code> .  Bei horizontaler Schreibrichtung wird ein angenäherter Wert für die Glyphenhöhe übergeben. Dieser Wert wird aus den Fontproportionen berechnet und ist deshalb für alle Glyphen in einem Font identisch. Es ist nicht garantiert, dass die sichtbare Glyphen exakt die gleiche Höhe hat wie der hier übergebene Wert.

Tabelle 10.16 Elemente der Struktur `TET_char_info` (C, C++, Ruby), äquivalente öffentliche Felder (Java, PHP, Objective-C), Schlüsselwörter (Perl) oder Eigenschaften (COM und .NET) mit Typ und Bedeutung. Für weitere Informationen siehe Abschnitt 6.2, »Geometrie von Seite und Text«, Seite 84 und Abschnitt 6.3, »Textfarbe«, Seite 91.

Eigenschaft/ Feldname	Beschreibung																										
<b>alpha</b>	(Double) Richtung des Textverlaufs in Grad, gemessen gegen den Uhrzeigersinn (bzw. im Uhrzeigersinn bei Top-Down-Koordinaten). Bei horizontaler Schreibrichtung ist das die Richtung der Grundlinie des Textes; bei vertikaler Schreibrichtung ist das die Abweichung von der vertikalen Standardrichtung. Der Winkel liegt im Bereich $-180^\circ < \alpha \leq +180^\circ$ . Für normalen horizontalen Text sowie für Standardtext in vertikaler Schreibrichtung ist der Winkel gleich $0^\circ$ .																										
<b>beta</b>	(Double) Neigung des Textes in Grad gegen den Uhrzeigersinn (bzw. im Uhrzeigersinn bei Top-Down-Koordinaten), relativ zur Senkrechten von alpha. Der Winkel ist $0^\circ$ für normalen senkrechten und negativ für kursiven Text (positiv bei Top-Down-Koordinaten). Der Winkel liegt im Bereich $-180^\circ < \beta \leq 180^\circ$ , ist aber nicht $\pm 90^\circ$ . Ist $\text{abs}(\beta) > 90^\circ$ , ist der Text an der Grundlinie gespiegelt.																										
<b>fontid</b>	(Integer) Index des Fonts im Pseudo-Objekt <code>fonts[]</code> (siehe die pCOS-Pfadreferenz). <code>fontid</code> ist nie negativ.																										
<b>fontsize</b>	(Double) Fontgröße (immer positiv); es besteht keine feste Beziehung dieses Wertes zur tatsächlichen Glyphenhöhe, diese kann je nach Fontdesign unterschiedlich sein. Bei den meisten Fonts ist die Fontgröße so gewählt, dass sie alle Oberlängen (inklusive der Akzente) und Unterlängen umfasst.																										
<b>textrendering</b>	(Integer) Darstellungsmodus für Text (Textrendering): <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; vertical-align: top;"> <table border="0"> <tr> <td style="width: 20px;">0</td> <td style="width: 20px;"></td> <td>Text füllen</td> </tr> <tr> <td>1</td> <td></td> <td>Umrisslinien zeichnen</td> </tr> <tr> <td>2</td> <td></td> <td>Umrisslinien zeichnen, füllen</td> </tr> <tr> <td>3</td> <td></td> <td>Unsichtbarer Text (für OCR)</td> </tr> </table> </td> <td style="width: 50%; vertical-align: top;"> <table border="0"> <tr> <td style="width: 20px;">4</td> <td style="width: 20px;"></td> <td>Text füllen, zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>5</td> <td></td> <td>Umrisslinien zeichnen und zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>6</td> <td></td> <td>Umrisslinien zeichnen, füllen und zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>7</td> <td></td> <td>Text zum Beschneidungspfad hinzufügen</td> </tr> </table> </td> </tr> </table> <p>Text in Type-3-Fonts: <code>textrendering=3</code> und <code>7</code> ergeben unsichtbaren Text; all anderen Werte von <code>textrendering</code> sind irrelevant und können ignoriert werden.</p>	<table border="0"> <tr> <td style="width: 20px;">0</td> <td style="width: 20px;"></td> <td>Text füllen</td> </tr> <tr> <td>1</td> <td></td> <td>Umrisslinien zeichnen</td> </tr> <tr> <td>2</td> <td></td> <td>Umrisslinien zeichnen, füllen</td> </tr> <tr> <td>3</td> <td></td> <td>Unsichtbarer Text (für OCR)</td> </tr> </table>	0		Text füllen	1		Umrisslinien zeichnen	2		Umrisslinien zeichnen, füllen	3		Unsichtbarer Text (für OCR)	<table border="0"> <tr> <td style="width: 20px;">4</td> <td style="width: 20px;"></td> <td>Text füllen, zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>5</td> <td></td> <td>Umrisslinien zeichnen und zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>6</td> <td></td> <td>Umrisslinien zeichnen, füllen und zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>7</td> <td></td> <td>Text zum Beschneidungspfad hinzufügen</td> </tr> </table>	4		Text füllen, zum Beschneidungspfad hinzufügen	5		Umrisslinien zeichnen und zum Beschneidungspfad hinzufügen	6		Umrisslinien zeichnen, füllen und zum Beschneidungspfad hinzufügen	7		Text zum Beschneidungspfad hinzufügen
<table border="0"> <tr> <td style="width: 20px;">0</td> <td style="width: 20px;"></td> <td>Text füllen</td> </tr> <tr> <td>1</td> <td></td> <td>Umrisslinien zeichnen</td> </tr> <tr> <td>2</td> <td></td> <td>Umrisslinien zeichnen, füllen</td> </tr> <tr> <td>3</td> <td></td> <td>Unsichtbarer Text (für OCR)</td> </tr> </table>	0		Text füllen	1		Umrisslinien zeichnen	2		Umrisslinien zeichnen, füllen	3		Unsichtbarer Text (für OCR)	<table border="0"> <tr> <td style="width: 20px;">4</td> <td style="width: 20px;"></td> <td>Text füllen, zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>5</td> <td></td> <td>Umrisslinien zeichnen und zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>6</td> <td></td> <td>Umrisslinien zeichnen, füllen und zum Beschneidungspfad hinzufügen</td> </tr> <tr> <td>7</td> <td></td> <td>Text zum Beschneidungspfad hinzufügen</td> </tr> </table>	4		Text füllen, zum Beschneidungspfad hinzufügen	5		Umrisslinien zeichnen und zum Beschneidungspfad hinzufügen	6		Umrisslinien zeichnen, füllen und zum Beschneidungspfad hinzufügen	7		Text zum Beschneidungspfad hinzufügen		
0		Text füllen																									
1		Umrisslinien zeichnen																									
2		Umrisslinien zeichnen, füllen																									
3		Unsichtbarer Text (für OCR)																									
4		Text füllen, zum Beschneidungspfad hinzufügen																									
5		Umrisslinien zeichnen und zum Beschneidungspfad hinzufügen																									
6		Umrisslinien zeichnen, füllen und zum Beschneidungspfad hinzufügen																									
7		Text zum Beschneidungspfad hinzufügen																									
<b>colorid</b>	(Integer) Index der Textfarbe, also der Kombination aus Füllfarbe, Linienfarbe und Textdarstellung. Gleiche Kombinationen in einem Dokument werden mit der gleichen Farb-ID dargestellt. Unterschiedliche Kombinationen werden mit unterschiedlichen Farb-IDs dargestellt. Die Farben mehrerer Glyphen können also anhand ihrer Farb-IDs auf Gleichheit geprüft werden. Durch den Vergleich der <code>colorid</code> -Werte aufeinanderfolgender Glyphen lassen sich zum Beispiel Änderungen in der Textfarbe erkennen. Der genaue Farbraum und die genauen Farbkomponenten für das Füllen und/oder Zeichnen von Text lassen sich mit <code>TET_get_color_info()</code> abfragen.																										

1. In der REALbasic/Xojo-Sprachbindung heißt dieses Feld `attr8`.

```

C++  const TET_color_info *get_color_info(int doc, int colorid, wstring optlist)
C# Java  int get_color_info(int doc, int colorid, String optlist)
Perl PHP  object get_color_info(long doc, long colorid, string optlist)
C  const TET_color_info *TET_get_color_info(TET *tet, int doc, int colorid, const char *optlist)

```

Abfrage von Farbinformationen für eine Farb-ID, die mit `TET_get_char_info` abgefragt wurde.

**doc** Gültiges Dokument-Handle, das mit `TET_open_document*()` erzeugt wurde.

- colorid** Gültige Farb-ID, die vom Element *colorid* von `TET_get_char_info()` erzeugt wurde.
- optlist** Optionsliste gemäß Tabelle 10.17 mit den Arten von Farbe, die extrahiert werden sollen.

Tabelle 10.17 Option für `TET_get_color_info()`

Option	Beschreibung
<b>usage</b>	(Schlüsselwort) Verwendung der Farbe (Standardwert: fill)
<b>fill</b>	Füllfarbe abfragen
<b>stroke</b>	Linienfarbe abfragen

**Rückgabe** Struktur mit Informationen zu Farbraum und Farbe, die abgefragt wurden.

**Bindungen** C und C++-Sprachbindungen: Zeiger auf eine Struktur `TET_color_info` mit Informationen zur abgefragten Füll- oder Linienfarbe. Die Elemente dieser Datenstruktur werden in Tabelle 10.18 dargestellt.

COM-, Java-, .NET- und Objective-C-Sprachbindungen: Farbinformationen können aus den Properties/Public fields von TET gemäß Tabelle 10.18 abgefragt werden.

Perl- und Python-Sprachbindungen: Farbinformationen können aus einem Hash abgefragt werden, der die in Tabelle 10.18 aufgeführten Schlüsselwörter enthält.

PHP-Sprachbindung: Farbinformationen können aus einem Objekt abgefragt werden, das die in Tabelle 10.18 aufgeführten Felder enthält.

REALbasic/Xojo-Sprachbindung: Farbinformationen können aus einem Objekt `TET_char_info` abgefragt werden, das die in Tabelle 10.18 aufgeführten Elemente enthält.

Ruby-Sprachbindung: Farbinformationen können aus einem Objekt `TET_char_info` abgefragt werden, das die in Tabelle 10.18 aufgeführten Elemente enthält.

Tabelle 10.18 Elemente der Struktur `TET_color_info` (C, C++, Ruby), äquivalente öffentliche Felder (Java, PHP, Objective-C), Schlüsselwörter (Perl) oder Eigenschaften (COM und .NET) mit Typ und Bedeutung. Für weitere Informationen siehe Abschnitt 6.2, »Geometrie von Seite und Text«, Seite 84 und »Textfarbe«, Seite 91.

Eigenschaft/ Feldname	Beschreibung
<b>colorspaceid</b>	(Integer) Index des Farbraums im Pseudo-Objekt <code>colorspaces[ ]</code> (siehe die <code>pCOS</code> -Pfadreferenz) oder -1, sofern auf die Glyphe keine Farbe angewendet wird.
<b>patternid</b>	(Integer) Index des Patterns im Pseudo-Objekt <code>patterns[ ]</code> (siehe die <code>pCOS</code> -Pfadreferenz) oder -1, sofern auf die Glyphe kein Pattern angewendet wird.
<b>components</b>	(Array mit Double-Werten) Farbkomponentenwerte, die in dem Farbraum interpretiert werden, der mit <code>colorspaceid</code> ausgegeben wird. C- und C++-Sprachbindungen: Die Anzahl an relevanten Array-Einträgen ist im Feld <code>n</code> angegeben.
<b>n</b>	(Integer; nur C- und C++-Sprachbindungen) Anzahl der Array-Einträge in <code>components</code>

## 10.6 Funktionen zur Abfrage von Bildern

---

**C++** `const TET_image_info *get_image_info(int page)`

**C# Java** `int get_image_info(int page)`

**Perl PHP** `object image_info get_image_info(long page)`

**C** `const TET_image_info *TET_get_image_info(TET *tet, int page)`

---

Liefert Informationen zum nächsten Bild auf der Seite (jedoch nicht die eigentlichen Pixeldaten).

**page** Gültiges Seiten-Handle, das mit `TET_open_page()` erzeugt wurde.

**Rückgabe** Sind keine Bilder mehr auf der Seite vorhanden, wird ein sprachspezifischer Wert zurückgegeben, andernfalls stehen sprachspezifische Informationen zum Bild zur Verfügung. Für weitere Informationen siehe Abschnitt *Sprachbindungen* unten.

**Details** Diese Funktion liefert das nächste Bild, das mit dem übergebenen Seiten-Handle verknüpft ist (oder gibt 0/NULL zurück, wenn keine weiteren Bilder mehr vorhanden sind) und gibt detaillierte Informationen zum Bild zurück. Die folgenden Bildtypen werden ignoriert:

- ▶ Als Masken verwendete Bilder werden ignoriert. Sie können mit pCOS und dem Pseudo-Objekt *maskid* abgefragt werden (siehe Abschnitt 8.5.2, »Bildmasken und Transparenzmasken«, Seite 144).
- ▶ Beim Zusammensetzen in ein größeres Bild überführte (d.h. *mergetype=consumed*) werden ignoriert.
- ▶ Kleine Bildfragmente, die bei der Bildzusammensetzung ausgefiltert wurden, werden ignoriert (siehe Abschnitt 8.4, »Filtern von Bildfragmenten«, Seite 141).
- ▶ Komplette außerhalb des Extraktionsbereichs liegende Bilder, die durch die Optionen *clippingarea*, *excludebox* und *includebox* definiert wurde, werden ignoriert.

Die Bildinformationen in der Struktur oder den Eigenschaften/Feldern sind gültig, bis `TET_get_image_info()` oder `TET_close_page()` das nächste Mal mit dem selben Seiten-Handle aufgerufen werden. Da für jedes TET-Objekt nur ein Satz von Eigenschaften/Feldern für Bildinformationen vorliegt, müssen diese vom Client komplett ausgewertet werden, bevor `TET_get_image_info()` erneut für dieselbe oder eine andere Seite aufgerufen wird.

**Bindungen** C- und C++-Sprachbindungen: Sind keine Bilder mehr auf der Seite verfügbar, wird ein NULL-Zeiger zurückgegeben. Andernfalls wird ein Zeiger auf eine Struktur vom Typ `TET_image_info` mit Bildinformationen zurückgegeben. Die Elemente dieser Datenstruktur werden in Tabelle 10.19 dargestellt.

COM-, Java-, .NET- und Objective-C-Sprachbindungen: Sind keine Bilder mehr auf der Seite verfügbar, wird -1 zurückgegeben, sonst 1. Einzelne Bildinformationen können aus den TET-Eigenschaften/Feldern gemäß Tabelle 10.19 abgefragt werden. Alle Eigenschaften/Felder enthalten den Wert -1, wenn auf sie zugegriffen wird, obwohl die Funktion -1 zurückgegeben hat.

Perl- und Python-Sprachbindungen: Sind keine Bilder mehr auf der Seite verfügbar, wird 0 zurückgegeben, sonst ein Hash mit den in Tabelle 10.19 aufgeführten Schlüssel-

wörtern. Einzelne Bildinformationen können mit den Schlüsselwörtern in diesem Hash abgefragt werden.

PHP-Sprachbindung: ein leeres (Null-)Objekt wird zurückgegeben, wenn keine Bilder mehr auf der Seite verfügbar sind, andernfalls ein Objekt vom Typ `TET_image_info`. Einzelne Bildinformationen können aus jenen Feldern gemäß Tabelle 10.19 abgefragt werden. Integer-Felder im Info-Objekt des Bildes sind in der PHP-Sprachbindung als `long` implementiert.

REALbasic/Xojo-Sprachbindung: Sind keine Bilder mehr auf der Seite verfügbar, wird `nil` zurückgegeben, sonst ein Objekt vom Typ `TET_image_info` mit den in Tabelle 10.19 aufgeführten Member-Feldern. Einzelne Bildinformationen können aus den Member-Feldern dieses Objekts abgefragt werden.

Ruby-Sprachbindung: Sind keine Bilder mehr verfügbar, wird `nil` (Null-Objekt) zurückgegeben, andernfalls ein Objekt vom Typ `TET_image_info`.

Tabelle 10.19 Elemente der Struktur `TET_image_info` (C, C++, Ruby), äquivalente öffentliche Felder (Java, PHP, Objective-C) und Eigenschaften (COM und .NET) mit Typ und Bedeutung. Für weitere Informationen siehe Abschnitt 8.1, »Grundlagen der Bildextraktion«, Seite 131.

Eigenschaft/ Feldname	Beschreibung
<code>x, y</code>	(Double) Position des Referenzpunkts des Bildes. Der Referenzpunkt ist die untere linke Ecke des Bildes.
<code>width, height</code>	(Double) Breite und Höhe des Bildes auf der Seite in Punkt, gemessen entlang den Kanten des Bildes
<code>alpha</code>	(Double) Richtung der Pixelzeilen. Der Winkel liegt im Bereich $-180^\circ < \alpha \leq +180^\circ$ . Für aufrecht stehende Bilder ist <code>alpha</code> gleich <code>0</code> .
<code>beta</code>	(Double) Richtung der Pixelspalten relativ zur Senkrechten von <code>alpha</code> . Der Winkel liegt im Bereich $-180^\circ < \beta \leq +180^\circ$ , ist aber nicht $\pm 90^\circ$ . Für aufrecht stehende Bilder liegt <code>beta</code> im Bereich $-90^\circ < \beta < +90^\circ$ . Ist $ \beta  > 90^\circ$ , wird das Bild an der Grundlinie gespiegelt.
<code>imageid</code>	(Integer) Index des Bildes im pCOS-Pseudo-Objekt <code>images[ ]</code> . Ausführliche Bild- und Maskeneigenschaften können über die Einträge in diesem Pseudo-Objekt abgefragt werden (siehe die pCOS-Pfadreferenz).

---

```

C++ Java int write_image_file(int doc, int imageid, String optlist)
Perl PHP long write_image_file(long doc, long imageid, string optlist)
C int TET_write_image_file(TET *tet, int doc, int imageid, const char *optlist)

```

---

Schreibt Bilddaten auf die Festplatte.

**doc** Gültiges Dokument-Handle, das mit `TET_open_document*`( ) erzeugt wurde.

**imageid** pCOS-ID des Bildes. Diese ID kann über das Feld `imageid` nach einem erfolgreichen Aufruf von `TET_get_image_info`( ) abgefragt werden oder beim Durchlauf aller Einträge im Pseudo-Objekt `images` (dieser Array enthält `length:images` Einträge).

**optlist** Optionsliste mit Seitenoptionen gemäß Tabelle 10.20. Folgende Optionen können verwendet werden:

`compression, dpi, filename, keepicprofile, keepxmp, preferredtiffcompression, typeonly, validatejpeg.`

Folgende Optionen anderer Funktionen beeinflussen die erzeugte Bildausgabe ebenfalls:

- ▶ **TET\_open\_document\***(*): allowjpeg2000, spotcolor* (siehe Tabelle 10.8)
- ▶ **TET\_open\_page/TET\_process\_page**(*): imageanalysis* (siehe Tabelle 10.10, Tabelle 10.13)

**Rückgabe** -1 im Fehlerfall, und sonst ein Wert größer o. Gibt die Funktion einen Fehler (-1) zurück, sollten Sie genauere Informationen zur Fehlerursache mit **TET\_get\_errmsg**(*)* abfragen. Im Fehlerfall wird keine Bildausgabe erzeugt. Sofern der Rückgabewert nicht -1 ist, kann das Bild in dem im Rückgabewert angegebenen Dateiformat extrahiert werden:

- ▶ -1: ein Fehler ist aufgetreten; es wird kein Bild extrahiert
- ▶ 10: das Bild wird als TIFF (*.tif*) extrahiert
- ▶ 20: das Bild wird als JPEG (*.jpg*) extrahiert
- ▶ 31: das Bild wird als einfaches JPEG 2000 (*.jp2*) extrahiert (nur wenn *allowjpeg2000=true*)
- ▶ 32: das Bild wird als erweitertes JPEG 2000 (*.jpf*) extrahiert (nur wenn *allowjpeg2000=true*)
- ▶ 33: das Bild wird als reiner JPEG-2000-Codestream (*.j2k*) extrahiert (nur wenn *allowjpeg2000=true*)
- ▶ 50: das Bild wird als JBIG2 (*.jbig2*) extrahiert

**Details** Diese Funktion wandelt die Pixeldaten für das Bild mit der angegebenen pCOS-ID in eins von mehreren Bildformaten um und speichert das Ergebnis auf der Festplatte. Wurde die Option *typeonly* übergeben, wird nur der Bildtyp zurückgegeben, aber keine Bilddatei erzeugt.

**Bindungen** C/C++: Makros für die Rückgabewerte sind in *tetlib.h* verfügbar.

Tabelle 10.20 Optionen für **TET\_write\_image\_file**(*)* und **TET\_get\_image\_data**(*)*

Option	Beschreibung
<b>compression</b>	(Schlüsselwort) Algorithmus für die Kompression von Pixeldaten (Standardwert: auto): <b>auto</b> Ein geeigneter Kompressionsalgorithmus wird automatisch ausgewählt. <b>none</b> (Nur relevant für TIFF-Bilder) Speichert die Pixeldaten unkomprimiert, falls möglich.
<b>dpi</b>	(Liste aus ein oder zwei nicht negativen Floatwerten) Ein oder zwei Werte, die die gewünschte Bildauflösung in Pixeln pro Zoll in horizontaler und vertikaler Richtung angeben. Wird nur ein einzelner Wert übergeben, wird er für beide Richtungen verwendet. Die übergebenen Werte werden in den erzeugten TIFF-Bildern gespeichert. Sie verändern die Anzahl von Pixeln im Bild nicht (d.h. kein Downsampling). Für Informationen zur Bestimmung der Bildauflösung siehe »Bildauflösung«, Seite 138. Sind ein oder zwei Werte gleich Null, wird keine Bildauflösung gespeichert. Standardwert: 72
<b>filename</b> <sup>1</sup>	(String; erforderlich, sofern nicht <i>typeonly</i> übergeben wird) Name der Bilddatei auf der Festplatte. An den Dateinamen wird ein Suffix für das Format der Bilddatei angehängt. Tabelle 2.1, Seite 19 beschreibt die Dateinamenskonventionen für das TET-Kommandozeilen-Werkzeug und für TETML. Wir empfehlen, das gleiche Dateinamensschema zu verwenden, wenn Bilder zusammen mit TETML verwendet werden.
<b>keepiccprofile</b>	(Boolean) Bei <i>true</i> und wenn dem Bild ein ICC-Profil zugewiesen ist, wird dieses in die extrahierten TIFF- und JPEG-Bilder eingebettet. Wird diese Option auf <i>false</i> gesetzt, führt dies auf Kosten des Color-Managements zu kleineren Bilddateien. Standardwert: <i>true</i>
<b>keepxmp</b>	(Boolean) Bei <i>true</i> und wenn das Bild über zugehörige XMP-Metadaten im PDF verfügt, werden die Metadaten in die extrahierten TIFF- und JPEG-Bilder eingebettet. Standardwert: <i>true</i>

Tabelle 10.20 Optionen für `TET_write_image_file()` und `TET_get_image_data()`

Option	Beschreibung
<b>preferredtiff-compression</b>	(Schlüsselwort) Kompressionsschema für die meisten extrahierten TIFF-Bilder (Standardwert: flate): <b>lzw</b> LZW-Kompression (TIFF-Kompressionsschema 5) <b>flate</b> Flate-Kompression, auch Adobe Deflate oder zlib-Kompression genannt (TIFF-Kompressionsschema 8)
<b>typeonly</b> <sup>1</sup>	(Boolean) Der Bildtyp wird anhand der übergebenen Bildoptionen bestimmt, aber es wird keine Bilddatei erzeugt. Dies ist nützlich, um den von <code>TET_get_image_data()</code> zurückgegebenen Bildtyp zu bestimmen, da er von der Funktion selbst nicht zurückgegeben wird. Standardwert: false
<b>validatejpeg</b>	(Boolean) Bei true werden extrahierte JPEG-Bilder überprüft, um eine korrekte Bildausgabe zu gewährleisten. Bei false ist die Verarbeitung etwas schneller, aber ungültige JPEG-Daten werden unverändert in die erzeugte Bilddatei kopiert. Standardwert: true

1. Nur bei `TET_write_image_file()`

---

**C++** `const char *get_image_data(int doc, size_t *length, int imageid, wstring optlist)`

**C# Java** `final byte[] get_image_data(int doc, int imageid, String optlist)`

**Perl PHP** `string get_image_data(long doc, long imageid, string optlist)`

**C** `const char *TET_get_image_data(TET *tet, int doc, size_t *length, int imageid, const char *optlist)`

---

Schreibt Bilddaten in den Arbeitsspeicher.

**doc** Gültiges Dokument-Handle, das mit `TET_open_document*()` erzeugt wurde.

**length** (Nur C- und C++-Sprachbindungen) C-Zeiger auf einen Speicherplatz, an dem die Länge des zurückgegebenen Strings in Bytes abgelegt wird.

**imageid** pCOS-ID des Bildes. Diese ID kann über das Feld `imageid` nach einem erfolgreichen Aufruf von `TET_get_image_info()` abgefragt werden oder beim Durchlauf aller Einträge im pCOS-Array `images` (dieser Array enthält `length:images` Einträge).

**optlist** Optionsliste mit Bildoptionen gemäß Tabelle 10.20. Folgende Optionen können verwendet werden: `compression`, `keepxmp`

**Rückgabe** Bilddaten gemäß der angegebenen Optionen. Im Fehlerfall wird in C and C++ ein NULL-Zeiger zurückgegeben und in allen anderen Sprachbindungen ein leerer String. Gibt die Funktion einen Fehler (-1) zurück, sollten Sie genauere Informationen zur Fehlerursache mit `TET_get_errmsg()` abfragen.

**Details** Diese Funktion wandelt die Pixeldaten für das Bild mit der angegebenen pCOS-ID in eins von mehreren Bildformaten um und stellt die Daten im Arbeitsspeicher zur Verfügung.

**Bindungen** COM: Client-Programme verwenden zur Speicherung der UTF-8-Daten meist den Typ Variant.

C- und C++-Sprachbindungen: Der zurückgegebene Datenpuffer kann bis zum nächsten Aufruf dieser Funktion verwendet werden.

REALbasic/Xojo: das Ergebnis wird als REALbasic/Xojo-String im Encoding -1 ausgegeben (Binärdaten).

## 10.7 Funktionen für TET Markup Language (TETML)

---

**C++ Java** *int process\_page(int doc, int pagenumber, String optlist)*  
**Perl PHP** *long process\_page(long doc, long pagenumber, string optlist)*  
**C** *int TET\_process\_page(TET \*tet, int doc, int pagenumber, const char \*optlist)*

---

Verarbeitet eine Seite und erzeugt TETML-Ausgabe.

**doc** Gültiges Dokument-Handle, das mit *TET\_open\_document\**() erzeugt wurde.

**pagenumber** Physikalische Nummer der zu verarbeitenden Seite. Die erste Seite hat die Seitennummer 1. Die Gesamtzahl der Seiten kann mit *TET\_pcos\_get\_number()* und dem pCOS-Pfad *length:pages* abgefragt werden. Der Parameter *pagenumber* kann 0 sein, falls *trailer=true*.

**optlist** Optionsliste mit Optionen aus folgenden Gruppen:

- ▶ Allgemeine seitenbezogene Optionen gemäß Tabelle 10.10 (diese werden ignoriert, falls *pagenumber=0*):  
*clippingarea, contentanalysis, excludebox, fontsizeange, granularity, ignoreinvisibletext, imageanalysis, includebox, layoutanalysis, skipengines*
- ▶ Option zu TETML-Details gemäß Tabelle 10.21: *tetml*

Tabelle 10.21 Zusätzliche Optionen für *TET\_process\_page()*

Option	Beschreibung
<b>tetml</b>	(Optionsliste) Steuert TETML-Details. Folgende Optionen sind verfügbar:
<b>elements</b>	(Optionsliste) Gibt optionale TETML-Elemente an:
<b>line</b>	(Boolean; nur bei <i>granularity=word</i> ) Bei true enthält die TETML-Ausgabe Line-Elemente zwischen dem Level Para und Word. Standardwert: false
<b>glyphdetails</b>	(Optionsliste; nur bei <i>granularity=glyph</i> und <i>granularity=word</i> ) Gibt an, welche Attribute für jedes Glyphenelement ausgegeben werden (Standardwert für alle Unteroptionen: false):
<b>all</b>	(Boolean) Alle Attribut-Unteroptionen sind aktiviert
<b>dehyphenation</b>	(Boolean) Gibt das Attribut <i>dehyphenation</i> zur Kennzeichnung getrennter Wörter aus.
<b>dropcap</b>	(Boolean) Gibt das Attribut <i>dropcap</i> zur Kennzeichnung mehrzeiliger Anfangsbuchstaben am Anfang eines Absatzes aus.
<b>font</b>	(Boolean) Gibt die Attribute <i>font, size, textrendering, unknown</i> aus.
<b>geometry</b>	(Boolean) Gibt die Attribute <i>x, y, width, alpha, beta</i> aus.
<b>sub</b>	(Boolean) Gibt das Attribut <i>sub</i> zur Kennzeichnung tiefgestellter Zeichen aus.
<b>sup</b>	(Boolean) Gibt das Attribut <i>sup</i> zur Kennzeichnung hochgestellter Zeichen aus.
<b>shadow</b>	(Boolean) Gibt das Attribut <i>shadow</i> zur Kennzeichnung von Schattentext oder künstlicher Fettschrift aus.
<b>textcolor</b>	(Boolean) Gibt die Attribute <i>fill</i> und <i>stroke</i> für die Glyphfarben (abhängig von <i>textrendering</i> ) und für die zugehörigen Color-Elemente aus.
<b>trailer</b>	(Boolean) Bei true werden die Trailer-Daten, d.h. Daten hinter der letzten Seite des Dokuments, ausgegeben (sie müssen an die zuvor ausgegebenen seitenbezogenen Daten angehängt werden). Um Trailer-Daten auszugeben, muss diese Option im letzten Aufruf dieser Funktion übergeben werden. Bei <i>pagenumber=0</i> werden nur Trailer-Daten ausgegeben (ohne seitenbezogene Daten). Sobald <i>trailer=true</i> übergeben wurde, darf <i>TET_process_page()</i> für dasselbe Dokument nicht mehr aufgerufen werden. Standardwert: false

**Rückgabe** Diese Funktion gibt immer 1 zurück. PDF-Probleme werden in einem TETML-Element *Exception* ausgegeben. Probleme bei der Analyse von Optionslisten lösen eine *Exception* aus.

**Details** Diese Funktion öffnet eine Seite, erzeugt TETML-Ausgabe gemäß der an `TET_open_document*`( ) übergebenen Formatoptionen und schließt die Seite. Die erzeugten Daten können mit `TET_get_tetml`( ) abgefragt werden.

Diese Funktion darf nur aufgerufen werden, wenn die Option `tetml` im entsprechenden Aufruf von `TET_open_document*`( ) übergeben wurde. Header-Daten, d.h. dokumentbezogene Daten vor der ersten Seite, werden vor den ersten seitenbezogenen Daten durch `TET_open_document*`( ) erzeugt. Sie können durch einen Aufruf von `TET_get_tetml`( ) vor dem ersten Aufruf von `TET_process_page`( ) separat oder in Kombination mit seitenbezogenen Daten abgefragt werden.

Trailer-Daten, d.h. dokumentbezogene Daten hinter der letzten Seite, müssen mit der Unteroption `trailer` angefordert werden, wenn diese Funktion zum letzten Mal für ein Dokument aufgerufen wird. Trailer-Daten können mit einem separaten Aufruf nach der letzten Seite (`pagenumber=0`) oder zusammen mit der letzten Seite aufgerufen werden (`pagenumber` ungleich 0). Seiten können in beliebiger Reihenfolge abgefragt werden und es kann eine beliebige Untermenge an Dokumentseiten abgefragt werden.

Der Aufruf von `TET_close_document`( ) ohne Abfrage der Trailer-Daten bzw. der erneute Aufruf von `TET_process_page`( ) nach Abfrage der Trailer-Daten führt zu einem Fehler.

---

**C++** `const char *get_tetml(int doc, size_t *length, wstring optlist)`

**C# Java** `final byte[] get_tetml(int doc, String optlist)`

**Perl PHP** `string get_tetml(long doc, string optlist)`

**C** `const char *TET_get_tetml(TET *tet, int doc, size_t *length, const char *optlist)`

---

Abfragen von TETML-Daten aus dem Arbeitsspeicher.

**doc** Gültiges Dokument-Handle, das mit `TET_open_document*`( ) erzeugt wurde.

**length** (Nur C- und C++-Sprachbindungen) Zeiger auf eine Variable zur Speicherung der Länge der zurückgegebenen Streamdaten in Bytes. Bei `length` wird das terminierende Null-Byte nicht mitgezählt.

**optlist** (Derzeit sind keine unterstützten Optionen verfügbar.)

**Rückgabe** Byte-Array, das das nächste Datenfragment enthält. Ist der Speicher leer, wird ein leerer String zurückgegeben (in C: ein NULL-Zeiger und `*len=0`).

**Details** Mit dieser Funktion können TETML-Daten abgefragt werden, die mit `TET_open_document*`( ) und einem oder mehreren Aufrufen von `TET_process_page`( ) erzeugt wurden. TETML-Daten werden, unabhängig von der Option `outputformat`, immer im Format UTF-8 kodiert. Der interne Speicher wird durch diesen Aufruf geleert. `TET_get_tetml`( ) muss deshalb nicht nach jedem `TET_process_page`( ) aufgerufen werden. Der Client kann Daten für eine oder mehrere Seiten oder für das gesamte Dokument im Speicher akkumulieren.

Im TETML-Modus muss diese Funktion mindestens einmal vor `TET_close_document`( ) aufgerufen werden, da sonst die Daten nicht länger zugänglich sind. Wird `TET_get_tetml`( ) genau einmal aufgerufen (ein solcher Aufruf muss zwischen dem letzten Aufruf von `TET_process_page`( ) und `TET_close_document`( ) erfolgen), enthält der Speicher für das

gesamte Dokument auf jeden Fall korrekte TETML-Ausgabe. Diese Funktion darf nicht aufgerufen werden, wenn die Unteroption *filename* an die Option *tetml* von *TET\_open\_document\*()* übergeben wurde.

**Bindungen** C- und C++-Sprachbindungen: das Ergebnis wird als null-terminierter UTF-8-String zurückgegeben. Bei i5/iSeries und zSeries wird EBCDIC-kodiertes UTF-8 zurückgegeben. Der zurückgegebene Datenpuffer kann bis zum nächsten Aufruf von *TET\_get\_tetml()* verwendet werden.

Java- und .NET-Sprachbindungen: das Ergebnis wird als Byte-Array mit UTF-8-Daten zurückgegeben.

COM: Client-Programme verwenden zur Speicherung der UTF-8-Daten meist den Typ *Variant*.

REALbasic/Xojo: das Ergebnis wird als REALbasic/Xojo-String im UTF-8-Encoding zurückgegeben.

PHP-Sprachbindung: das Ergebnis wird als UTF-8-String zurückgegeben.

Python: das Ergebnis wird als 8-Bit-String zurückgegeben (Python 3: *bytes*).

RPG-Sprachbindung: das Ergebnis wird als null-terminierter EBCDIC-UTF-8-String zurückgegeben.

---

**C++** *const char \*get\_xml\_data(int doc, size\_t \*length, wstring optlist)*

**C# Java** *final byte[] get\_xml\_data(int doc, String optlist)*

**Perl PHP** *string get\_xml\_data(long doc, string optlist)*

**C** *const char \*TET\_get\_xml\_data(TET \*tet, int doc, size\_t \*length, const char \*optlist)*

---

Veraltet, verwenden Sie stattdessen *TET\_get\_tetml()*.

## 10.8 pCOS-Funktionen

Die pCOS-Syntax zur Abfrage von Objektdaten aus einem PDF wird vollständig unterstützt. Für eine genaue Beschreibung siehe die pCOS-Pfadreferenz, die als separates Dokument zur Verfügung steht.

---

**C++ Java** `double pcos_get_number(int doc, String path)`  
**Perl PHP** `float pcos_get_number(int doc, string path)`  
**C** `double TET_pcos_get_number(TET *tet, int doc, const char *path, ...)`

---

Liefert den Wert eines pCOS-Pfades vom Typ *Zahl* oder *Boolean*.

**doc** Gültiges Dokument-Handle, das mit `TET_open_document*()` erzeugt wurde.

**path** Vollständiger pCOS-Pfad für ein Zahl- oder Boolean-Objekt.

**Weitere Parameter** (Nur C-Sprachbindung) Es können ein oder mehrere zusätzliche Parameter übergeben werden, sofern der Parameter *key* entsprechende Platzhalter enthält (%s für Strings oder %d für Integers; %% wird für ein einzelnes Prozentzeichen verwendet). Diese Parameter ersparen es Ihnen, komplexe Pfade mit mehreren Zahlen oder Strings explizit zu formatieren. Der Client muss sicherstellen, dass Anzahl und Typ der Platzhalter mit den zusätzlich übergebenen Parametern übereinstimmen.

**Rückgabe** Numerischer Wert des durch den pCOS-Pfad bezeichneten Objekts. Bei Booleschen Werten wird 1 zurückgegeben, wenn sie *true* sind, andernfalls 0.

---

**C++ Java** `String pcos_get_string(int doc, String path)`  
**Perl PHP** `string pcos_get_string(int doc, string path)`  
**C** `const char *TET_pcos_get_string(TET *tet, int doc, const char *path, ...)`

---

Liefert den Wert eines pCOS-Pfades vom Typ *Name*, *Zahl*, *String* oder *Boolean*.

**doc** Gültiges Dokument-Handle, das mit `TET_open_document*()` erzeugt wurde.

**path** Vollständiger pCOS-Pfad für ein String-, Name-, Zahl- oder Boolean-Objekt.

**Weitere Parameter** (Nur C-Sprachbindung) Es können ein oder mehrere zusätzliche Parameter übergeben werden, sofern der Parameter *key* entsprechende Platzhalter enthält (%s für Strings oder %d für Integers; %% wird für ein einzelnes Prozentzeichen verwendet). Diese Parameter ersparen es Ihnen, komplexe Pfade mit mehreren Zahlen oder Strings explizit zu formatieren. Der Client muss sicherstellen, dass Anzahl und Typ der Platzhalter mit den zusätzlich übergebenen Parametern übereinstimmen.

**Rückgabe** String mit dem Wert des durch den pCOS-Pfad bezeichneten Objekts. Bei Booleschen Werten wird einer der Strings *true* oder *false* zurückgegeben.

**Details** Diese Funktion löst eine Exception aus, wenn pCOS nicht im vollständigen Modus läuft und der Typ des Objekts *string* ist. Die Objekte `/Info/*` (Dokument-Infoschlüssel) können jedoch auch im eingeschränkten pCOS-Modus abgefragt werden, sofern `nocopy=false` oder `plainmetadata=true` ist; und `bookmarks[...]/Title` sowie alle Pfade beginnend mit `pages[...]/annots[...]/` können im eingeschränkten pCOS-Modus abgefragt werden, sofern `nocopy=false` ist.

Diese Funktion geht davon aus, dass die Strings, die aus dem PDF-Dokument abgefragt werden, Text-Strings sind. String-Objekte, die Binärdaten enthalten, sollten mit `TET_pcos_get_stream()` abgefragt werden, das die Daten unverändert zurückliefert.

**Bindungen** C-Sprachbindung: Der String wird im UTF-8-Format (bei zSeries und i5/iSeries: EBCDIC-UTF-8) ohne BOM zurückgegeben. Die zurückgegebenen Strings werden in einem Ring-Puffer mit bis zu 10 Einträgen gespeichert. Werden mehr als 10 Strings abgefragt, werden die Puffer wiederverwendet. Clients müssen die Strings deshalb kopieren, wenn sie auf mehr als 10 Strings gleichzeitig zugreifen möchten. Bis zu 10 Aufrufe dieser Funktion können zum Beispiel als Parameter für eine `printf()`-Anweisung verwendet werden, da die Rückgabe-Strings unabhängig voneinander sind, sofern nicht mehr als 10 Strings gleichzeitig verwendet werden.

C++-Sprachbindung: Der String wird in der standardmäßig aktiven `wstring`-Konfiguration des C++-Wrappers als `wstring` zurückgegeben. Im Kompatibilitätsmodus `string` bei zSeries und i5/iSeries wird das Ergebnis im EBCDIC-UTF-8-Format ohne BOM zurückgegeben.

Java- und .NET-Sprachbindungen: das Ergebnis wird als Unicode-String zurückgegeben. Ist kein Text mehr verfügbar, wird ein Null-Objekt zurückgegeben.

Perl-, PHP-, Python- und Ruby-Sprachbindungen: das Ergebnis wird als UTF-8-String zurückgegeben. Ist kein Text mehr verfügbar, wird ein Null-Objekt zurückgegeben.

RPG-Sprachbindung: das Ergebnis wird als EBCDIC-UTF-8-String zurückgegeben.

---

```
C++ const unsigned char *pcos_get_stream(int doc, int *length, string optlist, wstring path)
C# Java final byte[] pcos_get_stream(int doc, String optlist, String path)
Perl PHP string pcos_get_stream(int doc, string optlist, string path)
C const unsigned char *TET_pcos_get_stream(TET *tet, int doc, int *length, const char *optlist,
    const char *path, ...)
```

---

Liefert den Inhalt eines pCOS-Pfades vom Typ `stream`, `fstream` oder `string`.

**doc** Gültiges Dokument-Handle, das mit `TET_open_document*()` erzeugt wurde.

**length** (Nur C- und C++-Sprachbindung) Zeiger auf eine Variable zur Speicherung der Länge der zurückgegebenen Streamdaten in Bytes.

**optlist** Optionsliste mit Optionen zur Abfrage von Streamdaten gemäß Tabelle 10.22.

**path** Vollständiger pCOS-Pfad für ein Stream- oder String-Objekt.

**Weitere Parameter** (Nur C-Sprachbindung) Es können ein oder mehrere zusätzliche Parameter übergeben werden, sofern der Parameter `key` entsprechende Platzhalter enthält (%s für Strings oder %d für Integers; %% wird für ein einzelnes Prozentzeichen verwendet). Diese Parameter ersparen es Ihnen, komplexe Pfade mit mehreren Zahlen oder Strings explizit zu formatieren. Der Client muss sicherstellen, dass Anzahl und Typ der Platzhalter mit den zusätzlich übergebenen Parametern übereinstimmen.

**Rückgabe** Die im Stream bzw. String enthaltenen unverschlüsselten Daten. Die zurückgegebenen Daten sind leer (in C und C++: NULL), wenn der Stream bzw. String leer ist oder wenn In-

halte verschlüsselter Anhänge in einem unverschlüsselten Dokument abgefragt werden und für die angehängten Dokumente kein Kennwort übergeben wurde.

Ist das Objekt vom Typ *stream*, werden eventuell vorhandene Filter entfernt (d.h. die eigentlichen Rohdaten werden zurückgegeben), sofern nicht *keepfilter=true*. Ist das Objekt vom Typ *fstream* oder *string*, werden die Daten so zurückgegeben, wie sie im PDF vorgefunden wurden, mit Ausnahme der Filter ASCII85 und ASCIIHex, die entfernt werden.

Zusätzlich zur Dekompression der Daten und zum Entfernen der ASCII-Filter kann Textkonvertierung gemäß der Option *convert* angewendet werden.

JPX-komprimierte Streams werden wie folgt behandelt: Bilddaten mit 1...8 Bit pro Komponente werden mit 8 Bit pro Komponente zurückgegeben; Bilddaten mit 9...16 Bit pro Komponente werden mit 16 Bit pro Komponente zurückgegeben. Wenn kein PDF-Farbraum vorhanden ist und der JPX-komprimierte Stream eine interne Farbpalette enthält, wird die Palette vor der Rückgabe des unkomprimierten Datenstreams angewendet, um sicherzustellen, daß die Pixeldaten zum zurückgegebenen Farbraum und der Anzahl von Komponenten passen. Beachten Sie, dass die Palette nicht angewendet wird, wenn der PDF-Farbraum *Indexed* vorhanden ist.

**Details** Diese Funktion löst eine Exception aus, wenn pCOS nicht im vollständigen Modus läuft (siehe die pCOS-Pfadreferenz). Eine Ausnahme bildet das Objekt */Root/Metadata*, das auch im eingeschränkten pCOS-Modus abrufbar ist, sofern *nocopy=false* oder *plainmetadata=true*. Eine Exception wird zudem ausgelöst, wenn *path* nicht auf ein Objekt vom Typ *stream*, *fstream* oder *string* zeigt.

Ungeachtet ihres Namens kann diese Funktion auch zur Abfrage von Objekten vom Typ *string* eingesetzt werden. Im Gegensatz zu *TET\_pcos\_get\_string()*, das das Objekt als Text-String behandelt, verändert diese Funktion die zurückgegebenen Daten in keiner Weise. Strings mit Binärdaten werden in PDF selten verwendet und lassen sich nicht zuverlässig automatisch erkennen. Benutzer müssen deshalb selbst darauf achten, bei der Abfrage von String-Objekten die für Binärdaten oder Text geeignete Funktion zu verwenden.

**Bindungen** COM: Client-Programme verwenden zur Speicherung des Stream-Inhalts meist den Typ Variant. Bei JavaScript mit COM ist es nicht möglich, die Länge des zurückgegebenen Varianten-Arrays abzufragen (bei anderen Sprachen mit COM ist dies aber möglich).

C- und C++-Sprachbindungen: Der zurückgegebene Datenpuffer kann bis zum nächsten Aufruf dieser Funktion verwendet werden.

Python: das Ergebnis wird als 8-Bit-String zurückgegeben (Python 3: *bytes*).

**Hinweis** Mit dieser Funktion lassen sich eingebettete Fonts aus PDF extrahieren. Beachten Sie, dass Fonts den Lizenzvereinbarungen der jeweiligen Hersteller unterliegen und ohne explizite Genehmigung des Rechteinhabers nicht weiterverwendet werden dürfen. Kontaktieren Sie gegebenenfalls den Anbieter des Fonts, um Lizenzfragen zu klären.

Tabelle 10.22 Option für `TET_pcos_get_stream()`

Option	Beschreibung
<b>convert</b>	<p>(Schlüsselwort; wird bei Streams ignoriert, die mit nicht unterstützten Filtern komprimiert sind) Steuert, ob die String- oder Stream-Inhalte konvertiert werden (Standardwert: none):</p> <p><b>none</b> Inhalte werden als Binärdaten ohne Konvertierung behandelt.</p> <p><b>unicode</b> Inhalte werden als Textdaten behandelt (d.h. genauso wie in <code>TET_pcos_get_string()</code>) und nach Unicode normalisiert. In nicht Unicode-fähigen Sprachbindungen werden Daten ins Format UTF-8 ohne BOM konvertiert. Diese Option wird für den selten verwendeten PDF-Datentyp »Text Stream« benötigt (er kann z.B. für JavaScript verwendet werden, obwohl der Großteil der JavaScripts in String- und nicht in Stream-Objekten enthalten ist).</p>
<b>keepfilter</b>	<p>(Boolean; nur empfohlen bei Streams mit Bilddaten; wird bei Streams ignoriert, die mit nicht unterstützten Filtern komprimiert sind) Bei <code>true</code> sind die Streamdaten mit dem im Pseudo-Objekt <code>filterinfo</code> des Bildes angegebenen Filter komprimiert (siehe die pCOS-Pfadreferenz). Bei <code>false</code> sind die Streamdaten nicht komprimiert. Standardwert: <code>true</code> bei nicht unterstützten Filtern, sonst <code>false</code></p>



# A TET-Kurzreferenz

Die folgende Tabelle enthält eine Übersicht über alle TET-API-Funktionen. Das Präfix (C) bedeutet C-Funktionsprototypen, die in der Java-Sprachbindung nicht verfügbar sind.

## Allgemeine Funktionen

<b>Funktionsprototyp</b>	<b>Seite</b>
(C) <i>TET *TET_new(void)</i>	182
<i>void delete()</i>	183

## Verarbeitung von Optionen

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>void set_option(String optlist)</i>	180

## PVF-Funktionen

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>void create_pvf(String filename, byte[] data, String optlist)</i>	183
<i>int delete_pvf(String filename)</i>	184
<i>int info_pvf(String filename, String keyword)</i>	184

## Funktionen zur Unicode-Konvertierung

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>String convert_to_unicode(String inputformat, byte[] input, String optlist)</i>	185

## Verarbeitung von Exceptions

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>String get_apiname()</i>	187
<i>String get_errmsg()</i>	187
<i>int get_errnum()</i>	187

## Dokumentfunktionen

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>int open_document(String filename, String optlist)</i>	190
(C) <i>int TET_open_document_callback(TET *tet, void *opaque, size_t filesize, size_t (*readproc)(void *opaque, void *buffer, size_t size), int (*seekproc)(void *opaque, long offset), const char *optlist)</i>	197
<i>void close_document(int doc)</i>	198

## Seitenfunktionen

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>int open_page(int doc, int pagenumber, String optlist)</i>	199
<i>void close_page(int page)</i>	208

## Funktionen zu Text- und Glyphendetails

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>String get_text(int page)</i>	209
<i>int get_char_info(int page)</i>	210

## Funktionen zur Bildabfrage

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>int get_image_info(int page)</i>	215
<i>int write_image_file(int doc, int imageid, String optlist)</i>	216
<i>final byte[] get_image_data(int doc, int imageid, String optlist)</i>	218

## Funktionen für TET Markup Language (TETML)

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>int process_page(int doc, int pagenumber, String optlist)</i>	219
<i>final byte[] get_image_data(int doc, int imageid, String optlist)</i>	218

## pCOS-Funktionen

<b>Funktionsprototyp</b>	<b>Seite</b>
<i>double pcos_get_number(int doc, String path)</i>	222
<i>String pcos_get_string(int doc, String path)</i>	222
<i>final byte[] pcos_get_stream(int doc, String optlist, String path)</i>	223

# B Änderungen an diesem Handbuch

## Änderungen an diesem Handbuch

<b>Datum</b>	<b>Änderung</b>
06. November 2015	▶ Deutsche Übersetzung des Handbuchs für TET 5.0
02. Februar 2015	▶ Deutsche Übersetzung des Handbuchs für TET 4.4
26. September 2014	▶ Deutsche Übersetzung des Handbuchs für TET 4.3
9. Januar 2006	▶ Deutsche Übersetzung des Handbuchs für TET 2.1.0 mit kleineren Korrekturen
14. Dezember 2005	▶ Erweiterungen und Korrekturen für TET 2.1.0; erweiterte Dokumentation für die PHP und RPG-Sprachbindungen
28. Juli 2005	▶ Deutsche Übersetzung des Handbuchs für TET 2.0.0 mit kleineren Erweiterungen und Korrekturen
20. Juni 2005	▶ Erweiterung und Umstrukturierung des Handbuchs für TET 2.0.0
14. Oktober 2003	▶ Aktualisierung des Handbuchs für TET 1.1
23. November 2002	▶ Hinzufügen der Beschreibung von <code>TET_open_doc_callback()</code> und eines Codebeispiels zur Bestimmung der Seitengröße für TET 1.0.2
4. April 2002	▶ Erste Ausgabe für TET 1



# Index

## A

Anmerkungen 81  
API-Referenz 173  
Arabisch 96  
Artefakte in Tagged PDF 201  
ascender 88

## B

Basic Multilingual Plane 108  
Beispiele  
    Textextraktion-Status 67  
    XSLT 169  
Benutzer-Kennwort 67  
Berechtigungskennwort 67  
Bereich der Textextraktion 84  
bidirektionaler Text 96  
Bilder  
    Anzahl in einem Dokument 135  
    Auflösung 138  
    Dateiformat ermitteln 132  
    Entfernen von kleinen Bildfragmenten 141  
    Extrahieren auf Festplatte oder  
    Arbeitsspeicher 131  
    Farbtreue 142  
    Geometrie 136  
    platzierte Bilder 134  
    Ressourcen 134  
    ressourcenbasierte Bildextraktion 136  
    seitenbasierte Extraktion 135  
    XMP-Metadaten 133  
    Zusammensetzen 139  
BMP 108  
Boolesche Werte in Optionslisten 177  
Byte Order Mark (BOM) 108

## C

C++ und .NET 37  
C++-Sprachbindung 32  
capheight 88  
CJK (Chinesisch, Japanisch, Koreanisch) 12, 93  
    Kompatibilitätsformen 94  
    Konfiguration 7  
    Wortgrenzen 93  
CLI 32  
Codeliste 126  
COM-Sprachbindung 34  
C-Sprachbindung 29  
CSV-Format 171

## D

Dateianhänge 82  
Dateisuche 71  
Dekomposition 116  
descender-Option 88  
DeviceN-Farbraum 142  
Dispose() 183  
Dokument- und Seitenfunktionen 190  
Dokumentdomänen 79  
Dokument-Infofelder 79  
Dokumentstile 103  
Double-Byte-Varianten 94

## E

Ebenen 83  
Einheiten 84  
Endpunkte von Glyphen und Wörtern 89  
Entfernen von kleinen Bildfragmenten 141  
Enttrennung 101  
EUDC-Fonts 123  
Evaluierungsversion 7

## F

Farbe des Textes 91  
Farbräume 142  
Filtern von Text 110  
Float- und Integer-Werte in Optionslisten 178  
Folding 112  
Font-Filterung (XSLT-Beispiel) 170  
FontReporter-Plugin 12, 124  
Font-Statistiken (XSLT-Beispiel) 170  
Formularfelder 81  
Füllfarbe des Textes 91

## G

Geometrie von Bildern 136  
geschützte Dokumente 67  
Glyphen 107  
Glyphen ohne Unicode-Werte 123  
Glyphenmetrik 85  
Glyphenregeln 128  
Glyphlisten 127  
Granularität 99

## H

Hebräisch 96  
Hervorhebung 89

HTML-Konverter (XSLT-Beispiel) 172

## I

ICC-Profil 142  
ideografischer Text: Wortgrenzen 93  
Filter für Microsoft-Produkte 59  
Index (XSLT-Beispiel) 171  
Inhaltsanalyse 99  
Installation von TET 7

## J

J2EE Application Servers 35  
Javadoc 36  
Java-Sprachbindung 35  
JBIG2 131  
JPEG 131  
JPEG 2000 131

## K

kanonische Dekomposition 116  
Kategorien von Ressourcen 70  
Kennwort für Dateianhänge 67  
Kennwörter 67  
Kommandozeilen-Tool 19  
Kommentare 81  
kommerzielle Lizenz 10  
Kompatibilitätsdekomposition 116  
Konkordanz (XSLT-Beispiel) 169  
Konnektor 49  
Koordinatensystem 84  
Kostenloses TET Plugin für Adobe Acrobat 49  
künstliche Fettschrift entfernen 101

## L

Layers 201  
Lesezeichen 81  
Ligaturen 109  
Linienfarbe des Textes 31  
Listenwerte in Optionslisten 174  
Lizenzschlüssel 8  
Logging 188  
Logo-Fonts 123  
Lucene-Suchmaschine 31

## M

Master-Kennwort 67  
MediaWiki 64  
Millimeter 84  
Minibeispiele 14

## N

Nachbearbeitung Unicode 110  
.NET-Sprachbindung 37  
Normalisierung 120

## O

Oberlänge 88  
Objective-C-Sprachbindung 38  
Optimierung der Verarbeitungsgeschwindigkeit 74  
Optionslisten 173  
    Syntax 174  
    verschachtelt 174  
Oracle Text 56

## P

Pakete 82  
pCOS  
    API-Funktionen 222  
    Cookbook 16  
PDF-Versionen 11  
Perl-Sprachbindung 40  
PHP-Sprachbindung 41  
platzierte Bilder 134  
Portfolios 82  
Private Use Area (PUA) 108  
PUA (Private Use Area) 108, 114, 123  
Punkt 84  
Python-Sprachbindung 43

## R

Rasterbilder  
    Extrahieren 131  
    Formate 131  
REALbasic/Xojo-Sprachbindung 44  
Rechtecke in Optionslisten 178  
resourcefile-Parameter 73  
Response-Datei 22  
ressourcenbasierte Schleife zur Bildextraktion 136  
Ressourcenkonfiguration 70  
Roadmap für Dokumentation und Beispiele 14  
Rohtext-Extraktion (XSLT-Beispiel) 172  
RPG-Sprachbindung 47  
Ruby-Sprachbindung 45

## S

Schattentext entfernen 101  
Schema 157  
Schlüsselwörter in Optionslisten 178  
Schmuckfarbe 142  
searchpath 71  
seitenbasierte Schleife zur Bildextraktion 135  
Seitenboxen 84  
Separation-Farbraum 142  
Sequenzen 109  
Servlets 35  
shrug-Feature 67  
Single-Byte-Varianten 94  
Solr-Suchmaschine 54  
Strings in Optionslisten 176

Suche nach verwendeten Fonts (XSLT-Beispiel) 170  
Surrogate 108  
Syntax für Optionslisten 174

## T

Tabellenerkennung 105  
Tabellen-Extraktion (XSLT-Beispiel) 171  
Tagged PDF 82, 201  
TET Cookbook 15  
TET Markup Language (TETML) 147  
TET\_CATCH() 187  
TET\_close\_document() 198  
TET\_close\_page() 208  
TET\_convert\_to\_unicode() 185  
TET\_create\_pvf() 183  
TET\_delete\_pvf() 184  
TET\_delete() 183  
TET\_EXIT\_TRY() 29, 187  
TET\_get\_apiname() 187  
TET\_get\_char\_info() 210  
TET\_get\_color\_info() 213  
TET\_get\_errmsg() 187  
TET\_get\_errnum() 187  
TET\_get\_image\_data() 218  
TET\_get\_image\_info() 215  
TET\_get\_tetml() 220  
TET\_get\_text() 209  
TET\_info\_pvf() 184  
TET\_new() 182  
TET\_open\_document\_callback() 197  
TET\_open\_document() 190  
TET\_open\_page() 199  
TET\_pcos\_get\_number() 222  
TET\_pcos\_get\_stream() 223  
TET\_pcos\_get\_string() 222  
TET\_RETHROW() 187  
TET\_set\_option() 180  
TET\_TRY() 187  
TET\_write\_image\_file() 216  
tet.upr 72  
TET-Funktionsumfang 11  
TET-Kommandozeilen-Tool 19  
TET-Konnektor 49  
    für Lucene 51  
    für MediaWiki 64  
    für Microsoft-Produkte 59  
    für Oracle 56  
    für Solr 54  
    für Tika 62  
TETML 147  
    Schema 157  
TETRESOURCEFILE-Umgebungsvariable 72  
TeX-Dokumente 77  
Textextraktion-Status 67  
Textfarbe 91  
TIFF 131  
Tika-Toolkit 62  
ToUnicode-CMap 127

## U

Umrisslinien zeichnen 213  
Unichar-Werte in Optionslisten 176  
Unicode  
    BOM 108  
    Dekomposition 116  
    Encoding-Formen 108  
    Encoding-Schemata 108  
    Folding 112  
    in Optionslisten 176  
    Konzepte 107  
    Mengen 177  
    Nachbearbeitung 110, 112  
    Normalform 120  
    Vorbereitung 110  
unsichtbarer Text 213  
Unterlänge 88  
UPR-Dateiformat 70  
UTF-32 122  
UTF-Formate 108

## V

Varianten mit halber Breite 94  
Varianten mit voller Breite 94  
Verarbeitung von Exceptions 27  
    in C 29  
Verarbeitungsgeschwindigkeit optimieren 74  
Versalhöhe 88  
verschachtelte Optionslisten 174  
vertikale Textausgabe 93  
Vorbereitung Unicode 110  
vorrotierte Glyphen 94

## W

Wordfinder 100  
Wortgrenzen-Erkennung 100

## X

XFA-Formulare 162  
xheight 88  
XMP-Metadaten 80  
    für Bilder 133  
    XSLT-Beispiel 171  
Xojo-Sprachbindung 44  
XSD-Schema für TETML 157  
XSLT 165  
    Beispiele 15, 169

## Z

Zahlen in Optionslisten 178  
Zeichen und Glyphen 107  
Zoll 84  
zusammengesetzte Zeichen 109

**PDFlib GmbH**

Franziska-Bilek-Weg 9  
D-80339 München  
www.pdflib.com  
Tel. +49 • 89 • 452 33 84-0  
Fax +49 • 89 • 452 33 84-99

Bei Fragen können Sie die PDFlib-Mailing-Liste abonnieren  
und sich deren Archiv ansehen unter [groups.yahoo.com/neo/groups/pdflib/info](http://groups.yahoo.com/neo/groups/pdflib/info)

**Vertriebsinformationen**

[sales@pdflib.com](mailto:sales@pdflib.com)

**Support**

[support@pdflib.com](mailto:support@pdflib.com) (*geben Sie bitte immer Ihre Lizenznummer an*)

