PDFlib GmbH

pCOS-Pfadreferenz

PDF Information Retrieval Tool

pCOS-Schnittstelle Version 9

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.

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

Inhaltsverzeichnis

Index *41*

1 Einführung

1.1 Was ist pCOS?

Die pCOS-Schnittstelle *(PDFlib Comprehensive Object Syntax)* bietet eine einfache und elegante Lösung für das Abrufen von technischen Informationen aus allen Bereichen eines PDF-Dokuments, mit Ausnahme der Seiteninhalte, wie zum Beispiel Seitengröße, Metadaten, interaktive Elemente usw. pCOS-Benutzer sollten über Grundkenntnisse von internen PDF-Strukturen und Dictionary-Schlüsseln verfügen, müssen sich aber nicht mit PDF-Syntax und Parsing-Details befassen. pCOS-Benutzer sollten sich unbedingt ein Exemplar der *PDF Reference* besorgen. Seit der Standardisierung von PDF 1.7 im Jahr 2008 ist die PDF Reference als ISO 32000-1 verfügbar. Dieses Dokument kann über *www.iso.org* bezogen werden. Wenn Sie die offizielle Version nicht erwerben wollen, können Sie eine kostenlose Ausgabe herunterladen, die inhaltlich identisch ist:

Document Management – Portable Document Format – Part 1: PDF 1.7 First Edition

Das PDF-Dokument kann hier heruntergeladen werden: *www.adobe.com/devnet/pdf/pdf_reference.html*.

1.2 Roadmap für Dokumentation und Beispiele

Unten aufgeführte Materialien stehen Ihnen zum erfolgreichen Einsatz von pCOS zur Verfügung.

Minibeispiel für alle Sprachbindungen. Das Minibeispiel *dumper* wird in allen Paketen und für alle Sprachbindungen mitgeliefert. Es enthält kurzen Beispielcode für die Verwendung von pCOS. Das Minibeispiel dient in erster Linie zum Testen Ihrer pCOS-Installation und für eine schnelle Übersicht über die Programmierung von pCOS-Anwendungen.

pCOS-Pfadreferenz. Die vorliegende *pCOS-Pfadreferenz* enthält Beispiele und eine genaue Beschreibung der pCOS-Pfadsyntax, die den Kern der pCOS-Schnittstelle bildet. Da die pCOS-Schnittstelle auch in weiteren Produkten von PDFlib GmbH enthalten ist, können Sie die pCOS-Pfadreferenz mit allen Produkten verwenden, die pCOS enthalten.

Zugehöriges Produkthandbuch. Die pCOS-Schnittstelle steht als Einzelprodukt oder als integrierter Teil von anderen Produkten von PDFlib GmbH zur Verfügung. Jedes Produkt wird mit einem oder mehreren projektspezifischen Handbüchern ausgeliefert, die die Verwendung der jeweiligen Programmbibliothek (z.B. pCOS oder TET) beschreiben sowie das entsprechende Kommandozeilen-Tool, falls vorhanden. Das Produkthandbuch deckt die verschiedenen, von einem Produkt unterstützten Programmiersprachen ab und beschreibt ausführlich die entsprechenden Funktionen.

pCOS Cookbook. Das *pCOS Cookbook* ist eine Sammlung von Codefragmenten für die pCOS-Schnittstelle. Es steht unter folgender Adresse zur Verfügung:

```
www.pdflib.com/pcos-cookbook/
```

Das pCOS Cookbook beschreibt die Verwendung von pCOS für verschiedene Applikationen. Studium des Cookbooks wird dringend empfohlen, da es eine Sammlung von nützlichen pCOS-Programmierbeispielen enthält.

1.3 Verfügbarkeit der pCOS-Schnittstelle

Die pCOS-Schnittstelle steht als separates Produkt unter dem Namen PDFlib pCOS zur Verfügung. Sie wird zudem auch als integriertes Feature in weiteren Produkten von PDFlib GmbH angeboten. Mit der Erweiterung der Schnittstelle und der Unterstützung von neueren Versionen der PDF-Eingabe wird die Versionsnummer der pCOS-Schnittstelle angehoben. Tabelle 1.1 führt die Versionsnummern der pCOS-Schnittstelle auf, die in den verschiedenen Produktversionen implementiert sind.

Tabelle 1.1 In den Produkten von PDFlib GmbH implementierte Version der pCOS-Schnittstelle

pCOS-Schnittstelle	***unterstützte Version der PDF-Eingabe / entsprechende Acrobat-Version***	***PDFlib GmbH Produktname und Version***
1	*PDF 1.6 / Acrobat 7*	*TET 2.0, 2.1*
2	*PDF 1.6 / Acrobat 7*	*pCOS 1.0*
3	*PDF 1.7 / Acrobat 8* *Identisch mit ISO 32000-1*	*PDFlib+PDI 7, PPS 7* *TET 2.2, pCOS 2.0, PLOP 3.0, TET 2.3*
4	*PDF 1.7 extension level 3 / Acrobat 9 außer AES-256-Verschlüsselung*	*PLOP 4.0, TET 3.0, TET PDF IFilter 3.0*
5	*PDF 1.7 extension level 3 / Acrobat 9*	*PDFlib+PDI 8, PPS 8*
6	*PDF 1.7 extension level 3 / Acrobat 9*	*TET 4.0, TET PDF IFilter 4.0*
7	*PDF 1.7 extension level 8 / Acrobat X* *Syntax und Verschlüsselungsmethoden sind identisch mit ISO 32000-2, auch PDF 2.0 genannt.*	*pCOS 3.0, PLOP 4.1,* *PDFlib+PDI 8,1, PPS 8,1*
8	*PDF 1.7 extension level 8 / Acrobat X/XI* *Syntax und Verschlüsselungsmethoden sind identisch mit ISO 32000-2, auch PDF 2.0 genannt.*	*TET 4.1+, TET PDF IFilter 4.1+* *PDFlib+PDI 9.0, PPS 9.0* *pCOS 4.0*
9	*PDF 1.7 extension level 8 / Acrobat X/XI* *Syntax und Verschlüsselungsmethoden sind identisch mit ISO 32000-2, auch PDF 2.0 genannt.*	*PLOP 5.0, PLOP DS 5.0*

Einige Features der pCOS-Schnittstelle stehen nur bei TET zur Verfügung. Auf diese wird im Handbuch explizit hingewiesen.

2 pCOS-Beispiele

Dieses Kapitel enthält Beispiele für pCOS-Pfade, mit denen die entsprechenden Werte aus PDF-Dokumenten abgefragt werden können. Weitere Beispiele, die zusätzliche Programmlogik erfordern, finden Sie im pCOS Cookbook auf der PDFlib-Website.

Sofern nicht anders angegeben, sind alle Programmierbeispiele in der Sprache Java abgefasst. Wenn Sie die Beispiele entsprechend anpassen (im wesentlichen nur die Syntax), können sie aber mit allen von pCOS unterstützten Programmiersprachen verwendet werden.

Die in diesem Kapitel aufgeführten Beispiele stellen nur eine Auswahl dar. Durch die Verwendung anderer PDF-Objekte sind viele weitere pCOS-Anwendungen möglich.

2.1 pCOS-Funktionen

Allgemeine pCOS-Funktionen. Folgende Funktionen stehen für die Abfrage von PDF-Dokumenten mit pCOS zur Verfügung:

- *pcos_get_number( )* Abfrage der Objekttypen *number* oder *boolean;*
- *pcos_get_string( )* Abfrage der Objekttypen *name, number, string* oder *boolean;*
- *pcos_get_stream( )* Abfrage der Objekttypen *stream, fstream* oder *string*.

Mit diesen Funktionen lassen sich Informationen mit Hilfe der pCOS-Pfadsyntax aus einem PDF-Dokument abfragen. Die allgemeine Struktur einer pCOS-Anwendung sieht folgendermaßen aus:

```
/* PDF-Dokument öffnen */
int doc = p.open_document(filename, "");
if (doc == -1)
        throw new Exception("Error: " + p.get_errmsg());

/* Wert eines pCOS-Pseudo-Objekts abfragen */
System.out.println(" PDF version: " + p.pcos_get_string(doc, "pdfversionstring"));

p.close_document(doc);
```

Die Parameter für die pCOS-Funktionen sind in allen Produkten gleich. Sie sind in den Referenzhandbüchern der entsprechenden Produkte dokumentiert; für einen schnellen Überblick über die pCOS-Funktionsaufrufe siehe Anhang A, »pCOS-Funktionsreferenz«.

Programmlogik hinzufügen. Viele pCOS-Objekte bestehen aus Arrays einer bestimmten Länge. Die Länge kann mit dem Präfix *length:* abgefragt werden. Das Array kann dann mit ganzzahligen Werten im Bereich von 0 bis *length-1* indiziert werden. Mit dem folgenden Code lässt sich die Anzahl der Fonts in einem Dokument abfragen sowie Typ und Name der einzelnen Fonts ausgeben:

```
count = (int) p.pcos_get_number(doc, "length:fonts");

for (i = 0; i < count; i++) {
    String fonts;

    System.out.print(p.pcos_get_string(doc, "fonts[" + i + "]/type") + " font ");
```

```
    System.out.println(p.pcos_get_string(doc, fonts[" + i + "]/name));
}
```

Formatieren von Platzhaltern in C. Die C-Sprachbindung bietet eine komfortable Funktion zur einfachen Verwendung von Parametern innerhalb eines pCOS-Pfades. Analog zu den Formatierungsparametern der Funktionsfamilie *printf( )* können Sie die jeweiligen Platzhalter *%s* und *%d* für die Parameter *string* und *integer* verwenden. Die Werte dieser Parameter müssen als zusätzliche Funktionsparameter nach dem pCOS-Pfad angegeben werden. pCOS ersetzt die Platzhalter dann durch die tatsächlichen Werte. Diese Funktion ist besonders nützlich für Pfade mit Array-Indizes.

So kann beispielsweise das obige Java-Idiom zur Auflistung aller Fonts in C wie folgt geschrieben werden:

```
count = (int) PDF_pcos_get_number(p, doc, "length:fonts");

for (i = 0; i < count; i++)
{
    printf("%s font ", PDF_pcos_get_string(p, doc, "fonts[%d]/type", i));
    printf("%s\n", PDF_pcos_get_string(p, doc, "fonts[%d]/name", i));
}
```

Da moderne Programmiersprachen leistungsfähige Funktionen zur Verarbeitung von Strings bieten, ist diese Funktion nur in der C-Sprachbindung, aber nicht in anderen Sprachbindungen verfügbar.

2.2 Dokument

In Tabelle 2.1 werden pCOS-Pfade für allgemeine und dokumentbezogene Objekte aufgeführt.

Tabelle 2.1 pCOS-Pfade für dokumentbezogene Elemente

pCOS-Pfad	*Typ*	*Beschreibung*
pcosmode	*number*	*pCOS-Modus des Dokuments, d.h. sein Verschlüsselungsstatus (siehe Abschnitt 4.9, »Verschlüsselte PDF-Dokumente und pCOS-Modus«, Seite 36)*
pdfversionstring	*string*	*String mit der PDF-Versionsnummer des Dokuments*
/Info/Title	*string*	*Dokument-Infofeld* `Title`; *Die folgenden Feldnamen sind in PDF vordefiniert und können auf ähnliche Art verwendet werden:* `Title, Author, Subject, Keywords, Creator, Producer, CreationDate, ModDate, Trapped`
/Info/ArticleNumber	*string*	*benutzerdefiniertes Dokument-Infofeld* `ArticleNumber` *(Dokument-Infofelder können beliebige Namen verwenden)*
/Root/Metadata	*stream*	*XMP-Stream mit den Dokument-Metadaten*
pdfa, pdfe, pdfua, pdfvt, pdfx	*string*	*Status der Konformität zum Standard PDF/A, PDF/E, PDF/UA, PDF/VT oder PDF/X*

Status der Verschlüsselung und pCOS-Modus. Den pCOS-Modus des Dokuments können Sie mit dem Pseudo-Objekt *pcosmode* abfragen. Damit lassen sich Exceptions vermeiden, wenn später Informationen abgerufen werden sollen, für die kein Zugriff gewährt wird (z.B. weil das Dokument verschlüsselt ist und kein Kennwort angegeben wurde). Für alle pCOS-Anwendungen wird die folgende allgemeine Struktur empfohlen, die auf den Werten von *pcosmode* basiert:

```
/* PDF-Dokument öffnen */
int doc = p.open_document(filename, "requiredmode=minimum");
if (doc == -1)
        throw new Exception("Error: " + p.get_errmsg());

int pcosmode = (int) p.pcos_get_number(doc, "pcosmode");
boolean plainmetadata = p.pcos_get_number(doc, "encrypt/plainmetadata") != 0;

// Immer vorhandene, universale Pseudo-Objekte abfragen
System.out.println(" PDF version: " + p.pcos_get_string(doc, "pdfversionstring"));
System.out.println("  Encryption: " + p.pcos_get_string(doc, "encrypt/description"));

// verschlüsseltes Dokument, aber Kennwort ist nicht vorhanden
if (pcosmode == 0)
{
    System.out.println("Minimaler Modus: keine Informationen mehr vorhanden\n");
    p.delete();
    return;
}

// andernfalls Abfrage weiterer Informationen
System.out.println("PDF/A status: " + p.pcos_get_string(doc, "pdfa"));

// kein Master-Kennwort angegeben; Metadaten können nicht abgefragt werden
```

```
if (pcosmode == 1 && !plainmetadata && p.pcos_get_number(doc, "encrypt/nocopy") != 0)
{
    System.out.println("Eingeschränkter Modus: keine Informationen mehr vorhanden\n");
    p.delete();
    return;
}

// andernfalls Abfrage von Dokument-Infofeldern und XMP-Metadaten
...

p.close_document(doc);
```

PDF-Version. Mit dem folgenden Codefragment lässt sich die PDF-Versionsnummer eines Dokuments ausgeben:

```
System.out.println(" PDF version: " + p.pcos_get_string(doc, "pdfversionstring"));
```

Dokument-Infofelder. Dokument-Infofelder lassen sich mit der folgenden Codesequenz abfragen. Um sicherzustellen, dass ein Objekt im PDF-Dokument tatsächlich existiert und dem erwarteten Typ entspricht, wird zuerst der Typ ausgewertet. Ist das Objekt vorhanden und vom Typ *string*, kann es abgefragt werden:

```
objtype = p.pcos_get_string(doc, "type:/Info/Title");
if (objtype.equals("string"))
{
        /* Dokument-Infoschlüssel gefunden */
        title = p.pcos_get_string(doc, "/Info/Title");
}
```

XMP-Metadaten. Ein Stream mit XMP-Metadaten lässt sich mit folgender Codesequenz abfragen:

```
objtype = p.pcos_get_string(doc, "type:/Root/Metadata");
if (objtype.equals("stream"))
{
        /* XMP-Metadaten gefunden */
        metadata = p.pcos_get_stream(doc, "", "/Root/Metadata");
}
```

PDF-Standards. Der Status der Konformität zu den Standards PDF/A, PDF/E, PDF/UA, PDF/VT oder PDF/X lässt sich mit einfachen pCOS-Pseudo-Objekten folgendermaßen abfragen:

```
System.out.println("PDF/A status: " + p.pcos_get_string(doc, "pdfa"));
System.out.println("PDF/E status: " + p.pcos_get_string(doc, "pdfe"));
System.out.println("PDF/UA status: " + p.pcos_get_string(doc, "pdfua"));
System.out.println("PDF/VT status: " + p.pcos_get_string(doc, "pdfvt"));
System.out.println("PDF/X status: " + p.pcos_get_string(doc, "pdfx"));
```

2.3 Seiten

In Tabelle 2.2 werden pCOS-Pfade für seitenbezogene Objekte aufgeführt.

Tabelle 2.2 pCOS-Pfade für seitenbezogene Objekte

pCOS-Pfad	*Typ*	*Beschreibung*
length:pages	*number*	*Anzahl der Seiten im Dokument*
pages[...]/width ***pages[...]/height***	*number*	*Breite und Höhe der im Array indizierten Seite (beachten Sie, dass Array-Indizes null-basiert sind)*

Anzahl der Seiten. Die Gesamtzahl der Seiten in einem Dokument lässt sich folgendermaßen abfragen:

```
pagecount = p.pcos_get_number(doc, "length:pages");
```

Seitengröße. Obwohl sich die Seiteneinträge *MediaBox*, *CropBox* und *Rotate* direkt in pCOS abfragen lassen, müssen sie zusammen ausgewertet werden, um die eigentliche Seitengröße zu ermitteln. Die Seitengröße lässt sich viel leichter mit den Schlüsselwörtern *width* und *height* des Pseudo-Objekts *pages* ermitteln. Mit dem folgenden Codefragment lässt sich die Breite und Höhe von Seite 3 ermitteln (beachten Sie, dass die Indizes für das Pseudo-Objekt *pages* bei 0 beginnen):

```
pagenum = 2;        // page 3 (0-based)
width = p.pcos_get_number(doc, "pages[" + pagenum + "]/width");
height = p.pcos_get_number(doc, "pages[" + pagenum + "]/height");
```

Transparenz. Die Seitentransparenz kann für den Druck und andere Prozesse relevant sein. Mit dem Schlüsselwort *usespagetransparency* des Pseudo-Objekts *pages* können Sie Seiten mit transparenten Elementen ermitteln:

```
pagenum = 0;        // page 1 (0-based)
if (p.pcos_get_number(doc, "pages[" + pagenum + "]/usespagetransparency"))
{
        ...Seite mit transparenten Elementen...
}
```

2.4 Fonts

In Tabelle 2.3 werden pCOS-Pfade für fontbezogene Objekte aufgeführt.

Tabelle 2.3 pCOS-Pfade für fontbezogene Objekte

pCOS-Pfad	Typ	Beschreibung
length:fonts	*number*	*Anzahl der Fonts im Dokument*
fonts[...]/name	*string*	*Fontname*
fonts[...]/vertical	*boolean*	*Font auf vertikale Schreibrichtung prüfen*
fonts[...]/embedded	*boolean*	*Status der Font-Einbettung*
fonts[...]/ascender ***fonts[...]/descender***	*number*	*Wert für Oberlänge/Unterlänge eines Fonts (nicht immer verfügbar, siehe Codebeispiel unten)*

Liste aller Fonts. Mit der folgenden Codesequenz kann eine Liste aller Fonts in einem Dokument zusammen mit dem Status ihrer Einbettung erzeugt werden:

```
count = p.pcos_get_number(doc, "length:fonts");
for (i=0; i < count; i++)
{
        fontname = p.pcos_get_string(doc, "fonts[" + i + "]/name");
        embedded = p.pcos_get_number(doc, "fonts[" + i + "]/embedded");
        /* ... */
}
```

Schreibrichtung. Das folgende Codefragment prüft, ob ein Font vertikale Schreibrichtung unterstützt: Der Font wird über seine ID identifiziert, d.h. über den Index im Array *fonts*. Diese *id* kann durch Aufzählen aller möglichen Indexwerte ermittelt werden:

```
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;
        }
}
```

TET Das Produkt TET liefert Font-IDs auch mit der Funktion get_char_info( ).

Fontmetriken. In PDF können Fonts ein Font-Deskriptor-Dictionary mit Metrikwerten und anderen Fontinformationen enthalten.

```
count = p.pcos_get_number(doc, "length:fonts");
for (i=0; i < count; i++)
{
        ascender = p.pcos_get_number(doc, "fonts[" + i + "]/ascender");
        descender = p.pcos_get_number(doc, "fonts[" + i + "]/descender");
        /* ... */
}
```

2.5 Rasterbilder

In Tabelle 2.4 werden pCOS-Pfade für Rasterbild-Objekte aufgeführt.

Tabelle 2.4 pCOS-Pfade für Rasterbild-Objekte

pCOS-Pfad	*Typ*	*Beschreibung*
length:images	*number*	*Anzahl der Rasterbilder im Dokument*
images[...]/Width	*number*	*Bildbreite in Pixeln*
images[...]/Height	*number*	*Bildhöhe in Pixeln*

Liste aller Rasterbilder. Genau wie bei Fonts können Sie auch eine Liste aller Rasterbilder im Dokument erzeugen:

```
count = p.pcos_get_number(doc, "length:images");
for (i=0; i < count; i++)
{
        width = p.pcos_get_string(doc, "images[" + i + "]/Width");
        height = p.pcos_get_number(doc, "images[" + i + "]/Height");
        bpc = p.pcos_get_number(doc, "images[" + i + "]/bpc");
}
```

2.6 Interaktive Elemente

In Tabelle 2.5 werden pCOS-Pfade für Objekte zur Beschreibung interaktiver Elemente aufgeführt.

Tabelle 2.5 pCOS-Pfade für verschiedene PDF-Objekte

pCOS-Pfad	Typ	Beschreibung
length:bookmarks	number	Anzahl der Lesezeichen im Dokument
bookmarks[...]/Title	string	Lesezeichen-Text
bookmarks[...]/destpage	number	Seitenzahl der Zielseite bei aktiviertem Lesezeichen oder -1, wenn das Lesezeichen auf keine Seite im Dokument springt
pages[...]/annots[...]/A/URI	string	Ziel-URL des Weblinks auf allen Seiten
length:fields	number	Anzahl der Formularfelder im Dokument

Lesezeichen. Mit dem folgenden Codefragment lassen sich die Lesezeichen im Dokument abfragen. Für jedes Lesezeichen wird seine Verschachtelungsebene, die Zielseite und der Titel angezeigt:

```
int count = (int) p.get_number(doc, "length:bookmarks");

for (int i = 0; i < count; ++i) {
    int level    = (int) p.get_number(doc, "bookmarks[" + i + "]/level");
    int destpage = (int) p.get_number(doc, "bookmarks[" + i + "]/destpage");

    for (int j = 0; j < level * 4; j += 1) {
        System.out.print(" ");
    }

    System.out.print(p.get_string(doc, "bookmarks[" + i + "]/Title"));

    if (destpage != -1) {
        System.out.print(": page " + destpage);
    }
}
```

3 pCOS-Datentypen

3.1 PDF-Basisdatentypen

pCOS bietet die drei Funktionen *pcos_get_number( )*, *pcos_get_string( )* und *pcos_get_stream( )*. Damit lassen sich alle Datentypen abfragen, die in PDF-Dokumenten vorkommen. Den Datentyp eines bestimmten Objekts in PDF können Sie mit Hilfe der PDF-Referenz herausfinden.

Zahlen. Objekte vom Typ *integer* und *real* lassen sich mit *pcos_get_number( )* abfragen. pCOS unterscheidet nicht zwischen Ganzzahlen und Fließkommazahlen. Zum Beispiel:

```
/* Anzahl der Seiten im Dokument ermitteln */
int n_pages = (int) p.pcos_get_number(doc, "length:pages");
```

Namen und Strings. Objekte vom Typ *name* und *string* lassen sich mit *pcos_get_string( )* abfragen. Zum Beispiel:

```
string title = p.pcos_get_string(doc, "/Info/Title");
```

In PDF können Objekte vom Typ *name* Nicht-ASCII-Zeichen und die Syntax *#xx* enthalten (hexadezimaler Wert mit Präfix), um bestimmte Sonderzeichen aufzunehmen. pCOS verarbeitet PDF-Namen folgendermaßen:

- Von Objekten des Typs *name* wird die Dekoration entfernt, bevor sie zurückgegeben werden (das heißt, die Syntax *#xx* wird aufgelöst).
- Objekte vom Typ *name* werden in den meisten Sprachbindungen als Unicode-Strings zurückgegeben. In der C-Sprachbindung werden sie allerdings als UTF-8-Werte ohne BOM zurückgegeben.

Da es sich bei den meisten Strings in PDF um Text-Strings handelt, verarbeitet *pcos_get_string( )* sie als solche. In seltenen Fällen enthalten Strings in PDF jedoch binäre Informationen. In diesem Fall sollten Sie Strings mit der Funktion *pcos_get_stream( )* abfragen, wodurch Binär-Strings erhalten und Inhalte unverändert bleiben. Zum Beispiel:

```
byte[] signature = p.pcos_get_stream(doc, "", "fields[0]/V/Contents");
```

Booleans. Objekte vom Typ *boolean* können mit *pcos_get_number( )* abgefragt werden und werden als 1 *(true)* oder 0 *(false)* zurückgegeben. Zum Beispiel:

```
int linearized_i = p.pcos_get_number(doc, "linearized");
```

Mit *pcos_get_string( )* lassen sich auch Boolesche Objekte abfragen; in diesem Fall werden sie als einer der Strings *true* und *false* zurückgegeben. Zum Beispiel:

```
string linearized_s = p.pcos_get_string(doc, "linearized");
```

Streams. Objekte vom Typ *stream* lassen sich mit *pcos_get_stream( )* abfragen. Zum Beispiel:

```
byte[] contents = p.pcos_get_stream(doc, "", "/Root/Metadata");
```

In PDF können Stream-Daten mit einem oder mehreren Kompressionsfiltern vorverarbeitet werden. Je nach pCOS-Datentyp *(stream* oder *fstream)* sind die Inhalte komprimiert oder nicht komprimiert. Mit der Option *keepfilter* von *pcos_get_stream( )* lassen sich komprimierte Daten auch für den Typ *stream* abfragen.

Die Liste der im Stream vorhandenen Filter kann vom Stream-Dictionary abgefragt werden; für Rasterbilder lässt sich auf diese Information leichter über das Bild-Dictionary *filterinfo* zugreifen. Enthält die Filterkette eines Streams nur unterstützte Filter, ist sie vom Typ *stream*. Bei der Abfrage eines Objekts vom Typ *stream* entfernt *pcos_get_stream( )* alle Filter und gibt die jeweiligen ungefilterten Daten zurück.

Hinweis pCOS unterstützt keine Stream-Filter vom Typ JBIG2 und JPX.

Enthält die Filterkette eines Streams mindestens einen nicht unterstützten Filter, ist der Objekttyp ein *fstream* (gefilterter Stream). Bei der Abfrage der Inhalte eines Objekts vom Typ *fstream* entfernt *pcos_get_stream( )* die unterstützten Filter am Anfang einer Filterkette, behält aber die nicht unterstützten Filter und gibt die Stream-Daten mit den immer noch angewendeten nicht unterstützten Filtern zurück. Eine Liste aller angewendeten Filter kann vom Stream-Dictionary abgefragt und die gefilterten Stream-Inhalte können mit *pcos_get_stream( )* abgerufen werden. Beachten Sie, dass die Namen unterstützter Filter bei der Abfrage von Namen der Stream-Filter nicht entfernt werden, deshalb sollte der Client sie ignorieren.

PDF-Streams enthalten in der Regel Binärdaten. In seltenen Fällen jedoch (Text-Streams) können sie stattdessen Text enthalten (z.B. JavaScript-Streams). Um die passende Textkonvertierung anzustoßen, verwenden Sie die Option *convert=unicode* von *pcos_get_stream( )*.

3.2 Zusammengesetzte Datenstrukturen

Objekte mit einem der Basisdatentypen können in zwei verschiedenen zusammengesetzten Datenstrukturen gruppiert werden: als Arrays und Dictionaries. pCOS bietet keine speziellen Funktionen für die Abfrage zusammengesetzter Objekte. Stattdessen können die in einem Dictionary oder Array enthaltenen Objekte einzeln adressiert und abgefragt werden.

Arrays. Arrays sind eindimensionale Sammlungen einer beliebigen Anzahl von Objekten beliebigen Typs. Da ein Array auch verschachtelte Arrays enthalten kann, können auch mehrdimensionale Datenstrukturen dargestellt werden.

Die Inhalte eines Arrays können durch die Abfrage der Anzahl *N* seiner Elemente (unter Angabe des Präfix~~es~~ *length* vor dem Array-Pfad) und dann über die Iteration aller Elemente vom Index 0 bis *N-1* aufgelistet werden.

Dictionaries. Dictionaries (auch assoziative Arrays genannt) enthalten eine beliebige Anzahl von Objektpaaren. Das erste Objekt in jedem Paar ist vom Typ *name* und wird als Schlüsselwort bezeichnet. Das zweite Objekt wird als Wert bezeichnet und kann von beliebigem Typ außer *null* sein.

Die Inhalte eines Dictionarys können durch die Abfrage der Anzahl *N* seiner Elementen (unter Angabe des Präfix *length* vor dem Dictionary-Pfad) und dann über die Iteration aller Elemente vom Index 0 bis *N-1* aufgelistet werden. Das Auflisten von Dictionaries führt alle Dictionary-Schlüssel mit Hilfe des Suffix *.key* am Ende des Dictionary-Pfades in der Reihenfolge auf, in der sie in der PDF-Datei gespeichert sind. Auf ähnliche Weise lassen sich die entsprechenden Werte mit dem Suffix *.val* auflisten. Vererbte Werte (siehe unten) und Pseudo-Objekte sind beim Auflisten der Dictionary-Schlüssel nicht sichtbar und werden im *length*-Count nicht berücksichtigt.

Einige seitenbezogene Dictionary-Einträge in PDF können über eine baumartige Datenstruktur vererbt werden, was die Abfrage erschwert. Zum Beispiel ist die *MediaBox* für eine Seite nicht immer im Seiten-Dictionary enthalten, sie kann aber von einem beliebig komplexen Seitenbaum vererbt werden. pCOS löst dieses Problem durch transparentes Einfügen alle vererbten Schlüssel und Werte in die Dictionaries im Pseudo-Objekt *pages[ ].* pCOS-Benutzer können also davon ausgehen, dass alle vererbbaren Einträge direkt im Dictionary verfügbar sind und müssen nicht in allen relevanten übergeordneten Elementen eines Baumes suchen. Das Zusammenführen von vererbten Einträgen ist nur möglich, wenn auf den Seitenbaum über das Pseudo-Objekt *pages[ ]* zugegriffen wird; der Zugriff auf den */Pages*-Baum, das Pseudo-Objekt *objects[ ]* oder das Auflisten der Schlüssel über *pages[ ][ ]* gibt nur die im jeweiligen Dictionary vorhandenen Einträge ohne Vererbung zurück.

Auslesen von Dictionary-Einträgen. Im folgenden Beispiel werden die Schlüssel-/Wert-Paare in der Dokument-Info des Dictionarys aufgelistet:

```
count = (int) p.pcos_get_number(doc, "length:/Info");

for (i = 0; i < count; i++) {
        String info;
        String key;

        info = "type:/Info[" + i + "]";
        objtype = p.pcos_get_string(doc, info);
```

```
            info = "/Info[" + i + "].key";
            key = p.pcos_get_string(doc, info);
            System.out.print(key + ": ");

            /* Infofelder können als Objekte vom Typ string oder name gespeichert werden */
            if (objtype.equals("name") || objtype.equals("string"))
            {
                info = "/Info[" + i + "]";
                System.out.println("'" + p.pcos_get_string(doc, info) + "'");
            }
        }
```

3.3 Object Identifiers (IDs)

pCOS-IDs für Dictionaries und Arrays. Im Gegensatz zu PDF-Objekt-IDs liefern pCOS-IDs einen eindeutigen Identifier für ein über einen pCOS-Pfad angesprochenes Element (da Arrays und Dictionaries verschachtelt werden können, kann ein Objekt die gleiche PDF-Objekt-ID wie sein übergeordneter Array oder Dictionary haben). pCOS-IDs lassen sich mit dem Präfix *pcosid* vor dem Dictionary- oder Array-Pfad abfragen.

Die pCOS-ID kann daher statt der expliziten Pfad-Adressierung als Shortcut für den wiederholten Zugriff auf Elemente verwendet werden Dies erhöht die Geschwindigkeit, wenn eine Schleife über alle Elemente eines großen Arrays durchlaufen wird. Mit dem Pseudo-Objekt *objects[]* lässt sich der Inhalt eines mit einer bestimmten ID identifizierten Elements abfragen.

4 pCOS-Pfadreferenz

4.1 pCOS-Pfadsyntax

Den Kern der pCOS-Schnittstelle bildet eine einfache Pfadsyntax zur Adressierung und Abfrage beliebiger Objekte in einem PDF-Dokument. Außer den Objektdaten selbst kann pCOS weitere Informationen über das Objekt, wie seinen Typ oder die Länge liefern. Je nach Objekttyp (der selbst auch abgefragt werden kann) kann mit einer der Funktionen *pcos_get_number( ), pcos_get_string( )* und *pcos_get_stream( )* der Wert eines Objekts abgefragt werden. Die allgemeine Syntax für pCOS-Pfade sieht folgendermaßen aus:

```
[<prefix>:][pseudoname[<index>]]/<name>[<index>]/<name>[<index>] ... [.key|.val]
```

Die Pfadkomponenten haben folgende Bedeutung:

- Das optionale *prefix* kann verschiedene Werte haben, siehe Tabelle 4.1.
- Der optionale *pseudo object name* kann den Namen des Pseudo-Objekts enthalten. Pseudo-Objekte sind bei PDF nicht vorhanden, werden aber in pCOS als Abkürzungen für den bequemen Zugriff auf Informationen unterstützt, die nicht einfach durch das Lesen eines einzelnen Wertes im PDF-Dokument abgerufen werden können. Die Einträge in Pseudo-Objekten vom Typ *dict* können nicht enumeriert werden.
- Bei den Komponenten *name* handelt es sich um im Dokument gefundene Dictionary-Schlüssel. Mehrere Namen werden durch das Zeichen / voneinander getrennt. pCOS-Pfade beginnen mit einem Eintrag im Trailer-Dictionary des Dokuments oder einem künstlichen Objekt, dem sogenannten Pseudo-Objekt. Dieses wird von pCOS zum vereinfachten Zugriff auf verschiedene Datenstrukturen hinzugefügt (z.B. Seiten). Jeder Name muss ein im vorangegangenen Dictionary vorhandener Dictionary-Schlüssel sein. Vollständige Pfade beschreiben die Folge der Dictionary-Schlüssel vom ersten Dictionary (entweder dem Trailer-Dictionary oder einem Pseudo-Objekt) bis zum Zielobjekt.
- Pfade oder Pfadkomponenten für ein Array oder Dictionary können einen numerischen Index enthalten, der in Dezimalformat mit umgebenden Klammern angegeben werden muss. Verschachtelte Arrays oder Dictionaries können über mehrere Indexeinträge adressiert werden. Der erste Eintrag in einem Array oder einem Dictionary hat den Index 0.
- Pfade oder Pfadkomponenten mit der Angabe eines Dictionarys können einen Index und eines der Suffixe *.key* oder *.val* enthalten. Damit lässt sich ein bestimmter Dictionary-Schlüssel oder der entsprechende Wert des indizierten Dictionary-Eintrags abfragen. Enthält ein Pfad für ein Dictionary einen Index, muss eines dieser Suffixe folgen.

Encoding für pCOS-Pfade. pCOS-Pfade bestehen meistens nur aus ASCII-Zeichen. In einigen Fällen jedoch (z.B. bei PDFlib-Blocknamen) können Nicht-ASCII-Zeichen erforderlich sein. pCOS-Pfade müssen nach den folgenden Regeln kodiert werden:

- Wenn eine Pfadkomponente eines der Zeichen */, [,]* oder # enthält, müssen diese durch das Nummernzeichen #, gefolgt von einem zweistelligen hexadezimalen ASCII-Wert ausgedrückt werden.

- In Unicode-fähigen Sprachbindungen besteht der Pfad aus einem Unicode-String mit oder ohne ASCII-Zeichen.
- In nicht Unicode-fähigen Sprachbindungen muss der Pfad in UTF-8 angegeben werden. Es macht keinen Unterschied, ob der String einen BOM enthält oder nicht. Ein BOM kann am Anfang des Pfades oder am Anfang bestimmter Pfadkomponenten (jeweils hinter dem Zeichen /) platziert werden. Auf EBCDIC-Systemen muss der Pfad generell im Encoding *ebcdic* angegeben werden. Zeichen außerhalb des ASCII-Zeichensatzes müssen als EBCDIC-UTF-8 (mit oder ohne BOM) angegeben werden.

4.2 Präfixe für Pfade

Mit Präfixen lassen sich verschiedene Attribute eines Objekts abfragen (und nicht sein Wert). Für eine Liste der unterstützten Präfixe siehe Tabelle 4.1.

Das Präfix *length* und die Auflistung von Inhalten über Indizes sind nur auf echte PDF-Objekte und Pseudo-Objekte vom Typ *array* anwendbar, aber nicht auf Pseudo-Objekte vom Typ *Dictionary*. Das Präfix *pcosid* kann nicht auf Pseudo-Objekte angewendet werden. Das Präfix *type* wird für alle Pseudo-Objekte unterstützt.

Tabelle 4.1 Präfixe für pCOS-Pfade

Präfix	Beschreibung
length	*(Zahl) Länge eines Objekts, abhängig vom Objekttyp:* ***array*** *Anzahl der Elemente im Array* ***dict*** *Anzahl der Schlüssel/Wertpaare im Dictionary* ***stream*** *Anzahl der Schlüssel/Wertpaare im Stream-Dictionary (verwenden Sie für die Bestimmung der Länge des Streams in Bytes den Schlüssel* `Length`*)* ***fstream*** *wie Stream* ***sonst*** *0*
pcosid	*(Zahl) Eindeutige pCOS-ID für ein Objekt vom Typ Dictionary oder Array.* *Beschreibt der Pfad ein Objekt, das im PDF nicht vorkommt, wird -1 zurückgegeben. Damit lässt sich prüfen, ob eine Objekt vorhanden ist und sofern es vorhanden ist, wird die ID ausgegeben.*
type	*(Zahl oder String) Objekttyp als Zahl oder String:* ***0, null*** *Null-Objekt bzw. Objekt nicht vorhanden (zur Prüfung, ob ein Objekt vorhanden ist). pCOS-Schnittstelle 9: Dieser Wert wird auch zurückgegeben, wenn ein PDF-Syntaxfehler beim Versuch auftritt, auf das im Pfad angegebene Objekt zuzugreifen.* ***1, boolean*** *Boolesches Objekt* ***2, number*** *Ganze Zahl oder Fließkommazahl* ***3, name*** *Objekt vom Typ Name* ***4, string*** *String-Objekt* ***5, array*** *Array-Objekt* ***6, dict*** *Dictionary-Objekt (außer Stream)* ***7, stream*** *Stream-Objekt, das nur unterstützte Filter verwendet* ***8, stream*** *Stream-Objekt, das ein oder mehrere nicht unterstützte Filter verwendet* *Für C- und C++-Entwickler stehen enum-Definitionen für diese Typen zur Verfügung.*

4.3 Universale Pseudo-Objekte

Universale Pseudo-Objekte stehen für alle Level von *pcosmode* zur Verfügung, unabhängig von der Verschlüsselung und der Verfügbarkeit eines Kennworts. Für eine Auflistung der universalen Pseudo-Objekte siehe Tabelle 4.2, Tabelle 4.3 und Tabelle 4.4.

4.3.1 Allgemeine Dokument-Informationen

Tabelle 4.2 Universale Pseudo-Objekte für allgemeine Dokument-Informationen

Objektname	Beschreibung
encrypt	*(Dictionary) Dictionary mit Schlüsseln, die den Verschlüsselungsstatus des Dokuments angeben:* ***length*** *(Zahl) Länge des Schlüssels für die Verschlüsselung in Bits* ***algorithm*** *(Zahl)* ***description*** *(String) Nummer oder Beschreibung des Verschlüsselungsalgorithmus:* ***-1*** *Unbekannte Verschlüsselung* ***0*** *Keine Verschlüsselung* ***1*** *40-Bit RC4 (Acrobat 2-4)* ***2*** *128-Bit RC4 (Acrobat 5)* ***3*** *128-Bit RC4 (Acrobat 6)* ***4*** *128-Bit AES (Acrobat 7)* ***5*** *Public Key in Kombination mit 128-Bit RC4 (Acrobat 5)[1]* ***6*** *Public Key in Kombination mit 128-Bit AES (Acrobat 7)[1]* ***7*** *Adobe Policy Server (Acrobat 7)[1]* ***8*** *Adobe Digital Editions (EBX)[1]* ***9*** *(pCOS-Schnittstelle 5) 256-Bit AES (Acrobat 9)* ***10*** *(pCOS-Schnittstelle 5) Public Key in Kombination mit 256-Bit AES (Acrobat 9)[1]* ***11*** *(pCOS-Schnittstelle 7) 256-Bit AES (Acrobat X/XI)* ***master*** *(Boolean)* `True`*, wenn für das Ändern der Sicherheitseinstellungen im PDF ein Master-Kennwort erforderlich ist (Berechtigungen, Benutzer- oder Master-Kennwort), sonst* `false` ***user*** *(Boolean)* `True`*, wenn für das Öffnen der PDF-Datei ein Benutzerkennwort erforderlich ist, sonst* `false` ***attachment*** *(Boolean; pCOS-Schnittstelle 8)* `True`*, wenn für das Öffnen eines Anhang der PDF-Datei, nicht der PDF-Datei selbst, ein Benutzer-Kennwort erforderlich ist, sonst* `false` ***noaccessible, noannots, noassemble, nocopy, noforms, nohiresprint, nomodify, noprint*** *(Boolean)* `True`*, wenn der entsprechende Zugriffsschutz gesetzt ist, sonst* `false` ***plainmetadata*** *(Boolean)* `True`*, wenn die PDF-Datei verschlüsselt ist, aber unverschlüsselte Metadaten enthält, sonst* `false`
filename	*(String) Name der PDF-Datei*
filesize	*(Zahl) Größe der PDF-Datei in Bytes*
linearized	*(Boolean)* `True`*, wenn das PDF-Dokument linearisiert ist, sonst* `false`
pcosmode ***pcosmode-name***	*(Zahl oder String) pCOS-Modus als Zahl oder String:* ***0, minimum*** ***1, restricted*** ***2, full***

Tabelle 4.2 Universale Pseudo-Objekte für allgemeine Dokument-Informationen

Objektname	Beschreibung
revisions	*(Zahl; pCOS-Schnittstelle 9) Anzahl der Dokumentrevisionen im PDF, wobei jede Revision durch ein inkrementelles PDF-Update beschrieben wird. Enthält ein Dokument mehrere Signaturen, wird jede Signatur als separates PDF-Update angehängt. Ein PDF-Update kann jedoch auch weitere Änderungen enthalten, z.B. hinzugefügte oder entfernte Anmerkungen, ausgefüllte Formularfelder usw. Die erste Signatur fügt nicht zwangsläufig ein PDF-Update hinzu.*
shrug	*(Boolean; nur in den Produkten TET, PDFlib+PDI, PPS, PLOP, PLOP DS)* `True`*, wenn beim Öffnen des PDF-Dokuments die Sicherheitseinstellungen ignoriert wurden; die Anwendung muss sich unbedingt an die Vorgaben des Dokumentautors halten. Der Wert* `true` *wird zurückgeliefert, wenn alle der folgenden Bedingungen erfüllt sind:* ► *Der Shrug-Modus wurde mit der Option* `shrug` *aktiviert.* ► *Das für das Dokument erforderliche Master-Kennwort wurde nicht übergeben.* ► *Das Benutzerkennwort (sofern für das Dokument erforderlich) wurde übergeben.* ► *Nur für das Produkt TET: die Sicherheitseinstellungen im Dokument erlauben kein Extrahieren von Inhalten.*

1. Mit diesem Algorithmus verschlüsselte Dokumente können identifiziert, aber nicht entschlüsselt werden.

4.3.2 PDF-Versionsangaben

Tabelle 4.3 Universale Pseudo-Objekte für PDF-Versionsinformation

Objektname	Beschreibung
extension-level	*(Zahl) Adobe Extension Level basierend auf ISO 32000 oder* `0` *(Null), wenn kein Extension Level vorhanden ist. Acrobat 9 erzeugt Dokumente mit Extension Level 3; Acrobat X/XI erzeugt Extension Level 8.*
fullpdf-version	*(Zahl) Numerischer Wert der PDF-Versionsnummer. Der Wert* `100 * BaseVersion + ExtensionLevel` *wird zurückgegeben, z.B.:* ***150*** *PDF 1.5 (Acrobat 6)* ***160*** *PDF 1.6 (Acrobat 7)* ***170*** *PDF 1.7 (Acrobat 8) = ISO 32000-1* ***173*** *PDF 1.7 Adobe Extension Level 3 (Acrobat 9)* ***178*** *(pCOS-Schnittstelle 5) PDF 1.7 Adobe Extension Level 8 (Acrobat X/XI)* ***200*** *(pCOS-Schnittstelle 5) PDF 2.0 = ISO 32000-2*
pdfversion	*(Zahl) PDF-Versionsnummer multipliziert mit 10, z.B. 17 für PDF 1.7*
pdfversion-string	*(String) Vollständiger PDF-Versions-String wie von verschiedenen API-Funktionen zur Einstellung der Kompatibilität der PDF-Ausgabe erwartet, z.B.* `1.5`, `1.6`, `1.7`, `1.7ext3`, `1.7ext8`

Im minimalen pCOS-Modus (d.h. für eine verschlüsselte Datei ist keine Benutzer-Kennwort vorhanden) kann der *ExtensionLevel* bei den Versionsangaben fehlen (z.B. wird statt 1.7ext3 nur 1.7 ausgegeben), weil die Angaben zum Extension Level nicht entschlüsselt werden können.

4.3.3 Identifikation der Bibliothek

Tabelle 4.4 Universale Pseudo-Objekte für die Identifikation der Bibliothek

Objektname	Beschreibung
major ***minor*** ***revision***	*(Zahl) Haupt-, Korrektur- oder Revisionsnummer der Bibliothek*
pcosinterface	*(Zahl) Versionsnummer der Schnittstelle der zugrunde liegenden pCOS-Implementierung. Für Informationen zur Version der pCOS-Schnittstelle, die in einer bestimmten Produktversion implementiert ist, siehe Abschnitt 1.3, »Verfügbarkeit der pCOS-Schnittstelle«, Seite 6.*
version	*(String) Gibt die vollständige Bibliotheksversion als String im Format <major>.<minor>.<revision> aus, an den gegebenenfalls noch weitere Kennungen angefügt sind (beta, rc usw.).*

4.4 Pseudo-Objekte zur Identifizierung von PDF-Standards

Tabelle 4.5 listet Pseudo-Objekte zur Identifizierung von PDF-Standards auf. Die Werte dieser Pseudo-Objekte werden auf Basis der entsprechenden Einträge im Dokument zur Identifizierung des PDF-Standards erzeugt. Eine Validierung gegen den Standard findet nicht statt.

Tabelle 4.5 Pseudo-Objekte zur Identifizierung von PDF-Standards

Objektname	*Beschreibung*
pdfa	*(String) PDF/A (ISO 19005-1 und 19005-2) Konformitätsstufe des Dokuments. Mögliche Werte:* none PDF/A-1a:2005, PDF/A-1b:2005 PDF/A-2a, PDF/A-2b, PDF/A-2u PDF/A-3a, PDF/A-3b, PDF/A-3u *(pCOS-Schnittstelle 8)*
pdfe	*(String; pCOS-Schnittstelle 5) PDF/E-Konformitätsstufe (ISO 24517-1 und 24517-2) des Dokuments. Mögliche Werte:* none PDF/E-1 PDF/E-2 *(pCOS-Schnittstelle 7)*
pdfua	*(String; pCOS-Schnittstelle 7) PDF/UA-Konformitätsstufe (ISO 14289) des Dokuments. Mögliche Werte:* none PDF/UA-1
pdfvt	*(String; pCOS-Schnittstelle 7) PDF/UA-Konformitätsstufe (ISO 14289) des Dokuments. Mögliche Werte:* none PDF/VT-1 PDF/VT-2
pdfx	*(String) PDF/X-Konformitätsstufe (ISO 15930-1 usw.) des Dokuments. Mögliche Werte:* none PDF/X-1:2001, PDF/X-1a:2001, PDF/X-1a:2003 PDF/X-2:2003 PDF/X-3:2002, PDF/X-3:2003 PDF/X-4, PDF/X-4p PDF/X-5g, PDF/X-5n, PDF/X-5p

4.5 Pseudo-Objekte für Seiten

Tabelle 4.6 listet Pseudo-Objekte für Seiteninformationen auf.

Tabelle 4.6 Pseudo-Objekte für Seiten

Objektname	Beschreibung
pages	*(Array von Dictionaries) Jedes Array-Element adressiert eine Seite des Dokuments. Durch die Indizierung mit der dezimalen Darstellung der Seitenzahl minus eins kann diese Seite adressiert werden (die erste Seite hat den Index 0). Mit dem Präfix* `length` *kann die Anzahl der Seiten im Dokument bestimmt werden. Ein auf diese Weise adressiertes Seitenobjekt enthält alle Attribute, die über den* `/Pages`*-Baum vererbt werden. Die Einträge* `/MediaBox` *und* `/Rotate` *sind immer vorhanden. Zusätzlich zu den PDF-Standardeinträgen in einem Page-Dictionary sind auf jeder Seite die folgenden Pseudo-Einträge verfügbar:* ***colorspaces, extgstates, fonts, images, patterns, properties, shadings, templates*** *(Array von Dictionaries) Seitenressourcen gemäß Tabelle 4.9.* ***annots*** *(Array von Dictionaries) Zusätzlich zu den PDF-Standardschlüsseln im Array* `Annots` *unterstützt pCOS den folgenden Pseudo-Schlüssel für Dictionaries im Array* `Annots`*:* ***destpage*** *(Zahl; nur für* `Subtype=Link` *und wenn der Eintrag* `Dest` *oder eine* `GoTo`*-Aktion vorhanden ist) Seitenzahl der Zielseite (erste Seite ist 1)* ***blocks*** *(Dictionary von Dictionaries) Abkürzung für* `pages[]/PieceInfo/PDFlib/Private/Blocks`*, d.h. die Liste von Blöcken auf der Seite für die Verwendung mit PDFlib Personalization Server (PPS). Zusätzlich zu den vorhandenen PDF-Schlüsseln unterstützt pCOS die folgenden Pseudo-Schlüssel für Dictionaries im Array* `blocks`*:* ***rect*** *(Rechteck) Ähnlich wie* `Rect`*, außer dass relevante Einträge zu* `CropBox`*/*`MediaBox` *und* `Rotate` *berücksichtigt werden und die Reihenfolge der Koordinaten normalisiert wird.* ***height*** *(Zahl) Seitenhöhe. Zur Bestimmung der Seitenhöhe werden MediaBox oder CropBox herangezogen (sofern vorhanden). Auch Einträge zur Seitendrehung werden angewendet.* ***fields*** *(Array von Dictionaries; pCOS-Schnittstelle 9) Array mit Dictionaries für die Formularfelder auf der Seite. Es werden die gleichen Pseudo-Dictionary-Schlüssel wie für den globalen Array* `fields[]` *unterstützt (siehe Tabelle 4.7, Seite 29). Für Signaturfelder werden ebenfalls die gleichen Pseudo-Dictionary-Schlüssel wie für den Array* `signaturefields[]` *unterstützt (Tabelle 4.8, Seite 31).* ***isempty*** *(Boolean)* `True`*, wenn die Seite leer ist.* ***label*** *(String) Seitenlabel der Seite (einschließlich eventuell vorhandener Präfixe). Labels werden so dargestellt wie in Acrobat. Ist kein Label vorhanden (oder das PageLabel-Dictionary ungültig), enthält der String die dezimale Seitenzahl. Römische Ziffern werden im Stil von Acrobat erzeugt (z.B. VL) und nicht im davon abweichenden, klassischen Stil (z.B. XLV). Ist* `/Root/PageLabels` *nicht vorhanden, enthält das Dokument keine Seitenlabel.* ***usespagetransparency***[1] *(Boolean; pCOS-Schnittstelle 8)* `True`*, wenn es Transparenzelemente auf der Seite gibt, sonst* `false`*.* ***usesanytransparency***[1] *(Boolean; pCOS-Schnittstelle 8)* `True`*, wenn es Transparenzelemente auf der Seite oder in den Anmerkungen auf der Seite gibt, sonst* `false`*.* ***width*** *(Zahl) Seitenbreite (es gelten die selben Regeln wie für* `height`*)* *Die folgenden Einträge werden vererbt:* `CropBox`*,* `MediaBox`*,* `Resources`*,* `Rotate`*.*

1. Diese Überprüfung gibt Transparenz in den Ressourcen einer Seite aus (z.B. Form XObjects, Rasterbilder), unabhängig davon, ob die Ressourcen tatsächlich zur Erzeugung von sichtbarem Seiteninhalt verwendet werden. Transparenz wird so definiert wie im PDF/VT-Standard.

4.6 Pseudo-Objekte für interaktive Elemente

Tabelle 4.7 listet Pseudo-Objekte auf, die zur Abfrage von PDF-Objekten oder als Shortcuts zu verschiedenen interaktiven Elementen verwendet werden können.

Tabelle 4.7 Pseudo-Objekte für PDF-Objekte, Seiten und interaktive Elemente

Objektname	Beschreibung
articles	*(Array von Dictionaries) Array mit Dictionaries der Article-Threads für das Dokument. Enthält das Dokument keine Article-Threads, hat das Array eine Länge von 0. Zusätzlich zu den PDF-Standardschlüsseln unterstützt pCOS den folgenden Pseudo-Schlüssel für Dictionaries im Array* `articles`*:* ***beads*** *(Array von Dictionaries) Bead-Verzeichnis mit PDF-Standardschlüsseln, sowie den folgenden:* ***destpage*** *(Zahl) Seitenzahl der Zielseite (erste Seite ist 1)*
bookmarks	*(Array von Dictionaries) Array mit Dictionaries der Lesezeichen (outlines) für das Dokument. Zusätzlich zu den PDF-Standardschlüsseln unterstützt pCOS die folgenden Pseudo-Schlüssel für Dictionaries im Array* `bookmarks`*:* ***destpage*** *(Zahl; nur für* `Subtype=Link` *und wenn der Eintrag* `Dest` *oder eine* `GoTo`*-Aktion vorhanden ist) Seitenzahl der Zielseite (erste Seite ist 1), wenn das Lesezeichen ein Ziel oder eine* `GoTo`*-Aktion enthält, die auf eine Seite im selben Dokument verweist, sonst -1.* ***level*** *(Zahl) Verschachtelungstiefe der Lesezeichen-Hierarchie*
destpage	*(Zahl; pCOS-Schnittstelle 9) Seitenzahl der Zielseite (erste Seite ist 1), die beim Öffnen des Dokuments angezeigt wird. Der Wert wird von der Öffnen-Aktion oder der Zielseite des Dokuments genommen, sofern vorhanden, sonst -1.*
fields	*(Array von Dictionaries) Array mit Dictionaries der Formularfelder für das Dokument. Zusätzlich zu den PDF-Standardschlüsseln im Formularfeld-Dictionary und den Einträgen im Dictionary der zugehörigen Widget-Annotation unterstützt pCOS die folgenden Pseudo-Schlüssel für Dictionaries im Array* `fields`*:* ***exportvalue***[1] *(String; pCOS-Schnittstelle 9) Exportwert des Formularfelds* ***fullname*** *(String) Vollständiger Name des Formularfelds. Für unbenannte Felder wird der Name des übergeordneten Felds verwendet. Gibt es mehrere unbenannte Elemente des selben Typs auf der selben Ebene, wird das Suffix* `#N` *an den Namen gehängt, wobei* `N` *ein fortlaufende Ganzzahl ist (beginnend bei 0).* ***level*** *(Zahl) Ebene in der Formularfeld-Hierarchie (bestimmt anhand des Separators ».«)* ***parent*** *(Zahl; pCOS-Schnittstelle 9) Index des dem Formularfeld übergeordneten Knoten im Array* `fields[]`*; ist dem Feld keine anderes Feld übergeordnet, ist der Wert -1.* ***type*** *(String; pCOS-Schnittstelle 9) Feldtyp:* `barcode`*,* `container` *(Knoten im Formularfeld-Baum, der nicht als eigenes Feld, sondern nur als Container für andere Felder außer Optionsfeldern (radio buttons) dient),* `checkbox`*,* `combobox`*,* `listbox`*,* `pushbutton`*,* `radiobutton`*,* `radiogroup` *(Container für Optionsfelder),* `signature`*,* `textfield` ***value***[1] *(Verschiedene Typen; pCOS-Schnittstelle 9) Wert für das Feld (ermittelt aus dem Schlüssel* `V` *oder vom Array* `Opt` *für Optionsfelder und Kontrollkästchen)* ***visible*** *(Boolean; pCOS-Schnittstelle 9)* `True`**,** ***wenn das Feld sichtbar ist.***

Tabelle 4.7 Pseudo-Objekte für PDF-Objekte, Seiten und interaktive Elemente

Objektname	Beschreibung
names	*(Dictionary) Dictionary, bei dem jeder Eintrag einen einfachen Zugriff auf einen Namensbaum bietet. Die folgenden Namensbäume werden unterstützt:* AP, AlternatePresentations, Dests, EmbeddedFiles, IDS, JavaScript, Pages, Renditions, Templates, URLS. *Auf jeden Namensbaum kann mit dem Namens als Schlüssel zugegriffen werden, um den entsprechenden Wert abzurufen, z.B.:* names/Dests[0].key *ruft den Namen eines Ziels ab* names/Dests[0].val *ruft das zugehörige Ziel-Dictionary ab* *Zusätzlich zu den PDF-Standardeinträgen in einem Dictionary unterstützt pCOS den folgenden Pseudo-Schlüssel für Dictionaries im Namensbaum* Dests*:* ***destpage*** *(Zahl) Seitenzahl der Zielseite (erste Seite ist 1), wenn die Zielpunkte auf eine Seite im selben Dokument verweisen, sonst -1.* *Andere Einträge im Namensbaum müssen direkt über* /Root/Names/Dests *usw. abgefragt werden, da sie in den Pseudo-Objekten des Namensbaum nicht vorhanden sind.*
objects	*(Array) Adressiert ein Element, für das bereits eine pCOS-ID mit dem Präfix* pcosid *abgefragt wurde. Die ID muss als Array-Index in Dezimalformat übergeben werden; als Ergebnis wird das PDF-Objekt mit der übergebenen ID adressiert. Das Präfix* length *ist für dieses Array nicht zulässig.*
tagged	*(Boolean)* True*, wenn das PDF-Dokument mit Tags versehen ist, sonst* false

1. Diese Pseudo-Objekt steht nicht immer zur Verfügung. Es muss mit dem Präfix type *geprüft werden, ob es vorhanden ist.*

4.7 Pseudo-Objekte für Signaturen

Tabelle 4.8 listet Pseudo-Objekte zur Abfrage von Informationen zu Signaturen auf.

Tabelle 4.8 Pseudo-Objekte für Signaturen

Objektname	Beschreibung
signaturefields	*(Array von Dictionaries; pCOS-Schnittstelle 9) Array mit allen signierten und unsignierten Signaturfeld-Dictionaries im Dokument. Im Array werden erst die signierten Felder in der Reihenfolge ihrer Signierung aufgelistet, und dann alle unsignierten Felder. Zusätzlich zu den Einträgen im entsprechenden Eintrag des Signaturfelds im Pseudo-Objekt* `fields[ ]` *unterstützt pCOS folgende Pseudo-Schlüssel für Dictionaries in diesem Array:* ***cades*** *(Boolean)* `True`*, wenn das Feld signiert ist und eine CAdES-Signatur enthält, sonst* `false` ***field*** *(Integer) Index des zugehörigen Formularfelds im Array* `fields[]` ***fillablefields*** *(Boolean; nur für* `sigtype=certification`*)* `True`*, wenn die Zertifizierungssignatur Formularfelder schützt, sonst* `false` ***permissions*** *(String; nur für* `sigtype=certification`*) Erlaubte Änderungen am Dokument, die die Zertifizierungssignatur nicht ungültig machen:* `nochanges`, `formfilling`, `formsandannotations` ***preventchanges*** *(Boolean; nur für* `sigtype=certification`*)* `True`*, wenn die Zertifizierungssignatur Acrobat dazu veranlasst, Elemente der Benutzeroberfläche zu verstecken, die die Signatur ungültig machen würden* ***sigtype*** *(String) Signaturtyp:* `none` *(für unsignierte Felder),* `approval`, `certification` *oder* `doctimestamp`
usagerights	*(Boolean; pCOS-Schnittstelle 9)* `True`*, wenn das Dokument signierte Benutzerrechte enthält; solche Dokumente werden auch als PDF mit erweiterten Reader-Funktionen bezeichnet.*

4.8 Pseudo-Objekte für Ressourcen

Ressourcen sind ein zentrales Mittel zur Verwaltung verschiedener Arten von Daten, die erforderlich sind, um den Inhalt einer Seite vollständig zu beschreiben. Das Ressourcen-Konzept in PDF ist sehr leistungsfähig und effizient, erschwert aber den Zugang mit verschiedenen technischen Konzepten wie Rekursion und Vererbung. pCOS vereinfacht die Abfrage von Ressourcen und liefert mehrere Gruppen von Pseudo-Objekten für die direkte Abfrage von Ressourcen. Einige dieser Dictionaries für Pseudo-Ressourcen enthalten zusätzlich zu den PDF-Standardschlüsseln Einträge, mit denen sich die Abfrage von Ressourcen-Informationen weiter vereinfachen lässt. Die Pseudo-Ressourcen von pCOS sind aus Benutzersicht entwickelt und unterscheiden sich von nativen PDF-Ressourcen:

- Einige Einträge können hinzugefügt (z.B. Inline-Bilder, einfache Farbräume) oder gelöscht worden sein (z.B. Fonts, die auf keiner Seite verwendet werden).
- Zusätzlich zu den originalen PDF-Dictionary-Schlüsseln können pCOS-Ressourcen-Dictionaries benutzerfreundliche Schlüssel mit zusätzlichen Angaben enthalten (z.B. Status der Font-Einbettung, Anzahl der Komponenten eines Farbraums).

pCOS unterstützt zur Abfrage von Ressourcen zwei Arten von Pseudo-Objekten. Arrays für globale Ressourcen enthalten alle Ressourcen eines bestimmten Typs in einem PDF-Dokument, seitenbezogene Ressourcen enthalten dagegen nur die auf einer bestimmten Seite verwendeten Ressourcen. Die entsprechenden Pseudo-Arrays sind für alle Typen von Ressourcen verfügbar, die in Tabelle 4.9 aufgelistet sind:

- Eine Liste aller Ressourcen im Dokument finden Sie im Array für globale Ressourcen (z.B. *images[ ]*). Wird die Länge eines Pseudo-Arrays für globale Ressourcen abgerufen, hat dies einen Ressource-Scan über alle Seiten zur Folge.
- Eine Liste aller Ressourcen auf einer Seite finden Sie im Array für seitenbezogene Ressourcen (z.B. *pages[ ]/images[ ]*). Der Zugriff auf die Länge eines Pseudo-Arrays der Seiten-Ressourcen führt zu einem Ressource-Scan für diese Seite (um alle Ressourcen zu sammeln, die auf der Seite verwendet werden, und um Bilder auf dieser Seite zusammenzuführen).

Tabelle 4.9 Pseudo-Objekte für Ressourcen; jede Ressourcen-Kategorie `P` *erzeugt zwei Ressourcen-Arrays* `P[ ]` *und* `pages[ ]/P[ ]`.

Objektname	*Beschreibung*
colorspaces	*(Array von Dictionaries; bei* `name=ICCBased` *handelt es sich um den Typ stream) Array mit Dictionaries oder Streams für alle Farbräume auf der Seite oder im Dokument. Farbraum-Ressourcen umfassen alle Farbräume, die von einem beliebigen Objekttyp referenziert werden können, einschließlich der Farbräume ohne native PDF-Ressourcen (d.h.* `DeviceGray`, `DeviceRGB` *und* `DeviceCMYK`*). Zusätzlich zu den PDF-Standardschlüsseln in Farbraum-Dictionaries (sofern der Farbraum in PDF durch ein Dictionary dargestellt wird) und in Dictionaries von ICC-Profilen unterstützt pCOS die folgenden Pseudo-Schlüssel:* ***alternateid*** *(Integer; nur wenn* `name=Separation` *und* `DeviceN`*) Index des zugrunde liegenden alternativen Farbraums im Pseudo-Objekt* `colorspaces[]` ***baseid*** *(Integer; nur wenn* `name=Indexed`*) Index des zugrunde liegenden Basis-Farbraums im Pseudo-Objekt* `colorspaces[]` ***colorantname*** *(Name; nur wenn* `name=Separation`*) Name der Schmuckfarbe. CJK-Farbnamen, die nicht aus ASCII-Zeichen bestehen, werden nach Unicode konvertiert.* ***colorantnames*** *(Array von Namen; nur bei* `name=DeviceN`*) Namen der Schmuckfarben* ***components*** *(Integer) Anzahl der Komponenten des Farbraums* ***name*** *(String) Name des Farbraums:* `CalGray`, `CalRGB`, `DeviceCMYK`, `DeviceGray`, `DeviceN`, `DeviceRGB`, `ICCBased`, `Indexed`, `Lab`, `Separation` ***csarray*** *(Array; nicht wenn* `name=DeviceGray/RGB/CMYK`*) Array, das den zugrunde liegenden nativen Farbraum beschreibt, d.h. das Farbraum-Objekt im PDF.*
extgstates	*(Array von Dictionaries) Array mit Dictionaries für alle erweiterten Grafikzustände (ExtGStates) auf der Seite oder im Dokument*

Tabelle 4.9 Pseudo-Objekte für Ressourcen; jede Ressourcen-Kategorie `P` *erzeugt zwei Ressourcen-Arrays* `P[ ]` *und* `pages[ ]/P[ ]`.

Objektname	*Beschreibung*
fonts	*(Array von Dictionaries) Array mit Dictionaries für alle Fonts auf der Seite oder im Dokument. Zusätzlich zu den PDF-Standardschlüsseln in Font-Dictionaries unterstützt pCOS die folgenden Pseudo-Schlüssel:*
	ascender *(Float; pCOS-Schnittstelle 6) Oberlänge des Fonts. Je nach Verfügbarkeit wird der Wert aus dem FontDescriptor-Dictionary im PDF genommen oder ein geschätzter Wert verwendet. Der Wert wird relativ zu einem Font mit dem Skalierungsfaktor von 1000 ausgedrückt, d.h. 1000 Einheiten beziehen sich auf die vollständige Fontgröße.* *Produkt TET: Neben Dictionary-Werten im PDF werden auch eingebettete Fonts und unter OSX oder Windows installierte Fonts analysiert, um die Font-Metrik zu ermitteln. Die Ergebnisse der Font-Analyse sind nur nach dem Aufruf von TET_get_char_info( ) mit einer Glyphe in diesem Font verfügbar. 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-FontDescriptor.*
	capheight *(Float; pCOS-Schnittstelle 6) Versalhöhe des Fonts; siehe* `ascender`
	italicangle *(Float; pCOS-Schnittstelle 6) Neigungswinkel kursiver Fonts in Grad*
	name *(String) PDF-Fontname ohne Präfix. CJK-Fontnamen, die nicht aus ASCII-Zeichen bestehen, werden nach Unicode konvertiert.*
	descender *(Float; pCOS-Schnittstelle 6) Unterlänge des Fonts; siehe* `ascender`
	embedded *(Boolean) Status der Fonteinbettung*
	fullname *(String; pCOS-Schnittstelle 5) PDF-Fontname mit Subset-Präfix, sofern vorhanden. CJK-Fontnamen, die nicht aus ASCII-Zeichen bestehen, werden nach Unicode konvertiert.*
	type *(String) Fonttyp:* `(unknown)`, `Composite`, `Multiple Master`, `OpenType`, `TrueType`, `TrueType (CID)`, `Type 1`, `Type 1 (CID)`, `Type 1 CFF`, `Type 1 CFF (CID)`, `Type 3`
	vertical *(Boolean)* `true` *für Fonts mit vertikaler Schreibrichtung, sonst* `false`
	weight *(Float; pCOS-Schnittstelle 6) Fontgewicht im Bereich 0...900: 0=keine Information vorhanden, 400=normal, 700=fett*
	xheight *(Float; pCOS-Schnittstelle 6) X-Höhe des Fonts; siehe* `ascender`

Tabelle 4.9 Pseudo-Objekte für Ressourcen; jede Ressourcen-Kategorie `P` *erzeugt zwei Ressourcen-Arrays* `P[ ]` *und* `pages[ ]/P[ ]`.

Objektname	Beschreibung
images	*(Array von Streams) Array mit Dictionaries für alle Rasterbilder auf der Seite oder im Dokument. Das Produkt TET fügt zusammengeführte (künstliche) Bilder dem Array* `images[]` *hinzu.* *Zusätzlich zu den PDF-Standardschlüsseln werden die folgenden Pseudo-Schlüssel unterstützt:* **bpc** *(Integer) Anzahl der Bits pro Komponente. Dieser Eintrag ist in der Regel der gleiche wie der PDF-Schlüssel* `BitsPerComponent`*, steht aber anders als dieser immer zur Verfügung (insbesondere Bildmasken enthalten diesen Schlüssel manchmal nicht). Für JPEG-2000-Bilder kann* `bpc` *den Wert -1 haben, da die Anzahl von Bits pro Komponente eventuell nicht in den PDF-Strukturen verfügbar ist.* **colorspaceid** *(Integer) Index des Farbraums des Bildes im Pseudo-Objekt* `colorspaces[]`*. Damit lassen sich Farbraum-Eigenschaften im Detail abfragen: Für JPEG-2000-Bilder kann der Farbraum den Wert -1 haben, da der Farbraum eventuell nicht in den PDF-Strukturen verfügbar ist. Für Bildmasken wird die ID von* `DeviceGray` *zurückgegeben.* **filterinfo** *(Dictionary) Beschreibt den übrigen Filter für Streams mit nicht unterstützten Filtern oder beim Abrufen von Stream-Daten, für die die Option* `keepfilter` *auf* `true` *gesetzt wurde. Ist kein solcher Filter vorhanden, steht kein* `filterinfo`*-Dictionary zur Verfügung. Das Dictionary enthält die folgenden Einträge:* **name** *(Name) Name des Filters* **supported** *(Boolean)* `True`*, sofern der Filter unterstützt wird* **decodeparms** *(Dictionary) Das Dictionary* `DecodeParms`*, sofern für den Filter vorhanden* **maskid** *(Integer; pCOS-Schnittstelle 9; nur im Produkt TET) Index der Bildmaske im Pseudo-Objekt* `images[]`*, sofern das Bild einen Eintrag* `Mask` *oder* `SMask` *hat; sonst -1. Damit lassen sich Bildmasken sowie Eigenschaften der Maske identifizieren.* **mergetype** *(Integer; nur im Produkt TET) Die folgenden Typen beschreiben den Bildstatus:* **0** *(normal) Das Bild entspricht einem Bild im PDF.* **1** *(artificial) Das Bild ist das Ergebnis der Zusammenführung mehrerer verarbeiteter Bilder (d.h. Bilder mit* `mergetype=2`*) zu einem einzigen Bild. Das resultierende künstliche Bild existiert in den PDF-Datenstrukturen nicht als Objekt.* **2** *(consumed) Das Bild sollte ignoriert werden, da es in ein größeres Bild überführt wurde. Obwohl das Bild im PDF vorhanden ist, sollte es nicht extrahiert werden, da es Teil eines künstlichen Bildes ist (d.h. eines Bildes mit* `mergetype=1`*).* *Dieser Eintrag enthält Informationen über alle bisher verarbeiteten Seiten. Der Wert kann sich während der Verarbeitung anderer Seiten im Dokument ändern. Ist die finale (konstante) Information erforderlich, müssen alle Seiten verarbeitet worden sein oder der Wert des pCOS-Pfads* `length:images` *muss abgefragt worden sein.* **stencilmask** *(Boolean; pCOS-Schnittstelle 9; nur im Produkt TET) Flag für die Bildmaske. Dieser Eintrag ist in der Regel der gleiche wie der PDF-Schlüssel* `ImageMask`*, sofern vorhanden, steht aber anders als dieser immer zur Verfügung.*
patterns	*(Array von Dictionaries) Array mit Dictionaries für alle Patterns auf der Seite oder im Dokument*
properties	*(Array von Dictionaries) Array mit Dictionaries für alle Eigenschaften auf der Seite oder im Dokument*
shadings	*(Array von Dictionaries) Array mit Dictionaries für Schattierungen auf der Seite oder im Dokument. Zusätzlich zu den PDF-Standardschlüsseln in Shading-Dictionaries unterstützt pCOS de folgenden Pseudo-Schlüssel:* **colorspaceid** *(Integer) Index des zugrunde liegenden Farbraums im Pseudo-Objekt* `colorspaces[]`
templates	*(Array von Dictionaries) Array mit Dictionaries für alle Templates (Form XObjects) auf der Seite oder im Dokument*

4.9 Verschlüsselte PDF-Dokumente und pCOS-Modus

pCOS unterstützt sowohl verschlüsselte als auch unverschlüsselte PDF-Dokumente als Eingabe. Die vollständige Abfrage von Objekten aus einem verschlüsselten Dokument erfordert jedoch die Übergabe des entsprechenden Master-Kennworts beim Öffnen des Dokuments. Abhängig von der Verfügbarkeit von Benutzer-und Master-Kennwort können verschlüsselte Dokumente in einem der unten beschriebenen pCOS-Modi verarbeitet werden.

Vollständiger pCOS-Modus (Modus 2). Unverschlüsselte Dokumente werden immer im vollständigen pCOS-Modus geöffnet. Dokumente mit verschlüsselten Inhalten können ohne Einschränkung verarbeitet werden, wenn das Master-Kennwort beim Öffnen der Datei übergeben wurde. Alle Objekte werden unverschlüsselt zurückgegeben.

Wenn für ein unverschlüsseltes Dokument mit verschlüsselten Dateianhängen kein Kennwort für die Anhänge übergeben wurde, führt das Abrufen der folgenden pCOS-Pfade (d.h. der Inhalt der Dateianhänge) zu einem leeren Rückgabewert (in C und C++: NULL):

```
pages[...]/annots[...]/FS/EF/F
names/EmbeddedFiles[...]/EF/F
```

Eingeschränkter pCOS-Modus (Modus 1). Wenn das Dokument ohne das passende Master-Kennwort geöffnet wurde und kein Benutzer-Kennwort erfordert (oder wenn nur das Benutzer-Kennwort übergeben wurde), können Objekte vom Typ *string*, *stream* oder *fstream* nicht abgefragt werden. Wenn die Extraktion von Seiteninhalten erlaubt ist, d.h. wenn *nocopy=false,* sind die in Tabelle 4.10 aufgeführten Objekte trotzdem zugänglich.

Tabelle 4.10 Objekte, die im eingeschränkten pCOS-Modus zugänglich sind, wenn Textextraktion erlaubt ist, d.h. wenn `nocopy=false`

object	*pCOS-Pfad*
Dokument-Metadaten[1]	`/Root/Metadata` *(XMP-Metadaten)* `/Root/Lang` *(pCOS-Schnittstelle 8)* `/Info/*` *(Dokument-Infofelder)*
bookmarks	`bookmarks[...]/Title`
Inhalte von Anmerkungen	*alle Pfade beginnend mit* `pages[...]/annots[...]`
Dateianhänge auf Dokumentebene (pCOS-Schnittstelle 7)	*alle Pfade beginnend mit* `names/EmbeddedFiles[...]`

1. Diese Objekte können auch abgefragt werden, wenn `plainmetadata=true`

Minimaler pCOS-Modus (Modus 0). Unabhängig vom Verschlüsselungsstatus und den verfügbaren Kennwörtern sind die in Tabelle 4.2, Tabelle 4.3 und Tabelle 4.4 aufgeführten universalen pCOS-Pseudo-Objekte immer verfügbar. Mit dem Pseudo-Objekt *encrypt* lässt sich zum Beispiel der Verschlüsselungsstatus eines Dokuments abfragen. Verschlüsselte Objekte können im minimalen pCOS-Modus nicht abgefragt werden.

Im minimalen pCOS-Modus kann der *ExtensionLevel* bei den Versionsangaben fehlen, die von den Pseudo-Objekten *extensionlevel, fullpdfversion, pdfversion* und *pdfversionstring* zurückgegeben werden (z.B. wird statt 1.7ext3 nur 1.7 ausgegeben). Möglicherweise

ist die zurückgegebene Version auch zu niedrig, weil die Angaben zum Extension Level nicht entschlüsselt werden können.

Zusammenfassung der Kennwort-Kombinationen. Tabelle 4.11 listet die resultierenden pCOS-Modi für geschützte Dokumente und verschiedene Kennwort-Kombinationen auf. Abhängig vom Verschlüsselungsstatus des Dokuments und dem übergebenen Kennwort zum Öffnen der Datei können PDF-Objektpfade im minimalen, eingeschränkten oder vollständigen pCOS-Modus verfügbar sein. Der Versuch, einen für den jeweiligen Modus unpassenden pCOS-Pfad abzurufen, löst eine Exception aus.

Tabelle 4.11 Resultierende pCOS-Modi für geschützte Dokumente und verschiedene Kennwort-Kombinationen

Kennen Sie...	***...läuft pCOS in folgendem Modus...***
keins der Kennwörter	► *für das Dokument ist ein Benutzerkennwort erforderlich: minimaler pCOS-Modus* ► *für das Dokument ist kein Benutzerkennwort erforderlich: eingeschränkter pCOS-Modus* ► *das Dokument enthält verschlüsselte Dateianhänge: vollständiger pCOS-Modus, aber Dateianhänge können nicht abgefragt werden*
Benutzer-Kennwort	*eingeschränkter pCOS-Modus*
Master-Kennwort oder Kennwort für die verschlüsselten Dateianhänge eines Dokuments	*vollständiger pCOS-Modus*

A pCOS-Funktionsreferenz

Die folgende Tabelle gibt einen Überblick über die pCOS-Funktionen. Weitere Informationen zu bestimmten Programmiersprachen finden Sie im entsprechenden Produkthandbuch.

pCOS-Funktionsprototypen
double pcos_get_number(int doc, String path)
String pcos_get_string(int doc, String path)
final byte[ ] pcos_get_stream(int doc, String optlist, String path)

B Änderungen

Änderungen an diesem Handbuch

Datum	*Änderungen*
17. Dezember 2014	► *Deutsche Übersetzung des Handbuchs für die pCOS-Schnittstelle 9*
14. März 2014	► *Deutsche Übersetzung des Handbuchs für die pCOS-Schnittstelle 8*

Index

PDFlib GmbH

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

Vertriebsinformationen
sales@pdflib.com

Support
support@pdflib.com *(geben Sie bitte immer Ihre Lizenznummer an)*