PDFlib GmbH

pCOS Path Reference

PDF Information Retrieval Tool

pCOS Interface Version 12

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

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

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

Contents

1 Introduction

1.1 What is pCOS?

The pCOS *(PDFlib Comprehensive Object Syntax)* interface provides a simple and elegant facility for retrieving technical information from all sections of a PDF document which do not describe page contents, such as page dimensions, metadata, interactive elements, etc. pCOS users are assumed to have some basic knowledge of internal PDF structures and dictionary keys, but do not have to deal with PDF syntax and parsing details. We strongly recommend that pCOS users obtain a copy of the *PDF Reference*. Since the standardization of PDF 1.7 in 2008 the PDF Reference is available as ISO 32000-1. This standard document can be purchased from *www.iso.org*. If you don't want to purchase the official version you can download a free edition which is identical in content:

Document Management – Portable Document Format – Part 1: PDF 1.7, First Edition
Downloadable PDF from *www.adobe.com/devnet/pdf/pdf_reference.html*.

1.2 Roadmap to Documentation and Samples

We provide the material listed below to assist you in using pCOS successfully.

Mini sample for all language bindings. The *dumper* mini sample is available in all packages and for all language bindings. It provides minimal sample code for using pCOS. The mini sample is useful for testing your pCOS installation and for getting a quick overview of writing pCOS applications.

pCOS Path Reference. The *pCOS Path Reference* (this manual) contains examples and a concise description of the pCOS path syntax which forms the heart of the pCOS interface. Since the pCOS interface is included in several PDFlib GmbH products, the pCOS Path Reference can be used with all products that include pCOS.

Corresponding Product Manual. The pCOS interface is available as an integrated part the following PDFlib GmbH products:

- PDFlib+PDI
- PDFlib Personalization Server (PPS)
- PDFlib TET (Text and Image Extraction Toolkit)
- PDFlib TET PDF IFilter
- PDFlib PLOP
- PDFlib PLOP DS

Each product comes with one or more additional product-specific manuals which describe the use of the respective programming library and the corresponding command-line tool if applicable. The product manual covers the programming languages which are supported by a product and discusses the API in detail.

pCOS Cookbook. The *pCOS Cookbook* is a collection of code fragments for the pCOS interface. It is available at the following URL:

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

The pCOS Cookbook details the use of pCOS for a variety of applications. It is highly recommended because it serves as a repository of useful pCOS programming idioms.

1.3 Availability of the pCOS Interface

As the pCOS interface is extended and new features are added, the pCOS interface number is increased. Table 1.1 details the pCOS interface numbers which are implemented in various product versions

Table 1.1 pCOS interface versions implemented in PDFlib GmbH products

pCOS interface	***highest supported PDF input version / corresponding Acrobat version***	***PDFlib GmbH product name and version***
8	*PDF 1.7 extension level 8 / Acrobat X/XI*	*TET 4.1+, TET PDF IFilter 4.1+ PDFlib+PDI 9.x, PPS 9.x pCOS 4.0*
9	*PDF 1.7 extension level 8 / Acrobat X/XI PDF 2.0 (ISO 32000-2)*	*PLOP 5.0, PLOP DS 5.0*
10	*PDF 1.7 extension level 8 / Acrobat X/XI/DC PDF 2.0 (ISO 32000-2)*	*TET 5.0, TET PDF IFilter 5.0*
11	*PDF 1.7 extension level 8 / Acrobat X/XI/DC including certificate security PDF 2.0 (ISO 32000-2)*	*PLOP 5.1+, PLOP DS 5.1+ TET 5.1, TET PDF IFilter 5.1*
12	*PDF 1.7 extension level 8 / Acrobat X/XI/DC including certificate security PDF 2.0 (ISO 32000-2)*	*PLOP 5.3, PLOP DS 5.3*

The pCOS programming interface is available in the products listed above. The PLOP and PLOP DS product packages additionally contain the pCOS command-line tool which allows you to use pCOS without the need for any programming.

Some aspects of the pCOS interface are available only in the TET product, but not in other products. These features are explicitly marked in this manual.

2 pCOS Examples

This chapter provides examples for pCOS paths which can be used to retrieve the corresponding values from PDF documents. More elaborate examples which require additional program logic are available in the pCOS Cookbook on the PDFlib Web site.

Except where noted otherwise all programming examples are presented in the Java language. However, with the obvious changes (mostly of syntactic nature) the examples can be used with all programming languages supported by pCOS.

The examples shown in this chapter are not comprehensive. Many more pCOS applications are possible by using other PDF objects.

2.1 pCOS Functions

Basic pCOS function calls. The following functions are the workhorses for querying PDF documents with pCOS:

- *pcos_get_number( )* retrieves objects of type number or boolean;
- *pcos_get_string( )* retrieves objects of type name, number, string, or boolean;
- *pcos_get_stream( )* retrieves objects of type stream, fstream, or string.

These functions can be used to retrieve information from a PDF document using the pCOS path syntax. The basic structure of a pCOS application looks as follows:

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

/* Retrieve the value of a pCOS pseudo object */
System.out.println(" PDF version: " + p.pcos_get_string(doc, "pdfversionstring"));

p.close_document(doc);
```

The parameters for the pCOS functions are the same in all products. They are documented in the respective product reference manuals; a quick overview of pCOS function prototypes is available in Appendix A, »Revision History«.

Adding program logic. Many pCOS objects consist of arrays of some length. The length can be retrieved with the *length:* prefix. The array can then be indexed with integer values in the range 0 up to *length-1*. The following code queries the number of fonts in a document and emits the type and name of each font:

```
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));
}
```

Formatting placeholders in C. The C language binding offers a convenience feature to facilitate the use of parameters within a pCOS path. Analogous to the formatting parameters of the *printf( )* family of functions you can use *%s* and *%d* placeholders for string and integer parameters, respectively. The values of these parameters must be added as additional function parameters after the pCOS path. pCOS will replace the placeholders with the actual values. This feature is particularly useful for paths containing array indices.

For example, the Java idiom above for listing all fonts can be written in C as follows:

```
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));
}
```

Since modern programming languages offer more sophisticated string handling functions this feature is only available in the C language binding, but not any other language binding.

2.2 Document

Table 2.1 lists pCOS paths for general and document-related objects.

Table 2.1 pCOS paths for document-related items

pCOS path	*type*	*explanation*
pcosmode	*number*	*pCOS mode of the document, i.e. its encryption status (see Section 4.10, »Protected PDF Documents and pCOS Mode«, page 39)*
pdfversionstring	*string*	*string representing the PDF version number of the document*
/Info/Title	*string*	*Document info field* `Title`; *The following field names are predefined in PDF and can be used in a similar manner:* `Title, Author, Subject, Keywords, Creator, Producer, CreationDate, ModDate, Trapped`
/Info/ArticleNumber	*string*	*custom document info field* `ArticleNumber` *(document info entries can use arbitrary names)*
/Root/Metadata	*stream*	*XMP stream with the document's metadata*
pdfa, pdfe, pdfua, pdfvt, pdfx	*string*	*PDF/A, PDF/E, PDF/UA, PDF/VT or PDF/X standard conformance status*

Encryption status and pCOS mode. You can query the *pcosmode* pseudo object to determine the pCOS mode for the document. This is important to avoid an exception when an attempt is made at retrieving information for which no access is granted (e.g. because the document is encrypted and no suitable password has been supplied). The following general structure based on values of *pcosmode* is recommended for all pCOS applications:

```
/* Open the PDF document */
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;

// Retrieve universal pseudo objects which are always available
System.out.println(" PDF version: " + p.pcos_get_string(doc, "pdfversionstring"));
System.out.println("  Encryption: " + p.pcos_get_string(doc, "encrypt/description"));

// encrypted document, but suitable password or digital ID was not supplied
if (pcosmode == 0)
{
    System.out.println("Minimum mode: no more information available\n");
    p.delete();
    return;
}

// otherwise query more information
System.out.println("PDF/A status: " + p.pcos_get_string(doc, "pdfa"));

// no master password supplied; we cannot retrieve metadata
if (pcosmode == 1 && !plainmetadata && p.pcos_get_number(doc, "encrypt/nocopy") != 0)
{
```

```
    System.out.print("Restricted mode: no more information available");
    p.delete();
    return;
}

// otherwise we can query document information fields and XMP metadata
...

p.close_document(doc);
```

PDF version. The following code fragment emits the PDF version number of a document:

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

Document info fields. Document information fields can be retrieved with the following code sequence. In order to make sure that an object actually exists in the PDF document and has the expected type we first check its type. If the object is present and has type *string* we can retrieve it:

```
objtype = p.pcos_get_string(doc, "type:/Info/Title");
if (objtype.equals("string"))
{
        /* Document info key found */
        title = p.pcos_get_string(doc, "/Info/Title");
}
```

XMP metadata. A stream containing XMP metadata can be retrieved with the following code sequence:

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

PDF standards. The PDF/A, PDF/E, PDF/UA, PDF/VT or PDF/X standard conformance status can be queried with simple pCOS pseudo objects as follows:

```
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"));
```