13 PPS und das PDFlib Block-Plugin

Der PDFlib Personalization Server (PPS) bietet einen PDF-Workflow zur Verarbeitung variabler Daten auf Basis von Templates. Mit Hilfe des Blockkonzepts können importierte Seiten mit variablen Mengen von ein- oder mehrzeiligen Texten, Rasterbildern, PDF-Seiten oder Vektorgrafiken angereichert werden. Damit lassen sich auf einfache Art Anwendungen implementieren, die individualisierte PDF-Dokumente erfordern, zum Beispiel:

- Serienbriefe
- flexible Erstellung personalisierter Massenbriefe
- Berichtswesen und Rechnungserstellung
- Personalisierung von Visitenkarten

Mit dem PDFlib Block-Plugin können Sie interaktiv Blöcke erstellen und bearbeiten und mit dem Plugin zur Konvertierung von Formularfeldern PDF-Formularfelder in PDFlib-Blöcke konvertieren. Blöcke können mit PPS gefüllt werden. Die Ergebnisse der Block-Befüllung mit PPS können Sie in einer Vorschau in Acrobat ansehen, da das Block-Plugin eine integrierte Version des PPS enthält.

Hinweis *Zur Blockverarbeitung ist der PDFlib Personalization Server (PPS) erforderlich. PPS ist zwar in allen PDFlib-Paketen enthalten, Sie müssen dafür aber einen eigenen Lizenzschlüssel erwerben; ein Lizenzschlüssel für PDFlib oder PDFlib+PDI reicht nicht aus. Außerdem ist das PDFlib Block-Plugin für Adobe Acrobat erforderlich, um Blöcke in PDF-Templates interaktiv anzulegen.*

Cookbook *Codebeispiele zur Verarbeitung variabler Daten mit Blöcken finden Sie in der Kategorie* `blocks` *im PDFlib Cookbook.*

13.1 Installation des Block-Plugins

Das Block-Plugin funktioniert mit folgenden Versionen von Acrobat (es funktioniert nicht mit Acrobat Reader):

- Windows: Acrobat 8/9/X/XI
- Windows: Acrobat Standard und Pro 2017/2020
- Windows: Acrobat Pro DC (32-bit und 64-bit)
- macOS: Acrobat Standard und Pro 2017/2020
- macOS: Acrobat DC (Universal Binary für Intel und M1)

Da Acrobat DC in 32- und 64-Bit-Versionen verfügbar ist, gibt es zwei verschiedene Installationspakete für das Block Plugin. Es ist wichtig, das Paket zu verwenden, das zur installierten Acrobat-Version passt.

Installation des PDFlib Block-Plugins unter Windows. Zur Installation des Block-Plugins sowie des Plugins zur Konvertierung von PDF-Formularfeldern in Acrobat kopieren Sie die Plugin-Dateien in ein Unterverzeichnis des Acrobat-Plugin-Verzeichnisses. Dies wird von der Installationsroutine des Plugins automatisch durchgeführt, kann aber auch manuell erfolgen. Unter Windows heißen die Dateien *Block.api* und *AcroForm-Conversion.api*.

Das Plugin-Verzeichnis für Acrobat 32-bit lautet üblicherweise wie folgt:

```
C:\Programme (x86)\Adobe\Acrobat DC\Acrobat\plug_ins\PDFlib Block Plugin
```

Das Plugin-Verzeichnis für Acrobat 64-bit lautet üblicherweise wie folgt:

```
C:\Programme\Adobe\Acrobat DC\Acrobat\plug_ins\PDFlib Block Plugin
```

Installation des Block-Plugins für Acrobat DC auf macOS. Um das Plugin für alle Benutzer zu installieren, gehen Sie folgendermaßen vor:

- Durch Doppelklick auf das Disk Image extrahieren Sie die Plugin-Dateien in einen Ordner.
- Kopieren Sie den Plugin-Ordner an die folgende Stelle im Ordner *Library* des Systems (erstellen Sie den Ordner *Plug-ins*, falls er noch nicht existiert):

```
/Library/Application Support/Adobe/Acrobat/XXX/Plug-ins
```

Um das Plugin für einen einzelnen Benutzer zu installieren, gehen Sie folgendermaßen vor:

- Klicken Sie auf den Desktop, um sicherzustellen, dass Sie im Finder sind, halten Sie die Taste *Option* gedrückt und wählen Sie *Gehe zu, Library*, um den Benutzerordner *Library* zu öffnen.
- Kopieren Sie den Plugin-Ordner an die folgende Stelle im Benutzerordner *Library* (erstellen Sie den Ordner *Plug-ins*, falls er noch nicht existiert):

```
/Benutzer/<Benutzername>/Library/Application Support/Adobe/Acrobat/XXX/Plug-ins
```

Mehrsprachige Benutzeroberfläche. Das PDFlib Block-Plugin unterstützt mehrere Sprachen in der Benutzeroberfläche. Die Sprache für das Block-Plugin wird automatisch eingestellt und richtet sich nach der Sprache der Acrobat-Benutzeroberfläche. Derzeit stehen Englisch, Deutsch und Japanisch zur Verfügung. Wenn Acrobat in einer anderen Sprache läuft, wird die englische Benutzeroberfläche für das Block-Plugin gewählt.

Sandbox-Schutz in Acrobat DC auf Windows. Acrobat DC 2020 führte ein neues Sicherheitsmodell mit der Bezeichnung *Sandbox-Schutz* ein. Es kann via *Bearbeiten, Voreinstellungen, [Allgemein...], Sicherheit (erweitert), Geschützter Modus* und *Geschützte Ansicht* aktiviert werden. Sobald die neue Schutzmethode aktiviert ist, sind bestimmte Operationen eingeschränkt und oberhalb des Dokuments erscheint ein gelber Streifen mit einem Sicherheitshinweis. Weitere Informationen zum Sandbox-Schutz finden Sie unter

```
helpx.adobe.com/acrobat/using/whats-new/2020-august.html
www.adobe.com/devnet-docs/acrobatetk/tools/AppSec/sandboxprotections.html
```

Falls der Sandbox-Schutz aktiv ist, beeinträchtigt er die Vorschau-Funktion des Block-Plugins. In der Voreinstellung erlaubt die Geschützte Ansicht Zugriff auf das *AppData*-Verzeichnis von Acrobat, das temporäre Verzeichnis sowie einige andere Verzeichnisse, jedoch nicht auf beliebige Anwenderverzeichnisse. Das Block-Plugin kann nur von solchen Verzeichnissen lesen oder darauf schreiben, die in der voreingestellten Liste des Geschützten Modus enthalten sind oder die in einer Policy-Datei konfiguriert wurden. Diese Datei finden Sie an folgender Stelle (für 32-bit und 64-bit-Versionen von Acrobat):

```
C:\Programme (x86)\Adobe\Acrobat DC\Acrobat\PDFlibBlockCustomPolicies.txt
C:\Programme\Adobe\Acrobat DC\Acrobat\PDFlibBlockCustomPolicies.txt
```

Per Voreinstellung erlaubt die Policy-Datei Zugriff zu den folgenden Verzeichnissen; weitere Verzeichnisse können vom Administrator zu dieser Whitelist hinzugefügt werden:

```
; Protected Path Section
FILES_ALLOW_ANY = C:\Users\<username>\*.*
FILES_ALLOW_ANY = C:\Users\Public\*.*
```

Falls der Geschützte Modus oder die Geschützte Ansicht aktiviert ist und es werden Verzeichnisse benutzt, die nicht in der Whitelist enthalten sind, können einige Funktionen des Block Plugins, insbesondere Vorschau und Importieren/Exportieren von Blöcken fehlschlagen.

Fehlerbehebung. Falls das PDFlib Block-Plugin nicht wie erwartet funktioniert, überprüfen Sie Folgendes:

- Stellen Sie sicher, dass unter *Bearbeiten, Voreinstellungen, [Allgemein...], Allgemein* das Kontrollkästchen *Nur zertifizierte Zusatzmodule verwenden* deaktiviert ist. Die Plugins werden nicht geladen, wenn Acrobat im zertifizierten Modus ausgeführt wird.
- Das Block-Plugin und auch andere Acrobat-Plugins funktionieren manchmal nicht korrekt bei PDF-Formularen, die mit Adobe Designer erstellt wurden, da sie mit dem internen Sicherheitsmodell von Acrobat kollidieren. Sie sollten die statischen PDF-Formulare von Adobe Designer vermeiden und als Input für das Block-Plugin nur dynamische PDF-Formulare nutzen.

13.2 Überblick über das Blockkonzept

13.2.1 Trennung von Dokumentdesign und Programmcode

Mit PDFlib-Blöcken ist es auf einfache Art möglich, variablen Text, Rasterbilder, PDF-Seiten oder Vektorgrafiken auf importierte Seiten zu platzieren. Im Gegensatz zu einfachen PDF-Seiten enthalten Seiten mit Blöcken Informationen über die Art der Verarbeitung, die später auf der Serverseite stattfindet. Das Blockkonzept von PDFlib trennt die folgenden Aufgaben voneinander:

- Der Designer erstellt das Seitenlayout und legt dabei die Position von variablem Text und Bildelementen sowie deren Eigenschaften wie Fontgröße, Farbe oder Bildskalierung fest. Nach der Erstellung des Layouts als PDF-Dokument legt der Designer mit dem Block-Plugin für Acrobat die variablen Blöcke und ihre Eigenschaften fest.
- Der Programmierer schreibt den Code, der die Informationen in den PDF-Blöcken auf den importierten PDF-Seiten mit dynamischen Daten wie zum Beispiel Datenbankfeldern verknüpft. Der Programmierer benötigt keine genaueren Kenntnisse über einen Block (ob dieser einen Namen oder eine Postleitzahl enthält, wo genau er sich auf der Seite befindet oder wie er formatiert ist usw.). Er ist deshalb von Layout-Änderungen unabhängig. PPS kümmert sich um alle blockspezifischen Details, die aus den in der Datei abgelegten Blockeigenschaften ermittelt werden.

Anders ausgedrückt ist der vom Programmierer entwickelte Code »datenblind«, das heißt, er ist generisch und hängt nicht von blockspezifischen Eigenschaften ab. Der Designer kann zum Beispiel den Block mit dem Namen des Empfängers auf der Seite verschieben oder die Fontgröße verändern. Der generische Code zur Blockverarbeitung braucht dann nicht geändert zu werden. Die Ausgabe wird korrekt generiert, sobald der Designer die Blockeigenschaften mit dem Acrobat-Plugin entsprechend geändert hat.

Als Zwischenschritt kann das Füllen der Blöcke in einer Vorschau in Acrobat angezeigt werden, um Entwicklungs-und Testzyklen zu beschleunigen. Die Block-Vorschau basiert auf Voreinstellungen (z. B. Text oder ein Bilddateiname), die in den Blockdefinitionen angegeben sind.

13.2.2 Blockeigenschaften

Das Verhalten von Blöcken lässt sich über die Blockeigenschaften steuern. Diese werden einem Block mit dem Block-Plugin zugeordnet.

Vordefinierte Blockeigenschaften. PDFlib-Blöcke sind als Rechtecke auf der Seite definiert, denen ein Name, ein Typ und eine offene Menge von Eigenschaften zugewiesen sind, die später vom PPS verarbeitet werden. Der Name zur Identifikation des Blocks kann beliebig gewählt werden, zum Beispiel *firstname*, *lastname* oder *zipcode*. PDFlib unterstützt unterschiedliche Blocktypen:

- Blöcke vom Typ *Textline* enthalten eine einzelne Textzeile, die mit der Textline-Ausgabemethode in PPS verarbeitet wird.
- Blöcke vom Typ *Textflow* enthalten eine oder mehrere Zeilen mit Textdaten. Mehrzeiliger Text wird mit dem Textflow-Formatierer in PPS formatiert. Textflow-Blöcke lassen sich so verbinden, dass Text von einem Block in den nächsten fließt (siehe »Verbinden von Textflow-Blöcken«, Seite 425).
- Blöcke vom Typ *Image* enthalten ein Rasterbild. Dies entspricht dem Einfügen eines TIFF- oder JPEG-Bildes in einer DTP-Anwendung.

- Blöcke vom Typ *PDF* enthalten beliebige PDF-Inhalte, die aus einer Seite eines anderen PDF-Dokuments importiert wurden. Dies entspricht dem Einfügen einer PDF-Seite in einer DTP-Anwendung.
- Blöcke vom Typ *Graphics* enthalten Vektorgrafiken. Dies entspricht dem Einfügen einer SVG-Datei in einer DTP-Anwendung.

Blöcke haben je nach Typ bestimmte vordefinierte Eigenschaften. Blockeigenschaften können mit dem Block-Plugin erstellt und verändert werden (siehe Abschnitt 13.3.2, »Bearbeiten von Blockeigenschaften«, Seite 411). Eine Liste aller vordefinierten Blockeigenschaften finden Sie in Abschnitt 13.7, »Blockeigenschaften«, Seite 428. Für einen Textblock kann zum Beispiel die Schriftart und -größe des Texts und für einen Bild- oder PDF-Block der Skalierungsfaktor oder die Drehung festgelegt werden. Für jeden Blocktyp bietet PPS eine eigene Verarbeitungsfunktion, zum Beispiel *PDF_fill_textblock()*. Anhand des Blocknamens suchen die Funktionen auf einer eingefügten PDF-Seite nach dem Block, analysieren seine Eigenschaften und platzieren die vom Client übergebenen Daten (ein- oder mehrzeiligen Text, Rasterbild, Vektorgrafik oder PDF-Seite) entsprechend der jeweiligen Blockeigenschaften auf der neuen Seite. Als Programmierer können Sie die Blockeigenschaften durch Angabe der entsprechenden Blockoptionen in den Block-Füllfunktionen überschreiben.

Blockeigenschaften für Vorgabewerte. Für die Vorgabewerte eines Blocks können spezielle Blockeigenschaften definiert werden. Dies sind Text-, Bild-, PDF- oder Grafik- Inhalte, die in dem Block platziert werden, wenn keine variablen Daten an die Block-Füllfunktionen geliefert werden oder wenn die Blockinhalte derzeit zwar konstant sind, sich in der nächsten Auflage aber ändern können.

Vorgabewerte werden auch von der Vorschaufunktion des Block-Plugins verwendet (siehe Abschnitt 13.5, »Block-Vorschau in Acrobat«, Seite 418).

Benutzerdefinierte Blockeigenschaften. Die vordefinierten Blockeigenschaften ermöglichen eine schnelle Implementierung von Anwendungen zur Verarbeitung variabler Daten. Designer sind damit jedoch auf die Eigenschaften beschränkt, die PPS intern kennt und automatisch verarbeitet. Um mehr Flexibilität zu bieten, wird deshalb zusätzlich die Möglichkeit geboten, einen Block mit selbst definierten Eigenschaften zu versehen.

Durch diese Erweiterung wird das Blockkonzept selbst Anwendungen gerecht, die höhere Anforderungen an die Verarbeitung variabler Daten stellen. Für selbst definierte Eigenschaften gibt es keinerlei Regeln, da sie von PPS in keiner Weise verarbeitet, sondern lediglich dem Client verfügbar gemacht werden, der sie inspizieren und auf geeignete Art darauf reagieren kann. Auf Basis der selbst definierten Eigenschaften eines Blocks könnte zum Beispiel über Layout oder Datenerfassung entschieden werden. So könnte für eine wissenschaftliche Anwendung festgelegt werden, mit wie vielen Stellen eine Zahl ausgegeben wird, oder die Blockeigenschaft könnte den Namen eines Datenbankfeldes definieren, dessen Inhalt dann zum Füllen des Blocks benutzt wird.

13.2.3 Was spricht gegen PDF-Formularfelder?

Erfahrene Acrobat-Benutzer mögen sich fragen, warum wir ein neues Blockkonzept implementiert haben, statt die bestehenden Formularfelder von PDF zu nutzen. Der entscheidende Unterschied besteht darin, dass PDF-Formularfelder in erster Linie dafür konzipiert wurden, vom Benutzer ausgefüllt zu werden, während PDFlib-Blöcke auto-

matisch ausgefüllt werden. Anwendungen, die sowohl interaktives als auch automatisches Ausfüllen benötigen, können PDF-Formulare und PDFlib-Blöcken mit Hilfe des Plugins zur Konvertierung von Formularfeldern kombinieren (sieheAbschnitt 13.4, »Konvertieren von PDF-Formularfeldern in Blöcke«, Seite 415).

Wenngleich die beiden Konzepte in vielen Punkten übereinstimmen, bieten PDFlib-Blöcke gegenüber PDF-Formularfeldern doch einige Vorteile. Diese werden in Tabelle 13.1 aufgezeigt.

Tabelle 13.1 Vergleich zwischen PDF-Formularfeldern und PDFlib-Blöcken

Funktion	**PDF-Formularfelder**	**PDFlib-Blöcke**
Konzeption	*für interaktiven Gebrauch*	*für automatisches Ausfüllen*
typografische Funktionen (neben Schriftart und -größe)	*–*	*Unterschneiden, Wort- und Zeichenabstand, Unterstreichen/Überstreichen/Durchstreichen*
OpenType-Features	*–*	*viele OpenType Layout-Funktionen, z.B. Ligaturen, Swash-Zeichen, Mediävalziffern*
Unterstützung für komplexe Schriftsysteme	*eingeschränkt*	*Anpassung der Zeichenform und bidirektionale Formatierung, z.B. für Arabisch und Devanagari*
Font-Behandlung	*Font-Einbettung*	*Font-Einbettung, Einbettung von Untergruppen, Encoding*
Textformatierung	*linksbündig, rechtsbündig, mittig*	*linksbündig, rechtsbündig, mittig, Blocksatz; verschiedene Formatierungsalgorithmen und Steuermöglichkeiten; Inline-Optionen können zur Steuerung der Textdarstellung verwendet werden*
Wechsel des Fonts und anderer Textattribute innerhalb des Texts	*–*	*ja*
kombiniertes Ergebnis ist Bestandteil der PDF-Seitenbeschreibung	*–*	*ja*
Benutzer können kombinierte Feldinhalte editieren	*ja*	*nein*
Funktionalität erweiterbar	*–*	*ja (selbst definierte Blockeigenschaften)*
Verwendung von Bilddateien zum Füllen	*–*	*BMP, CCITT, GIF, PNG, JPEG, JBIG2, JPEG 2000, TIFF*
Verwendung von Vektorgrafik zum Füllen	*–*	*SVG*
Farbunterstützung	*RGB*	*Graustufen, RGB, CMYK, Lab, Schmuckfarbe (HKS- und PANTONE-Schmuckfarben im Block-Plugin integriert), DeviceN*
PDF-Standards	*–*	*PDF/A, PDF/X, PDF/VT, PDF/UA*
Eigenschaften von Text und Grafik beim Füllen veränderbar	*–*	*ja*
transparente Inhalte	*–*	*ja*
Verbinden von Textblöcken möglich	*–*	*ja*

13.3 Bearbeiten von Blöcken mit dem Block-Plugin

13.3.1 Anlegen von Blöcken

Aktivieren des Blockwerkzeugs. Das Block-Plugin zur Erzeugung von PDFlib-Blöcken ist dem Acrobat-Formularwerkzeug recht ähnlich. Solange das Blockwerkzeug ausgewählt ist, sind alle Blöcke auf der Seite sichtbar. Wenn Sie ein anderes Acrobat-Werkzeug auswählen, werden die Blöcke ausgeblendet, sind aber nach wie vor vorhanden. Sie können das Blockwerkzeug auf folgende Arten aktivieren:

- Klicken Sie auf das Blocksymbol  , das folgendermaßen in Acrobat DC zu finden ist: *Werkzeuge, Erweiterte Bearbeitung*.
- Wählen Sie den Menübefehl *PDFlib Block, PDFlib Block-Werkzeug*.

Anlegen und Ändern von Blöcken. Nach der Auswahl des Blockwerkzeugs legen Sie einen Block an, indem Sie mit dem Fadenkreuz-Zeiger einfach an der gewünschten Position auf der Seite ein Rechteck geeigneter Größe aufziehen. Blöcke sind immer Rechtecke, deren Kanten parallel zu den Seitenrändern verlaufen. (Verwenden Sie für Blockinhalte, die nicht parallel zu den Seitenrändern verlaufen, die Eigenschaft *rotate*). Bei der Erstellung eines neuen Block-Rechtecks erscheint der Dialog PDFlib Blockeigenschaften, in dem Sie die Eigenschaften einstellen können (siehe Abschnitt 13.3.2, »Bearbeiten von Blockeigenschaften«, Seite 411). Das Block-Plugin erzeugt einen Blocknamen, den Sie im Dialog PDFlib Blockeigenschaften jederzeit ändern können. Der Blockname muss innerhalb einer Seite eindeutig sein, kann aber auf einer anderen Seite wiederverwendet werden.

Sie können den Blocktyp im oberen Bereich zum Typ *Textline, Textflow, Image, PDF* oder *Graphics* ändern. Für die Darstellung der Blocktypen werden verschiedene Farben verwendet (siehe Abbildung 13.1). Im Dialog *PDFlib Blockeigenschaften* sind die Eigenschaften je nach Blocktyp in Gruppen und Untergruppen gegliedert.

Hinweis *Benutzen Sie nach dem Hinzufügen neuer Blöcke oder Bearbeiten existierender Blöcke den Acrobat-Befehl »Speichern unter« (und nicht »Speichern«), um die Dateigröße zu minimieren.*

Selektieren von Blöcken. Blockoperationen wie Kopieren, Verschieben, Bearbeiten oder Löschen beziehen sich auf einen oder mehrere selektierte Blöcke. Mit dem Blockwerkzeug lassen sich ein oder mehrere Blöcke wie folgt auswählen:

- Um einen einzelnen Block auszuwählen, klicken Sie einfach darauf.
- Um mehrere Blöcke auszuwählen, halten Sie die Shift-Taste gedrückt und klicken Sie dabei auf die gewünschten Blöcke.
- Um alle Blöcke auf der Seite auszuwählen, drücken Sie Strg-A (Windows) bzw. Cmd-A (macOS) oder *Bearbeiten, Alles auswählen*.

Das Kontextmenü. Sind einer oder mehrere Blöcke selektiert, können Sie das Kontextmenü zum schnellen Zugriff auf blockspezifische Funktionen nutzen (die sich auch im Menü *PDFlib Block* befinden). Zum Öffnen des Kontextmenüs klicken Sie mit der rechten Maustaste (Windows) oder klicken bei gedrückter Strg-Taste (macOS) auf den oder die selektierten Blöcke. Um zum Beispiel einen Block zu löschen, selektieren Sie ihn mit dem Blockwerkzeug und drücken die *Entf*-Taste oder wählen Sie den Befehl *Bearbeiten, Löschen* im Kontextmenü.

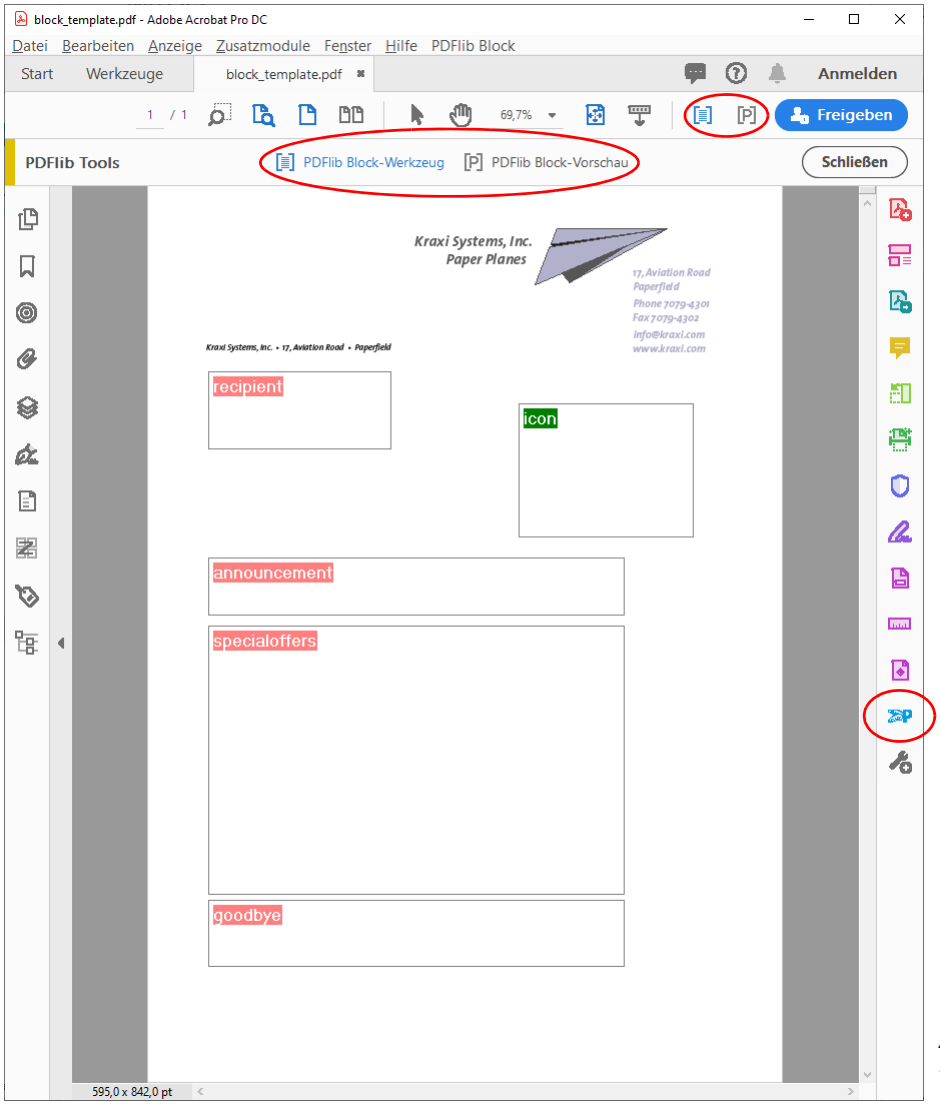

Abb. 13.1 Darstellung von Blöcken

Wenn Sie in einen Seitenbereich ohne Blöcke rechtsklicken (oder unter macOS bei gedrückter Strg-Taste klicken), können Sie eine Block-Vorschau erstellen oder die Vorschau-Funktion konfigurieren.

Blockgröße und -position. Mit dem Blockwerkzeug können Sie einen oder mehrere Blöcke an eine andere Position bewegen. Wenn Sie die Umschalttaste (Shift) während des Ziehens gedrückt halten, sind nur horizontale und vertikale Bewegungen möglich, was die exakte Ausrichtung von Blöcken erleichtert. Wenn Sie den Zeiger in die Nähe einer Blockecke bewegen, verwandelt er sich in einen Doppelpfeil, mit dem Sie die Blockgröße ändern können. Wenn Sie den Zeiger in die Nähe einer Blockecke bewegen, verwandelt er sich in einen Pfeil, mit dem Sie die Blockgröße ändern können.

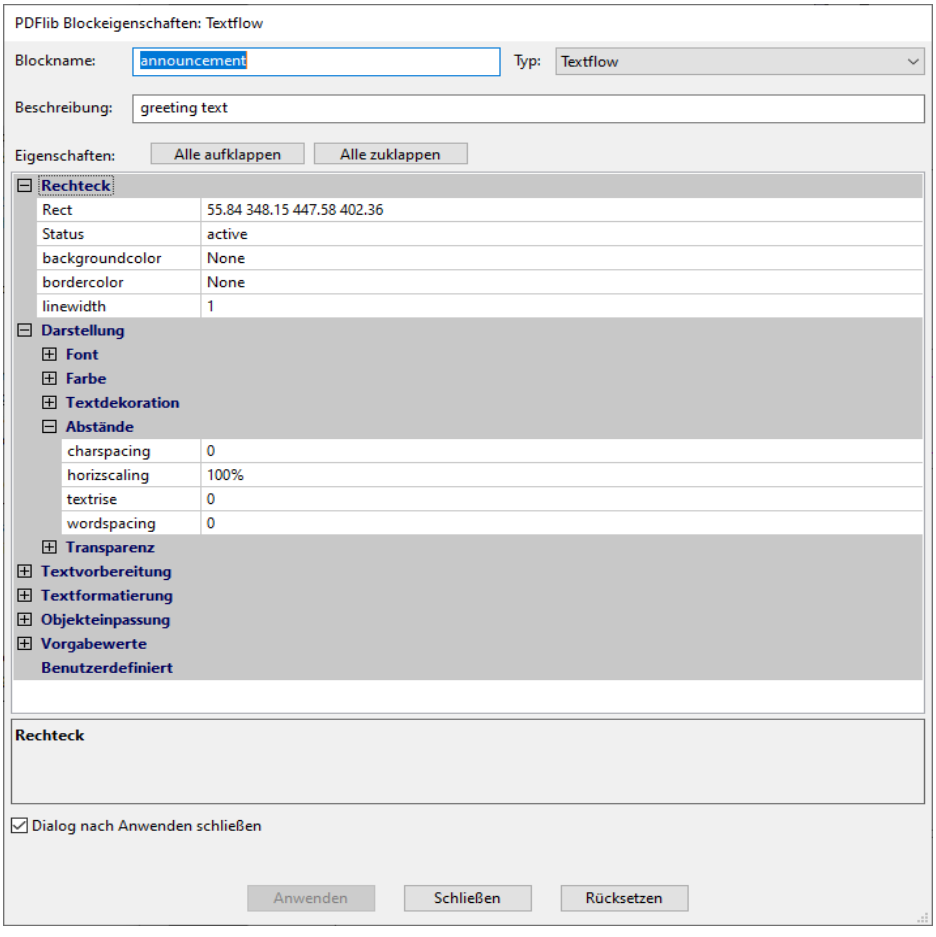

Abb. 13.2 Dialog PDFlib-Blockeigenschaften

Um die Position oder Größe mehrerer Blöcke anzupassen, wählen Sie diese aus und wählen im Menü *PDFlib Block* oder im Kontextmenü die Befehle aus den Untermenüs *Ausrichten, Zentrieren, Verteilen* und *Skalieren*. Die Position eines oder mehrerer Blöcke lässt sich auch feinstufig mit den Pfeiltasten ändern.

Alternativ dazu können Sie im Blockeigenschaften-Dialog numerische Blockkoordinaten eingeben. Der Ursprung des Koordinatensystems liegt in der linken oberen Ecke der Seite. Die Koordinaten werden in der Einheit angezeigt, die in Acrobat gerade eingestellt ist:

- Zum Ändern der Einheit in Acrobat DC gehen sie folgendermaßen vor: Wählen Sie den Menübefehl *Bearbeiten, Voreinstellungen, [Allgemein...], Einheiten und Hilfslinien* und selektieren Sie unter Einheit, Seiten- und Linealeinheiten nach Bedarf Punkt, Millimeter, Zoll, Pica oder Zentimeter.
- Zum Anzeigen der Cursor-Koordinaten wählen Sie den Menübefehl *Anzeige, Ein-/Ausblenden, Cursorkoordinaten*.

Beachten Sie, dass die gewählte Einheit ausschließlich auf die Eigenschaft *Rect* und sonst keine numerische Eigenschaft (z.B. *fontsize)* angewandt wird.

Blöcke mit Hilfe eines Rasters positionieren. Mit Hilfe der Raster-Funktion von Acrobat können Sie Blöcke exakt positionieren und in der Größe anpassen:

- Raster einblenden: *Anzeige, Ein-/Ausblenden, Lineale und Raster*;
- Raster-Ausrichtung einschalten: *Anzeige, Ein-/Ausblenden, Lineale und Raster, Am Raster ausrichten*;
- Raster ändern: *Bearbeiten, Voreinstellungen..., [Allgemein...], Einheiten und Hilfslinien*. Ändern Sie dort Abstand, Position und Farbe des Rasters.

Wenn *Am Raster ausrichten* aktiviert ist, werden Größe und Position der Blöcke am konfigurierten Raster ausgerichtet. *Am Raster ausrichten* beeinflusst neu erzeugte Blöcke sowie bestehende Blöcke, die verschoben oder mit dem Block-Plugin angepasst werden.

Anlegen von Blöcken durch Auswahl eines Rasterbildes oder einer Vektorgrafik. Statt das Rechteck für einen Block manuell aufzuziehen, können Sie die Blockgröße auch anhand von vorhandenem Seiteninhalt definieren. Dazu müssen Sie zunächst sicherstellen, dass der Menüpunkt *PDFlib Block, Zur Blockdefinition Objekt anklicken* aktiv ist. Wenn Sie nun mit dem Blockwerkzeug auf ein Rasterbild auf der Seite klicken, wird ein Block in derselben Größe und Position wie das Bild erzeugt. Sie können ebenso auf andere grafische Objekte klicken. Das Blockwerkzeug versucht, die umgebende Grafik (zum Beispiel ein Logo) zu selektieren. Diese *Objekt-Klick*-Funktion ist als Hilfsmittel zur Definition von Blöcken gedacht. Sie können einen Block auch nachträglich ohne Einschränkungen verschieben oder in der Größe verändern. Der Block ist nicht an das Rasterbild oder das grafische Objekt gebunden, das als Hilfsmittel verwendet wurde.

Die *Objekt-Klick*-Funktion versucht zu erkennen, welche Vektorgrafiken oder Rasterbilder ein logisch zusammengehörendes Element auf der Seite bilden. Wenn Sie auf einen Seiteninhalt klicken, so wird dessen Bounding Box (das umschließende Rechteck) ermittelt und selektiert, sofern das Objekt nicht weiß oder sehr groß ist. Im nächsten Schritt werden weitere Objekte, die sich teilweise im ermittelten Rechteck befinden, zum selektierten Bereich hinzugenommen und so weiter. Auf der Grundlage des endgültigen Bereichs wird das Blockrechteck generiert. Die *Objekt-Klick*-Funktion versucht letztendlich, vollständige Grafiken und nicht nur einzelne Linien zu selektieren.

Automatische Erkennung von Fonteigenschaften. Das Block-Plugin ist in der Lage, den Font zu analysieren, der sich an der Stelle befindet, an der der Textline- oder Textflow-Block positioniert wird. Darauf aufbauend kann es die folgenden entsprechenden Blockeigenschaften automatisch füllen:

```
fontname, fontsize, fillcolor, charspacing, horizscaling, wordspacing,
textrendering, textrise
```

Da die automatische Erkennung von Fonteigenschaften zu unerwünschten Ergebnissen führen kann, wenn der Hintergrund ignoriert werden soll, kann diese Funktion mit *PDFlib Block, Automatische Font- und Farberkennung* aktiviert oder deaktiviert werden. Standardmäßig ist diese Funktion deaktiviert.

Sperren von Blöcken. Blöcke können gegen versehentliches Bewegen, Verkleinern/Vergrößern oder Löschen gesperrt werden. Dazu wählen Sie das Blockwerkzeug, selektieren den gewünschten Block und wählen *Sperren* aus dem Kontextmenü. Ist ein Block gesperrt, lässt er sich weder bewegen, in der Größe ändern oder löschen noch können seine Eigenschaften bearbeitet werden.

13.3.2 Bearbeiten von Blockeigenschaften

Beim Anlegen eines neuen Blocks, beim Doppelklick auf einen vorhandenen Block und bei der Selektion von *Eigenschaften* im Kontextmenü des Blocks erscheint der Dialog PDFlib Blockeigenschaften, in dem Sie die Eigenschaften des selektierten Blocks einstellen können (siehe Abbildung 13.2). Wie in Abschnitt 13.7, »Blockeigenschaften«, Seite 428 beschrieben, gibt es je nach Blocktyp mehrere Gruppen von Eigenschaften.
Die Schaltfläche *Anwenden* ist nur aktiv, wenn Sie eine oder mehrere Eigenschaften in dem Dialog geändert haben. Bei gesperrten Blöcken ist sie deaktiviert.

Hinweis *Abhängig vom Blocktyp und bestimmten Einstellungen bei den Eigenschaften können manche Eigenschaften deaktiviert sein. Zum Beispiel ist die Eigenschaftsgruppe* `Tabulatoren` *für* `Textflow` *für* `hortabmethod=ruler` *zur Tabulatoreinstellung nur aktiv, wenn die Eigenschaft* `hortabmethod` *in der Gruppe* `Textformatierung`, `Tabulatoren` *auf* `ruler` *gesetzt ist.*

Hinweis *Wenn Sie Text für eine Blockeigenschaft eintippen, werden einige Zeichen eventuell während der Eingabe ersetzt, z.B. gerade Anführungszeichen durch typografische Anführungszeichen. Diese Ersetzung wird durch das Betriebssystem durchgeführt und kann wie folgt deaktiviert werden: »Systemeinstellungen«, »Tastatur«, »Text«, »Typografische Interpunktion«.*

Um den Wert einer Eigenschaft zu ändern, geben Sie die gewünschte Zahl oder den gewünschten Text in das Eingabefeld der Eigenschaft ein (z.B. bei *linewidth)*, selektieren den Wert aus einem Klappmenü (z.B. bei *fitmethod, orientate)* oder selektieren mit dem »...«-Button rechts einen Font, Farbwert oder Dateinamen (z.B. bei *background color, defaultimage)*. Bei der Eigenschaft *fontname* können Sie die Schrift aus einer Liste mit allen im System installierten Schriften auswählen oder einen Schriftnamen eingeben. Unabhängig von der Auswahlmethode muss der gewählte Font auf dem System installiert sein, auf dem die Blöcke mit PPS gefüllt werden.

Geänderte Eigenschaften werden im Blockeigenschaften-Dialog fett dargestellt. Wenn Sie eine Eigenschaft in einem Block geändert haben, wird das Suffix *(*)* an den Blocknamen gehängt. Klicken Sie auf die Schaltfläche *Anwenden*, um Ihre Änderungen zu speichern und den Block zu aktualisieren. Die soeben definierten Eigenschaften werden als Bestandteil der Blockdefinition in der PDF-Datei gespeichert.

Übereinander liegende Blöcke. Überlappende Blöcke lassen sich oft nur schwer selektieren, da beim Klicken in einen Bereich nur der oberste Block selektiert wird. In solchen Fällen kann der Befehl *Block auswählen* im Kontextmenü genutzt werden, um einen der Blöcke anhand seines Namens zu selektieren. Die nächste Aktion, die im Bereich des selektierten Blocks so durchgeführt wird, bezieht sich dann nur auf diesen Block. Drücken Sie zum Beispiel die *Enter*-Taste, um die gewählten Blockeigenschaften zu bearbeiten. Auf diese Weise lassen sich Blöcke problemlos bearbeiten, selbst wenn sie teilweise oder vollständig von anderen Blöcken verdeckt werden.

Verwendung und Wiederherstellung von wiederverwendeten Werten. Um sich Tipp- und Klickaufwand zu ersparen, merkt sich das Blockwerkzeug die Werte, die im Blockeigenschaften-Dialog für den zuletzt definierten Block eingegeben wurden. Diese werden beim Anlegen eines neuen Blocks wiederverwendet. Natürlich können Sie diese Werte jederzeit umdefinieren.

Mit dem Button *Alle rücksetzen* im Fenster PDFlib Blockeigenschaften werden die meisten Eigenschaften auf ihre Standardwerte zurückgesetzt. Folgende Eigenschaften bleiben unverändert:

- die Eigenschaften *Blockname, Typ, Rect* und *Beschreibung*
- alle selbst definierten Eigenschaften

Hinweis *Verwechseln Sie die Standardwerte von vordefinierten Blockeigenschaften nicht mit den Vorgabewerten in den Eigenschaften* `defaulttext`, `defaultimage`, `defaultpdf` *und* `defaultgraphics`, *die Platzhalter-Daten zur Generierung der Vorschau enthalten (siehe »Vorgabewerte«, Seite 418).*

Gleichzeitige Bearbeitung mehrerer Blöcke. Um Zeit zu sparen, können Sie mehrere Blöcke gleichzeitig bearbeiten. So wählen Sie mehrere Blöcke aus:

- Aktivieren Sie das Block-Werkzeug über das Menü *PDFlib Block, PDFlib Block-Werkzeug*.
- Klicken Sie auf den ersten Block, den Sie auswählen möchten. Dies ist der Master-Block. Klicken Sie bei gedrückter Shift-Taste auf weitere Blöcke, die Sie bearbeiten möchten. Um alle Blöcke auf der aktuellen Seite auszuwählen, können Sie auch den Befehl *Alle Auswählen* aus dem Menü *Bearbeiten* verwenden.
- Doppelklicken Sie auf einen der Blöcke, um den Blockeigenschaften-Dialog zu öffnen. Dieser Block wird zum neuen Master-Block.
- Alternativ klicken Sie auf einen einzelnen Block, um ihn zum Master-Block zu machen und drücken dann die *Enter*-Taste, um den Blockeigenschaften-Dialog zu öffnen.

Der Blockeigenschaften-Dialog zeigt die gemeinsamen Eigenschaften aller ausgewählten Blöcke. Es werden die Werte des Master-Blocks verwendet. Folgendermaßen übertragen Sie Eigenschaften auf alle ausgewählten Blöcke:

- Um nur einzelne Eigenschaften zu übertragen: Ändern Sie die Werte im *Blockeigenschaften*-Dialog wie gewünscht. Die geänderten Werte werden fett dargestellt. Deaktivieren Sie *Alle Eigenschaften des Master-Blocks anwenden*. Klicken Sie auf *Anwenden*. Nur die geänderten Werte werden auf die ausgewählten Blöcke übertragen.
- Um alle Eigenschaften des Master-Blocks zu übertragen: Ändern Sie gegebenenfalls Werte im *Blockeigenschaften*-Dialog wie gewünscht. Die geänderten Werte werden fett dargestellt. Aktivieren Sie *Alle Eigenschaften des Master-Blocks anwenden*. Klicken Sie auf *Anwenden*. Alle Werte des Master-Blocks, einschließlich der geänderten, werden auf die ausgewählten Blöcke übertragen.

Die folgenden Eigenschaften können nicht für mehrere Blöcke gleichzeitig bearbeitet oder übertragen werden:

```
Name, Description, Subtype, Type, Rect, Status
```

13.3.3 Kopieren von Blöcken zwischen Seiten und Dokumenten

Das Block-Plugin bietet mehrere Methoden zum Verschieben oder Kopieren von Blöcken auf der aktuellen Seite, innerhalb des aktuellen Dokuments oder zwischen Dokumenten:

- Blöcke können durch Ziehen mit der Maus oder durch Einfügen auf einer anderen Seite oder in ein anderes offenes Dokument verschoben oder kopiert werden.
- Blöcke können auf eine oder mehrere Seiten desselben Dokuments durch normales Kopieren/Einfügen dupliziert werden.
- Blöcke können in eine neue Datei (mit leeren Seiten) oder in ein vorhandenes Dokument (auf die vorhandenen Seiten) exportiert werden.
- Blöcke können aus anderen Dokumenten importiert werden.

Um den Seiteninhalt unter Beibehaltung der Blockdefinitionen zu aktualisieren, können Sie die zugrunde liegenden Seiten ohne Veränderung der Blöcke ersetzen. Dazu verwenden Sie den Menübefehl *Werkzeuge, Seiten verwalten, Ersetzen*.

Kopieren von Blöcken. Sie können Blöcke kopieren, indem Sie einen oder mehrere Blöcke selektieren und bei gedrückter Strg-Taste (Windows) bzw. Alt-Taste (macOS) an eine neue Position ziehen. So lange die Taste gedrückt ist, nimmt der Mauszeiger eine andere Form an. Ein kopierter Block hat die gleichen Eigenschaften wie der ursprüngliche Block mit Ausnahme des Namens, der automatisch im neuen Block geändert wird.

Ebenso können Sie Blöcke mit Kopieren/Einfügen an eine andere Position auf der selben Seite, auf einer anderen Seite des selben Dokuments oder eines anderen in Acrobat geöffneten Dokuments kopieren:

- Wählen Sie das Blockwerkzeug und selektieren Sie die zu kopierenden Blöcke.
- Kopieren Sie die selektierten Blöcke mit Strg-C (Windows) bzw. Cmd-C (macOS) oder *Bearbeiten, Kopieren* in die Zwischenablage.
- Navigieren Sie zur gewünschten Zielseite.
- Stellen Sie sicher, dass das Block-Werkzeug aktiviert ist und fügen Sie mit Strg-V (Windows) bzw. Cmd-V (macOS) oder *Bearbeiten, Einfügen* die Blöcke aus der Zwischenablage an der gewünschten Position auf der Seite oder im Dokument ein.

Duplizieren von Blöcken auf andere Seiten. Sie können einen oder mehrere Blöcke gleichzeitig auf eine beliebige Anzahl von Seiten im Dokument duplizieren:

- Wählen Sie das Blockwerkzeug und selektieren Sie die zu duplizierenden Blöcke.
- Wählen Sie *Import und Export, Duplizieren...* im Menü *PDFlib Block* oder im Kontextmenü.
- Wählen Sie aus, welche Blöcke dupliziert werden sollen *(Ausgewählte Blöcke* oder *Alle Blöcke auf dieser Seite)* und auf welche Zielseiten sie kopiert werden sollen.

Import und Export von Blöcken. Mit den Funktionen zum Import und Export von Blöcken können alle Blockdefinitionen auf der Seite oder in einem Dokument über mehrere PDF-Dateien verteilt werden. Dies ist zum Beispiel bei der Aktualisierung des Seiteninhalts sinnvoll, wenn vorhandene Blockdefinitionen erhalten bleiben sollen. Um die Blockdefinitionen in eine eigene Datei zu exportieren, gehen Sie wie folgt vor:

- Wählen Sie das Blockwerkzeug und selektieren Sie die zu exportierenden Blöcke.
- Wählen Sie den Befehl *Import und Export, Export...* im Menü *PDFlib Block* oder im Kontextmenü. Geben Sie den Seitenbereich sowie einen Namen für die PDF-Datei ein, die die Blockdefinitionen enthalten soll.

Mit dem Befehl *Import und Export, Import...* im Menü *PDFlib Block* oder im Kontextmenü importieren Sie Blockdefinitionen. Dabei können Sie auswählen, ob die importierten Blöcke auf alle Seiten oder nur einen Seitenbereich im Dokument platziert werden. Bei mehreren selektierten Seiten werden die Blockdefinitionen unverändert auf diese kopiert. Enthält der Seitenbereich des Zieldokuments mehr Seiten als die Datei mit den zu importierenden Blockdefinitionen, können Sie die Checkbox *Vorlage wiederholen* selektieren. Die Blöcke in der zu importierenden Datei werden dann so lange wiederholt auf das Zieldokument übertragen, bis das Dokumentende erreicht ist.

Kopieren von Blöcken in ein anderes Dokument beim Export. Beim Exportieren von Blöcken können Sie diese unmittelbar auf die Seiten eines anderen Dokuments übertra-

gen. Dazu wählen Sie ein bereits existierendes Dokument als Exportdatei. Wenn Sie die Checkbox *Vorhandene Blöcke löschen* selektieren, werden alle im Zieldokument bereits vorhandenen Blöcke gelöscht, bevor die neuen Blöcke in das Dokument kopiert werden.

13.3.4 Anpassen der Benutzeroberfläche des Block-Plugins mit XML

Einige Aspekte der Block-Plugin-Benutzeroberfläche werden bei jeder Acrobat-Sitzung gestartet/neu geladen und können über eine XML-Konfigurationsdatei gesteuert werden. Das Produktpaket enthält die Beispiel-Konfigurationsdatei *factory settings.xml*. Wenn die Konfiguration geändert wurde, werden die neuen Einstellungen in der Datei *user settings.xml* gespeichert. Die geänderte Konfiguration wird jedes Mal beim Start von Acrobat geladen und beim Beenden von Acrobat gespeichert. Die Konfigurationsdatei befindet sich in einem Verzeichnis, das in etwa folgendermaßen lautet:

```
Windows:    \<Benutzer>\AppData\Local\Adobe\Acrobat\DC\PDFlib\Block Plugin 5
macOS:      <home>/Library/Application Support/Adobe/Acrobat/DC/PDFlib/Block Plugin 5
```

Zur manuellen Bearbeitung der Konfigurationsdatei können Sie die folgenden XML-Elemente verwenden:

- Das Element */Block_Plugin/MainDialog/CloseOnApply* steuert den Ausgangsstatus des Kontrollkästchens *Dialog nach Anwenden schließen* im Blockeigenschaften-Dialog. Dieses Kontrollkästchen bestimmt, ob der Blockeigenschaften-Dialog nach dem Erstellen eines Blocks oder dem Ändern von Blockeigenschaften geöffnet bleibt.
- Das Element */Block_Plugin/MainDialog/ApplyAllProps* steuert den Ausgangsstatus des Kontrollkästchens *Alle Eigenschaften des Master-Blocks anwenden* im Blockeigenschaften-Dialog. Dieses Kontrollkästchen bestimmt, ob alle Eigenschaften des Master-Blocks oder nur solche, die in dem Dialog geändert wurden, auf die ausgewählten Blöcke angewendet werden.
- Das Element */Block_Plugin/FontDialog/ShowBaseFonts* steuert, ob die 14 Basisfonts in der Fontliste des *Blockeigenschaften*-Dialogs angezeigt werden *(Eigenschaftengruppe Darstellung, Eigenschaft fontname),* zusätzlich zu den auf dem System installierten Fonts.
- Das Element */Block_Plugin/Command/ControlByClick* steuert den Ausgangsstatus des Menüpunkts *PDFlib Block, Zur Blockdefinition Objekt anklicken*.
- Das Element */Block_Plugin/Command/DetectFonts* steuert den Ausgangsstatus des Menüpunkts *PDFlib Block, Automatische Font- und Farberkennung*.
- Das Element */Block_Plugin/Command/KeyAccelerator* mit den möglichen Werten *control* (Bezeichnung für die Strg-Taste auf Windows und die Command-Taste unter macOS),*shift* für die Shift-Taste, *control+shift* oder *none* steuert die Zugriffstaste für die folgenden Tastaturkürzel:

  ```
  A (Alles auswählen), C (Kopieren), I (Dialog PPDFlib-Blockeigenschaften),
  V (Einfügen), X (Ausschneiden)
  ```

  Die Änderung wirkt beim nächsten Start von Acrobat, weil Tastenkombinationen nicht zur Laufzeit geändert werden können. Wenn dieser Eintrag nicht vorhanden ist, stehen keine Tastaturkürzel zur Verfügung. Der Standardwert ist *control*.
- Das Element *Configuration/Preferences/PreviewStatusMessage* legt fest, ob nach jeder Preview-Operation ein Hinweisdialog angezeigt wird (z.B. »10 Blöcke verarbeitet: ...«).

13.4 Konvertieren von PDF-Formularfeldern in Blöcke

Statt PDFlib-Blöcke manuell zu erstellen, können Sie PDF-Formularfelder auch automatisch in PDFlib-Blöcke konvertieren. Dies ist insbesondere dann von Vorteil, wenn Sie bereits über komplexe PDF-Formulare mit zahlreichen Feldern verfügen, die in Zukunft automatisch mit PPS gefüllt werden sollen oder eine große Anzahl vorhandener PDF-Formulare umwandeln müssen, damit sie automatisch ausfüllbar sind. Um alle Formularfelder auf einer Seite in PDFlib-Blöcke zu konvertieren, wählen Sie *PDFlib Block, Formularfelder konvertieren, Aktuelle Seite*. Um alle Formularfelder in einem Dokument zu konvertieren, verwenden Sie stattdessen *Alle Seiten*. Um nur die selektierten Formularfelder zu konvertieren, verwenden Sie *Ausgewählte Formularfelder*; zur Selektion eines oder mehrerer Formularfelder wählen Sie das *Objektauswahl*-Werkzeug von Acrobat: *Werkzeuge, Rich Media*.

Konvertierung einzelner Eigenschaften. Bei der automatischen Konvertierung von Formularfeldern werden Formularfelder der im Dialogfeld *PDFlib Block, Formularfelder konvertieren, Konvertierungseinstellungen...* in Blöcke vom Typ *Textline* oder *Textflow* umgewandelt. Standardmäßig werden alle Formularfeldtypen konvertiert. Die Eigenschaften der Formularfelder werden gemäß Tabelle 13.3 in entsprechende Blockeigenschaften umgewandelt.

Formularfelder mit gleichen Namen. Gleichnamige Formularfelder auf der Seite sind erlaubt, Blocknamen dagegen müssen auf einer Seite eindeutig sein. Bei der Formularkonvertierung werden deshalb der Eindeutigkeit halber Zahlen an die generierten Blocknamen angehängt (siehe auch »Zu welchem Formularfeld gehört ein Block?«, Seite 415).

Beachten Sie, dass aufgrund eines Fehlers in Acrobat die Attribute von Formularfeldern gleichen Namens nicht korrekt wiedergegeben werden. Haben mehrere Felder denselben Namen, aber unterschiedliche Attribute, werden diese Unterschiede nicht korrekt in die generierten Blöcke übernommen. Bei der Konvertierung erscheint in solchen Fällen eine Warnung mit den Namen der betroffenen Formularfelder. Sie sollten die Eigenschaften in den generierten Blöcken dann genau überprüfen.

Zu welchem Formularfeld gehört ein Block? Da die Formularfeldnamen nicht unverändert übernommen werden, wenn mehrere Felder gleichen Namens konvertiert werden (zum Beispiel bei Radiobuttons), lässt sich schwer nachvollziehen, welcher Block eigentlich zu welchem Formularfeld gehört. Dies wäre aber insbesondere dann wichtig, wenn die Blöcke aus einer FDF- oder XFDF-Datei gefüllt werden, und das Ergebnis auch wie das ausgefüllte Formular aussehen soll.

Zur Lösung dieses Problems speichert das AcroFormConversion-Plugin bei der Erstellung des Blocks detaillierte Informationen über das zugrunde liegende Formularfeld in benutzerdefinierten Eigenschaften. Tabelle 13.2 zeigt alle benutzerdefinierten Eigenschaften, die zur zuverlässigen Ermittlung eines Blocks verwendet werden können; alle Eigenschaften haben den Typ *string*.

Binden von Blöcken an zugehörige Formularfelder. Um PDF-Formularfelder und die daraus generierten PDFlib-Blöcke aufeinander abgestimmt zu halten, können die generierten Blöcke an die entsprechenden Formularfelder gebunden werden. Das Blockwerkzeug erhält dann die Beziehung zwischen Formularfeldern und Blöcken aufrecht.

Tabelle 13.2 Benutzerdefinierte Eigenschaften zur Identifizierung des einem Block zugrunde liegenden Formularfelds

benutzerdefinierte Eigenschaft	***Bedeutung***
PDFlib:field:name	*Vollständig qualifizierter Name des Formularfelds*
PDFlib:field:pagenumber	*Nummer der Seite (als String) im Originaldokument, auf der sich das Formularfeld befand*
PDFlib:field:type	*Typ des Formularfelds; mögliche Werte sind* `pushbutton`, `checkbox`, `radiobutton`, `listbox`, `combobox`, `textfield`, `signature`
PDFlib:field:value	*(Nur für* `type=checkbox`*) Exportwert des Formularfelds*

Tabelle 13.3 Konvertierung von PDF-Formularfeldern in PDFlib-Blöcke

PDF-Formularfeldeigenschaft...	***...wird konvertiert in PDFlib-Blockeigenschaft***
Alle Felder	
Position	`Rect`
Name	`Name`
Tooltip	`Description`
Darstellung, Text, Font	`fontname`
Darstellung, Text, Fontgröße	`fontsize`*; die Fontgröße* `auto` *wird in eine feste Größe von 2/3 der Blockhöhe konvertiert, außerdem wird für die Eigenschaft* `fitmethod` *der Wert* `auto` *eingetragen. Bei mehrzeiligen Feldern/Blöcken ergibt diese Kombination eine passende Fontgröße, die kleiner als der Anfangswert von 2/3 der Blockhöhe sein kann.*
Darstellung, Text, Textfarbe	`strokecolor` *und* `fillcolor`
Darstellung, Umrandung und Farbe, Umrandungsfarbe	`bordercolor`
Darstellung, Umrandung und Farbe, Füllfarbe	`backgroundcolor`
Darstellung, Umrandung und Farbe, Linienstärke	`linewidth`*: Thin=1, Medium=2, Thick=3*
Allgemein, Allgemeine Eigenschaften, Formularfeld	`Status:` *Sichtbar*=`active` *Unsichtbar*=`ignore` *Sichtbar, aber Drucken nicht möglich*=`ignore` *Unsichtbar, aber Drucken ist möglich*=`active`
Allgemein, Allgemeine Eigenschaften, Ausrichtung	`orientate:` *0*=`north`, *90*=`west`, *180*=`south`, *270*=`east`
Textfelder	
Optionen, Standardwert	`defaulttext`
Optionen, Ausrichtung	`position:` *Links*=`{left center}` *Zentriert*=`{center center}` *Rechts*=`{right center}`
Optionen, Mehrere Zeilen	*erstellt einen Textflow-Block wenn aktiviert* *erstellt einen Textline-Block wenn deaktiviert*

Tabelle 13.3 Konvertierung von PDF-Formularfeldern in PDFlib-Blöcke

PDF-Formularfeldeigenschaft...	***...wird konvertiert in PDFlib-Blockeigenschaft***
Optionsfelder und Kontrollkästchen	
Wenn »Schaltfläche ist standardmäßig aktiviert« angeklickt ist: Optionen, Schaltflächenstil bzw. Optionen, Kontrollkästchenstil	`defaulttext:` *Häkchen=4* *Kreis=l* *Kreuz=8* *Karo=u* *Quadrat=n* *Stern=H* *(Diese Zeichen stellen die jeweiligen Symbole im Font ZapfDingbats dar.)*
Kombinationsfelder und Listenfelder	
Optionen, ausgewähltes (Standard)-Element	`defaulttext`
Schaltflächen	
Optionen, Symbol und Beschriftung, Beschriftung	`defaulttext`

Wird der Konvertierungsprozess erneut durchgeführt, so wird ein gebundener Block gemäß der Eigenschaften des zugehörenden PDF-Formularfeldes aktualisiert. Gebundene Blöcke verhindern, dass dieselbe Arbeit doppelt erledigt werden muss: Bei der Aktualisierung eines Formulars zur interaktiven Verwendung kann der entsprechende Block automatisch mit aktualisiert werden.

Wenn Sie die Formularfelder nach der Konvertierung nicht mehr benötigen, wählen Sie im Dialogfeld *PDFlib Block, Formularfelder konvertieren, Konvertierungseinstellungen...* die Option *Konvertierte Formularfelder löschen*. Bei dieser Option werden die Formularfelder nach der Konvertierung gelöscht. Alle Aktionen (zum Beispiel JavaScript), die den betroffenen Feldern zugeordnet sind, werden ebenfalls aus dem Dokument entfernt.

Stapelkonvertierung. Wenn Sie die Formularfelder vieler PDF-Dokumente in PDFlib-Blöcke konvertieren möchten, können Sie die Stapelkonvertierung nutzen, die automatisch eine beliebige Anzahl von Dokumenten verarbeitet. Der Dialog zur Stapelverarbeitung kann über *PDFlib Block, Formularfelder konvertieren, Stapelkonvertierung...* geöffnet werden:

- Es können einzelne Dateien oder der vollständige Inhalt eines Verzeichnisses zur Verarbeitung ausgewählt werden.
- Die Ausgabedateien können im Verzeichnis der Eingabedateien oder in einem anderen Verzeichnis abgelegt werden. Sie können mit einem Präfix versehen werden, um sie von den Eingabedateien zu unterscheiden.
- Bei der Verarbeitung sehr vieler Dokumente sollte eine Log-Datei angegeben werden. In dieser werden alle verarbeiteten Dateien aufgelistet und zu jeder Datei Einzelheiten zur Konvertierung einschließlich eventueller Fehlermeldungen protokolliert.

Während der Konvertierung sind die PDF-Dokumente in Acrobat sichtbar, es können aber keine Acrobat-Menüfunktionen oder Werkzeuge verwendet werden, bis die Konvertierung abgeschlossen ist.

13.5 Block-Vorschau in Acrobat

Hinweis *Sie können die Vorschau-Funktion mit dem Dokument* `block_template.pdf` *aus dem PDFlib-Paket ausprobieren. Die erforderlichen Daten (z.B. Font und Rasterbild) sind ebenfalls im Paket enthalten.*

PDFlib-Blöcke werden von PPS verarbeitet, wobei das Füllen eines Blocks bezüglich der Datenquellen (z.B. Text aus einer Datenbank, Bilddateien auf der Festplatte) sowie visuelle und interaktive Aspekte der generierten Dokumente angepasst werden können. Für weitere Informationen siehe Abschnitt 13.6, »Füllen von Blöcken mit PPS«, Seite 423.

Außerdem enthält das Block-Plugin eine integrierte Version von PPS, mit der Sie Vorschau-Versionen der gefüllten Blöcke in Acrobat ohne Programmierkenntnisse erzeugen können. Obwohl diese Vorschaufunktion nicht die gleiche Flexibilität wie eine maßgeschneiderte Programmierung bieten kann, gibt sie doch einen schnellen Überblick über die Verarbeitung der Blöcke. Die Block-Vorschau können Sie zur Verbesserung der Position und Größe der Blöcke als auch für die Überprüfung der Block-Eigenschaften verwenden (z. B. Schriftart und Größe). Sie können die Blöcke ändern und eine neue Vorschau erstellen, bis Sie mit dem in der Vorschau angezeigten Ergebnis zufrieden sind. Vorschauen können für die aktuelle Seite oder das gesamte Dokument erzeugt werden.

Die Vorschau wird immer in einem neuen PDF-Dokument angezeigt. Das Originaldokument mit den Blöcken wird durch Erzeugen einer Vorschau nicht verändert. Sie können das Vorschau-Dokument nach Ihren Bedürfnissen speichern oder verwerfen.

Vorgabewerte. Da die serverseitigen Datenquellen (z.B. eine Datenbank) für Text, Vektorgrafiken, Rasterbilder oder PDF-Inhalte eines Blocks nicht im Plugin verfügbar sind, wird in der Vorschau-Funktion immer der Vorgabewert des Blocks verwendet, das heißt die Werte der Eigenschaften *defaulttext*, *defaultimage*, *defaultpdf* oder *defaultgraphics*. Üblicherweise wird ein Beispiel-Datensatz für die Vorgabewerte verwendet, der für den in PPS verwendeten realen Blockinhalt charakteristisch ist. Blöcke ohne Vorgabewerte werden bei der Erzeugung der Vorschau ignoriert, ebenso wie Blöcke mit *Status = ignoredefault*.

Die Vorgabewerte für neue Blöcke sind leer. Bevor Sie die Vorschau-Funktion verwenden, müssen Sie die Eigenschaften *defaulttext*, *defaultimage*, *defaultpdf* oder *defaultgraphics* (je nach Blocktyp) in der Eigenschaftengruppe *Vorgabewerte* festlegen oder geeignete Werte für gleichnamige Objekte im Dialog *Erweiterte PPS-Optionen* angeben.

Hinweis *Den Vorgabewert für Symbolfonts einzugeben, kann etwas knifflig sein, siehe »Verwendung von Symbolfonts für Vorgabetext«, Seite 422.*

Einrichten der Block-Vorschau. Sie können die Block-Vorschau auf folgende Arten erstellen:

- Durch Anklicken des Symbols für die PDFlib Block-Vorschau  , das folgendermaßen in Acrobat DC zu finden ist: *Werkzeuge, Erweiterte Bearbeitung*.
- Über den Menübefehl *PDFlib Block, Vorschau, Vorschau erstellen*.
- Bei aktiviertem Blockwerkzeug können Sie auch außerhalb eines Blocks rechtsklicken und das Kontextmenü mit den Einträgen *Vorschau erstellen* und *Vorschau-Konfiguration* öffnen.

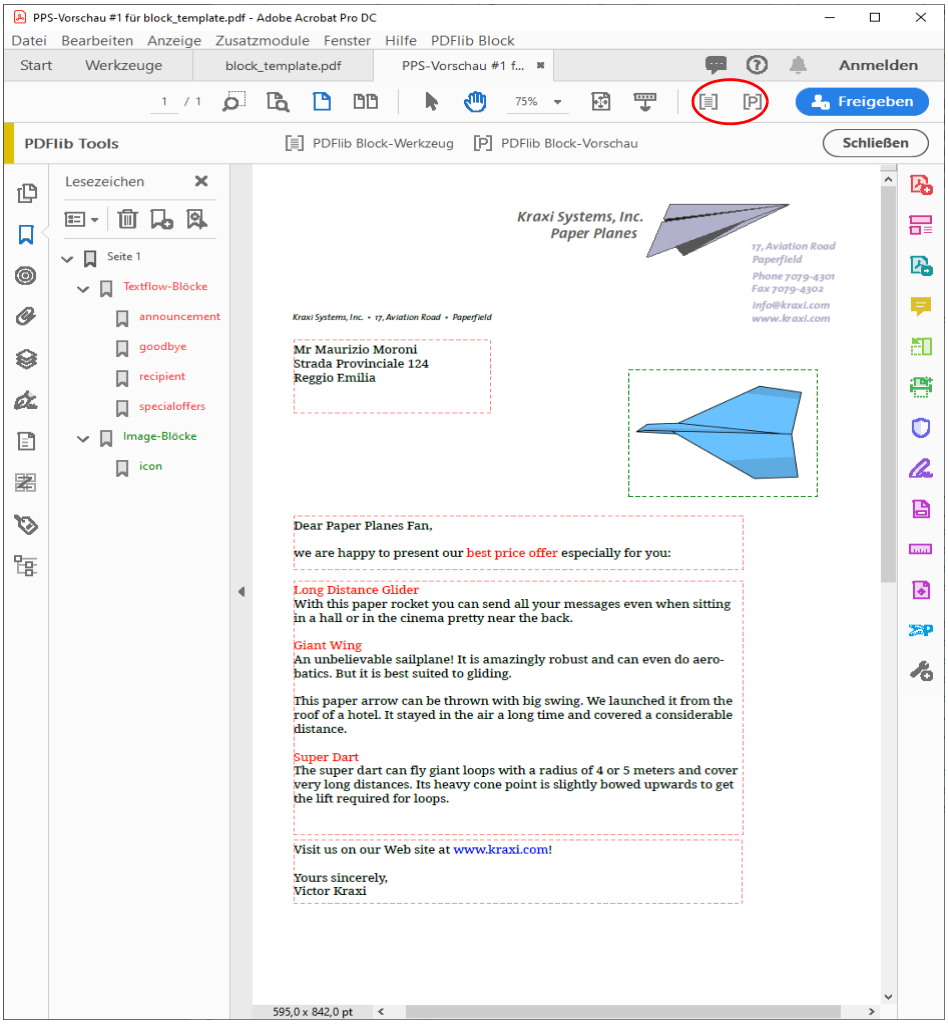

Abb. 13.3 Vorschau für das Container-Dokument aus Abbildung 13.1 mit Info-Ebenen und Anmerkungen

Die Vorschau wird auf der Grundlage der PDF-Datei auf der Festplatte erstellt. Alle Änderungen, die Sie in Acrobat vorgenommen haben, werden nur in der Vorschau berücksichtigt, wenn Sie die Block-PDF mit *Datei*, *Speichern* oder *Datei, Speichern unter...* vorher auf der Festplatte gespeichert haben. Modifizierte Blöcke können Sie an dem Stern hinter dem Blocknamen erkennen. Sie können die Vorschau-Funktion so konfigurieren, dass die Blockdatei automatisch gespeichert wird, bevor Sie eine Vorschau erstellen. So können Sie sicherstellen, dass ihre Änderungen unverzüglich in der Vorschau berücksichtigt werden.

Konfigurieren der Vorschau. Sie können die Erstellung der Block-Vorschau und des zugrunde liegenden PPS-Verhaltens über den Menübefehl *PDFlib Block, Vorschau, Vorschau-Konfiguration...* konfigurieren:

- Vorschau für die aktuelle Seite oder das gesamte Dokument;

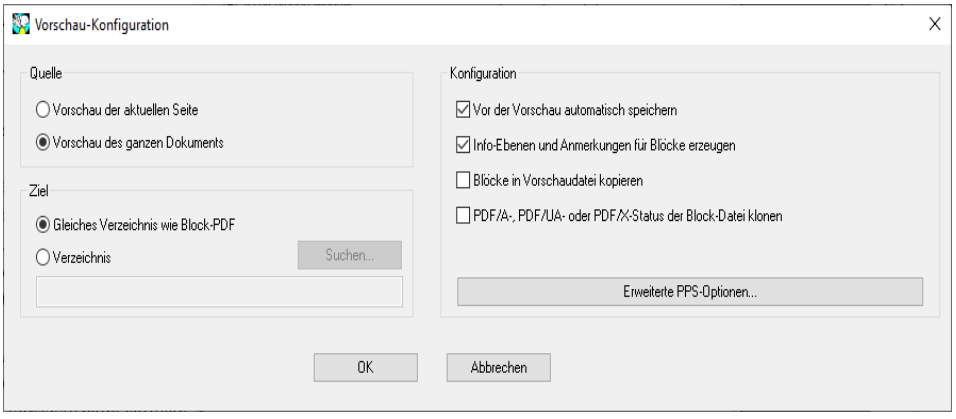

Abb. 13.4 Konfiguration der Block-Vorschau

- Ausgabe-Verzeichnis für die erstellten Vorschau-Dokumente;
- Automatisches Speichern der Blockdatei vor dem Erstellen der Vorschau;
- Hinzufügen von Info-Ebenen und Anmerkungen für Blöcke;
- Blöcke in die erzeugte Ausgabe kopieren;
- *PDF/A-, PDF/UA- oder PDF/X-Status der Block-Datei klonen:* Da diese Standards die Verwendung von Ebenen und Anmerkungen einschränken, kann die Option *Info-Ebenen und Anmerkungen erzeugen* nicht gleichzeitig aktiviert sein.
- *Kopiere Blöcke in Vorschaudatei* erlaubt das Kopieren von PDFlib-Blöcken in die generierte Vorschau beim Befüllen. Alle Blöcke werden kopiert, unabhängig davon, ob sie erfolgreich gefüllt werden konnten.
- Im Dialog *Erweiterte PPS-Optionen* können Sie zusätzliche Optionslisten für die PPS-Funktionen auf Basis der PPS-Programmschnittstelle erstellen. Zum Beispiel kann die Option *searchpath* für *PDF_set_option( )* verwendet werden, um ein Verzeichnis anzugeben, in dem sich Fonts und Rasterbilder für das Füllen der Blöcke befinden. Wir empfehlen, erweiterte Optionen in Zusammenarbeit mit dem Entwickler des PPS-Codes festzulegen.

Reihenfolge der Blöcke. Sobald das Dokument mit »Sichern als...« in Acrobat gespeichert wird, werden die Blöcke alphabetisch nach Blocknamen sortiert. In dieser Reihenfolge werden die Blöcke auch bei der Vorschau verarbeitet und von pCOS geliefert. Anwendungen werden Blöcke jedoch typischerweise auf Basis des Namens füllen, anstatt sich auf die Speicherreihenfolge in der Datei zu verlassen. Daher spielt die Speicherreihenfolge gewöhnlich keine Rolle.

Mit der Vorschau bereitgestellte Informationen. Die erzeugten Vorschau-Dokumente enthalten die ursprünglichen Seiteninhalte (Hintergrund), die gefüllten Blöcke und gegebenenfalls verschiedene andere Informationen. Diese Informationen können nützlich sein zur Überprüfung und Verbesserung von Block- und PPS-Konfiguration. Die folgenden Elemente werden für jeden aktiven Block mit Standardinhalt erstellt:

- *Fehlermarkierungen:* Blöcke, die nicht erfolgreich gefüllt werden konnten, werden durch ein durchgestrichenes Rechteck gekennzeichnet, so dass sie leicht identifiziert werden können. Fehlermarkierungen werden immer erzeugt, wenn ein Block nicht verarbeitet werden konnte.

- *Lesezeichen:* Die verarbeiteten Blöcke werden in Lesezeichen zusammengefasst, die nach Seitenzahl, Blocktyp und möglichen Fehlern strukturiert sind. Die Lesezeichen können Sie über den Menübefehl *Anzeige, Ein-/Ausblenden, Navigationsfenster, Lesezeichen* anzeigen. Lesezeichen werden immer erstellt.
- *Anmerkungen:* Für jeden verarbeiteten Block wird eine Anmerkung auf der Seite neben dem eigentlichen Blockinhalt erstellt. Das Anmerkungsrechteck visualisiert die ursprüngliche Blockgröße (abhängig von Vorgabewert und Füllmodus kann dies von der Ausdehnung des Blockinhalts abweichen). Die Anmerkung enthält den Namen des Blocks und eine Fehlermeldung, wenn der Block nicht gefüllt werden konnte. Anmerkungen werden standardmäßig generiert, können aber in der Vorschau-Konfiguration deaktiviert werden. Da die Verwendung von Anmerkungen in den Standards PDF/A und PDF/X beschränkt ist, werden Anmerkungen nicht erstellt, wenn die Option *PDF/A-, PDF/UA- oder PDF/X-Status der Blockdatei klonen* aktiviert ist.
- *Ebenen:* Seiteninhalte werden auf Ebenen platziert, um Analyse und Fehlersuche zu erleichtern. Eine separate Ebene wird für den Seitenhintergrund erstellt (das heißt für den Inhalt der ursprünglichen Seite), ebenso für jeden Blocktyp, für Fehlerblöcke, die nicht gefüllt werden konnten, sowie für die Anmerkungen mit den Block-Informationen. Wenn eine Ebene leer bleibt (z.B. weil keine Fehler aufgetreten sind), wird sie nicht erstellt. Die Ebenen können Sie über den Menübefehl *Anzeige, Ein-/Ausblenden, Navigationsfenster, Ebenen* anzeigen lassen. Standardmäßig werden alle Ebenen auf der Seite angezeigt. Um den Inhalt einer Ebene auszublenden, klicken Sie links neben dem Namen der Ebene auf das Augensymbol. Sie können das Erstellen von Ebenen in der Vorschau-Konfiguration deaktivieren. Da die Verwendung von Ebenen in den Standards PDF/A-1 und PDF/X-3 beschränkt ist, werden die Ebenen nicht erstellt, wenn die Option *PDF/A-, PDF/UA- oder PDF/X-Status der Block-Datei klonen* aktiviert ist.

PDF/A-, PDF/UA- oder PDF/X-Status klonen. Mit der Option *PDF/A-, PDF/UA- oder PDF/X-Status der Block-Datei klonen* können Sie PDF-Dateien gemäß eines dieser Standards erstellen. Der Klon-Modus kann aktiviert werden, wenn die Quelldatei einem der folgenden Standards entspricht:

```
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

PDF/UA-1

PDF/X-3:2003
PDF/X-4, PDF/X-4p
PDF/X-5n
```

Wenn eine Vorschau im Klon-Modus erstellt wird, dupliziert PPS die folgenden Aspekte der Block-PDF in der generierten Vorschau:
- die PDF-Standard-Identifizierung;
- Druckausgabebedingung;
- Seitengrößen einschließlich aller Seitenangaben;
- Tagged PDF: Dokumentsprache (falls vorhanden);
- XMP-Metadaten des Dokuments.

Beim Klonen standardkonformer PDF-Dokumente müssen alle Block-Füllvorgänge dem jeweilige Standard entsprechen. Wenn zum Beispiel keine Druckausgabebedingung vorhanden ist, können keine RGB-Bilder ohne ICC-Profil verwendet werden. Ebenso müssen alle verwendeten Fonts eingebettet werden. Die vollständige Liste der Anforderungen finden Sie in Abschnitt 12.3, »PDF/A zur Archivierung«, Seite 361 und Abschnitt 12.4, »PDF/X zur Druckproduktion«, Seite 373. Wenn ein Block-Füllvorgang im Klon-Modus den ausgewählten Standard verletzen würde (z. B. weil ein Vorgabebild RGB-Farben verwendet, aber das Dokument keine geeignete Druckausgabebedingung enthält), erscheint eine Fehlermeldung und es wird keine Vorschau erzeugt. Auf diese Weise können Benutzer potenzielle Standard-Verletzungen sehr früh im Workflow erkennen.

Verwendung von Symbolfonts für Vorgabetext. Sie können Symbolfonts auf zwei Arten im Vorgabetext verwenden:

- Bei der Arbeit mit 8-Bit-Legacy-Codes, wie z. B. in der Windows-Zeichentabelle gezeigt: Erstellen Sie die 8-Bit-Codes für *defaulttext* entweder durch Eingabe der entsprechenden 8-Bit-Zeichen (z.B. durch Kopieren/Einfügen aus der Windows-Zeichentabelle) oder als numerische Escape-Sequenz. In diesem Fall müssen Sie den Vorgabewert der Eigenschaft *charref* in der Eigenschaftsgruppe *Textvorbereitung* als *false* beibehalten, und Sie können nicht mit Character-Referenzen arbeiten. Zum Beispiel erstellt der folgende Vorgabetext die Glyphe »Smiley« aus dem Symbolfont Wingdings, wenn *charref=false* gesetzt ist:

  ```
  J
  \x4A
  \112
  ```

- Bei der Arbeit mit Unicode-Werten oder Glyphnamen aus dem verwendeten Font: Stellen Sie die Eigenschaft *charref* der Eigenschaftsgruppe *Textvorbereitung* auf *true* und geben Sie die Character-Referenzen oder Glyphnamen für die Symbole an (siehe Abschnitt 5.6.2, »Character-Referenzen«, Seite 136). Zum Beispiel erstellt der folgende Standardtext die Glyphe »Smiley« aus dem Symbolfont Wingdings, wenn *charref =true* gesetzt ist:

  ```
  &#xF04A;
  &.smileface;
  ```

Beachten Sie, dass bei beiden Methoden statt der tatsächlichen symbolischen Glyphen eine alternative Darstellung im Blockeigenschaften-Dialog angezeigt wir.

13.6 Füllen von Blöcken mit PPS

Um Blöcke mit PPS zu füllen, müssen Sie zunächst die Seite mit den Blöcken mit der Funktion *PDF_fit_pdi_page( )* auf der Ausgabeseite platzieren. Nachdem Sie die Seite platziert haben, können Sie sie mit den Funktionen *PDF_fill_* block( )* füllen.

Einfaches Beispiel: Hinzufügen von variablem Text zu einem Template. Eine häufige Aufgabe besteht darin, ein PDF-Template mit dynamischem Text anzureichern. Das folgende Codefragment öffnet eine Seite in einem Eingabe-Dokument (dem PDF-Template oder Block-Container), platziert diese auf der Ausgabeseite und füllt einen Textblock namens *firstname* mit variablem Text:

```
doc = p.open_pdi_document(filename, "");
if (doc == -1)
        throw new Exception("Error: " + p.get_errmsg());

page = p.open_pdi_page(doc, pageno, "");
if (page == -1)
        throw new Exception("Error: " + p.get_errmsg());

p.begin_page_ext(width, height, "");
/* Platzieren der importierten Seite */
p.fit_pdi_page(page, 0.0, 0.0, "");

/* Füllen eines einzelnen Blocks auf der platzierten Seite */
p.fill_textblock(page, "firstname", "Serge", "encoding=winansi");

p.close_pdi_page(page);
p.end_page_ext("");
p.close_pdi_document(doc);
```

Cookbook *Ein vollständiges Codebeispiel finden Sie im Cookbook-Topic* `blocks/starter_block`.

Überschreiben von Blockeigenschaften. In manchen Situationen möchte der Programmierer nur gewisse Eigenschaften in einer Blockdefinition verwenden, andere dagegen mit eigenen Werten überschreiben. Dies kann in verschiedenen Situationen nützlich sein:

- Bestimmte Überschreibungen können von der Business-Logik her sinnvoll sein.
- Der Skalierungsfaktor für ein Bild oder eine PDF-Seite wird nicht der Blockdefinition entnommen, sondern berechnet.
- Die Blockkoordinaten werden vom Programm angepasst, etwa um eine Rechnung mit einer variablen Zahl von Einträgen zu erstellen.
- Im Programm werden individuelle Schmuckfarbnamen angegeben, um bei der Erstellung von Drucksachen den Kundenanforderungen gerecht zu werden.

Blockeigenschaften können überschrieben werden, indem man den Namen der Blockeigenschaft und die gewünschten Werte in der Optionsliste der *PDF_fill_*block( )*-Funktionen wie folgt angibt:

```
p.fill_textblock(page, "firstname", "Serge", "fontsize=12");
```

Diese Anweisung überschreibt die interne Eigenschaft *fontsize* des Blocks mit dem Wert 12. Fast alle Namen von Blockeigenschaften können als Optionen benutzt werden.

Das Überschreiben von Blockeigenschaften wirkt sich nur auf den jeweiligen Funktionsaufruf aus. Die Änderung wird nicht in der Blockdefinition gespeichert.

Verschieben von Textflow-Blöcken beim Füllen. Die fixe Größe eines Textflow-Blocks passt unter Umständen nicht zu den variablen Textinhalten. Falls ein Block mit wenig Text gefüllt wird, bleibt eventuell eine Lücke zwischen dem aktuellen und nächsten Block; falls ein Block mit zu viel Text gefüllt werden soll, passt er nicht in das Block-Rechteck. In dieser Situation können Sie das Resultat des Textflow-Formatierungsprozesses abfragen, um die Position des nächsten Blocks anzupassen:

- Der Standardwert für *fitmethod* ist *auto*, d.h. der Text wird zwangsweise in das Block-Rechteck formatiert. Damit überschüssiger Text aus dem Block-Rechteck herausfließen kann, müssen Sie *fitmethod* auf *nofit* setzen. Dies kann zur Designzeit in den Blockeigenschaften angegeben werden oder mit der Option *fitmethod* von *PDF_fill_textblock( )*,
- Übergeben Sie die Dummy-Option *textflowhandle=-1* (in PHP: *textflowhandle=0)* an *PDF_fill_textblock( )*, damit diese Methode einen Textflow-Handle für den Inhalt des Blocks zurückliefert.
- Der zurückgegebene Textflow-Handle wird jetzt an *PDF_info_textflow( )* übergeben, um die Endposition des Textes mit Hilfe des Schlüsselworts *textendy* abzufragen.
- Nun fragen Sie die untere vertikale Position des Blocks mit *PDF_pcos_get_number( )* und dem pCOS-Pfad *pages[..]/blocks/<blockname>/Rect[1]* ab.
- Berechnen Sie nun die Differenz der beiden Werte. Ist dieser Versatz positiv, so hat der Textflow den Block nicht ganz ausgefüllt; ist er negativ, floss der Textflow aus dem Block-Rechteck heraus. In beiden Fällen können Sie den nächsten Block um den Betrag des Versatzes nach oben oder unten verschieben. Dies erreichen Sie mit der Option *refpoint* von *PDF_fill_textblock( )*, die die Eigenschaft *Rect* überschreibt. Da diese Option absolute Koordinaten erfordert, müssen Sie die vertikale Position des Blocks abfragen (analog zum vorigen Schritt) und die Summe der ursprünglichen Position und des Versatzes an die Option *refpoint* übergeben.
- Sie können diese Methode durch Akkumulieren der Versatzwerte mehrerer Blöcke auf eine beliebige Anzahl von Textflow-Blöcken anwenden. Jeder Block wird dann abhängig von seinem Inhalt nachfolgende Blöcke um einen geeigneten Betrag nach oben oder unten verschieben.

Hinweis Ein vollständiges Code-Beispiel finden Sie im Beispiel `starter_block`.

Platzieren importierter Seiten über gefüllten Blöcken. Bevor Blöcke gefüllt werden können, muss die importierte Seite auf der Ausgabeseite platziert werden. Das bedeutet in der Regel, dass sich die Originalseite unterhalb der Blockinhalte befindet. Manchmal ist es jedoch wünschenswert, dass die Seite oberhalb der gefüllten Blöcke zu liegen kommt. Dies erreichen Sie folgendermaßen: Platzieren Sie die Seite mit der Option *blind* von *PDF_fit_pdi_page( )*, damit die Blöcke und ihre Position für PPS bekannt sind. Platzieren Sie sie nach dem Füllen der Blöcke dann erneut, um die tatsächlichen Seiteninhalte anzuzeigen:

```
/* Zur Vorbereitung der Blöcke Seite unsichtbar im Blind-Modus positionieren */
p.fit_pdi_page(page, 0.0, 0.0, "blind");

/* Blöcke füllen */
p.fill_textblock(page, "firstname", "Serge", "encoding=winansi");
/* ... weitere Blöcke füllen ... */
```

```
/* Seite erneut platzieren, diesmal sichtbar */
p.fit_pdi_page(page, 0.0, 0.0, "");
```

Cookbook *Ein vollständiges Codebeispiel hierzu finden Sie im Cookbook-Topic* `blocks/block_below_contents`.

Ignorieren der Container-Seite beim Füllen von Blöcken. Importierte Blöcke können manchmal auch als Platzhalter ohne Bezug zum darunterliegenden Seiteninhalt sinnvoll sein. Beispielsweise können Sie eine Container-Seite mit Blöcken im *blind*-Modus, d.h. mit der Option *blind* von *PDF_fit_pdi_page( )* auf mehreren Ausgabeseiten platzieren und dann die Blöcke füllen. Auf diese Art können Sie die Blockeigenschaften nutzen und sogar Blöcke auf vielen Seiten (oder derselben Ausgabeseite) duplizieren, ohne die Container-Seite auf die Ausgabeseite zu platzieren.

Cookbook *Ein vollständiges Codebeispiel hierzu finden Sie im Cookbook-Topic* `blocks/duplicate_block`.

Verbinden von Textflow-Blöcken. Textflow-Blöcke können so verbunden werden, dass ein Block den übrigen Text des Vorgängerblocks aufnimmt. Wenn Sie langen variablen Text unter Umständen auf einer zweiten Seite fortsetzen müssen, können Sie zwei Blöcke verbinden und den verbleibenden Text des ersten Blocks im zweiten Block platzieren.

PPS erzeugt aus dem an *PDF_fill_textblock( )* übergebenen Text sowie den Blockeigenschaften intern einen Textflow. Ohne verbundene Blöcke wird der Textflow im Block platziert und das zugehörige Textflow-Handle am Ende des Aufrufs gelöscht; übriger Text geht dabei verloren.

Bei verbundenen Textflow-Blöcken kann übrig bleibender Text des ersten Blocks im nächsten Block platziert werden. Statt einen neuen Textflow zu erzeugen, wird der übrige Textflow als Blockinhalt verwendet. Textflow-Blöcke werden wie folgt verbunden:

- Beim ersten Aufruf von *PDF_fill_textblock( )* innerhalb einer Folge von verbundenen Textflow-Blöcken wird in der Option *textflowhandle* der Wert -1 (in PHP: 0) übergeben. *PDF_fill_textblock( )* gibt ein intern erzeugtes Textflow-Handle zurück, das von der Anwendung gespeichert werden muss.
- Beim nächsten Aufruf von *PDF_fill_textblock( )* kann in der Option *textflowhandle* das im vorigen Schritt erhaltene Textflow-Handle übergeben werden (der im Parameter *text* übergebene Text wird in diesem Fall ignoriert und sollte leer sein). Der Block wird mit dem Rest des Textflows gefüllt.
- Dieser Vorgang lässt sich mit weiteren Textflow-Blöcken wiederholen.
- Das zurückgegebene Textflow-Handle kann an *PDF_info_textflow( )* übergeben werden, um das Resultat des Vorgangs zu ermitteln, z.B. den Zustand oder die Textposition bei Vorgangsende.

Beachten Sie, dass die Eigenschaft *fitmethod* auf *clip* gesetzt werden sollte (dies ist ohnehin die Standardeinstellung, wenn *textflowhandle* übergeben wird). Das grundlegende Codefragment zum Verbinden von Textflow-Blöcken lautet wie folgt:

```
p.fit_pdi_page(page, 0.0, 0.0, "");
tf = -1;

for (i = 0; i < blockcount; i++)
{
        String optlist = "encoding=winansi textflowhandle=" + tf;
```

```
            int reason;
            tf = p.fill_textblock(page, blocknames[i], text, optlist);
            text = null;

            if (tf == -1)
                break;

            /* Ergebnis des letzten Aufrufs von fit_textflow() untersuchen */
            reason = (int) p.info_textflow(tf, "returnreason");
            result = p.get_string(reason, "");

            /* Schleife beenden, wenn der Text vollständig platziert wurde */
            if (result.equals("_stop"))
            {
                p.delete_textflow(tf);
                break;
            }
        }
```

Cookbook *Ein vollständiges Codebeispiel finden Sie im Cookbook-Topic* `blocks/linked_textblocks`.

Reihenfolge der Block-Befüllung. Die Blockfunktionen *PDF_fill_*block()* verarbeiten Blockeigenschaften und Blockinhalte in der folgenden Reihenfolge:

- Hintergrund: Wenn die Eigenschaft *backgroundcolor* vorhanden ist und das Schlüsselwort *colorspace* einen Wert ungleich *None* hat, wird der Blockbereich mit der festgelegten Farbe gefüllt.
- Rahmen: Wenn die Eigenschaft *bordercolor* vorhanden ist und das Schlüsselwort *colorspace* einen Wert ungleich *None* hat, wird der Blockrand in der festgelegten Farbe und Linienstärke gezeichnet.
- Inhalt: Die Blockinhalte und alle anderen Eigenschaften außer *bordercolor* und *linewidth* werden verarbeitet.
- Textline- und Textflow-Blöcke: Wenn weder Text noch Vorgabetext übergeben wird, gibt es keinerlei Ausgabe, auch nicht für die Hintergrundfarbe oder die Blockränder.

Verschachtelte Blöcke. Bevor Blöcke gefüllt werden können, muss die importierte Seite auf der Ausgabeseite platziert werden (da PPS sonst die Lage der Blöcke nach dem Skalieren, Drehen und Verschieben der Seite nicht erkennen würde). Wenn die Seite nur als Block-Container ohne statische Inhalte auf der neue Seite verwendet wird, können Sie die importierten Seite auch mit der Option *blind* platzieren.

Für eine erfolgreiche Block-Befüllung spielt es keine Rolle, wie die importierte Seite auf der Ausgabeseite platziert wurde:

- Die Seite kann direkt mit *PDF_fit_pdi_page()* platziert werden.
- Die Seite kann indirekt in einer Tabellenzelle mit *PDF_fit_table()* platziert werden.
- Die Seite kann als Inhalt eines anderen PDF-Blocks mit *PDF_fill_pdfblock()* platziert werden.

Die dritte Methode, das heißt das Füllen eines PDF-Blocks mit einer anderen Seite mit Blöcken, ermöglicht verschachtelte Block-Container. Damit lassen sich interessante Anwendungsfälle implementieren. Zum Beispiel können Sie Bogenmontage und Personalisieren in einem zweistufigen Block-Füllvorgang implementieren:

- Die Block-Containerseite auf erster Ebene enthält mehrere große PDF-Blöcke, die die Hauptbereiche auf dem zu bedruckenden Papier markieren. Die Anordnung der PDF-

Blöcke spiegelt die geplante Nachbearbeitung des Papiers wider (z. B. Falten oder Schneiden).

- Jeder der PDF-Blöcke auf erster Ebene wird dann mit einer PDF-Containerseite auf zweiter Ebene gefüllt. Diese enthält Text-, Bild-, PDF- oder Grafik-Blöcke, die zur individuellen Gestaltung mit variablem Text befüllt werden können.

Mit diesem Verfahren können Block-Container verschachtelt werden. Obwohl Blöcke beliebig tief verschachtelt werden können, wird eine Verschachtelungstiefe von drei oder mehr Ebenen nur selten benötigt werden.

Die Block-Container auf zweiter Ebene (z.B. eine Vorlagenseite für einen Brief) können für jede montierte Seite identisch oder verschieden sein. Wenn sie identisch sind, müssen erst die Blöcke auf der Briefvorlage gefüllt werden, bevor die Briefvorlage selbst in den nächsten Block erster Ebene platziert wird, da PPS immer den Ort der letzten Platzierung der Vorlagenseite verwendet.

Cookbook *Ein vollständiges Codebeispiel finden Sie im Cookbook-Topic* `blocks/nested_blocks`.

Block-Koordinaten. Die Rechteck-Koordinaten für einen Block beziehen sich auf das Standardkoordinatensystem von PDF. Wird die Seite, die den Block enthält, mit PPS auf der Ausgabeseite platziert, können an *PDF_fit_pdi_page()* verschiedene Optionen zur Positionierung und Skalierung übergeben werden. Diese Parameter werden bei der Verarbeitung des Blocks berücksichtigt. Damit wird es möglich, eine Template-Seite mehrfach auf der Ausgabeseite zu platzieren, wobei deren Blöcke jedes Mal mit Daten gefüllt werden. So könnte das Template für eine Visitenkarte zum Beispiel viermal auf ein Blatt montiert werden. Die Blockfunktionen kümmern sich um Transformationen des Koordinatensystems und die korrekte Platzierung des Texts für alle Blöcke bei allen Aufrufen der Seite. Vorausgesetzt wird hier lediglich, dass der Client die Seite platziert und alle Blöcke auf der platzierten Seite verarbeitet. Die Seite kann dann auf der Ausgabeseite an anderer Stelle erneut platziert werden, wobei weitere Blockverarbeitung an der neuen Position stattfindet usw.

Das Block-Plugin zeigt die Blockkoordinaten anders an, als sie intern in der PDF-Datei gespeichert werden. Während das Plugin mit der bei Acrobat üblichen Notation arbeitet und den Koordinatenursprung in der linken oberen Ecke hat, beziehen sich die internen Koordinaten (die im Block gespeichert werden) auf die PDF-Konvention und haben den Ursprung in der linken unteren Ecke der Seite. Die Koordinatenanzeige im Blockeigenschaften-Dialog ist abhängig von den in Acrobat festgelegten Einheiten (siehe »Blockgröße und -position«, Seite 408).

Schmuckfarben in Blockeigenschaften. Um in einer Blockeigenschaft eine Schmuckfarbe zu verwenden, können Sie mit der Schaltfläche »...« eine Liste aller HKS- und Pantone-Schmuckfarben anzeigen. Die angezeigten Farbnamen sind in PPS integriert und können ohne weitere Vorkehrungen verwendet werden. Für benutzerdefinierte Schmuckfarben kann im Block-Plugin eine Alternativfarbe definiert werden. Ist in den Blockeigenschaften keine Alternativfarbe spezifiziert, muss die benutzerdefinierte Schmuckfarbe in der PPS-Anwendung bereits früher mit *PDF_makespotcolor()* oder einer geeigneten Farboptionsliste definiert worden sein. Andernfalls schlagen die Blockfunktionen fehl.

13.7 Blockeigenschaften

PPS und das Block-Plugin bieten einige allgemeine Eigenschaften (»Properties«), die jedem Blocktyp zugeordnet werden können. Daneben gibt es Eigenschaften, die spezifisch für die Blocktypen *Textline, Textflow*, *Image, PDF* und *Graphics* sind.

Eigenschaften unterstützen dieselben Datentypen wie Optionslisten mit Ausnahme von Handles und Aktionslisten. Viele Blockeigenschaften heißen wie die Optionen für API-Funktionen wie *PDF_fit_image( ), PDF_fit_textline( )* (zum Beispiel *fitmethod, charspacing)*. In solchen Fällen verhält sich die Eigenschaft genau so wie die gleichnamige Option.

13.7.1 Administrative Eigenschaften

Administrative Eigenschaften beziehen sich auf alle Arten von Blöcken. Alle erforderlichen Einträge werden vom Block-Plugin automatisch generiert. Tabelle 13.4 gibt eine Übersicht über die administrativen Eigenschaften.

Tabelle 13.4 Administrative Blockeigenschaften

Schlüsselwort	Mögliche Werte und Erklärung
Description	*(String) Vom Benutzer lesbare Beschreibung der Funktion des Blocks, die in PDFDocEncoding oder Unicode kodiert ist (und im letzteren Fall mit einem BOM beginnt). Diese Eigenschaft dient nur zur Information des Benutzers und wird von PPS ignoriert.*
Locked	*(Boolean) Ist diese Eigenschaft gleich* `true`*, lassen sich der Block und seine Eigenschaften nicht mit dem Block-Plugin bearbeiten. Diese Eigenschaft wird von PPS ignoriert. Standardwert:* `false`
Name	*(String; erforderlich) Name des Blocks. Blocknamen müssen auf der Seite, aber nicht innerhalb des Dokuments eindeutig sein. Die drei Zeichen* `[ ] /` *sind in Blocknamen nicht erlaubt. Blocknamen dürfen maximal 125 Zeichen lang sein.*
Subtype	*(Schlüsselwort; erforderlich) Je nach Blocktyp entweder* `Text`, `Image`, `PDF` *oder* `Graphics`*. Beachten Sie, dass Textline- und Textflow-Blöcke als Subtyp beide* `Text` *haben, aber durch die Eigenschaft* `textflow` *unterschieden werden.*
textflow	*(Boolean) Steuert die ein- oder mehrzeilige Verarbeitung. Diese Eigenschaft ist nicht explizit in der Benutzeroberfläche des Block-Plugin verfügbar, sondern als Blocktyp Textline oder Textflow dargestellt (Standardwert:* `false`*):* ***false*** *Textline-Block: Text muss einzeilig sein und wird mit PDF_fit_textline( ) verarbeitet.* ***true*** *Textflow-Block: Text kann mehrzeilig sein und wird mit PDF_fit_textflow( ) verarbeitet. Neben den vordefinierten Eigenschaften für Text können alle Textflow-spezifischen Eigenschaften festgelegt werden (siehe Tabelle 13.9).*
Type	*(Schlüsselwort; erforderlich) Immer* `Block`

13.7.2 Eigenschaften für Rechtecke

Eigenschaften für Rechtecke beziehen sich auf alle Arten von Blocktypen. Sie legen das Aussehen des Blockrechtecks fest. Alle erforderlichen Einträge werden vom Block-Plugin automatisch generiert. Tabelle 13.5 gibt eine Übersicht über die Eigenschaften für Rechtecke.

Tabelle 13.5 Blockeigenschaften für Rechtecke

Schlüsselwort	Mögliche Werte und Erklärung
backgroundcolor	*(Color) Ist diese Eigenschaft vorhanden und enthält sie ein von* None *verschiedenes Schlüsselwort zur Festlegung des Farbraums, wird ein Rechteck gezeichnet und mit der angegebenen Farbe gefüllt. Damit lassen sich vorhandene Seiteninhalte verdecken. Standardwert:* None
bordercolor	*(Color) Ist diese Eigenschaft vorhanden und enthält sie ein von* None *verschiedenes Schlüsselwort zur Festlegung des Farbraums, wird ein Rechteck mit einem Rand in der angegebenen Farbe gezeichnet. Standardwert:* None
linewidth	*(Float; muss größer* 0 *sein) Breite der Linie, mit der das Blockrechteck gezeichnet wird; wird nur verwendet, wenn* `bordercolor` *gesetzt ist. Standardwert: 1*
Rect	*(Rechteck; erforderlich) Blockkoordinaten. Der Ursprung des Koordinatensystems liegt in der linken unteren Ecke der Seite. Das Block-Plugin zeigt die Koordinaten aber in der Notation von Acrobat an, das heißt, mit dem Ursprung in der linken oberen Ecke der Seite. Die Koordinaten werden in der Einheit angezeigt, die in Acrobat gerade ausgewählt ist. In der PDF-Datei werden sie immer in Punkt gespeichert.*
Status	*(Schlüsselwort) Beschreibt, wie der Block von PPS und der Vorschau-Funktion verarbeitet wird. (Standardwert:* `active`*):* ***active*** *Der Block wird entsprechend seiner Eigenschaften komplett verarbeitet.* ***ignore*** *Der Block wird ignoriert.* ***ignoredefault*** *Wie* `active`*, außer dass die Eigenschaften und Optionen* `defaulttext/image/pdf/graphics` *ignoriert werden, d.h. der Block bleibt leer, wenn keine variablen Inhalte verfügbar sind (vor allem in der Vorschau). Damit können Sie sicherstellen, dass die Block-Vorgabewerte nicht zum serverseitigen Füllen der Blöcke verwendet werden, obwohl der Block-Vorgabewerte zur Erzeugung der Vorschau enthalten kann. Es kann auch verwendet werden, um die Vorgabewerte für die Block-Vorschau zu deaktivieren, ohne die Vorgabewerte aus den Blockeigenschaften zu entfernen.* ***static*** *Es wird kein variabler Inhalt platziert, sondern der Vorgabewert für Text, Rasterbild, PDF oder Grafik wird verwendet, sofern vorhanden.*

13.7.3 Darstellungsspezifische Eigenschaften

Eigenschaften für die Darstellung legen Formatdetails fest:

- Tabelle 13.6 legt transparenzspezifische Eigenschaften fest.
- Tabelle 13.7 legt textspezifische Eigenschaften für Blöcke vom Typ Textline und Textflow fest.

Tabelle 13.6 Transparenzspezifische Eigenschaften für alle Blocktypen

Schlüsselwort	Mögliche Werte und Erklärung
blendmode	*(Liste von Schlüsselwörtern; Im PDF/A-1-Modus muss der Wert* `Normal` *verwendet werden) Name des Farbmischmodus:* None, `Color`, `ColorDodge`, `ColorBurn`, `Darken`, `Difference`, `Exclusion`, `HardLight`, `Hue`, `Lighten`, `Luminosity`, `Multiply`, None, `Normal`, `Overlay`, `Saturation`, `Screen`, `SoftLight`. *Standardwert:* None
opacityfill	*(Float; Im PDF/A-1-Modus muss der Wert 1 verwendet werden) Deckkraft für das Füllen von Flächen im Bereich 0… 1. Der Wert 0 bedeutet vollständig transparent; 1 bedeutet völlig undurchsichtig.*
opacitystroke	*(Float; Im PDF/A-1- Modus muss der Wert 1 verwendet werden) Deckkraft für das Durchziehen von Linien im Bereich 0… 1. Der Wert 0 bedeutet vollständig transparent; 1 bedeutet völlig undurchsichtig.*

Tabelle 13.7 Textspezifische Eigenschaften für Blöcke vom Typ Textline und Textflow

Schlüsselwort	Mögliche Werte und Erklärung
charspacing	*(Float oder Prozentwert) Zeichenabstand. Prozentwerte basieren auf* `fontsize`. *Standardwert: 0*
decoration-above	*(Boolean) Bei* `true` *wird die mit den Optionen* `underline`, `strikeout` *und* `overline` *aktivierte Textdekoration über dem Text gezeichnet und sonst unter den Text. Wird die Reihenfolge der Textausgabe geändert, wirkt sich dies auf die Sichtbarkeit der Dekorationslinien aus. Standardwert:* `false`
fillcolor	*(Color) Füllfarbe des Textes. Standardwert:* `gray 0` *(=schwarz)*
fontname[1]	*(String) Name des Fonts, so wie bei PDF_load_font( ) erforderlich. Das Block-Plugin zeigt alle installierten Systemfonts in einer Liste an. Beachten Sie jedoch, dass diese Fontnamen nicht unbedingt zwischen macOS, Windows und Unix portierbar sind. Wenn* `fontname` *mit einem* `@`*-Zeichen beginnt, wird der Font in vertikalem Textausgabemodus angewendet.* *Der Encoding für den Text muss beim Füllen der Blöcke als Option für PDF_fill_textblock( ) angegeben werden, es sei denn, die Option* `font` *wurde benutzt.*
fontsize[1]	*(Float) Größe des Fonts in Punkt*
horizscaling	*(Float oder Prozentwert) Horizontale Skalierung von Text. Standardwert: 100%*
italicangle	*(Float) Neigungswinkel von kursivem Text in Grad. Standardwert: 0*
kerning	*(Boolean) Unterschneidung. Standardwert:* `false`
overline	*(Boolean) Modus für Überstreichen. Standardwert:* `false`
shadow	*(Composite) Schatteneffekt erzeugen (Standardwert: kein Schatteneffekt). Es gibt folgende Untereigenschaften:* **fillcolor** *(Color) Füllfarbe des Texts. Standardwert:* `{gray 0.8}` **offset** *(Liste aus Floats oder Prozentwert) Bestimmt den Schatten-Offset vom Referenzpunkt des Textes in Benutzerkoordinaten oder als Prozentwert der Fontgröße. Standardwert:* `{5% -5%}`
strikeout	*(Boolean) Modus für Durchstreichen. Standardwert:* `false`
strokecolor	*(Color) Farbe, in der Text gezeichnet wird. Standardwert:* `gray 0` *(=schwarz)*
strokewidth	*(Float, Prozentwert oder Schlüsselwort; nur wirksam, wenn* `textrendering` *auf* `Umrisslinien zeichnen` *gesetzt ist) Linienstärke für Umrisslinien (in Benutzerkoordinaten oder als Prozentwert von* `fontsize`*). Das Schlüsselwort* `auto` *oder der äquivalente Wert 0 verwenden einen eingebauten Standardwert. Standardwert:* `auto`
textrendering	*(Integer) Darstellungsmodus für Text. Nur der Wert 3 hat einen Einfluss auf Typ-3-Fonts (Standardwert:* `0`*):* **0**  *Text füllen** **1**  *Umrisslinien zeichnen** **2**  *Umrisslinien zeichnen, füllen** **3** *Unsichtbarer Text** **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 (nicht für Blöcke)**
textrise	*(Float oder Prozentwert) Wert für den vertikalen Textversatz. Prozentwerte basieren auf* `fontsize`. *Standardwert: 0*
underline	*(Boolean) Modus für Unterstreichen. Standardwert:* `false`

Tabelle 13.7 Textspezifische Eigenschaften für Blöcke vom Typ Textline und Textflow

Schlüsselwort	Mögliche Werte und Erklärung
underline-position	*(Float, Prozentwert oder Schlüsselwort) Position der Linie für unterstrichenen Text, relativ zur Grundlinie. Prozentwerte basieren auf* fontsize. *Standardwert:* auto
underline-width	*(Float, Prozentwert oder Schlüsselwort) Stärke der Linie für unterstrichenen Text. Prozentwerte basieren auf* fontsize. *Standardwert:* auto
wordspacing	*(Float oder Prozentwert) Wortabstand. Prozentwerte basieren auf* fontsize. *Standardwert: 0*

1. Diese Eigenschaft wird für Textline- und Textflow-Blöcke benötigt; sie wird vom Block-Plugin unterstützt.

13.7.4 Eigenschaften zur Textvorbereitung

Eigenschaften zur Textvorbereitung legen Vorverarbeitungsschritte für Textline- und Textflow-Blöcke fest. Tabelle 13.8 gibt eine Übersicht über Eigenschaften zur Textvorbereitung für Textline- und Textflow-Blöcke.

Tabelle 13.8 Eigenschaften zur Textvorbereitung für Textline- und Textflow-Blöcke

Schlüsselwort	Mögliche Werte und Erklärung
charref	*(Boolean) Bei* true *werden numerische Referenzen, Character-Referenzen und Glyphnamen-Referenzen ersetzt. Standardwert: globale Option* charref
escape-sequence	*(Boolean) Mit* true *wird die Ersetzung von Escape-Sequenzen in Content-Strings, Hypertext-Strings und Name-Strings aktiviert. Standardwert: globale Option* escapesequence
features	*(Liste von Schlüsselwörtern) Bestimmt, welche typografischen Features eines OpenType-Fonts auf den Text angewendet werden, entsprechend der Optionen* script *und* language. *Schlüsselwörter für im Font nicht vorhandene Features werden ignoriert. Die folgenden Schlüsselwörter können übergeben werden:* **_none** *Kein Feature auf den Font anwenden. Als Ausnahme muss das Feature* vert *explizit mit dem Schlüsselwort* novert *deaktiviert werden.* **<name>** *Aktivierung eines Features durch Angabe seiner vier Zeichen langen OpenType-Bezeichnung. Einige häufig benutzte Feature-Namen sind* liga, ital, tnum, smcp, swsh, zero. *Die vollständige Liste der Namen und Beschreibungen aller unterstützten Features finden Sie in Abschnitt 7.3.1, »Unterstützte OpenType-Features«, Seite 187.* ***no<name>*** *Das Präfix* no *vor einem Feature-Namen (z.B.* noliga*) deaktiviert dieses Feature.* *Standardwert:* _none *für horizontalen Textausgabemodus. Im vertikalen Textausgabemodus wird automatisch* vert *verwendet.* *Für die Unterstützung von OpenType-Features ist die Option* readfeatures *in PDF_load_font( ) erforderlich.*
language	*(Schlüsselwort; nur wenn* script *übergeben wird) Der Text wird bezüglich der angegebenen Sprache verarbeitet, was für die Optionen* features *und* shaping *relevant ist. Eine vollständige Liste aller Schlüsselwörter finden Sie in Abschnitt 7.4.2, »Schrift und Sprache«, Seite 197, zum Beispiel* ARA *(Arabisch),* JAN *(Japanisch),* HIN *(Hindi). Standardwert:* _none *(Sprache nicht definiert)*
script	*(Schlüsselwort; erforderlich, falls* shaping=true*) Der Text wird bezüglich des verwendeten Schriftsystems verarbeitet, das für die Optionen* features, shaping *und* advancedlinebreaking *festgelegt wurde. Die gängigsten Schlüsselwörter für Schriftsysteme sind:* _none *((Schriftsystem nicht definiert),* latn, grek, cyrl, armn, hebr, arab, deva, beng, guru, gujr, orya, taml, thai, laoo, tibt, hang, kana, han. *Das Schlüsselwort* _auto *wählt das Schriftsystem, zu dem die meisten Zeichen im Text gehören, wobei* _latn *und* _none *ignoriert werden. Eine vollständige Liste aller Schlüsselwörter finden Sie in Abschnitt 7.4.2, »Schrift und Sprache«, Seite 197. Standardwert:* _none
shaping	*(Boolean) Mit* true *wird der Text entsprechend der Optionen* script *und* language *formatiert (shaped). Die Option* script *muss einen Wert ungleich* _none *haben und die erforderlichen Shaping-Tabellen müssen im Font vorhanden sein. Standardwert:* false

13.7.5 Eigenschaften für die Textformatierung

Tabelle 13.9 gibt eine Übersicht über Eigenschaften, die nur mit Textflow-Blöcken verwendet werden können, mit Ausnahme der Eigenschaft *stamp*, die auch mit Textline-Blöcken verwendet werden kann. Anhand dieser Eigenschaften wird die anfängliche Optionsliste zur Textflow-Verarbeitung zusammengestellt (entsprechend dem Parameter *optlist* für *PDF_create_textflow( )*). Mit dem Block-Plugin können keine Inline-Optionslisten für Textflow festgelegt werden. Diese lassen sich jedoch auf dem Server als Bestandteil des Textinhalts übergeben, wenn der Block mit *PDF_fill_textblock( )* gefüllt wird. Auch der Vorgabetext in der Eigenschaft *defaulttext* kann Inline-Optionen enthalten.

Tabelle 13.9 Eigenschaften für die Textformatierung (fast alle nur für Textflow-Blöcke)

Schlüsselwort	Mögliche Werte und Erklärung
adjustmethod	*(Schlüsselwort) Methode zur Anpassung von Textteilen, die nach einer Vergrößerung oder Verkleinerung des Wortabstands* `minspacing` *und* `maxspacing` *nicht mehr in die Zeile passen. (Standardwert:* `auto`*):* **auto** — *Folgende Methoden werden in der angeführten Reihenfolge angewandt:* `shrink`, `spread`, `nofit`, `split`. **clip** — *Wie* `nofit`*, nur dass der längere Teil am rechten Rand der Fitbox (unter Berücksichtigung der Option* `rightindent`*) abgeschnitten wird.* **nofit** — *Das letzte Wort wird in die nächste Zeile verschoben, sofern die verbleibende (kurze) Zeile nicht kürzer als der in der Option* `nofitlimit` *festgelegte Prozentwert ist. Auch Absätze im Blocksatz sehen bei dieser Methode leicht ausgefranst aus.* **shrink** — *Passt ein Wort nicht in die Zeile, wird der Text so lange gestaucht, bis das Wort hineinpasst, sofern die Option* `shrinklimit` *dies zulässt. Falls der Text immer noch nicht passt, wird die Methode* **nofit** *angewendet.* **split** — *Das letzte Wort wird nicht in die nächste Zeile verschoben, sondern zwangsweise getrennt. Bei Textfonts (nicht aber bei Symbolfonts) wird ein Trennzeichen eingefügt.* **spread** — *Das letzte Wort wird in die nächste Zeile gestellt und die verbleibende kurze Zeile durch Sperren des Textes auf Blocksatz gebracht, sofern die Option* `spreadlimit` *dies zulässt. Kann kein Blocksatz erreicht werden, kommt die Methode* `nofit` *zum Einsatz.*
advanced-linebreak	*(Boolean) Aktiviert einen Algorithmus für fortgeschrittenen Zeilenumbruch bei komplexen Schriftsystemen. Dies ist für den Zeilenumbruch bei solchen Schriftsystemen erforderlich, die Wortgrenzen nicht mit Leerzeichen kennzeichnen, wie zum Beispiel bei Thai. Die Optionen* `locale` *und* `script` *werden berücksichtigt. Standardwert:* `false`
alignment	*(Schlüsselwort) Legt die Formatierung für die Zeilen eines Absatzes fest. Standardwert:* `left`. **left** — *linksbündig, beginnend bei* `leftindent` **center** — *mittig zwischen* `leftindent` *und* `rightindent` **right** — *rechtsbündig, bei* `rightindent` *endend* **justify** — *links- und rechtsbündig (Blocksatz)*
avoidempty-begin	*(Boolean) Bei* `true` *werden Leerzeilen am Anfang einer Fitbox gelöscht. Standardwert:* `false`
fixedleading	*(Boolean) Bei* `true` *wird der beim ersten Zeichen einer Zeile geltende Zeilenabstand verwendet. Andernfalls wird der größte Zeilenabstand in der Zeile verwendet. Standardwert:* `false`

Tabelle 13.9 Eigenschaften für die Textformatierung (fast alle nur für Textflow-Blöcke)

Schlüsselwort	Mögliche Werte und Erklärung
hortab-method	*(Schlüsselwort) Legt die Interpretation von horizontalen Tabulatoren im Text fest. Liegt die berechnete Position links der aktuellen Textposition, so wird der Tabulator ignoriert (Standardwert:* `relative`*):* ***relative*** *Die Position wird um den in* `hortabsize` *festgelegten Betrag vorgerückt.* ***typewriter*** *Die Position wird auf das nächste Vielfache von* `hortabsize` *vorgerückt.* ***ruler*** *Die Position wird auf den n-ten in der Option* ***ruler*** *verfügbaren Tabulatorwert gesetzt, wobei* ***n*** *die Anzahl der bislang in der Textzeile vorgekommenen Tabs bezeichnet. Ist n größer als die Anzahl der Tabulatorpositionen, kommt die Methode* ***relative*** *zum Einsatz.*
hortabsize	*(Float oder Prozentwert) Legt die Breite eines horizontalen Tabulators fest[1]. Die Interpretation wird von der Option* `hortabmethod` *gesteuert. Standardwert: 7.5%*
lastalignment	*(Schlüsselwort) Bestimmt die Formatierung der letzten Zeile eines Absatzes. Neben allen Schlüsselwörtern der Option* `alignment` *wird das folgende Schlüsselwort unterstützt (Standardwert:* `auto`*):* ***auto*** *Es wird der Wert der Option* `alignment` *verwendet. Nur bei* `justify` *wird left verwendet.*
leading	*(Float oder Prozentwert) Zeilenabstand in Benutzerkoordinaten oder als prozentualer Anteil der Fontgröße. Standardwert: 100%*
locale	*(Schlüsselwort) Bestimmt die Spracheinstellung für lokalisierte Zeilenumbruch-Methoden, wenn* `advancedlinebreak=true`*. Das Schlüsselwort besteht aus einer oder mehreren Komponenten, wobei die optionalen Komponenten durch einen Unterstrich '_' getrennt sind. (Die Syntax unterscheidet sich geringfügig von NLS/POSIX locale IDs):* ► *Ein erforderlicher Sprachcode aus zwei oder drei Kleinbuchstaben nach ISO 639-2 (siehe* `www.loc.gov/standards/iso639-2`*), z.B.* `en`*, (Englisch),* `de` *(Deutsch),* `ja` *(Japanisch). Dieser unterscheidet sich von der Option* `language`*.* ► *Ein optionaler vierbuchstabiger Schriftsystem-Code nach ISO 15924, z.B.* `Hira` *(Hiragana),* `Hebr` *(Hebräisch),* `Arab` *(Arabisch),* `Thai` *(Thai).* ► *Ein optionaler Ländercode aus zwei Großbuchstaben nach ISO 3166, z.B.* `DE` *(Deutschland),* `CH` *(Schweiz),* `GB` *(Großbritannien)* *Für den erweiterten Zeilenumbruch ist keine Spracheinstellung erforderlich: mit dem Schlüsselwort* `_none` *wird keine Sprache eingestellt. Standardwert:* `_none` *Beispiele:* `de_DE`, `en_US`, `en_GB`
maxspacing minspacing	*(Float oder Prozentwert) Bestimmt den maximalen bzw. minimalen Abstand zwischen Wörtern (in Benutzerkoordinaten oder als prozentualer Anteil der Breite eines Leerzeichens). Der berechnete Wortabstand wird durch die hier übergebenen Werte begrenzt, aber der Wert der Option* `wordspacing` *wird noch addiert. Standardwerte:* `minspacing=50%`, `maxspacing=500%`
minlinecount	*(Integer) Mindestanzahl von Zeilen im letzten Absatz der Fitbox. Wenn es weniger Zeilen sind, werden sie in die nächste Fitbox verschoben. Mit dem Wert 2 werden einzelne Zeilen eines Absatzes am Ende einer Fitbox verhindert (»Schusterjunge«). Standardwert: 1*
nofitlimit	*(Float oder Prozentwert) Minimal erlaubte Länge einer Zeile bei der Methode* `nofit` *(in Benutzerkoordinaten oder als Prozentwert der Breite der Fitbox). Standardwert: 75%*
parindent	*(Float oder Prozentwert) Legt den linken Einzug der ersten Zeile eines Absatzes fest[1]. Der Wert wird zu* `leftindent` *addiert. Wird diese Option innerhalb der Zeile angegeben, so wirkt sie wie ein Tabulator. Standardwert: 0*
rightindent leftindent	*(Float oder Prozentwert) Bestimmt den rechten bzw. linken Einzug aller Textzeilen[1]. Wird* `leftindent` *innerhalb der Zeile angegeben und befindet sich die definierte Position links der aktuellen Textposition, so wird die Option für die aktuelle Zeile ignoriert. Standardwert: 0*
ruler[2]	*(Liste von Floats oder Prozentwerten) Liste der absoluten Tabulatorpositionen für* `hortabmethod=ruler`[1]*. Die Liste darf maximal 32 nicht negative Einträge in aufsteigender Reihenfolge enthalten. Standardwert: Vielfache von* `hortabsize` *als Integers*

Tabelle 13.9 Eigenschaften für die Textformatierung (fast alle nur für Textflow-Blöcke)

Schlüsselwort	Mögliche Werte und Erklärung
shrinklimit	*(Prozentwert) Untere Grenze für das Stauchen von Text mit der Methode* `shrink`*. Der berechnete Stauchungsfaktor wird durch den hier übergebenen Wert begrenzt, aber noch mit dem Wert der Option* `horizscaling` *multipliziert. Standardwert: 85%*
spreadlimit	*(Float oder Prozentwert) Obere Grenze für den Abstand zwischen zwei Zeichen bei der Methode* `spread` *(in Benutzerkoordinaten oder als prozentualer Anteil der Fontgröße); der berechnete Zeichenabstand wird durch den hier übergebenen Wert begrenzt, aber der Wert der Option* `charspacing` *wird noch addiert. Standardwert: 0*
stamp	*(Schlüsselwort; Textline- und Textflow-Blöcke) Mit dieser Option lässt sich ein diagonaler Stempel im Blockrechteck erzeugen. Der Stempeltext wird dabei so groß wie möglich gesetzt. Bei der Positionierung des Stempeltexts in der Box werden die Optionen* `position`*,* `fitmethod` *und* `orientate` *(nur* `north` *und* `south`*) berücksichtigt. Standardwert:* none. **ll2ur** *Der Stempel verläuft diagonal von der linken unteren zur rechten oberen Ecke.* **ul2lr** *Der Stempel verläuft diagonal vom der linken oberen zur rechten unteren Ecke.* **none** *Es wird kein Stempel erzeugt.*
tabalignchar	*(Integer) Unicode-Wert des Zeichens, an dem dezimale Tabulatoren ausgerichtet werden. Standardwert: das Zeichen '.' (U+002E)*
tabalignment[2]	*(Liste aus Schlüsselwörtern) Ausrichtung für Tabulatoren. Jeder Listeneintrag definiert die Ausrichtung des entsprechenden Eintrags in der Option* `ruler` *(Standardwert:* `left`*):* **center** *Text wird mittig an der Tabulatorposition ausgerichtet.* **decimal** *Das erste* `tabalignchar`*-Zeichen wird linksbündig an der Tabulatorposition ausgerichtet. Ist kein* `tabalignchar`*-Zeichen vorhanden, wird rechtsbündig ausgerichtet.* **left** *Text wird linksbündig an der Tabulatorposition ausgerichtet.* **right** *Text wird rechtsbündig an der Tabulatorposition ausgerichtet.*

1. In Benutzerkoordinaten oder als prozentualer Anteil an der Breite der Fitbox
2. Die Tabulatorpositionen werden im Blockeigenschaften-Dialog in der Eigenschaftengruppe `Tabulatoren für Textflow für hortabmethod=ruler` *festgelegt.*

13.7.6 Eigenschaften für die Objekteinpassung

Eigenschaften für die Objekteinpassung stehen für alle Blocktypen zur Verfügung, obwohl einige Eigenschaften nur auf bestimmte Blocktypen angewendet werden können. Sie steuern, wie Inhalte in die Blöcke eingepasst werden:

- Tabelle 13.10 gibt eine Übersicht über Einpassungseigenschaften für Blöcke vom Typ Textline, Image, PDF und Graphics.
- Tabelle 13.11 gibt eine Übersicht über Einpassungseigenschaften für Textflow-Blöcke (die meisten beziehen sich auf vertikale Einpassung).

Der Algorithmus für die Objekteinpassung verwendet das Block-Rechteck als Fitbox. Außer bei *fitmethod=clip* wird der Text nicht beschnitten. Um sicherzugehen, dass der Blockinhalt nicht über das Blockrechteck hinausgeht, vermeiden Sie die Methode *fitmethod =nofit*.

Tabelle 13.10 Einpassungseigenschaften für Blöcke vom Typ Textline, Image, PDF und Graphics

Schlüsselwort	Mögliche Werte und Erklärung
alignchar	*(Unichar oder Schlüsselwort) Ist das hier definierte Zeichen im Text vorhanden, wird dessen linke untere Ecke am Referenzpunkt ausgerichtet. Bei horizontalem Text mit* `orientate=north` *oder* `south` *definiert der erste in der Option* `position` *übergebene Wert die Position. Bei horizontalem Text mit* `orientate=west` *oder* `east` *definiert der zweite in der Option* `position` *übergebene Wert die Position. Diese Option wird ignoriert, wenn das hier definierte Zeichen im Text nicht vorhanden ist. Beim Wert 0 und dem Schlüsselwort* `none` *erfolgt keine Ausrichtung. Es wird die definierte* `fitmethod` *angewandt, selbst wenn der Text wegen der Ausrichtung nach* `alignchar` *nicht innerhalb der Fitbox platziert werden kann. Standardwert:* `none`
dpi	*(Float-Liste; nur für Blocktyp Image) Einer oder zwei Werte, die die gewünschte Bildauflösung in Pixeln pro Zoll in horizontaler und vertikaler Richtung angeben. Beim Wert 0 wird die interne Bildauflösung verwendet, sofern vorhanden, andernfalls 72 dpi. Diese Eigenschaft wird ignoriert, wenn die Eigenschaft* `fitmethod` *mit einem der Schlüsselwörter* `auto`, `meet`, `slice` *oder* `entire` *übergeben wurde. Standardwert: 0*
fitmethod	*(Schlüsselwort) Strategie für den Fall, dass der übergebene Inhalt nicht in die Box passt:* `auto`, `clip`, `entire`, `meet`, `nofit` *und* `slice` *(Standardwert:* `meet`*).*
margin	*(Float-Liste; nur für Blocktyp Textline) Ein oder zwei Float-Werte für eine zusätzliche horizontale und vertikale Verkleinerung des Blockrechtecks. Standardwert: 0*
orientate	*(Schlüsselwort) Legt fest, in welcher Ausrichtung der Inhalt platziert wird. Mögliche Werte sind* `north`, `east`, `south`, `west`. *Standardwert:* `north`
position	*(Float-Liste) Ein oder zwei Werte, die die Position des Referenzpunkts innerhalb des Inhalts festlegen. Die Position wird als Prozentwert innerhalb des Blocks angegeben. Nur für Blöcke vom Typ Textline: Das Schlüsselwort* `auto` *kann für den ersten Wert in der Liste verwendet werden. Es bedeutet* `rechts`*, wenn die Schreibrichtung von rechts nach links ist (z.B. in arabischen oder hebräischen Texten) und ansonsten* `links` *(z.B. für Texte in lateinischer Schrift).* *Standardwert: {0 0}, d.h. die linke untere Ecke*
rotate	*(Float) Drehwinkel in Grad, um den der Block gegen den Uhrzeigersinn gedreht wird, bevor die Verarbeitung beginnt. Das Rotationszentrum ist der Referenzpunkt. Standardwert: 0*
scale	*(Float-Liste; nur für Blöcke vom Typ Image, PDF und Graphics) Einer oder zwei Werte, die den oder die gewünschten Skalierungsfaktoren in horizontaler und vertikaler Richtung festlegen. Diese Eigenschaft wird ignoriert, wenn die Eigenschaft* `fitmethod` *mit einem der Schlüsselwörter* `auto`, `meet`, `slice` *oder* `entire` *übergeben wurde. Standardwert: 1*
shrinklimit	*(Float oder Prozentwert; nur für Blocktyp Textline) Untere Grenze für das Stauchen von Text mit der Methode* `fitmethod=auto`*. Standardwert: 0,75*

Tabelle 13.11 Einpassungseigenschaften für Textflow-Blöcke

Schlüsselwort	Mögliche Werte und Erklärung
firstlinedist	*(Float, Prozentwert oder Schlüsselwort) Abstand zwischen dem oberen Rand der Fitbox und der Grundlinie der ersten Textzeile. Angegeben wird er in Benutzerkoordinaten, als Prozentsatz der Fontgröße (der Größe der ersten in der Zeile auftretenden Schrift, wenn* `fixedleading=true`*, oder der maximal auftretenden Fontgröße andernfalls) oder als Schlüsselwort (Standardwert:* `leading`*):* **leading** *Für die erste Zeile ermittelter Zeilenabstand; diakritische Zeichen wie À berühren dabei den oberen Rand der Fitbox.* **ascender** *Für die erste Zeile ermittelte Oberlänge; Zeichen mit großer Oberlänge wie d oder h berühren den oberen Rand der Fitbox.* **capheight** *Für die erste Zeile ermittelte Versalhöhe; hohe Großbuchstaben wie H berühren dabei den oberen Rand der Fitbox.* **xheight** *Für die erste Zeile ermittelte x-Höhe; Kleinbuchstaben wie x berühren den oberen Rand der Fitbox.* *Ist* `fixedleading=false`*, wird der größte Wert verwendet, der für Zeilenabstand, Oberlänge, x-Höhe und Versalhöhe in der ersten Zeile ermittelt wurde.*
fitmethod	*(Schlüsselwort) Strategie für den Fall, dass der übergebene Inhalt nicht in die Box passt:* **auto** `fontsize` *und* `leading` *werden so lange reduziert, bis der Text passt.* **clip** *Der Text wird am Blockrand abgeschnitten.* **nofit** *Der Text fließt aus dem unteren Blockrand heraus.* *Standardwert:* `clip`*, falls die Option* `textflowhandle` *angegeben wurde, andernfalls* `auto`
lastlinedist	*(Float, Prozentwert oder Schlüsselwort; wird ignoriert bei* `fitmethod=nofit`*) Der kleinste Abstand zwischen der Grundlinie der letzten Textzeile und dem unteren Rand der Fitbox. Angegeben wird er in Benutzerkoordinaten, als Prozentsatz der Fontgröße (der ersten in der Zeile auftretenden Fontgröße, wenn* `fixedleading=true` *oder andernfalls der maximal in der Zeile auftretenden Fontgröße) oder als Schlüsselwort. Standardwert: 0, d.h. der untere Rand der Fitbox wird als Grundlinie verwendet und die normalen Unterlängen reichen aus der Fitbox hinaus.* **descender** *Für die letzte Zeile ermittelte Unterlänge; Zeichen mit Unterlängen wie* `g` *oder* `j` *berühren dabei den unteren Rand der Fitbox.* *Ist* `fixedleading=false` *wird der größte Wert verwendet, der in der letzten Zeile für die Unterlänge ermittelt wurde.*
linespread_limit	*(Float oder Prozentwert; nur für* `verticalalign=justify`*) Größter Wert in Benutzerkoordinaten oder als Prozentsatz des Zeilenabstands, um den der Zeilenabstand bei vertikaler Ausrichtung erhöht wird. Standardwert: 200%*
maxlines	*(Integer oder Schlüsselwort) Maximale Anzahl der Zeilen in der Fitbox oder das Schlüsselwort* `auto`*, bei dem möglichst viele Zeilen in der Fitbox platziert werden. Nach der Platzierung der maximalen Anzahl von Zeilen gibt PDF_fit_textflow( ) den String* `_boxfull` *zurück.*
minfontsize	*(Float oder Prozentwert) Mindestgröße des Fonts, in der der Text mit* `fitmethod=auto` *bei überschrittenem* `shrinklimit` *erstellt werden darf. Der Grenzwert wird in Benutzerkoordinaten oder als Prozentsatz der Blockhöhe angegeben. Wenn der Grenzwert erreicht ist, wird der Text mit der Fontgröße* `minfontsize` *erstellt. Standardwert: 0.1%*
orientate	*(Schlüsselwort) Legt fest, in welcher Ausrichtung der Inhalt platziert wird. Mögliche Werte sind* `north`*,* `east`*,* `south`*,* `west`*. Standardwert:* `north`

Tabelle 13.11 Einpassungseigenschaften für Textflow-Blöcke

Schlüsselwort	Mögliche Werte und Erklärung
rotate	*(Float) Dreht das Koordinatensystem, wobei die linke untere Ecke der Fitbox als Mittelpunkt und der übergebene Wert als Drehwinkel in Grad benutzt wird. Dabei werden die Box und der Text gedreht. Die Drehung wird gesetzt, nachdem der Text platziert wurde. Standardwert: 0*
verticalalign	*(Schlüsselwort) Vertikale Ausrichtung des Texts in der Fitbox (Standardwert:* `top`*):*
	top *Die Formatierung beginnt in der ersten Zeile und setzt sich nach unten fort. Füllt der Text die Fitbox nicht vollständig aus, bleibt Weißraum unter dem Text.*
	center *Der Text wird vertikal in der Fitbox zentriert. Füllt der Text die Fitbox nicht vollständig aus, bleibt Weißraum über und unter dem Text.*
	bottom *Die Formatierung beginnt in der letzten Zeile und setzt sich nach oben fort. Füllt der Text die Fitbox nicht vollständig aus, bleibt Weißraum über dem Text.*
	justify *Der Text wird am oberen und unteren Rand der Fitbox ausgerichtet. Dazu wird der Zeilenabstand bis zur durch* `linespreadlimit` *festgelegten Grenze erhöht. Die Höhe der ersten Zeile wird nur bei* `firstlinedist=leading` *vergrößert.*

13.7.7 Vorgabewerte

Eigenschaften für Vorgabewerte definieren den Blockinhalt, wenn keine spezifischen Inhalte bereitgestellt werden. Sie sind besonders nützlich für die Vorschau-Funktion, da die Blöcke dann mit ihren Vorgabewerten gefüllt werden. Tabelle 13.12 gibt eine Übersicht.

Tabelle 13.12 Blockeigenschaften für Vorgabewerte

Schlüsselwort	Mögliche Werte und Erklärung
default-graphics	*(String; nur für Blocktyp Graphics) Pfadname der Grafikdatei, die verwendet wird, wenn vom Client keine Grafik übergeben wird.*[1]
defaultimage	*(String; nur für Blocktyp Image) Pfadname des Bildes, das verwendet wird, wenn vom Client kein Bild übergeben wird.*[1]
defaultpdf	*(String; nur für Blocktyp PDF) Pfadname des PDF-Dokuments, das verwendet wird.*[1]
default-pdfpage	*(Integer; nur für Blocktyp PDF) Nummer der Seite im Vorgabe-PDF-Dokument. Standardwert: 1*
defaulttext	*(String; nur für Blocktyp Textline und Textflow) Text, der verwendet wird, wenn vom Client kein Ersatztext übergeben wird.*[2]

1. Dateinamen sollten ohne absolute Pfadangaben verwendet werden; stattdessen sollte in der PPS-Client-Anwendung mit der Funktionalität `SearchPath` *gearbeitet werden. Dies macht die Blockverarbeitung unabhängig von der Plattform und den dateisystemspezifischen Eigenheiten.*
2. Text wird im Zeichensatz `winansi` *oder Unicode interpretiert.*

13.7.8 Benutzerdefinierte Eigenschaften

Benutzerdefinierte Eigenschaften können für Blöcke beliebigen Typs definiert werden und werden von PPS und der Vorschau-Funktion ignoriert. Tabelle 13.13 gibt eine Übersicht über die Namensregeln für benutzerdefinierte Eigenschaften.

Tabelle 13.13 Benutzerdefinierte Blockeigenschaften für alle Blocktypen

Schlüsselwort	***Mögliche Werte und Erklärung***
beliebiger Name, der die drei Zeichen [ ] / nicht enthalten darf	*(String, Name, Float oder Float-Liste) Die Interpretation der Werte benutzerdefinierter Eigenschaften liegt vollständig bei der Client-Applikation; sie werden von PPS ignoriert.*

13.8 Abfragen von Blocknamen und -eigenschaften mit pCOS

Neben automatischer Blockverarbeitung mit PPS kann die integrierte pCOS-Schnittstelle genutzt werden, um Blocknamen aufzulisten sowie Standard- und benutzerdefinierte Eigenschaften abzufragen.

Cookbook *Ein vollständiges Codebeispiel zur Abfrage der Eigenschaften von Blöcken, die in einem importierten PDF-Dokument enthalten sind, finden Sie im Cookbook-Topic* `blocks/query_block_properties`.

Ermitteln der Anzahl und Namen von Blöcken. Die Namen und die Anzahl der Blöcke auf einer importierten Seite brauchen dem Clientcode nicht bekannt zu sein, da sie abgefragt werden können. Die folgende Anweisung gibt die Anzahl der Blöcke auf der Seite *pagenum* zurück:

```
blockcount = (int) p.pcos_get_number(doc, "length:pages[" + pagenum + "]/blocks");
```

Die folgende Anweisung gibt den Namen von Block Nummer *blocknum* auf der Seite *pagenum* (die Zählung von Seiten und Blöcken beginnt bei 0):

```
blockname = p.pcos_get_string(doc,
        "pages[" + pagenum + "]/blocks[" + blocknum + "]/Name");
```

Der zurückgegebene Blockname kann im weiteren Verlauf zur Abfrage von Blockeigenschaften oder zum Füllen des Blocks mit Text, Bild, PDF- oder Grafik-Inhalt verwendet werden. Existiert der angegebene Block nicht, wird eine Exception ausgelöst. Um dies zu vermeiden, können Sie die Anzahl der Blöcke und damit den höchsten Index im Array *blocks* mit dem Präfix *length* ermitteln. Beachten Sie, dass die Blockanzahl um eins höher ist als der größtmögliche Index, da die Array-Indizierung bei 0 beginnt.

Prüfen auf Vorhandensein von Blöcken. Für mehr Flexibilität bei der Client-Anwendung können Sie das Vorhandensein eines Blocks abprüfen, bevor Sie diesen versuchen zu füllen. Dadurch kann der Designer Blöcke auf andere Seiten verschieben, ohne die Anwendung zum Füllen von Blöcken zu beeinträchtigen.

Mit dem folgenden Codefragment lässt sich prüfen, ob ein Block mit dem Namen *foo* auf der Seite vorhanden ist:

```
/* Der pCOS-Objekttyp "dictionary" bedeutet, dass der Block vorhanden ist */
if (pcos_get_string(doc, "type:pages[" + pagenum + "]/blocks/" + "foo").equals("dict"))
{
        /* Block "foo" ist auf der Seite vorhanden */
}
```

Adressierung von Blöcken durch Nummer oder Namen. In der pCOS-Pfadsyntax zur Adressierung von Blockeigenschaften sind folgende Ausdrücke gleichbedeutend, wobei davon ausgegangen wird, dass der Block mit der Nummer 6 seine Eigenschaft *Name* auf *foo* gesetzt hat:

```
pages[...]/blocks[6]
pages[...]/blocks/foo
```

Abfragen von Blockkoordinaten. Die beiden Koordinatenpaare *(llx, lly)* und *(urx, ury)*, die die linke untere und die rechte obere Ecke eines Blocks namens *foo* beschreiben, lassen sich wie folgt abfragen:

```
llx = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[0]");
lly = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[1]");
urx = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[2]");
ury = p.pcos_get_number(doc, "pages[" + pagenum + "]/blocks/foo/rect[3]");
```

Beachten Sie, dass diese Koordinaten im Standard-Koordinatensystem (mit dem Ursprung in der linken unteren Ecke, eventuell modifiziert durch die *CropBox* der Seite) übergeben werden, wohingegen das Block-Plugin die Koordinaten gemäß des Koordinatensystems der Acrobat-Benutzeroberfläche mit dem Ursprung in der linken oberen Ecke der Seite anzeigt. Die mit dem pCOS-Pseudo-Objekt *rect* (nur Kleinbuchstaben) abgefragten Werte berücksichtigen alle relevanten *CropBox-/MediaBox*- und *Rotate*-Einträge und normalisieren die Reihenfolge der Koordinaten. Dagegen können die mit dem nativen PDF-Schlüsselwort *Rect* abgefragten Werte nicht direkt als neue Koordinaten verwendet werden, falls eine CropBox vorhanden ist.

Beachten Sie zudem, dass die Option *topdown* bei der Abfrage der Blockkoordinaten nicht berücksichtigt wird.

Abfrage von benutzerdefinierten Eigenschaften. Benutzerdefinierte Eigenschaften lassen sich wie in folgendem Beispiel abfragen, in dem die Eigenschaft *zipcode* von einem Block namens *b1* auf der Seite *pagenum* abgefragt wird:

```
zip = p.pcos_get_string(doc, "pages[" + pagenum + "]/blocks/b1/Custom/zipcode");
```

Wenn Sie nicht wissen, welche benutzerdefinierten Eigenschaften in einem Block eigentlich vorhanden sind, können Sie deren Namen zur Laufzeit abfragen. Mit folgendem Codefragment ermitteln Sie zum Beispiel den Namen der ersten benutzerdefinierten Eigenschaft des Blocks *b1:*

```
propname = p.pcos_get_string(doc, "pages[" + pagenum + "]/blocks/b1/Custom[0].key");
```

Wenn Sie den Index 0 erhöhen, erhalten Sie sukzessive die Namen aller weiteren benutzerdefinierten Eigenschaften. Mit dem Präfix *length* erhalten Sie die Anzahl der benutzerdefinierten Eigenschaften.

Nicht vorhandene Blockeigenschaften und Standardwerte. Mit dem Präfix *type* ermitteln Sie, ob ein Block oder eine Eigenschaft tatsächlich vorhanden sind. Ist der Typ eines Pfads gleich 0 oder *null*, so fehlt das entsprechende Objekt im PDF-Dokument. Beachten Sie, dass dann bei vordefinierten Eigenschaften der jeweilige Standardwert verwendet wird.

Namensraum für benutzerdefinierte Eigenschaften. Zur Vermeidung von Namenskonflikten beim Austausch von PDF-Dokumenten aus verschiedenen Quellen sollten die Namen benutzerdefinierter Eigenschaften aus einem firmeneigenen Präfix, gefolgt von einem Doppelpunkt ':' und dem eigentlichen Eigenschaftsnamen bestehen. Als firmenspezifisches Präfix empfehlen wir den jeweiligen Internet-Domainnamen. Die Property-Namen des Unternehmens ACME sähen dann zum Beispiel wie folgt aus:

```
acme.com:digits
acme.com:refnumber
```

Da Standard- und benutzerdefinierte Eigenschaften im Block unterschiedlich gespeichert werden, können PPS-Standardnamen (wie in Abschnitt 13.7, »Blockeigenschaften«, Seite 428 definiert) nie mit benutzerdefinierten Namen in Konflikt geraten.

13.9 Blöcke per Programm erzeugen und importieren

13.9.1 Erzeugen von PDFlib-Blöcken mit POCA

PDFlib-Blöcke können per Programm mit der POCA-Schnittstelle erstellt werden, die in PPS enthalten ist. Mit POCA können die erforderlichen PDF-Datenstrukturen für Blöcke vorbereitet werden und dann an die Option *blocks* von *PDF_begin/end_page_ext()* übergeben werden. Bei der Erstellung der Blockdefinitionen müssen Sie bestimmte Bedingungen befolgen, siehe Abschnitt 13.10, »Spezifikation für PDFlib-Blöcke«, Seite 444. Die Blockeigenschaften müssen gemäß der aufgeführten Datentypen erstellt werden, siehe Abschnitt 13.7, »Blockeigenschaften«, Seite 428.

Cookbook *Ein vollständiges Codebeispiel zur Erstellung der PDFlib-Blöcke mit PPS finden Sie in der Cookbook-Kategorie* `blocks`.

In der PDFlib Block-Spezifikation sind die Blöcke leider doppelt enthalten: einmal im Haupt-Block-Dictionary einer Seite und erneut unter dem Eintrag *Name* innerhalb eines bestimmten Block-Dictionary. Diese beiden Namen müssen identisch sein, um Probleme beim Füllen des Blocks mit PPS oder beim Anzeigen der Block-Vorschau mit dem Block-Plugin zu vermeiden. *PDF_*Durch *begin/end_page_ext()* wird daher eine Exception ausgelöst, wenn das mit der Option *blocks* bereitgestellte Dictionary eine Blockdefinition enthält, die diese Regel verletzt. Im folgenden Codebeispiel sind die entsprechenden Paare in blau hervorgehoben.

Das folgende Codefragment zeigt die Verwendung der POCA-Funktionen zur Erzeugung der Blockdefinition gemäß »Schlüssel in Block-Dictionaries«, Seite 444:

```
/* Block-Dictionary erzeugen */
blockdict = p.poca_new("containertype=dict usage=blocks");

/* ------------------------------------------------------
 * Textblock erzeugen
 * ------------------------------------------------------
 */
textblock = p.poca_new("containertype=dict usage=blocks type=name key=Type value=Block");

container1 = p.poca_new("containertype=array usage=blocks " +
        "type=integer values={70 640 300 700}");

p.poca_insert(textblock, "type=array key=Rect value=" + container1);
p.poca_insert(textblock, "type=name  key=Name value=job_title");
p.poca_insert(textblock, "type=name  key=Subtype value=Text");
p.poca_insert(textblock, "type=name  key=fitmethod value=auto");
p.poca_insert(textblock, "type=string  key=fontname value=Helvetica");
p.poca_insert(textblock, "type=float key=fontsize value=12");

/* Block im Block-Dictionary der Seite aufnehmen */
p.poca_insert(blockdict, "type=dict key=job_title direct=false value=" + textblock);

/* ------------------------------------------------------
 * Block vom Typ Image erzeugen
 * ------------------------------------------------------
 */
imageblock = p.poca_new("containertype=dict usage=blocks " +
        "type=name key=Type value=Block");
```

```
container2 = p.poca_new("containertype=array usage=blocks " +
        "type=integer values={70 440 300 600}");

p.poca_insert(imageblock, "type=array key=Rect value=" + container2);
p.poca_insert(imageblock, "type=name  key=Name value=logo");
p.poca_insert(imageblock, "type=name  key=Subtype value=Image");
p.poca_insert(imageblock, "type=name  key=fitmethod value=auto");

/* Block im Block-Dictionary der Seite aufnehmen */
p.poca_insert(blockdict, "type=dict key=logo direct=false value=" + imageblock);

/* ---------------------------------------------------------
 * Block vom Typ PDF erzeugen
 * ---------------------------------------------------------
 */
pdfblock = p.poca_new("containertype=dict usage=blocks " +
        "type=name key=Type value=Block");

container3 = p.poca_new("containertype=array usage=blocks " +
        "type=integer values={70 240 300 400}");

p.poca_insert(pdfblock, "type=array key=Rect value=" + container3);
p.poca_insert(pdfblock, "type=name key=Name value=pdflogo");
p.poca_insert(pdfblock, "type=name key=Subtype value=PDF");
p.poca_insert(pdfblock, "type=name key=fitmethod value=meet");

/* Block im Block-Dictionary der Seite aufnehmen */
p.poca_insert(blockdict, "type=dict key=pdflogo direct=false " + "value=" + pdfblock);


/* ---------------------------------------------------------
 * Block-Dictionary auf der aktuellen Seite aufnehmen
 * ---------------------------------------------------------
 */
p.end_page_ext("blocks=" + blockdict);


/* Bereinigen */
p.poca_delete(blockdict, "recursive");
```

13.9.2 Importieren von PDFlib-Blöcken

Sie können PDFlib-Blöcke aus dem Input-Dokument auf die aktuelle Ausgabeseite kopieren. Verwenden Sie dazu *PDF_process_pdi( )* und *action=copyallblocks* oder *action=copyblock* auf die folgende Weise:

```
if (p.process_pdi(p, doc, 0, "action=copyallblocks block={pagenumber=1}") != 1)
{
        /* Fehler */
}
```

So können Sie mehrstufige Block-Füllvorgänge implementieren. Beachten Sie, dass Blocknamen auf jeder Seite eindeutig sein müssen, d.h. Sie können nicht mehrere Blöcke mit dem gleichen Namen auf derselben Seite importieren. Verwenden Sie die Unteroption *outputblockname*, um Blöcke beim Kopieren umzubenennen.

13.10 Spezifikation für PDFlib-Blöcke

Die Blocksyntax geht konform mit der PDF-Referenz, die einen Erweiterungsmechanismus definiert, mit dem eine Applikation private Daten als Anhang an die Datenstrukturen einer PDF-Seite speichern kann. In diesem Abschnitt finden Sie eine Beschreibung des PDFlib Block-Syntax. Benutzer, die Blöcke mit dem Block-Plugin oder mit PDFlib erstellen, benötigen diese Informationen nicht.

PDF-Objektstruktur für PDFlib-Blöcke. Das Page-Dictionary enthält einen Eintrag *PieceInfo*, der als Wert ein weiteres Dictionary enthält. Das Page-Dictionary sollte auch den Schlüssel *LastModified* mit einem Zeitstempel für die Erstellung oder letzte Änderung an den Blockstrukturen enthalten. Dieses Dictionary enthält den Schlüssel *PDFlib*, der als Wert wiederum ein Dictionary mit Applikationsdaten enthält. Das Applikationsdaten-Dictionary enthält die beiden in Tabelle 13.14 aufgeführten Standardschlüssel.

Tabelle 13.14 Einträge in einem PDFlib-Applikationsdaten-Dictionary

Schlüsselwort	Wert
LastModified	*(Datum; erforderlich) Datum und Zeit der Erstellung oder letzten Änderung der Blöcke auf der Seite. Dieser Eintrag wird bei der Erstellung von Blöcken mit der POCA-Schnittstelle von PDFlib erzeugt.*
Private	*(Dictionary; erforderlich) Blockliste gemäß Tabelle 13.15*

Eine Blockliste ist ein Dictionary mit allgemeinen Information zur Blockverarbeitung sowie einer Liste aller Blöcke auf der Seite. Tabelle 13.15 enthält alle Schlüssel in einem Blocklisten-Dictionary.

Tabelle 13.15 Einträge in einem Blocklisten-Dictionary

Schlüsselwort	Wert
Blocks	*(Dictionary; erforderlich) Jeder Schlüssel ist ein Namensobjekt mit dem Namen eines Blocks; der zugehörige Wert ist das Block-Dictionary des Blocks (siehe Tabelle 13.17). Der Wert von* `Name` *im Block-Dictionary muss identisch mit dem Namen des Blocks in diesem Dictionary sein.*
BlockProducer[1]	*(String) Name der Software, mit der die Blöcke programmiert wurden. Dieser Eintrag wird bei der Erstellung von Blöcken mit der POCA-Schnittstelle von PDFlib erzeugt.*
PluginVersion[1]	*(String) Versionsnummer des Block-Plugins, mit dem die Blöcke erstellt wurden.*
pdfmark[1]	*(Boolean) Muss den Wert* `true` *haben, falls die Blockliste mit Hilfe von pdfmarks erstellt wurde.*
Version	*(Zahl; erforderlich) Die Versionsnummer der Blockspezifikation, gemäß der die Datei erstellt wurde. Dieses Dokument beschreibt Version 10 der Blockspezifikation.*

1. Genau einer der Schlüssel `BlockProducer`, `PluginVersion` *und* `pdfmark` *muss vorhanden sein.*

Datentypen für Blockeigenschaften. Für Blockeigenschaften werden mit Ausnahme von Handles und speziellen Listen wie Aktionslisten dieselben Datentypen wie für Optionslisten unterstützt. Tabelle 13.16 zeigt, wie diese Typen auf PDF-Datentypen abgebildet werden.

Schlüssel in Block-Dictionaries. Block-Dictionaries können die Schlüssel in Tabelle 13.17 enthalten.

Tabelle 13.16 Datentypen für Blockeigenschaften

Datentyp	***PDF-Typ und Anmerkungen***
Boolean	*(Boolean)*
String	*(String)*
Schlüsselwort (Name)	*(Name) Es führt zu einem Fehler, wenn Schlüsselwörter übergeben werden, die nicht in der Liste der Schlüsselwörter enthalten sind, die von einer bestimmten Eigenschaft unterstützt werden.*
Float, Integer	*(Zahl) Optionslisten akzeptieren zwar Punkt und Komma als Dezimalzeichen, bei PDF-Zahlen ist aber nur ein Punkt erlaubt.*
Prozent	*(Array mit zwei Elementen) Das erste Array-Element enthält die Zahl, das zweite Element einen String mit einem Prozentzeichen.*
Liste	*(Array)*
Farbe	*(Array mit zwei oder drei Elementen) Das erste Array-Element legt einen Farbraum fest und das zweite einen Farbwert. Um die Abwesenheit von Farben festzulegen, muss die entsprechende Eigenschaft weggelassen werden. Für das erste Array-Element werden folgende Einträge unterstützt:* ***/DeviceGray*** *Das zweite Element ist ein einzelner Graustufenwert.* ***/DeviceRGB*** *Das zweite Element ist ein Array aus drei RGB-Werten.* ***/DeviceCMYK*** *Das zweite Element ist ein Array aus vier CMYK-Werten.* ***[/Separation/spotname]*** *Das erste Element ist ein Array mit dem Schlüsselwort* `/Separation` *und einem Schmuckfarbnamen. Das zweite Element ist ein Wert für den Farbauftrag.* *Das optionale dritte Element im Array legt eine Alternativfarbe für die Schmuckfarbe fest, die selbst ein Farbarray in einem der Farbräume* `DeviceGray`, `DeviceRGB`, `DeviceCMYK` *oder* `Lab` *ist. Fehlt die Alternativfarbe, muss der Schmuckfarbname eine Farbe sein, die PPS intern bekannt ist oder von der Anwendung zur Laufzeit definiert wurde.* ***[/Lab]*** *Das erste Element ist ein Array mit dem Schlüsselwort* `Lab`*. Das zweite Element ist ein Array aus drei Lab-Werten.*
unichar	*(Text-String) Unicode-Strings im Format* `utf16be`*, der mit BOM U+FEFF beginnen.*

Tabelle 13.17 Einträge in einem Block-Dictionary

Eigenschaftsgruppe	***Wert***
Administrative Eigenschaften	*(Manche Schlüssel sind erforderlich) Administrative Blockeigenschaften gemäß Tabelle 13.4.*
Rechtecke	*(Manche Schlüssel sind erforderlich) Blockeigenschaften für Rechtecke gemäß Tabelle 13.5.*
Darstellung	*(Manche Schlüssel sind erforderlich) Darstellungsspezifische Eigenschaften für alle Blöcke gemäß Tabelle 13.6 und für Textline- und Textflow-Blöcke gemäß Tabelle 13.7.*
Textvorbereitung	*(Optional) Textvorbereitende Eigenschaften für Textflow- und Textline-Blöcke gemäß Tabelle 13.8.*
Textformatierung	*(Optional) Textformat-spezifische Eigenschaften für Textflow- und Textline-Blöcke gemäß Tabelle 13.9.*
Objekteinpassung	*(Optional) Einpassungseigenschaften für Blöcke vom Typ Textline, Raster, PDF und Graphics gemäß Tabelle 13.10 und Einpassungseigenschaften für Textflow-Blöcke gemäß Tabelle 13.11.*

Tabelle 13.17 Einträge in einem Block-Dictionary

Eigenschaftsgruppe	***Wert***
Vorgabewerte	*(Optional) Vorgabewerte für Blockeigenschaften gemäß Tabelle 13.12.*
Benutzerdefiniert	*(Dictionary; optional) Dictionary mit Schlüssel/Wert-Paaren für benutzerdefinierte Eigenschaften gemäß Tabelle 13.13.*