EGIZ

Dokumentation

Dokumentation EGIZ-Dokumentation

Allgemeine PDF-AS Dokumentation ab Version 5.0

Version 1.3.0, 30.04.2026

Thomas Lenz – thomas.lenz@egiz.gv.at
Jakob Heher – jakob.heher@a-sit.at

Zusammenfassung: PDF-AS ist ein Java Framework zur Erstellung von PDF Signaturen nach dem PAdES Standard. Dieses Dokument gibt eine Einführung in PDF-AS und beschreibt die verschiedenen Komponenten des Frameworks. Diese Komponenten umfassen mehrere Java Bibliotheken, eine Kommandozeilen Anwendung und eine Webanwendung.

E-Government Innovationszentrum
Sandgasse 38a, A-8010 Graz
Tel. +43 316 873 5514
Fax. +43 316 873 5520
E-Mail post@egiz.gv.at
www.egiz.gv.at

Das E-Government Innovationszentrum ist eine gemeinsame Einrichtung des Bundeskanzleramtes und der TU Graz

BUNDESKANZLERAMT ÖSTERREICH

TU Graz
Graz University of Technology

Inhaltsverzeichnis

3

4

2.1 PDF-AS Java Bibliothek4

2.2 PDF-AS Kommandozeilen Anwendung5

2.3 PDF-AS Webanwendung7

8

3.1 Grundsätzliches8

3.2 Basiseinstellungen für die Anbindung an die Bürgerkartenumgebung (BKU)9

3.3 Basiseinstellungen für die Anbindung an MOA-SS und MOA SP9

3.4 Basiseinstellungen für die Anbindung von MOCCA Online10

3.5 Basiseinstellungen für die Anbindung der Handy Signatur10

3.6 Spezifikation der Signatur Blöcke10

3.6.1 Design des Signatur Blocks (Tabelle)11

3.6.2 Positionierung von Signaturblöcken14

3.6.3 Dynamische Werte in der Signaturblockdefinition18

3.7 Signatur-Platzhalter im Dokument18

3.7.1 Aktivierung18

3.7.2 QR-Code Generierung19

3.7.3 Properties19

3.7.4 Signatur-Platzhalter Auswahl20

Abbildungsverzeichnis

1 Kurzbeschreibung

PDF-AS ist ein Java Framework mit dem PDF Dokumente digital signiert und verifiziert werden können. Ab Version 4.0 unterstützt PDF-AS nur Signaturen nach dem PDF Advanced Electronic Signatures (PAdES) Standard. Als Signaturerstellungseinheit kann PDF-AS eine Bürgerkartenumgebung (BKU), eine MOA-SS Instanz, eine PKCS12 Datei oder eine JavaKeyStore Datei verwenden. PDF-AS kann eigenständig PAdES Signaturen verifizieren. Dabei wird von PDF-AS standardmäßig allerdings keine Zertifikatsprüfung durchgeführt. PDF-AS kann auch eine Verifikation mit Zertifikatsprüfung durchführen, dazu benötigt PDF-AS allerdings eine MOA-SP Instanz.

2 Anwendungsbeschreibung

PDF-AS besteht aus drei Teilen:

- Java Bibliothek: Implementiert die Grundfunktionalität von PDF-AS
- Kommandozeilen Anwendung: Bietet ein Kommandozeilen Interface für PDF-AS.
- Webanwendung: Bietet eine Webschnittstelle für PDF-AS an.

2.1 **PDF-AS Java Bibliothek**

Die PDF-AS Java Bibliothek ist das Herzstück von PDF-AS. Sie ermöglicht die Signatur von PDF Dokumenten. Zu dieser Bibliothek gehören zwei weitere Java Bibliotheken „sigs-pades“ und „sigs-pkcs7detached“. Diese beiden Bibliotheken ermöglichen es verschiedene Signaturstandards „PAdES“ und „PCKS7 detached“ zur Signaturerstellung zu verwenden. Diese beiden Bibliotheken müssen zur Laufzeit von PDF-AS im Klassenpfad vorhanden sein.

PDF-AS kann verschiedene PDF Bibliotheken verwenden. Die Standard-Implementierung verwendet Apache PdfBox 3. Um diese zu verwenden, muss die Bibliothek pdf-as-pdfbox-3 im Klassenpfad eingebunden sein.

PDF-AS implementiert ein Pluginsystem um Vorverarbeitungsschritte von PDF Dokumenten durchzuführen. Um ein solche Plugin zu erzeugen, muss die Schnittstelle at.gv.egiz.pdfas.lib.api.preprocessor.PreProcessor implementiert werden. Die Plugins werden mittels Service Provider Interface (SPI) von PDF-AS geladen.

Ein Code Beispiel zur Signatur eines PDF Dokuments mittels der Java Bibliothek (Vorsicht: Es wird eine Dependency zum Modul *<groupId>at.gv.egiz.pdfas</groupId> <artifactId>sigs-pades</artifactId>* vorausgesetzt)

```
byte[] pdfDokument = …
String outputFile = …
PdfAs pdfas = PdfAsFactory.createPdfAs(new File(pdfas_dir));
Configuration config = pdfas.getConfiguration();
File outputPdfFile = new File(outputFile);
FileOutputStream fos = new FileOutputStream(outputPdfFile, false);
SignParameter para = PdfAsFactory.createSignParameter(config, new
ByteArrayDataSource(pdfDokument), fos);
para.setSignatureProfileId("SIGNATURBLOCK_DE");
para.setPlainSigner(new PAdESSigner(new BKUSLConnector(config)));
pdfas.sign(para);
System.out.println("Signed document " + outputFile);
```

Ein Beispiel Code zur Verifikation eines PDF Dokuments mittels der Java Bibliothek:

```
PdfAs pdfas = PdfAsFactory.createPdfAs(new File(pdfas_dir));
Configuration config = pdfas.getConfiguration();
VerifyParameter param = PdfAsFactory.createVerifyParameter(config, new
ByteArrayDataSource(pdf));
param.setSignatureVerificationLevel(SignatureVerificationLevel.INTEGRITY_ONLY
_VERIFICATION);
List<VerifyResult> vrs = pdfAs.verify(param);
Iterator<VerifyResult> resultIterator = vrs.iterator();
```

Es wurde eine Wrapper Bibliothek entwickelt, welche die API der PDF-AS 3 Bibliothek auf die neue API abbildet. Diese Bibliothek kann verwendet werden um den Umstieg auf PDF-AS in Version 4 zu erleichtern. Da sich die Funktionalität von Version 3 auf Version 4 stark verändert hat, werden allerdings einige alte API Aufrufe nicht mehr unterstützt. Es wird empfohlen Anwendungen die PDF-AS in Version 3 verwenden aktiv auf die API der Version 4 umzubauen. Eine detaillierte Dokumentation zu PDF-AS in Version 4 ist in [PDF-AS-API] verfügbar.

Wenn PDF-AS ein Dokument signiert prüft es automatisch die Signatur die aufgebracht werden soll. Dies ist erforderlich, da die eigentliche Signatur von externen Anwendungen, wie eine Bürgerkartenumgebung, MOA-SPSS oder der Handy-Signatur. Bei dieser Prüfung wird allerdings lediglich die Integrität der Signatur geprüft aber keine Zertifikatsprüfung durchgeführt.

2.2 PDF-AS Kommandozeilen Anwendung

Das Kommandozeilen Interface zu PDF-AS ermöglicht die Signatur sowie die Verifikation von PDF Dokumenten.

Zur Signatur stehen folgenden Optionen zur Verfügung:

Name	Wertebereich	Beschreibung
`-c, --connector <arg>`	bku, ks, moa	Mit dem connector Parameter wird festgelegt, welche Signaturerstellungseinheit verwendet werden soll. bku: verwendet eine lokale Bürgerkartenumgebung ks: verwendet eine PKCS12 Datei oder eine JavakeyStore moa: verwendet MOA-SS
`-conf, --configuration <arg>`	*PDF-AS Verzeichniss*	Pfad zum Ordner in dem die PDF-AS Konfiguration liegt.

Name	Wertebereich	Beschreibung
`-d, --deploy`		Spielt die Standard Konfiguration von PDF-AS ein.
`-m, --mode <arg>`	sign, verify	Der PDF-AS Modus bestimmt ob PDF-AS signieren oder verifizieren soll
`-o, --output <arg>`	*Datei*	Die Ausgabedatei von PDF-AS
`-p, --profile <arg>`	*Signaturprofil*	Das zu verwende Signaturprofil
`-pos, --position <arg>`	x:x_algo;y:y_algo;w:w_algo;p:p_algo;f:f_algo	Die Position der Signatur. Der x und y Wert beschreibt die Position auf der Seite. Der w Wert beschreibt die Breite des Signaturblocks. Der p Wert beschreibt die Seite des PDF Dokuments und der f Wert beschreibt bei automatischer Positionierung die zu beachtende Footerhöhe. Der r Wert gibt die Rotation des Signaturblocks in Grad (Vielfache von 90 Grad) an. Mögliche Werte sind: x_algo: „auto „, *nummerischerWert* y_algo: „auto „, *nummerischerWert* *w_algo: „auto“, nummerischerWert* p_algo: „auto“, „new“, „last“, *nummerischerWert* f_algo: *nummerischerWert*
`-vw, --verify_which <arg>`	*numerischerWert*	Legt die Signatur fest die geprüft werden soll. Die Signaturen im PDF Dokument sind 0 basiert nummeriert.
`-vl, --verify_level <arg>`	full, intOnly	Welche Form der Signaturprüfung durchgeführt werden soll. „full“ benötigt eine konfigurierte MOA-SP-Instanz.
`-ksa, --ks_alias <arg>`		Der Alias des Zertifikats in der PKCS12 Datei oder im JavaKeyStore, das zur Signatur verwendet werden soll.
`-ksf, --ks_file <arg>`	*Datei*	Die PKCS12 oder die JavaKeyStore Datei
`-kskp, --ks_keypass <arg>`		Das Passwort für den privaten Schlüssel in der PKCS12 oder JavaKeyStore Datei
`-kssp, --ks_storepass <arg>`		Das Passwort für die PKCS12 oder JavaKeyStore Datei
`-kst, --ks_type`	PKCS12, JKS	Der Typ des Keystores.

Name	Wertebereich	Beschreibung
`-sbp, --signature_block_parameter`	Schlüssel-Wert Paar	Definiert Schlüssel-Wert-Paare, auf die mittels ${sbp.key} im Signaturblock zugegriffen werden kann. Dieser Parameter kann mehrmals vorkommen. Bsp: -sbp subject=Test

Beispiel für Aufrufe um eine Signatur zu erstellen:

pdf-as –m sign –c bku –o test_dokument_signiert.pdf test_dokument.pdf

2.3 PDF-AS Webanwendung

PDF-AS Web ist eine Webanwendung, welche es ermöglicht PDF-Dokumente mit PDF-AS zu signieren. Eine detaillierte Dokumentation der Webanwendung finden Sie im Dokument [PDF-AS-WEB]

3 Konfiguration

Die Konfiguration der Version 5 ist rückwärtskompatibel mit der Konfiguration aus Version 4.

3.1 **Grundsätzliches**

Die Konfigurationsdateien befinden sich im Verzeichnis:

.pdfas/cfg

Die Konfigurationsdatei ist eine simple Java-property Datei. Neben der Standardkonfigurationsdatei (config.properties) befindet sich im oben genannten Verzeichnis auch noch die erweiterte Konfigurationsdatei (advancedconfig.properties). Diese Datei enthält fortgeschrittene Parameter, die bei Bedarf aktiviert werden können und wird von der Standardkonfigurationsdatei eingebunden.

Hinweis: Eine Java-property Datei muss im ISO-8859-1 (auch bekannt als ISO-Latin) Character encoding abgelegt sein. Dies Betrifft vor allem vorkommende Umlaute etc. Stellen Sie sicher, dass der Verwendete Text Editor beim Editieren der Konfigurationsdatei dieses Encoding verwendet.

Die Konfigurationsdatei wurde grundsätzlich hierarchisch aufgebaut, um ein einfache Gruppierung verschiedener Bereiche bewerkstelligen zu können.

Die hierarchischen Ebenen werden durch „.“ voneinander getrennt. So ergibt sich eine Baumartige Struktur von konfigurierbaren Werten.

Kommentare können zeilenweise eingefügt werden. Kommentarzeilen beginnen mit dem Raute Zeichen „#“. Kommentarzeilen werden von der Applikation nicht berücksichtigt und können verwendet werden um die Konfigurationsdatei besser lesbar zu machen oder Anmerkungen anzubringen. Beispiel:

 # Das ist ein Kommentar

Es wird empfohlen jede Konfigurationsdatei einleitend mit einem Kommentar zu versehen, welches Auskunft über die Herkunft der Konfiguration, den Verwendungszweck und die darin definierten Profile gibt. Zum Beispiel:

 # PDF-AS Konfigurationsdatei für den Gebrauch im Amt XYZ

3.2 Basiseinstellungen für die Anbindung an die Bürgerkartenumgebung (BKU)

Signierprozess: Request-Url

bku.sign.url=http://127.0.0.1:3495/http-security-layer-request

Prüf-Prozess: Request-URL

bku.verify.url=http://127.0.0.1:3495/http-security-layer-request

3.3 Basiseinstellungen für die Anbindung an MOA-SS und MOA SP

Um MOA-SS und MOA-SP zu verwenden, muss die Bibliothek pdf-as-moa in den Klassenpfad aufgenommen werden.

Signierprozess: Request-Url, Pfad des Signier Request-Template

moa.sign.url=http://129.27.153.100:18080/moa-spss/services/SignatureCreation

Der zu verwendende Sign Key und das Signaturzertifikat von MOA müssen folgendermaßen angegeben werden:

moa.sign.KeyIdentifier=TestKey2

moa.sign.Certificate=./moa_sign_certificate.crt

Prüf-Prozess: Request-URL

moa.verify.url=http://129.27.153.100:18080/moa-spss/services/SignatureVerification

Die von MOA zu verwendende TrustProfileID muss folgendermaßen angegeben werden:

moa.verify.TrustProfileID=Test-Signaturdienste

Falls gewünscht, kann ein Timeout für die Operationen angegeben werden:

moa.sign.timeout=5000 # (in Millisekunden)
moa.verify.timeout=5000

MOA-SS kann erst ab Version 2 zur Signaturpruefung verwendet werden.

3.4 Basiseinstellungen für die Anbindung von MOCCA Online

Signierprozess: Request-Url

moc.sign.url= http://129.27.153.100:18080/bkuonline/http-security-layer-request

3.5 Basiseinstellungen für die Anbindung der Handy Signatur

Signierprozess: Request-Url, Pfad des Signier Request-Template und der KeyboxIdentifier

mobile.sign.url=https://service.a-trust.at/mobile/https-security-layer-request/default.aspx

3.6 Spezifikation der Signatur Blöcke

Die Spezifikation legt fest, welche Bezeichner (key) im Signatur Block vorhanden sind und welche Werte (value) vordefiniert werden können. Bei der Erzeugung eines Signatur Blocks werden die notwendigen Daten (values) aus der Signaturanfrage Antwort von BKU oder MOA extrahiert und entsprechend zugeordnet. Die Spezifikation sagt jedoch nicht aus, in welcher Reihenfolge oder in welcher Anordnung die Bezeichner und Werte im Signatur Block eingetragen werden. Dafür gibt es eine separate Definition.

Der Wert der Description gibt dem Signaturprofil seinen Namen.

sig_obj.egov_graz_gv_at.description=EGOV Graz.gv.at

Definition der Bezeichner im Einzelnen:

Name des Zertifikat Inhabers:

sig_obj.egov_graz_gv_at.key.SIG_NAME=Inhaber

Signaturdatum:

sig_obj.egov_graz_gv_at.key.SIG_DATE=Datum

Name des Ausstellers:

sig_obj.egov_graz_gv_at.key.SIG_ISSUER=Aussteller

Seriennummer des Zertifikates:

sig_obj.egov_graz_gv_at.key.SIG_NUMBER=Seriennummer

Metadaten zum Zertifikat:

sig_obj.egov_graz_gv_at.key.SIG_META=Hinweis:

Der Bezeichner SIG_NAME kann verwendet werden um den Namen des Signators aus dem Signaturzertifikat in den Signaturblock einzufügen.

3.6.1 *Design des Signatur Blocks (Tabelle)*

Eine Signatur Block Tabelle besteht aus mindestens einer `main` Tabelle. Die Tabellen Reihen werden steigend durchnummeriert. Der Wert einer Zeile gibt an, was in dieser Zeile dargestellt werden soll.

D.h. Es werden die Synonyme der Bezeichner (laut Signatur Typen Spezifikation) eingetragen. Eine Ausnahme bildet das Synonym `TABLE`. Dieses verweist auf eine eingebettete Tabellendefinition (z.B. info).

Mit Hilfe der Felder –i(Image), -c(Caption=key), -v(Value=value) werden die jeweiligen Werte in die Tabellenzelle eingefügt. Die Trennung von Tabellenzellen erfolgt mit dem Zeichen „|“.

 # Signatur Tabellen Spezifikation

Z.B. Definition einer zweispaltigen Tabelle: links das Bild, rechts die Subtabelle info:

sig_obj.egov_graz_gv_at.table.main.1=SIG_LABEL-i|TABLE-info

sig_obj.egov_graz_gv_at.table.main.2=SIG_META-v

sig_obj.egov_graz_gv_at.table.main.3=SIG_ID-cv

Verhältnis der Aufteilung der Tabellen-Spalten:

sig_obj.egov_graz_gv_at.table.main.ColsWidth=1 4

Styledefinitionen→diese Vererben sich auch auf die Zellen:

Hintergrundfarbe:

sig_obj.egov_graz_gv_at.table.main.Style.bgcolor=222 222 200

Hinweis: Wenn ein Bild nicht transparent ist, so sollte die Hintergrundfarbe gleich der Bildhintergrundfarbe sein, um unschöne Farbsprünge zu vermeiden.

Innenabstand:

sig_obj.egov_graz_gv_at.table.main.Style.padding=3

Horizontalausrichtung:

sig_obj.egov_graz_gv_at.table.main.Style.halign=[left|center|right]

Vertikalausrichtung:

sig_obj.egov_graz_gv_at.table.main.Style.valign=[top|middle|bottom]

Ausrichtung ausschließlich für Werte-Zellen (Zellen, in denen signaturspezifische Daten wie z.B. der Name des Unterzeichners oder der Signaturzeitpunkt enthalten sind). Sind diese Werte nicht gesetzt, werden die entsprechenden Werte von `valign` bzw. `halign` übernommen:

Horizontalausrichtung für Werte-Zellen:

sig_obj.egov_graz_gv_at.table.main.Style.valuehalign=[left|center|lineCenter|right]

Vertikalausrichtung für Werte-Zellen:

sig_obj.egov_graz_gv_at.table.main.Style.valuevalign=[top|middle|bottom]

Ausrichtung ausschließlich für Bilder-Zellen. Sind diese Werte nicht gesetzt, werden die entsprechenden Werte von `valign` bzw. `halign` übernommen

Horizontalausrichtung für Bilder-Zellen:

sig_obj.egov_graz_gv_at.table.main.Style.imagehalign=[left|center|right]

Vertikalausrichtung für Bilder-Zellen:

sig_obj.egov_graz_gv_at.table.main.Style.imagevalign=[top|middle|bottom]

Rahmenstärke:

sig_obj.egov_graz_gv_at.table.main.Style.border=0.1

Schriftart: *face, height, weight*

default_font: HELVETICA,8,NORMAL

font_face: HELVETICA | TIMES_ROMAN | COURIER

font_height: float value

font_weight: NORMAL | BOLD | ITALIC | BOLDITALIC

sig_obj.egov_graz_gv_at.table.main.Style.font=HELVETICA,12,NORMAL

Definition einer Subtabelle – zum Beispiel:

sig_obj.egov_graz_gv_at.table.info.1=SIG_DATE-cv

sig_obj.egov_graz_gv_at.table.info.2=SIG_NAME-cv

sig_obj.egov_graz_gv_at.table.info.3=SIG_ISSUER-cv

sig_obj.egov_graz_gv_at.table.info.4=SIG_NUMBER-cv

sig_obj.egov_graz_gv_at.table.info.ColsWidth=1.5 4

3.6.2 *Positionierung von Signaturblöcken*

Ein Signaturblock kann entweder automatisch (Standardeinstellung) oder manuell positioniert werden.

Bei der automatischen Positionierung wird der Signaturblock auf die erste freie Stelle nach dem gesamten Dokumenttext einschließlich der Fußzeile platziert. Sollte auf der letzten Seite nicht mehr genügend Platz dafür sein, so wird eine neue Seite angelegt und der Signaturblock auf dieser platziert.

Hinweis: Mit Einführung der PDF Spezifikation ISO 32000-2 (2018) wurde die Manipulation an bereits signierten Dokumenten sehr eingeschränkt, wovon unter anderem auch das Hinzufügen neuer Seiten bei bereits signierten Dokumenten betroffen ist. In diesem Fall wird der Signaturblock bei automatischen Positionierung immer auf der letzten Seite des Dokuments platziert wobei, sofern möglich, der noch freie Platz verwendet wird. Anderenfalls wird der Signaturblock über einen bestehenden Inhalt platziert. Ab der Version 4.3.0 kann PDF-AS mittels des Konfigurationsparameters: `sigblock.placement.less.space.failing=true/false (Default: false)` so konfiguriert werden, dass der Signaturvorgang abgebrochen wird falls bei bereits signierten Dokumenten die freie Fläche zur Positionierung nicht mehr ausreichend ist.

Mittels manueller Positionierung kann in die Positionierung des Signaturblocks eingegriffen werden. Ein Signaturblock kann auf mehrere Arten manuell positioniert werden:

-pos x:x_algo;y:y_algo;w:w_algo;p:p_algo;f:f_algo;r:r_algo

x_algo := 'auto' ... automatische Positionierung

:= ... Absolutwert für x-Position

Default bei Fehlen des Parameters: 'auto'

y_algo := 'auto' ... automatische Positionierung

:= ... Absolutwert für y-Position

Default bei Fehlen des Parameters: 'auto'

w_algo := 'auto' ... automatische Breite

:= +[0..9] ... Absolutwert für Breite

Default bei Fehlen des Paramters: 'auto'

p_algo := 'auto' ... Automatisch am Ende des Dokuments (letzte Seite oder neue Seite falls die letzte Seite zu wenig freie Fläche aufweist)

:= 'new' ... Neue Seite am Ende des Dokuments

:= 'last' ... Letzte Seite des Dokuments

:= +[0..9] ... Seitennummer

Default bei Fehlen des Parameters: 'auto'

f_algo := +[0..9] ... y-Offset für Footer

Default bei Fehlen des Parameters: '0'

Wird nur bei y:auto berücksichtigt, andernfalls ignoriert!

r_algo := +[0..9] ... Drehung des Signaturblocks

Default bei Fehlen des Parameters: '0'

Ist nur bei manueller Positionierung sinnvoll

Default ist somit: *-pos x:auto;y:auto;w:auto;p:auto;f:0*

Variationen zum Beispiel:

-pos "x:auto;y:auto;w:auto"

-pos "x:10.0;y:10.0;w:100.0;r:270"

-pos "x:10.0;y:10.0;w:100.0;p:new;f:10"

-pos "x:auto;y:10.0;w:auto;p:last"

-pos "x:auto;y:auto;p:last"

-pos "x:22.0;y:auto;w:450.0;p:2;f:25"

-pos "x:auto;y:auto;w:auto;p:auto;f:25.0"

-pos "f:25.0"

-pos "x:150;y:22;w:400"

-pos „x:10.0;w:155.0"

Parameter *p* gibt dabei die Seite an, auf welcher der Signaturblock angebracht werden soll. Eine gültige Seitenzahl als Parameter bedeutet, dass der Signaturblock auf der angegebenen Seite absolut positioniert wird. *p=last* bedeutet, dass auf der letzten Seite des Dokuments eine absolute Positionierung vorgenommen wird. *p=new* bedeutet, dass eine neue, leere Seite an das Dokument angefügt und auf dieser dann eine absolute Positionierung vorgenommen wird. *p=auto* bedeutet, dass die Signatur eigentlich automatisch positioniert werden soll, bei Berechnung des Endes des Textes allerdings die Fußzeile ggf. ausgenommen wird. Mit diesem Mechanismus ist es möglich einen Signaturblock automatisch zwischen Text und Fußzeile zu platzieren, sofern dort genügend Platz vorhanden ist.

Bei absoluter Positionierung geben die Parameter *x* und *y* die Koordinaten der linken oberen Ecke des Signaturblocks auf der gewählten Seite an. *x* wird von links nach rechts gemessen. *y* wird von unten nach oben gemessen. Der Koordinaten Ursprung liegt in der linken unteren Ecke der Seite. Der Parameter *w* gibt zudem die Breite des Signaturblocks an. Diese wird von links nach rechts gemessen und muss eine Zahl größer als 0 sein.

Bei automatischer Positionierung unter Berücksichtigung der Fußzeile ist der Parameter *y:auto* zusammen mit dem Parameter *f* zu setzen. Ist zwischen dem Ende des Fließtextes und der Oberkante der Fußzeile genügend Platz für den Signaturblock, so wird dieser dort platziert. Ansonsten wird der Signaturblock auf eine neue Seite gesetzt.

Alle Koordinaten werden in PDF User Space Einheiten gemessen. Eine A4 Seite im Hochformat ist demnach 595 Einheiten breit und 842 Einheiten hoch.

Für weitere Informationen siehe auch den *–pos* Parameter des Kommandozeilen Tools.

Ein Signaturblock kann alternativ auch mit einem Platzhalter-Bild in einem Dokument positioniert werden. Siehe dazu: Signatur-Platzhalter im Dokument

Hinweise:

- Es ist durchaus möglich den Signaturblock so zu positionieren, dass er nicht sichtbar ist. Weiters kann er durch die Wahl einer sehr kleinen Breite unschön entstellt werden. Es liegt in der Verantwortung der User, eine ansprechende Darstellung und vernünftige Werte für die absolute Positionierung zu wählen.
- Beachten Sie bitte, dass die Angabe der Positionsparameter abhängig vom zugrundeliegenden Betriebssystem beim Aufruf aus der Commandline/Shell mit

Hochkomma versehen werden muss. So wird z.B. das Semikolon (";") unter Linux/Unix/MacOS als Trennzeichen zwischen zwei Kommandos betrachtet. Deshalb muss bei diesen Betriebssystemen der Parameter *-pos* zusammen mit Hochkommas verwendet werden:

... -pos "x:10.0;y:10.0;w:100.0;p:new;f:10"

3.6.3 *Dynamische Werte in der Signaturblockdefinition*

Es ist möglich in den Werte (value) von eigens definierten Tabellenspalten dynamisch auf Teile des verwendeten Zertifikats zuzugreifen. Konkret kann auf die einzelnen RDNs Teile des Issuer DN und des Subject DN des Signaturzertifikats wie im folgenden Beispiel illustriert zugegriffen werden:

```
sig_obj.BAIK_URKUNDE_SIGNATUR.value.SIG_SIG_LABEL=
${subject.CN}${subject.O != null ? ("\n" + subject.O) : ""}${subject.L != null ?
("\nKanzleisitz: " + subject.L) : ""}
```

Die Notation `${..}` ermöglicht die dynamische Auswertung eines Ausdrucks. Verfügbar sind `subject` und `issuer` und die im Zertifikat DN vorhandenen RDNs. Wie im Beispiel illustriert sind einfache String Operationen und Bedingungsauswertungen ebenfalls verfügbar.

Es können auch von außen Parametern als Schlüssel-Wert-Paare übergeben und im Signaturblock referenziert werden. Über die Kommandozeile beispielsweise über den Prefix „-sbp“. Die Schlüssel und Werte werden gegen einen regulären Ausdruck geprüft, um ungültige Zeichen zu verhindern. Standardmäßig verwendet PDF-AS die Regex "^([A-Za-z]){1,20}$" für den Schlüssel und "^([\p{Print}ß§€äöüÄÖÜ]){1,100}$" für den Wert. Die beiden regulären Ausdrücke können bei Bedarf in der Konfiguration mittels *sigblockparameter.key.regex* bzw. *sigblockparameter.value.regex* angepasst werden.

3.7 Signatur-Platzhalter im Dokument

In zu signierenden PDF Dokumenten können spezielle Bilder als Platzhalter positioniert werden. Ein solcher Platzhalter muss einen speziellen QR-Code (ein 2D Barcode) enthalten, damit er erkannt werden kann. Von diesem Platzhalter wird die Positionierung (linke obere Ecke) sowie die Breite für den Signaturblock übernommen. Zusätzlich kann über eine Property im QR-Code das Signaturprofil gewählt werden.

3.7.1 *Aktivierung*

Da das Scannen nach Platzhaltern in großen Dokumenten sehr zeitaufwändig sein kann, muss dieses Feature in der Konfiguration explizit aktiviert werden:

enable_placeholder_search=true aktiviert die Suche nach Platzhaltern für alle Profile. Der Standardwert für diesen Parameter ist *false*.

sig_obj.[PROFILNAME].enable_placeholder_search=[true|false] aktiviert, beziehungsweise deaktiviert die Suche nach Platzhaltern für ein bestimmtes Profil. Berücksichtigt wird hier das Profil, welches als SignParameter übergeben wurde, beziehungsweise - falls keines übergeben wurde - das default-Profil.

3.7.2 *QR-Code Generierung*

QR Codes können auf diversen Seiten kostenlos online generiert werden.

Beispiele:

https://pdf.egiz.gv.at/pdf-as-wai/qrCodeGeneration

Um eine Verwechslung mit eventuell bereits in einem Dokument vorhandenen anderen QR-Codes zu vermeiden, muss der im QR-Code eingebettete Text einem speziellen Format folgen:

*PDF-AS-POS[;property=value]**

Im einfachsten Fall wäre das also der String: *PDF-AS-POS*

3.7.3 *Properties*

Folgende Properties können wie oben beschrieben im QR-Code übergeben werden:

profile ein in der Konfiguration existierendes Signaturprofil

Beispiele:

PDF-AS-POS;profile=SIGNATURBLOCK_DE

PDF-AS-POS;profile=SIGNATURBLOCK_EN

id ein Identifikator für diesen Platzhalter

Beispiele:

PDF-AS-POS;id=0001

PDF-AS-POS;id=signature01

Die Properties aus dem QR-Code überschreiben auf jeden Fall die eventuell bei der Signatur mit übergebenen entsprechenden Signatur Parameter.

placeholder_profile_overwrite=false deaktiviert serverseitig die Übername von Signaturprofilen (z.B. *profile=SIGNATURBLOCK_DE*) aus Platzhalterinformationen. Der Standardwert für diesen Parameter ist *true*.

3.7.4 ***Signatur-Platzhalter Auswahl***

Befinden sich mehrer Platzhalter in einem Dokument kann PDF-AS in verschiedenen Modi die Auswahl treffen, welcher Platzhalter für den aktuellen Signaturvorgang zu verwenden ist. Welcher Modi verwendet wird lässt sich in der Datei „advancedconfig.properties" unter „placeholder_mode" konfigurieren. Folgende Modi sind verfügbar:

„0" in diesem Modus schlägt der Signaturvorgang fehl, wenn kein Platzhalter mit einem Identifikator gefunden wird der zu dem Identifikator „placeholder_id" passt. Dieser Modus wird "strict" genannt.

„1" in diesem Modus wird der Platzhalter mit dem Identifikator welcher zu „placeholder_id" passt verwendet. Wird dieser nicht gefunden, wird ein Platzhalter ohne Identifikator verwendet. Wird auch kein solcher Platzhalter gefunden, verläuft der Signaturvorgang ohne Platzhalter. Dieser Modus wird "moderate" genannt. Dies ist der Standardmodus.

„2" dieser Modus funktioniert gleich wie Modus „1". Allerdings werden, wenn nur Platzhalter mit Identifikatoren gefunden werden, der erste dieser verwendet. Dieser Modus wird "lenient" genannt.

„3" in diesem Modus wird der Platzhalter mit dem Identifikator welcher zu „placeholder_id" passt verwendet. Wird dieser nicht gefunden, wird der erste Platzhalter mit Identifikator in aufsteigender Reihenfolge verwendet. Numerische Platzhalteridentifikatoren werden numerisch verglichen, ansonsten textuell ohne Groß-/Kleinschreibung. Wurde kein Platzhalter gefunden, wird ein Platzhalter ohne Identifikator verwendet. Wird auch kein solcher Platzhalter gefunden, verläuft der Signaturvorgang ohne Platzhalter. Dieser Modus wird "sorted" genannt.

Dokumentenhistorie

Version	Datum	Autor(en)	Anmerkung
0.1	06.02.2014	Andreas Fitzek	Initiale Version
0.2	19.02.2014	Christian Maierhofer	Review
0.3	18.08.2014	Andreas Fitzek	Automatische Signaturprüfung
0.4	10.10.2014	Andreas Fitzek	pdf-as-moa Bibliothek
0.6	26.03.2015	Andreas Fitzek	Review update
0.7	12.02.2016	Andreas Fitzek	Platzhalter Modi
0.8	10.08.2017	Bianca Schnalzer	Parameter-Update
0.9	03.06.2019	Emina Ahmetovic	Code-Update
1.0	14.04.2021	Alexander Marsalek	sbp Parameter ergänzt
1.1	03.03.2023	Thomas Lenz	Auto-Positionierung update
1.2	20.12.2023	Thomas Lenz	Platzhalter updates
1.2.1	22.03.2024	Thomas Lenz	Page-Positionierung ‚last' hinzugefügt
1.3.0	30.04.2026	Jakob Heher	PDF-AS 5.0

Referenzen

[PDF-AS-WEB] Anbindung einer externen Webanwendung an PDF-AS-WEB 5.0 AnbindungExterneWebanwendung.pdf

[PDF-AS-API] PDF-AS API Dokumentation (https://joinup.ec.europa.eu/site/pdf-as/releases/4.0.7/docs/api)