EGIZ E-Government Innovationszentrum

www.egiz.gv.at
E-Mail: post@egiz.gv.at
Telefon: ++43 (316) 873 5514
Fax: ++43 (316) 873 5520
Inffeldgasse 16a / 8010 Graz / Austria

Dokumentation PDF-AS 4.0

PDF-AS 4.0 Dokumentation

Version 0.1, 05. Februar 2014

Andreas Fitzek – andreas.fitzek@egiz.gv.at

Zusammenfassung:

PDF-AS ist ein Java Framework zur Erstellung von PDF Signaturen nach de 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.

Inhaltsverzeichnis:

Abbildungsverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**Revision History**

Version	Datum	Autor(en)	
0.1	04.02.2014	Andreas Fitzek	Initiale Version

1 Kurzbeschreibung

PDF-AS ist eine 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. Zur Verifikation von PDF Dokumenten benötigt PDF-AS 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 Signatur zu verwenden.

Im Anhang findet sich ein Java Beispiel zur Signatur eines PDF Dokuments mittel BKU.

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 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 der PDF-AS in Version 4 ist in [PDF-AS-API] verfügbar.

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 lokal Bürgerkartenumgebung ks: verwendet eine PKCS12 Datei oder eine JavakeyStore
`-conf, --configuration <arg>`	*PDF-AS Verzeichniss*	Die PDF-AS Konfiguration die verwendet werden soll.
`-d, --deploy`		Spielt die Standard Konfiguration von PDF-AS ein.

Name	Wertebereich	Beschreibung
`-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 Wert beschreibt bei automatischer Positionierung die zu beachtende Footerhöhe. Mögliche Werte sind: x_alog: „auto „, *nummerischerWert* y_alog: „auto „, *nummerischerWert* *w_algo: „auto“, nummerischerWert* p_algo: „auto“, „new“, *nummerischerWert* f_algo: *nummerischerWert*
`-vw, --verify_wich <arg>`	*nummerischerWert*	Legt die Signatur fest die geprüft werden soll. Die Signaturen im PDF Dokument sind 0 basiert nummeriert.
`-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 Type des Keystores.

Beispiele 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 4 ist rückwärtskompatibel mit der Konfiguration aus Version 3. In Version 4 wurden die Standard Signaturprofile angepasst.

3.1 Grundsätzliches

Die Konfigurationsdateien befinden sich im Verzeichnis:

.pdfas/cfg

Die Konfigurationsdatei ist eine simple Java-property Datei.

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.

Diese 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

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

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://www.handy-signatur.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:

Das 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 – ab Version 3.1.1:

Horizontalausrichtung für Werte-Zellen:

sig_obj.egov_graz_gv_at.table.main.Style.valuehalign=[left|center|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 – ab Version 3.1.1:

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 | UNDERLINE | STRIKETHRU

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.

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

x_algo := 'auto' ... automatische Positionierung

:= ... Absolutwert für x-Position

Default bei Fehlen des Paramters: 'auto'

y_algo := 'auto' ... automatische Positionierung

:= ... Absolutwert für y-Position

Default bei Fehlen des Paramters: 'auto'

w_algo := 'auto' ... automatische Breite
:= +[0..9] ... Absolutwert für Breite

Default bei Fehlen des Paramters: 'auto'

p_algo := 'auto' ... Automatisch letzte Seite
:= 'new' ... Neue Seite am Ende des Dokumentes
:= +[0..9] ... Seitennummer

Default bei Fehlen des Paramters: 'auto'

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

Default bei Fehlen des Paramters: '0'
Wird nur bei y:auto berücksichtigt, andernfalls ignoriert!

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
-pos x:10.0;y:10.0;w:100.0;p:new;f:10
-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=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 die *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 hochformatige A4 Seite ist demnach 595 Einheiten breit und 842 Einheiten hoch.

Für weitere Informationen siehe auch den *–pos* Parameter des command line 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 des Users 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 Commandlin/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

Seit PDF-AS Version 3.2 ist es möglich in die 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.

3.7 Signatur-Platzhalter im Dokument

Seit Version 3.2 können in zu signierenden PDF Dokumenten 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 können im QR-Code einige Properties übergeben werden, über die Signaturprofil, Signaturtyp und Signaturkey gewählt werden können.

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 deaktiviert 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:

http://zxing.appspot.com/generator/

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

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

Referenzen

[PDF-AS-WEB]	Anbindung einer externen Webanwendung an PDF-AS-WEB 4.0Ö AnbindungExterneWebanwendung.pdf
[PDF-AS-API]	PDF-AS API Dokumentation

Anhang

Beispiel Aufruf der PDF-AS Bibliothek zur Signatur eine PDF Dokuments:

```
byte[] pdfDokument = …
PdfAs pdfas = PdfAsFactory.createPdfAs(new File(pdfas_dir));
Configuration config = pdfas.getConfiguration();
SignParameter para = PdfAsFactory.createSignParameter(config, new ByteArrayDataSource(pdfDokument));
ByteArrayDataSink bads = new ByteArrayDataSink();
para.setSignatureProfileId("SIGNATURBLOCK_DE");
para.setOutput(bads);
para.setPlainSigner(new PAdESSigner(new BKUSLConnector(config)));
pdfas.sign(para);
byte[] signedPDF = bads.getData();
```