

# Anbindung externer Webanwendung an PDF- AS-WEB 5.0

## Anbindung einer externen Webanwendung an PDF-AS-WEB 5.0

Version 1.2, 30.04.2025

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

**Zusammenfassung:** Dieses Dokument beschreibt die Anbindung externer Webanwendung an PDF-AS-WEB ab der Version 5.0. Es beschreibt die verfügbaren Parameter der PDF-AS-WEB Komponenten, sowie die Parameter die von PDF-AS-WEB an die externe Webanwendung übergeben werden.

## Inhaltsverzeichnis

1 Beschreibung der Schnittstelle .....	3
1.1 Übergabe per User Agent .....	3
1.2 Upload per SOAP Schnittstelle.....	5
1.3 Serversignatur per SOAP Schnittstelle .....	6
1.4 JSON-API .....	6
2 Beschreibung der Implementation.....	7
2.1 PDF-AS-WEB Servlets .....	7
2.1.1 Sign Servlet .....	7
2.1.2 ProvidePDF Servlet .....	9
2.1.3 PDFData Servlet.....	9
2.1.4 Verifikations Servlet .....	11
2.2 Parameterübergabe an die externe Webanwendung .....	11
2.2.1 Abrufen des originalen PDF Dokuments .....	11
2.2.2 Fehlerseite der externen Webanwendung .....	12
2.2.3 Benachrichtigung nach erfolgreicher Signatur.....	12

## Abbildungsverzeichnis

Abbildung 1 - Prozessablauf zur Anbindung einer externen Webanwendung und PDF-AS-WEB via User Agent<sup>4</sup>

Abbildung 2 - Prozessablauf zur Anbindung einer externen Webanwendung und PDF-AS-WEB via SOAP Schnittstelle<sup>5</sup>

# 1 Beschreibung der Schnittstelle

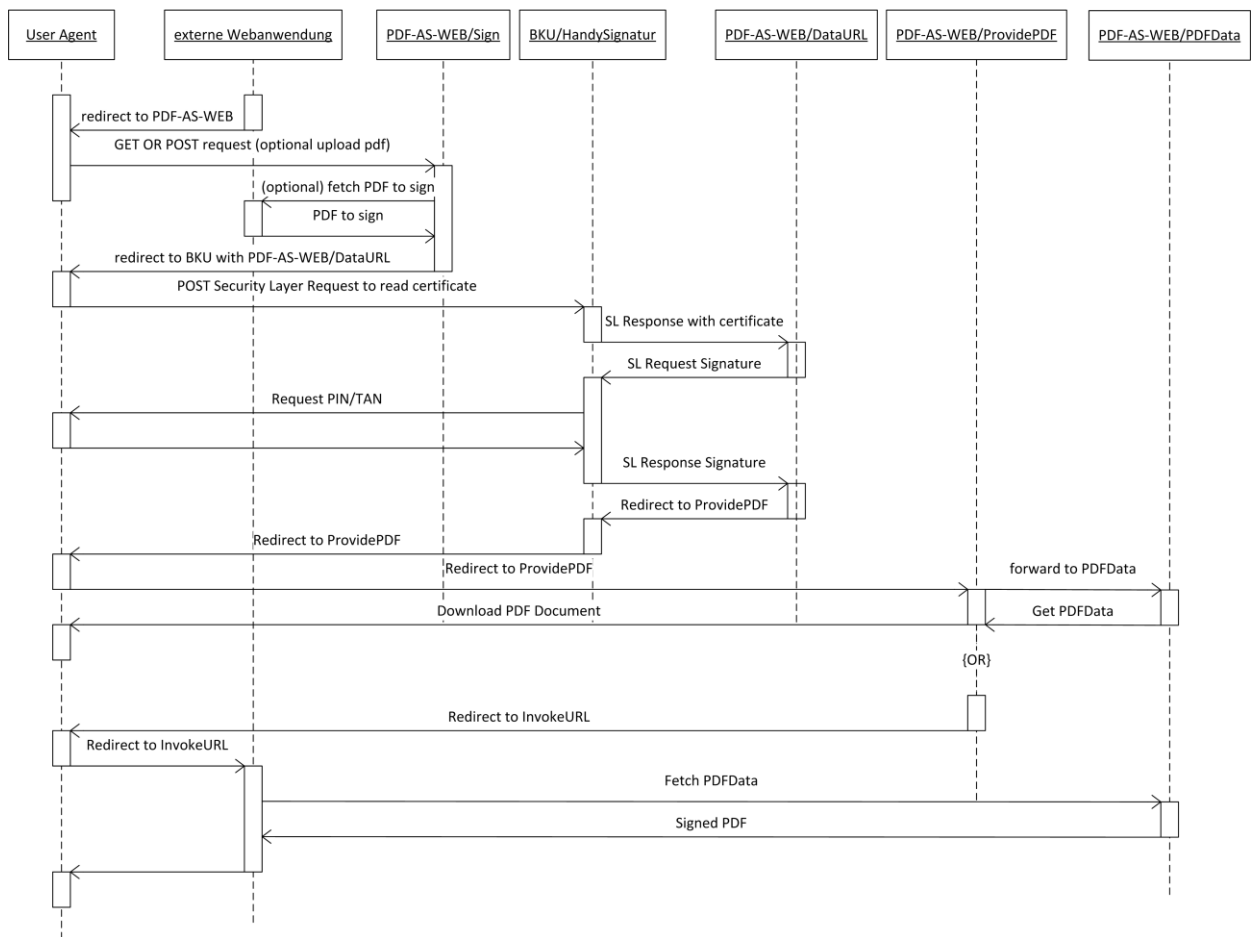
---

PDF-AS-WEB bietet unterschiedliche Möglichkeiten zur Anbindung einer externen Webanwendung:

- **Übergabe per User Agent:** In dieser Möglichkeit wird der Benutzer von der externen Webanwendung an PDF-AS-WEB weitergeleitet. PDF-AS-WEB bezieht in diesem Fall das zu signierende PDF von der externen Webanwendung.
- **Upload per SOAP Schnittstelle:** In dieser Möglichkeit wird das zu signierende PDF per SOAP Service von der externen Webanwendung direkt an PDF-AS-WEB hochgeladen. Als Antwort erhält die externe Webanwendung eine URL an die sie den Benutzer weiterleiten muss.
- **Serversignatur per SOAP Schnittstelle:** In dieser Möglichkeit wird das zu signierende PDF per SOAP Service von der externen Webanwendung direkt an PDF-AS-WEB hochgeladen. Als Antwort erhält die externe Webanwendung das signierte Dokument. Diese Möglichkeit funktioniert nur wenn das PDF Dokument direkt vom Server signiert werden soll.

## 1.1 Übergabe per User Agent

In Abbildung 1 - Prozessablauf zur Anbindung einer externen Webanwendung und PDF-AS-WEB wird der Ablauf der Signatur eines PDF Dokumentes einer externen Webanwendung durch PDF-AS-WEB dargestellt.



**Abbildung 1 - Prozessablauf zur Anbindung einer externen Webanwendung und PDF-AS-WEB via User Agent**

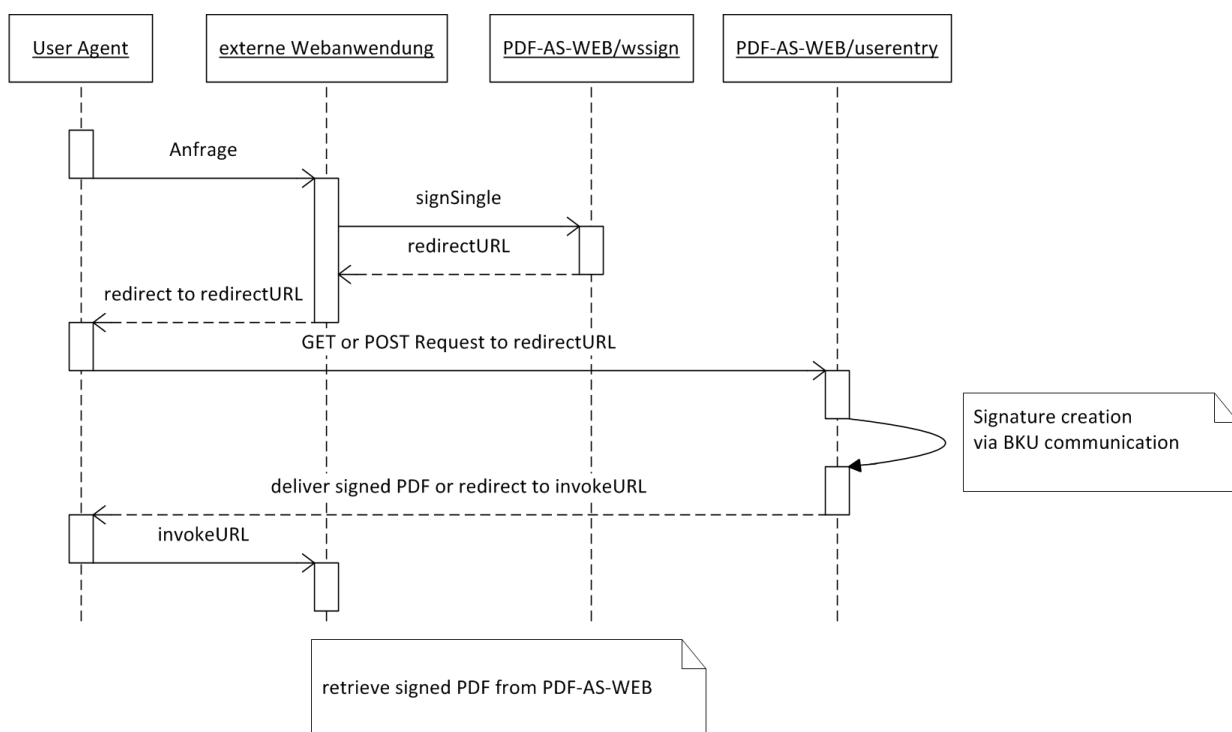
Im ersten Schritt leitet die externe Webanwendung den Benutzer zu PDF-AS-WEB um. Im Zuge dieser Weiterleitung werden PDF-AS-WEB diverse Parameter übergeben. Eine detaillierte Beschreibung der Parameter folgt später. Im ersten Request des Benutzers an PDF-AS-WEB kann entweder das zu signierende PDF Dokument per POST Request hochgeladen werden oder eine URL angegeben werden von der PDF-AS-WEB das PDF Dokument herunterladen soll. Sobald PDF-AS-WEB im Besitz des zu signierenden PDF Dokumentes ist wird die Signaturerstellung vorbereitet. PDF-AS-WEB bereitet einen Security Layer Request vor und leitet den Benutzer zur der ausgewählten Bürgerkartenumgebung um. Die Bürgerkartenumgebung bearbeitet die Security Layer Requests und sendet das Ergebnis per Data URL an PDF-AS-WEB. Wenn die Signatur erzeugt wurde antwortet PDF-AS-WEB mit einer Weiterleitung an das ProvidePDF Servlet mit entsprechender Data URL. Somit wird der Benutzer von der Bürgerkartenumgebung auf das ProvidePDF Servlet von PDF-AS-WEB weitergeleitet. Je nachdem ob die externe Webanwendung eine InvokeURL angegeben hat, wird der Benutzer auf diese URL weitergeleitet, oder es wird dem Benutzer angeboten das PDF Dokument direkt von PDF-AS herunter zu laden. Ein signiertes PDF Dokument kann nur ein einziges Mal von

PDF-AS-WEB heruntergeladen werden. Wird der Benutzer wieder an die externe Webanwendung weitergeleitet, so bekommt diese einen Link um das signierte PDF von PDF-AS-WEB herunterzuladen. In diesem letzten Schritt hat die Webanwendung auch die Möglichkeit den SHA-256 Hash der originalen Signatur Daten an PDF-AS-WEB zu schicken. Nur wenn die Originaldaten des signierten PDF Dokumentes zu diesem Hash passen, wird das PDF-Dokument von PDF-AS geliefert. Diese Überprüfung ist allerdings optional.

## 1.2 Upload per SOAP Schnittstelle

Die Endpunkt für das SOAP Webservice in PDF-AS-WEB befindet sich unter „services/wssign“. Unter der URL „services/wssign?wsdl“ kann die Web Service Description abgefragt werden. Dieses WSDL dient als Dokumentation für das SOAP Service.

In Abbildung 2 - Prozessablauf zur Anbindung einer externen Webanwendung und PDF-AS-WEB via SOAP Schnittstelle ist der Ablauf einer Signatur per SOAP Schnittstelle beschrieben.



**Abbildung 2 - Prozessablauf zur Anbindung einer externen Webanwendung und PDF-AS-WEB via SOAP Schnittstelle**

Die externe Webanwendung startet den Signaturvorgang in dem ein SOAP Request an PDF-AS-WEB geschickt wird. Dieser Request enthält das zu signierende PDF Dokument,

sowie den zu verwendenden *connector*, die *invoke-url*, die *invoke-error-url* und die Signaturposition. Also Antwort empfängt die Webanwendung eine URL. Der Benutzer muss an diese URL weitergeleitet werden damit PDF-AS-WEB eine Benutzerschnittstelle für die Bürgerkartenumgebung zur Verfügung hat. Nachdem das Dokument durch den Benutzer signiert wurde, kann dieser das PDF Dokument von PDF-AS-WEB herunterladen, bzw. falls eine *invoke-url* definiert wurde, wird der Benutzer an diese URL weitergeleitet. Somit hat die Webanwendung die Möglichkeit das signierte Dokument von PDF-AS-WEB zu beziehen.

Mit der Version 4.3.0 von PDF-AS wurde zwei weitere Methoden „signMultiple“ und „getMultiple“ eingeführt welche Bulk-Light Signaturen auch für clientbasierte Signature (z.B. MOBILEBKU) ermöglichen. Der allgemeine Prozessfluss ist weitestgehend identisch zu Abbildung 2, jedoch wird im Falle der Verwendung einer *invoke-url*, das als *invoke-url* übergebene Endpunkt inklusive eines HTTP GET Parameters „token“ aufgerufen, z.B. <http://localhost/sp?token=46f01421-1ac6-4640-9027-ebd1323bbab3>. Mittels diesen Parameterwert und der Verwendung der SOAP Methode „getMultiple“ können anschließend alle signierten Dokumente via SOAP Webservice bezogen werden. Falls keine *invoke-url* definiert wurde werden die signierten Dokumente als ZIP Archiv zum Download angeboten.

### 1.3 Serversignatur per SOAP Schnittstelle

Es wird dasselbe Web Service wie unter 1.2 verwendet um Serversignaturen zu erstellen. Der einzige Unterschied ist in diesem Fall, dass als Connector Parameter im SOAP Request „jks“ oder „moa“ angegeben werden müssen. PDF-AS-WEB verwendet dann einen Software Keystore, bzw. eine MOA-SS Instanz im Hintergrund zur Erzeugung der Signatur.

### 1.4 JSON-API

Zusätzlich den in 2.1 beschriebenen SOAP-Servlets bietet PDF-AS-Web auch eine funktionell äquivalente JSON-API. Diese ist unter folgenden Endpunkten erreichbar:

- **„/api/v2/sign/single“** – entspricht der „signSingle“-Operation auf „wssign“
- **„/api/v2/sign/bulk“** – entspricht der „signBulk“-Operation
- **„/api/v2/sign/multiple“** – entspricht der „signMultiple“-Operation
- **„/api/v2/sign/multiple/get-result“** – entspricht der „getMultiple“-Operation
- **„/api/v2/verify“** – entspricht der „verify“-Operation auf „wsverify“

Diese JSON-API ist auch über OpenAPI 3 maschinenlesbar dokumentiert.

## 2 Beschreibung der Implementation

---

### 2.1 PDF-AS-WEB Servlets

Hier werden die Parameter beschrieben, die PDF-AS-WEB externen Webanwendungen zur Steuerung anbietet.

#### 2.1.1 Sign Servlet

Das Sign Servlet ist der Haupteinstiegspunkt in PDF-AS-WEB. Dieses Servlet kann entweder mittels GET oder POST Methode aufgerufen werden. Es muss entweder der **pdf-file** Parameter oder der **pdf-url** Parameter angegeben werden.

- **„connector“**: Dieser Parameter gibt an welches Signaturgerät verwendet werden soll. (**Pflichtparameter**) Mögliche Werte sind:
  - **„bku“**: Es soll die lokale Bürgerkartenumgebung verwendet werden.
  - **„onlinebku“**: Es die MOCCA online als Bürgerkartenumgebung verwendet werden.
  - **„mobilebku“**: Es soll die Handy Signatur verwendet werden.
  - **„moa“**: Es soll eine MOA-SS Instanz verwendet werden.
  - **„jks“**: Es soll ein Keystore verwendet werden.
- **„keyId“**: Dieser Parameter gibt für MOA/Keystore-Signaturen den zu verwenden Key-Identifizier an. Zur Konfiguration von Key-Identifiern wird auf die Dokumentation der Konfigurationsdatei für PDF-AS-Web verwiesen.
- **„invoke-app-url“**: Dieser Parameter gibt die Fertigstellungsurl der externen Webanwendung an. Der Benutzer wird nach Abschluss des Signaturvorgangs wieder auf diese URL umgeleitet. (**optionaler Parameter**)
- **„invoke-app-error-url“**: Dieser Parameter gibt die Fehlerurl der externen Webanwendung an. Im Fehlerfall wird der Benutzer auf diese URL umgeleitet. (**optionaler Parameter**)
- **„locale“**: Die Locale die PDF-AS-WEB verwenden soll. Wird kein Parameter angegeben werden die Locale vom Benutzerbrowser übernommen. (**optionaler Parameter**) Mögliche Werte sind:
  - **„DE“**
  - **„EN“**
- **„sig-type“**: Dieser Parameter gibt das zu verwendete Signaturprofil von PDF-AS an. Wird dieser Parameter nicht angegeben, wird das Standardsignaturprofil verwendet. (**optionaler Parameter**) Mögliche Standardwerte sind:
  - SIGNATURBLOCK\_DE
  - SIGNATURBLOCK\_EN
  - SIGNATURBLOCK\_SMALL\_DE
  - SIGNATURBLOCK\_SMALL\_EN

- AMTSSIGNATURBLOCK\_DE
  - AMTSSIGNATURBLOCK\_EN
  - AMTSSIGNATURBLOCK\_DE\_SMALL
  - AMTSSIGNATURBLOCK\_EN\_SMALL
- „**pdf-file**“: Dieser Parameter gibt den File Parameter an beim upload des PDF Dokuments mittels POST Methode. (**optional Parameter**)
- „**pdf-url**“: Dieser Parameter gibt die URL an, von der das zu signierende PDF Dokument heruntergeladen werden soll. (**optional Parameter**)
- „**sig-pos-p**“: Mit diesem Parameter kann die Seite auf der die Signatur positioniert werden soll festgelegt werden. (**optional Parameter**)
- „**sig-pos-w**“: Mit diesem Parameter kann die Breite des Signaturblocks festgelegt werden. (**optional Parameter**)
- „**sig-pos-x**“: Mit diesem Parameter kann die X Koordinate der Position des Signaturblocks festgelegt werden. (**optional Parameter**)
- „**sig-pos-y**“: Mit diesem Parameter kann die Y Koordinate der Position des Signaturblocks festgelegt werden. (**optional Parameter**)
- „**sig-pos-r**“: Mit diesem Parameter kann die Rotation (in Grad) des Signaturblocks festgelegt werden. (**optional Parameter**)
- „**sig-pos-f**“: Mit diesem Parameter kann die Footerhöhe bei automatischer Platzierung des Signaturblocks vorgegeben werden. (**optional Parameter**)
- „**invoke-app-url-target**“: Mit diesem Parameter lässt sich der Frame festlegen in dem der Benutzer nach Abschluss des Signaturvorgangs an die „**invoke-app-url**“ umgeleitet wird. Dieser Parameter wird als target Attribute eines HTML Formulars eingetragen. Daher sind alle Werte gültig die für dieses Attribut gültig sind. Mögliche Werte für dieses Attribut sind:
  - **\_blank**: Die „**invoke-app-url**“ wird in einem neuen Fenster oder Tab geöffnet.
  - **\_self**: Die „**invoke-app-url**“ wird im aktuellen Frame geöffnet. Das ist das Standardverhalten von PDF-AS Web.
  - **\_parent**: Die „**invoke-app-url**“ wird im parent frame geöffnet.
  - **\_top**: Die „**invoke-app-url**“ wird der komplette Inhalt des aktuellen Fensters.
  - **Framename**: Die „**invoke-app-url**“ wird in einem benannten iframe angezeigt.
- „**verify-level**“: Mit diesem Parameter lässt sich festlegen, welche Art von Signaturverifikation nach dem Signaturvorgang durchgeführt werden soll.
  - „**full**“: Dies bedeutet, dass eine vollständige Signaturprüfung inklusive Zertifikatsprüfung durchgeführt werden soll. Diese Option erfordert eine korrekte Konfiguration von MOA-SP.



- „intOnly“: Dies bedeutet, dass eine Signaturprüfung durchgeführt wird, allerdings ohne Zertifikatsprüfung.
- „filename“: Mit diesem Parameter lässt sich der Name der unterschriebenen Datei festlegen. (**optionaler Parameter**)
- „qrcode“: Mit diesem Parameter lässt sich ein Text übergeben, welcher von als QR Code als Signaturmarke verwendet werden soll. (**optionaler Parameter**)
- „placeholder\_web\_enabled“: Mit diesem Parameter kann die Suche nach Placeholdern für diesen Request aktiviert (*true*) oder deaktiviert (*false*) werden. (**optionaler Parameter**)
- „placeholder\_web\_id“: Mit diesem Parameter kann ein spezifischer QR-Code Placeholder anhand dessen ID ausgewählt werden. (**optionaler Parameter**)
- „pp“: Dies ist ein Prefix für Preprocessor Argumente. Soll beispielsweise das Argument „testid“ übergeben werden, so kann ein Parameter „pp:testid“ verwendet werden. (**optionaler Parameter**)
- „ov“: Dies ist ein Prefix für das Überschreiben von Konfigurationseinstellung. Soll beispielsweise das Argument „testid“ übergeben werden so kann ein Parameter „ov:testid“ verwendet werden. (**optionaler Parameter**)
- „sbp“: Mit diesem Parameter können dynamische Signaturblockparameter übergeben werden. Es muss sich dabei um Key-Value Paare handeln, die in der Form „sbp:keyA=valueA“ bzw. „sbp:keyB=valueB“ übergeben werden. (**optionaler Parameter**)

### 2.1.2 ProvidePDF Servlet

Das Provide PDF Servlet lässt keine Parameter zu. Das Verhalten wird durch die Parameter gesteuert, die beim Aufruf des SignServlet übergeben wurden.

### 2.1.3 PDFData Servlet

Das PDFData Servlet returniert, falls vorhanden, das signierte PDF Dokument. Vorsicht: Dieses Servlet kann nur ein einziges Mal pro Signaturvorgang aufgerufen werden. Nachdem das signierte Dokument abgeholt wurde, werden die Daten des signierten PDF Dokumentes gelöscht.

- „origdigest“: Dieser Parameter kann den Hex kodierten SHA256 Wert des Originaldokuments enthalten. Wenn dieser Parameter angegeben wird, prüft PDF-AS-WEB, ob der Wert mit dem Wert der originalen Daten übereinstimmt. Nur wenn dieser übereinstimmt, werden die Signaturdaten returniert. Wenn nicht, wird ein Fehler generiert. (**optionaler Parameter**)

Das PDFData Servlet returniert auch das Zertifikat mit welchem die Signatur durchgeführt wurde. Dieses Zertifikat ist als Server Header „Signer-Certificate“ in der Response zu finden. Auch das Ergebnis der automatisch durchgeführten Signaturprüfung nach dem Signaturvorgang wird über die Server Header übergeben. Die Server Header „CertificateCheckCode“ und „ValueCheckCode“ sind in der Antwort des PDFData Servlet enthalten. Die Werte dieser beiden Header entsprechen den Werten eine MOA Signaturprüfung:

„ValueCheckCode“ Werte sind:

Wert	Bedeutung
0	Die Überprüfung des Werts der Signatur konnte erfolgreich durchgeführt werden.
1	Bei der Überprüfung des Werts der Signatur ist ein Fehler aufgetreten.

„CertificateCheckCode“ Werte sind:

Wert	Bedeutung
0	Eine formal korrekte Zertifikatskette vom Signatorzertifikat zu einem vertrauenswürdigen Wurzelzertifikat konnte konstruiert werden. Jedes Zertifikat dieser Kette ist zum in der Anfrage angegebenen Prüfzeitpunkt gültig.
1	Es konnte keine formal korrekte Zertifikatskette vom Signatorzertifikat zu einem vertrauenswürdigen Wurzelzertifikat konstruiert werden.
2	Eine formal korrekte Zertifikatskette vom Signatorzertifikat zu einem vertrauenswürdigen Wurzelzertifikat konnte konstruiert werden. Für zumindest ein Zertifikat dieser Kette fällt der Prüfzeitpunkt nicht in das Gültigkeitsintervall.
3	Eine formal korrekte Zertifikatskette vom Signatorzertifikat zu einem vertrauenswürdigen Wurzelzertifikat konnte konstruiert werden. Für alle Zertifikate dieser Kette fällt der Prüfzeitpunkt in das jeweilige Gültigkeitsintervall. Für zumindest ein Zertifikat konnte der Zertifikatsstatus nicht festgestellt werden.
4	Eine formal korrekte Zertifikatskette vom Signatorzertifikat zu einem vertrauenswürdigen Wurzelzertifikat konnte konstruiert werden. Für alle Zertifikate dieser Kette fällt der Prüfzeitpunkt in das jeweilige Gültigkeitsintervall. Zumindest ein Zertifikat ist zum Prüfzeitpunkt widerrufen.
5	Eine formal korrekte Zertifikatskette vom Signatorzertifikat zu einem vertrauenswürdigen Wurzelzertifikat konnte konstruiert werden. Für alle Zertifikate dieser Kette fällt der Prüfzeitpunkt in das jeweilige Gültigkeitsintervall. Kein Zertifikat dieser Kette ist zum Prüfzeitpunkt

	widerrufen. Zumindest ein Zertifikat ist zum Prüfzeitpunkt gesperrt.
99	Die Prüfung der Signaturprüfdaten wurde nicht durchgeführt, da bei der Prüfung der Gültigkeit der Signatur ein Fehler aufgetreten ist.

### 2.1.4 Verifikations Servlet

Das Verifikations Servlet ist unter „**/Verify**“ erreichbar. Mit diesem Servlet können Signaturen in PDF Dokumenten verifiziert werden. Dieses Servlet unterstützt folgende Parameter:

- **„pdf-file“**: Dieser Parameter gibt den File Parameter an beim upload des PDF Dokuments mittels POST Methode. (**optionaler Parameter**)
- **„pdf-url“**: Dieser Parameter gibt die URL an, von der das zu verifizierende PDF Dokument heruntergeladen werden soll. (**optionaler Parameter**)
- **„verify-level“**: Mit diesem Parameter lässt sich festlegen, welche Art von Signaturverifikation nach dem Signaturvorgang durchgeführt werden soll.
  - **„full“**: Dies bedeutet, dass eine vollständige Signaturprüfung inklusive Zertifikatsprüfung durchgeführt werden soll. Diese Option erfordert eine korrekte Konfiguration von MOA-SP.
  - **„intOnly“**: Dies bedeutet, dass eine Signaturprüfung durchgeführt wird, allerdings ohne Zertifikatsprüfung.
- **„format“**: Mit diesem Parameter kann das Ausgabeformat festgelegt werden. Wird diese Parameter nicht übergeben, wird eine HTML Seite generiert. (**optionaler Parameter**)
  - **„html“**: Dies erzeugt eine einfache HTML Seite mit den Verifikationsresultaten.
  - **„json“**: Dies erzeugt eine einen JSON String mit den Verifikationsresultaten.

## 2.2 Parameterübergabe an die externe Webanwendung

Hier werden die Parameter beschrieben, die PDF-AS-WEB an die externe Webanwendung übergibt.

### 2.2.1 Abrufen des originalen PDF Dokuments

Wenn PDF-AS-WEB das PDF Dokument von der externen Webanwendung herunterladen soll. Werden von PDF-AS keine Parameter übergeben, sondern nur die URL die mit **pdf-url** an das Sign Servlet übergeben wurde aufgerufen. PDF-AS-WEB bietet ein Whitelisting um pdf-urls zu filtern.

### 2.2.2 Fehlerseite der externen Webanwendung

Wenn in PDF-AS-WEB eine externen Fehlerurl mit dem Parameter ***invoke-app-error-url*** übergeben wurde und in PDF-AS-WEB ein Fehler auftritt, wird der Benutzer auf diese URL weitergeleitet. Diese URL wird mittels GET Methode aufgerufen und 2 Parameter werden übergeben:

- „**error**“: Dieser Parameter enthält einen Text mit einer einfachen Fehlermeldung
- „**cause**“: Dieser Parameter enthält einen Text mit einer detaillierteren Fehlerbeschreibung

PDF-AS-WEB bietet ein Whitelisting um Fehlerurls zu filtern.

### 2.2.3 Benachrichtigung nach erfolgreicher Signatur

Wenn in PDF-AS-WEB eine Fertigstellungurl mit dem Parameter ***invoke-app-url*** übergeben wurde, leitet PDF-AS-WEB nach einer erfolgreichen Signatur den Benutzer an diese URL weiter. Dieser URL wird mittels GET Methode aufgerufen und 2 Parameter werden übergeben:

- „**pdfurl**“: Die URL unter der das signierte Dokument abgerufen werden kann.
- „**pdflength**“: Die Anzahl der Bytes die das signierte Dokument hat.

PDF-AS-WEB bietet ein Whitelisting um Fertigstellungurl zu filtern.

## Dokumentenhistorie

Version	Datum	Autor(en)	Anmerkung
0.1	21.01.2014	Andreas Fitzek	Initiale Version
0.2	19.02.2014	Christian Maierhofer	Review
0.3	05.06.2014	Andreas Fitzek	Erweiterungen SOAP Schnittstelle
0.4	06.06.2014	Andreas Fitzek	sig-type Werte hinzugefügt
0.5	15.07.2014	Andreas Fitzek	Erweiterung invoke-app-url-target, verify-level, Server Headers in PDFData
0.6	19.08.2014	Andreas Fitzek	VerifyServlet
0.7	26.09.2014	Andreas Fitzek	filename parameter
0.8	26.03.2015	Andreas Fitzek	qrcontent, pp: und ov: Parameter
0.9	11.06.2021	Alexander Marsalek	sbp Parameter
1.0	02.03.2023	Thomas Lenz	Bulk-Light Funktion hinzugefügt
1.1	20.12.2023	Thomas Lenz	SignServlet Parameter hinzugefügt
1.2	30.04.2026	Jakob Heher	JSON-API hinzugefügt, PDF-AS 5.0