Dokumentation

PDF-AS Webanwendung Dokumentation

Dokumentation zur PDF-AS Webanwendung ab Version 4

Version 1.0, 05.12.2023

Thomas Lenz – thomas.lenz@egiz.gv.at
Andreas Fitzek – andreas.fitzek@egiz.gv.at

Emina Ahmetovic – emina.ahmetovic@egiz.gv.at

Zusammenfassung: Dieses Dokument beschreibt die Konfiguration der PDF-AS Webanwendung. Die PDF-AS Webanwendung erlaubt es, PDF Dokumente über ein Web Interface zu unterschreiben. Des Weiteren ermöglicht die PDF-AS Webanwendung externen Webanwendungen PDF Dokumente zu signieren.

E-Government Innovationszentrum
Inffeldgasse 16/a, 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

1 Deployment

1.1 Konfigurationsparameter

Im Java Anwendungsserver, zum Beispiel dem Apache Tomcat Server, muss eine Java Umgebungsvariable „pdf-as-web.conf“ definiert sein. Der Wert dieser Umgebungsvariablen ist der Dateipfad zur PDF-AS Web Konfigurationsdatei. Ein Beispielparameter für den Apache Tomcat Server:

```
-Dpdf-as-web.conf=“/…/pdf-as-web.properties“
```

1.2 Security Layer 2.0

Die Versionen von PDF-AS enthalten ab 4.1.5 die Security Layer 2.0-Funktionalität, die nur für PDF-AS Web verfügbar ist.

1.3 SOAP-Schnittstelle

PDF-AS Web bietet zwei SOAP-Schnittstellen um Dokumente zu unterschreiben bzw. zu verifizieren.

Die SOAP-Schnittstelle um Signaturen zu erzeugen ist definiert durch eine WSDL Datei. Die WSDL Datei kann unter *„{pdf-as-web-url}/services/wssign?wsdl“* abgerufen werden. Anwendungen können die SOAP-Schnittstelle nutzen um PDF-Dokumente direkt an PDF-AS Web hochzuladen. Bei diesem Request können diverse Signaturoptionen, wie beispielsweise der „Connector“, übergeben werden. Mit Hilfe der „Connector“ Option wird bestimmt womit die Signatur erzeugt werden soll. Hier können serverbasierte Signaturen, wie „JKS“ oder „MOA“ verwendet werden, oder clientbasierte Signaturen, wie „BKU“, „MOBILEBKU“ oder „ONLINEBKU“. Wird eine serverbasierte Signatur verwendet, ist das signierte Dokument in der Antwort enthalten. Wird eine clientbasierte Signatur verwendet, benötigt PDF-AS Web eine Benutzerschnittstelle um direkt mit dem Benutzer zu kommunizieren. In diesem Fall ist eine URL in der Antwort enthalten, auf die der Benutzer weitergeleitet werden muss. Das signierte Dokument wird dann entweder direkt an den Benutzer übermittelt oder die aufrufende Anwendung kann per „invoke-url“ informiert werden. „Abbildung 1.: Ablauf einer Signatur über die SOAP-Schnittstelle“ zeigt den Ablauf einer clientbasierten Signatur mit Hilfe der SOAP-Schnittstelle.

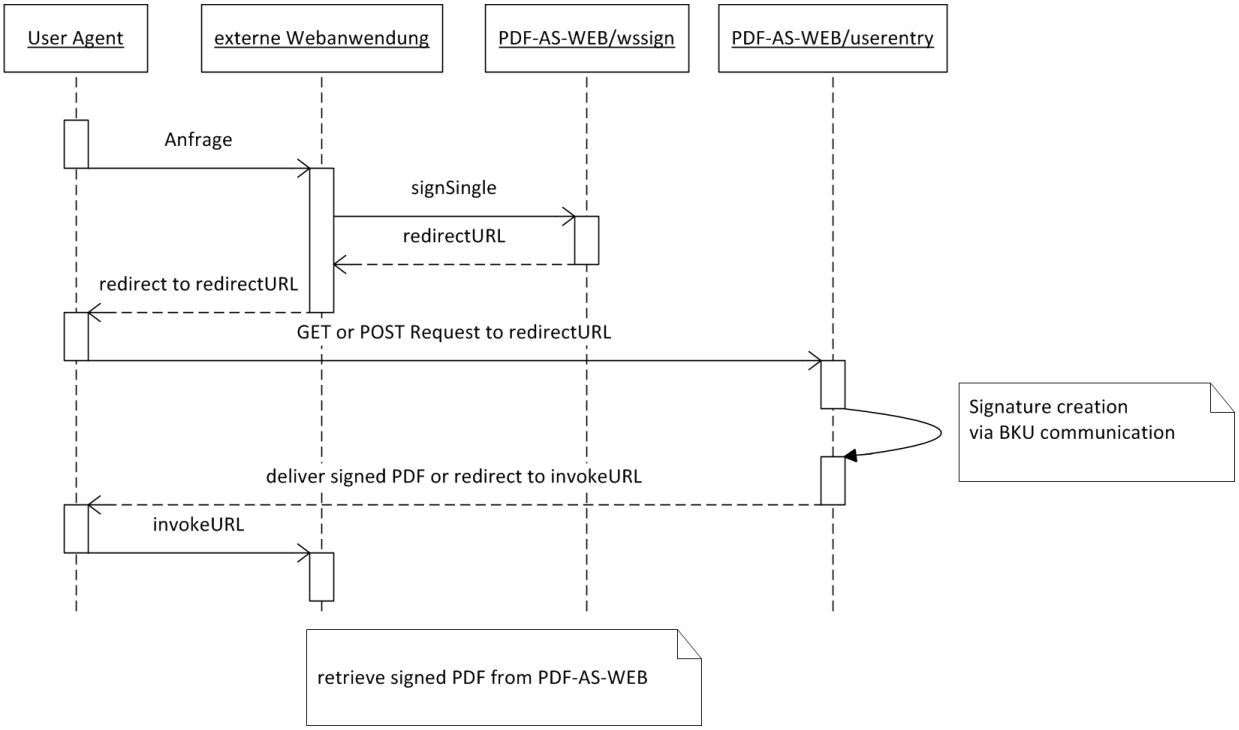

Abbildung 1.: Ablauf einer Signatur über die SOAP-Schnittstelle

Die SOAP-Schnittstelle bietet vier Methoden „signSingle“, „signBulk“, „signMultiple“ und „getMultiple“. Mit der Methode „signSingle“ lässt sich ein Dokument signieren. Mit der Methode „signBulk“ lassen sich mehrere Signaturanfragen im Bulk erstellen, wobei diese Methode nur für Serversignaturen (JKS, oder MOA) zur Verfügung steht.

Mit der Version 4.3.0 von PDF-AS wurden zwei weitere Methoden „signMultiple“ und „getMultiple“ eingeführt welche Bulk-Light Signaturen auch für clientbasierte Signature (z.B. MOBILEBKU) ermöglichen. Hierbei können PDF-AS mehrere zu signierende Dokumente übergeben werden welche anschließend dem Benutzer zur Unterzeichnung vorgelegt werden. Im Unterschied zur mehrfachen Verwendung von „signSingle muss der Benutzer*innen* im Falle von bulk-light nur ein Mal seinen Benutzernamen / Passwort bekannt geben und kann anschließend alle Dokumente via zweiten Faktor, z.B. Mobile App, freigeben. Die signierten Dokumente können anschließend entweder als ZIP Archiv, identisch zum „signSingle“ Prozess, von PDF-AS abgeholt werden, oder via eines als HTTP GET Parameter an die *invokeURL* returniertes token (z.B. http://localhost/sp?token=46f01421-1ac6-4640-9027-ebd1323bbab3), welches anschließend an der SOAP Schnittstelle unter Verwendung der Methode „getMultiple“ eingelöst werden kann. In diesem Fall werden die signierten Dokumente via SOAP Response ausgeliefert.

Die SOAP-Schnittstelle um Dokumente zu verifizieren ist definiert durch eine WSDL Datei. Die WSDL Datei kann unter *„{pdf-as-web-url}/services/wsverify?wsdl“*

abgerufen werden. Die SOAP-Schnittstelle bietet eine Methode „verify“ an. Diese Methode verifiziert ein Dokument und antwortet mit einer Liste von Signaturergebnissen. Für jede Signatur im Dokument wird ein Signaturergebnis gelistet.

1.4 PDF-AS Web Clusterbetrieb

PDF-AS Web lässt sich in einem Cluster betreiben. Vorrausetzung für die korrekte Funktion von PDF-AS Web im Cluster ist, dass der Lastverteiler bestehende Sitzungen immer zur selben Instanz weiterleitet. Es muss also eine Session Fixation im Load Balancer geschehen. Ein Beispiel dafür mit zwei Tomcat Instanzen als Anwendungsserver und einem Apache Webserver.

In den beiden Tomcat Instanzen muss der Parameter jvmRoute für die Catalina engine festgelegt werden. Der Wert dieses Parameters muss sich in beiden Instanzen unterscheiden. In unserem Beispiel ist die „node1“ und „node2“

Dazu wird in der der conf/server.xml die Zeile „<Engine name=“Catalina“ defaultHost=“localhost“>“ geändert zu „<Engine name=“Catalina“ defaultHost=“localhost“ jvmRoute=“node1“>“ bzw. „<Engine name=“Catalina“ defaultHost=“localhost“ jvmRoute=“node2“>“.

Wenn beide Tomcat Instanz auf demselben Server ausgeführt werden, müssen auch die AJP Ports unterschiedlich sein. In unserem Beispiel sind diese auf 8009 und 8029 festgelegt.

Im Apache Webserver müssen die Module „proxy_ajp“ und „proxy_balancer“ aktiviert sein. Als erstes wird ein Balancer definiert:

```
<Proxy balancer://testcluster stickysession=JSESSIONID|jsessionid
scolonpathdelim=On>
BalancerMember ajp://127.0.0.1:8009 min=10 max=100 route=node1 loadfactor=1
BalancerMember ajp://127.0.0.1:8029 min=10 max=100 route=node2 loadfactor=1
</Proxy>
```

Als letzer Schritt muss noch ein ProxyPass konfiguriert werden, welcher auf den Balancer zeigt:

```
ProxyPass /pdf-as-web balancer://testcluster/pdf-as-web
```

Wenn PDF-AS Web im Clusterbetrieb verwendet wird und mittels SOAP-Schnittstelle clientbasierte Signaturen erzeugt werden, benötigt PDF-AS Web eine Datenbank um die Clusterfähigkeit sicherzustellen. Nur in diesem Fall ist es zu empfehlen eine Datenbank für PDF-AS Web zu konfigurieren.

1.5 PDF-AS Web Konfigurationsdatei

Die PDF-AS Web Konfigurationsdatei ist eine simple Java Properties Datei mit folgenden Einträgen:

Name	Werteberiech	Beschreibung
pdfas.dir	*Verzeichnis*	Das Basisverzeichnis für die PDF-AS Konfiguration
error.showdetails	*true* | *false*	Legt fest ob PDF-AS im Fehlerfall eine detaillierte Fehlermeldung liefern soll.
bku.sign.url | bku.local.url	*URL*	Die URL die PDF-AS für die lokale Bürgerkartenumgebung verwenden soll. Ist dieser Parameter nicht definiert, so wird der Parameter bku.sign.url aus der Basis PDF-AS Konfiguration verwendet.
bku.sign.enabled	*true* | *false*	PDF-AS-WEB verbietet die Benutzung einer lokalen Bürgerkartenumgebung wenn dieser Parameter auf false gesetzt ist.
moc.sign.url | bku.online.url	*URL*	Die URL die PDF-AS für die online Bürgerkartenumgebung verwenden soll. Ist dieser Parameter nicht definiert, so wird der Parameter moc.sign.url aus der Basis PDF-AS Konfiguration verwendet.
moc.sign.enabled	*true* | *false*	PDF-AS-WEB verbietet die Benutzung einer online Bürgerkartenumgebung wenn dieser Parameter auf false gesetzt ist.
mobile.sign.url | bku.mobile.url	*URL*	Die URL die PDF-AS für die Handy Signatur verwenden soll. Ist dieser Parameter nicht definiert, so wird der Parameter mobile.sign.url aus der Basis PDF-AS Konfiguration verwendet.
mobile.sign.enabled	*true* | *false*	PDF-AS-WEB verbietet die Benutzung der Handy Signatur wenn dieser Parameter auf false gesetzt ist.
public.url	*URL*	Legt den öffentlichen Zugangspunkt für PDF-AS Web fest. (optional)
public.data.url	*URL*	Legt einen spezifischen Zugangspunkt für die DataURL für die Kommunikation via Security-Layer fest. (option) Dieser Zugangspunkt wird ausschließlich für die DataURL verwendet und überschreibt die globale Konfiguration des öffentlichen Zugangspunkts (*public.url*)
moa.enabled	*true* | *false*	Legt fest ob PDF-AS die Signatur mit einem MOA-SS Server zulassen soll. Der MOA-SS Server wird in der Basiskonfiguration von PDF-AS konfiguriert.

ks.enabled	*true* | *false*	Legt fest ob PDF-AS die Signatur mit einem KeyStore am Server zulassen soll.
ks.file	*Dateipfad*	Die bei einer Signatur mit einem Keystore zu verwendende KeyStore Datei.
ks.type	*JKS* | *PKCS12*	Die Art der Keystore Datei
ks.pass	*Passwort*	Das Passwort für die Keystore Datei
ks.key.alias		Der Bezeichner des Schlüssels in der KeyStore Datei mit dem die Signatur durchgeführt werden soll.
ks.key.pass	*Passwort*	Das Passwort für den Schlüssel in der KeyStore Datei.
ksl.(id).*		Mit ksl wird eine Liste von Keystore Einträgen ermöglicht. Die Zusammensetzung ergibt sich als ksl.(identifier).(property). Der Identifier identifiziert den Eintrag in der Keystoreliste. Für jeden Eintrag müssen die Eigenschaften für diesen Keystore definiert werden. Diese Eigenschaften sind äquivialent zu den Keystore Eigenschaften ks.*.
moal.(id).*		Mit moal wird eine Liste von MOA-SS Einträgen ermöglicht. Die Zusammensetzung ergibt sich als moal.(identifier).(property). Der Identifier identifiziert den Eintrag in der Liste. Für jeden Eintrag müssen die Eigenschaften für diese MOA-SS Anbindung definiert werden. Diese Eigenschaften sind „enabled“, „url“, „KeyIdentifier“, „Certificate“. Diese Eigenschaften verhalten sich äquivialent zu den Einstellungen „3.3 Basiseinstellungen für die Anbindung an MOA-SS“ aus der PDF-AS Dokumentation.
soap.sign.enabled	*true* | *false*	Aktiviert oder Deaktiviert die SOAP-Schnittstelle. Standardmäßig ist die SOAP-Schnittstelle deaktiviert.
Soap.verify.enabled	*true* | *false*	Aktiviert oder Deaktiviert die SOAP-Schnittstelle für Verifikation. Standardmäßig ist die SOAP-Schnittstelle deaktiviert.
whitelist.enabled	*true* | *false*	Legt fest ob PDF-AS externe URLS mit der Whitelist vergleicht.
whitelist.url.*	*JavaRegularExpression*	Eine Liste von Regular Expressions die eine Whitelist für externe URLS in PDF-AS festlegen.
request.store	*at.gv.egiz.pdfas.web.store .InMemoryRequestStore* | *at.gv.egiz.pdfas.web.store .DBRequestStore*	Legt die Implementierung des Requeststores fest. Nur wenn PDF-AS im Cluster verwendet wird und per SOAP-Schnittstelle eine clientbasierte Signatur erstellt werden soll, sollte dieser Parameter vom InMemoryRequestStore auf den DBRequestStore gestellt werden. Wenn hier der DBReqeustStore verwendet wird, speichert PDF-AS Web Signaturrequests, die per SOAP-Schnittstelle kommen und vom Benutzer bearbeitet werden müssen in eine Datenbank. In diesem Fall müssen die Eigenschaften hibernate.props.* entsprechenden festgelegt werden.

request.db.timeout	*Number*	Gibt die Anzahl der Sekunden an, die ein Signaturrequest in der Datenbank gespeichert wird.
hibernate.props.*	*Sammlung von Eigenschaften*	Alle Eigenschaften die unter diesem Prefix eingetragen sind dienen der Konfiguration von Hibernate für die Datenbankverbindung des DBRequestStore. Wenn dieser nicht konfiguriert ist, benötigt PDF-AS Web keine Datenbank. Ein komplettes Beispiel für diese Parameter ist in der Beispielkonfiguration in diesem Dokument zu finden.
reload.pwd	*Text*	Passwort um die Konfiguration neu zu laden.
reload.enabled	*true* | *false*	Aktiviert bzw. Deaktiviert das Servlet um die Konfiguration neu zu laden.
allow.ext.overwrite	*true* | *false*	Ist dieser Paramter auf true gesetzt, erlaubt PDF-AS-WEB externen Anwendungen Konfigurationseinträge zu überschreiben. **Achtung**: Es können nur Einträge der PDF-AS Konfiguration (cfg/config.properties), nicht jedoch aus der PDF-AS-WEB Konfiguration (pdf-as-web.properties) überschrieben werden.
ext.overwrite.wl.*	*JavaRegularExpression*	Eine Liste von Regular Expressions die eine Whitelist für Konfigurationseinträge die von externen Anwendungen überschrieben werden können festlegen.
qr.placeholder.generator.enabled	*true* | *false*	Ist dieser Paramter auf true gesetzt, kann das Platzhalter Servlet verwendet werden.
sl20.sign.enabled	*true* | *false*	Ist dieser Paramter auf true gesetzt, kann die Security-Layer 2.0 Schnittstelle verwendet werden.
sl20.mobile.url	*URL*	URL zum SL 2.0 Endpunkt der Handysignatur.
sl20.keystore.file	*Dateipfad*	Keystore mit Schlüsselmaterial für SL 2.0.
sl20.keystore.type	*JKS* | *PKCS12*	Type des Keystores.
sl20.keystore.pass	*Passwort*	Das Passwort für den Keystore.
sl20.keystore.sign.key.alias		Der Bezeichner des Signature-Schlüssels im KeyStore mit dem die Signatur durchgeführt werden soll.
sl20.keystore.sign.key.pass	*Passwort*	Das Passwort für den Signature Schlüssel in der KeyStore Datei.
sl20.keystore.enc.key.alias		Der Bezeichner des Verschlüsselung - Schlüssels im KeyStore mit dem die Signatur durchgeführt werden soll.
sl20.keystore.enc.key.pass	*Passwort*	Das Passwort für den Verschlüsselung - Schlüssel im KeyStore.
sl20.debug.validation.disable	*true* | *false*	Deaktivierung der Datenvalidierung. **Default: false**

sl20.debug.signed.result.enabled	*true* | *false*	Deaktivierung von signierten Kommandos. **Default:** false
sl20.debug.signed.result.required	*true* | *false*	Deaktivierung von verpflichtend signierten Kommandos. **Default:** false
sl20.debug.encryption.enabled	*true* | *false*	Deaktivierung von Verschlüsselung, wodurch signierte Kommandos Schlüsselmaterial zur Verschlüsselung beinhalten. **Default:**false
sl20.debug.encryption.required	*true* | *false*	Deaktivierung von verpflichtender Verschlüsselung. **Default:**false

Die Konfiguration kann zur Laufzeit neu geladen werden. Dies betrifft die Webkonfiguration und die Basiskonfiguration der PDF-AS-Bibliothek. Um die Konfiguration neu zu laden muss das Servlet /Reload mit der GET-Methode aufgerufen werden. Diesem Servlet muss der GET-Parameter „PASSWD“ übergeben werden. Der Wert dieses Parameters muss dem konfigurierten Wert in „reload.pwd“ entsprechen.

Wenn als Requeststore der Datenbankspeicher *DBRequestStore* verwendet werden soll, muss die Bibliothek pdf-as-web-db zum Klassenpfad hinzugefügt werden.

1.6 Statistik

PDF-AS-WEB kann statistische Informationen sammeln. Um eine möglichst gut Integration zu ermöglichen, können einfach selbstentwickelte Statistikmodule in PDF-AS-WEB integriert werden. Dazu muss die Schnittstelle at.gv.egiz.pdfas.web.stats. StatisticBackend implementiert werden. PDF-AS-WEB lädt Implementierungen dieser Schnittstelle mit Hilfe des Java Service Provider Interface. Eine einfache Implementierung dieser Schnittstelle ist bereits in PDF-AS-WEB integriert und liefert auf dem Logger „at.gv.egiz.pdfas.web.statistics“ statistische Informationen im CSV Format. Die Spalten der CSV Einträge sind wie folgt zu interpretieren:

- Zeitstempel
- Operation („sign“ | „verify“)
- Signarturgerät
- Quelle („web“ | „soap“)
- Signturprofil

- Dateigröße
- User-Agent des Client
- Status („ok“ | „error“)
- Fehlermeldung
- Fehlercode
- Ausführungszeit

1.7 Kommunikation

PDF-AS Web hat eingehende Verbindungen nur über den Anwendungsserver.

Ausgehende Verbindungen von PDF-AS Web sind Aufrufe wenn PDF-AS Web ein zu signierendes PDF-Dokument an einer URL abholt. Um Beispielsweise Portscans des internen Netzwerks zu verhindern, können diese URLs mit Hilfe einer Whitelist gefiltert werden.

1.8 QR Platzhalter Generator

PDF-AS Web bietet ein Platzhalter Servlet an. Mit diesem können QR Platzhalter für PDF-AS erzeugt werden. Das Platzhalter Servlet muss explizit aktiviert werden. Dies geschieht durch den Konfigurationseintrag „qr.placeholder.generator.enabled“.

Ist das Platzhalter Servlet aktiv, kann unter der URL /placeholder ein QR Platzhalter erzeugt werden. Die erzeugten Platzhalter haben eine Größe von 250 mal 98 Pixel. Es werden die Http-Methoden GET und POST unterstützt. Die folgenden Parameter können an das Servlet übergeben werden:

- „**id**“: Dieser Parameter gibt an welcher Identifier in den Platzhalter integriert werden soll. Es wird jeglicher Text aktzeptiert, wobei alle Symbole außer a-z, A-Z und 0-9 durch „_“ersetzt werden. (**optionaler Parameter**)
- „**profile**“: Dieser Parameter gibt an welches Signaturprofile in den Platzhalter kodiert werden soll. Endet der Profilname mit „_EN“ wird davon ausgegangen, dass es sich um ein englisches Profil handelt. (**optionaler Parameter**)

2 Beispiele

PDF-AS Web Konfigurationsdatei Beispiel:

```
# PDF-AS Basis Konfiguration
pdfas.dir=/config/.pdfas

# Oeffentliche URL
public.url=https://pdfas.egiz.gv.at/pdfas

# Show error details in PDF-AS
error.showdetails=false

# MOA-SS Signatur
moa.enabled=false

# Lokale BKU URL
bku.local.url=http://127.0.0.1:3495/http-security-layer-request

# Online BKU URL
bku.online.url=http://192.168.1.1/bkuonline/http-security-layer-request

# Handy Signatur URL
#bku.mobile.url=

# Platzhalter Generator aktiv
qr.placeholder.generator.enabled=true

ks.enabled=true
ks.file=/config/test.p12
ks.type=PKCS12
ks.pass=123abc
ks.key.alias=test
ks.key.pass=123abc

soap.sign.enabled=true
soap.verify.enabled=true

reload.pwd=123456
reload.enabled=true

# Whitelist
whitelist.enabled=true
# Matches everything
whitelist.url.01=^.*$

#whitelist.url.02=^http://.*$

#Request Store
# Default In Memory Store
#request.store=at.gv.egiz.pdfas.web.store.InMemoryRequestStore
# DB Request Store for cluster
#request.store=at.gv.egiz.pdfas.web.store.DBRequestStore

# seconds Requests are kept in Database (default 600)
#request.db.timeout=600
```

```
#Hibernate Properties for DB Store

#hibernate.props.hibernate.dialect=org.hibernate.dialect.MySQLDialect
#hibernate.props.hibernate.connection.driver_class=com.mysql.jdbc.Driver
#hibernate.props.hibernate.connection.url=jdbc:mysql://localhost/pdfasweb
#hibernate.props.hibernate.connection.username=pdfasweb
#hibernate.props.hibernate.connection.password=pdfasweb
#hibernate.props.hibernate.connection.pool_size=5
#hibernate.props.hibernate.connection.autocommit=false
#hibernate.props.hibernate.show_sql=true
#hibernate.props.hibernate.hbm2ddl.auto=update

# Handy Signature mit SL 2.0

sl20.sign.enabled=true
sl20.mobile.url=https://test1.a-trust.at/securitylayer2
sl20.keystore.file=
sl20.keystore.type=JKS
sl20.keystore.pass=****
sl20.keystore.sign.key.alias=sl20signing
sl20.keystore.sign.key.pass=****
sl20.keystore.enc.key.alias=sl20encryption
sl20.keystore.enc.key.pass=****
sl20.debug.validation.disable=false
sl20.debug.signed.result.enabled=false
sl20.debug.signed.result.required=false
sl20.debug.encryption.enabled=false
sl20.debug.encryption.required=false
```

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	15.07.2014	Andreas Fitzek	SOAP-Schnittstelle, Clusterbetrieb
0.4	18.08.2014	Andreas Fitzek	Konfigurationsparamter update
0.5	10.10.2014	Andreas Fitzek	pdf-as-web-db
0.6	26.03.2015	Andreas Fitzek	Key identifier
0.7	12.02.2016	Andreas Fitzek	Platzhhalter Servlet
0.8	03.12.2018	Emina Ahmetovic	Security Layer 2.0
0.9	02.03.2023	Thomas Lenz	Bulk-Light Funktion hinzugefügt

Referenzen