RUHR-UNIVERSITÄT BOCHUM

RUB

Anwenderdokumentation

Installationsbeschreibung
&
Schnittstellendefinition

JPDFSigner Version 1.2 RC

PDF-Signatur in Webanwendungen per Smartcard

Stand 12.07.2013

Erstellt von:

Daniel Moczarski
Dezernat 6
Abteilung 3 – IT-Systeme & Softwareintegration

Ruhr-Universität Bochum
Universitätsstraße 150
44801 Bochum

Inhaltsverzeichnis

1 ZIELGRUPPE

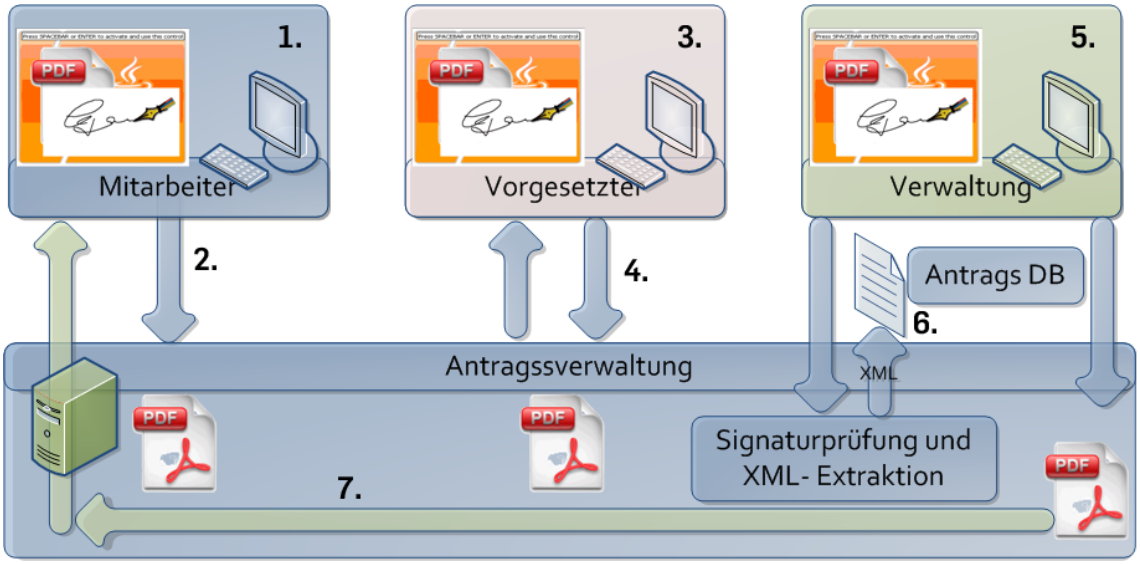
Diese Dokumentation richtet sich an den Anwender und Administrator der Software „JPDFSigner“.
Beschrieben werden sowohl Funktionalitäten der Software, als auch Schnittstellen und die Einbindung in bestehende Webapplikationen.

2 FUNKTIONSÜBERSICHT

Das „JPDFSigner“-Java Applet ist ein Werkzeug für Bedienstete und Studierende der Ruhr-Universität Bochum. Es ermöglicht die Signatur einzelner oder mehrerer PDF-Dokumente gleichzeitig anhand der RUBCard (Smartcard), die als Multifunktionsausweis an der RUB zur Verfügung gestellt wird. Da es sich um ein Java-Applet handelt, muss die Software von bereits existierenden Webapplikationen eingebunden werden. Diese wiederum starten JPDFSigner mit Parametern. Anhand der Parameter werden anschließend die PDF-Dokumente geladen und angezeigt. Nach erfolgreich durchgeführter Signatur der Dokumente, werden diese auf einen Zielserver geladen.

2.1 MÖGLICHER USE-CASE: ANTRAGSVERWALTUNG

Folgender typischer abstrakter Workflow soll das Einsatzszenario des JPDFSigners deutlich machen. Die PDF trägt dabei neben der optischen Repräsentation des Dokuments zusätzliche Nutzdaten in XML-Form, die zur elektronischen Weiterverarbeitung benötigt werden.



1. Mitarbeiter signiert Antrag mit personalisierter Smartcard.
2. Signierter Antrag wird in der Datenbank der Antragsverwaltung gespeichert oder dem Workflow hinzugefügt.
3. Vorgesetzter wird über den Antrag des Mitarbeiters informiert (z.B. automatisiert per Mail oder durch Workflowengine), prüft und signiert diesen wiederum mit seiner personalisierten Smartcard.
4. Zweifach signiertes PDF-Dokument wird in der Antragsverwaltung gespeichert bzw. dem Workflow zurückgeführt.
5. Verwaltungsmitarbeiter erhält Benachrichtigung über neuen Antrag. Dieser wird von einem Verwaltungsangestellten bearbeitet, geprüft und signiert.
6. XML-Nutzdaten werden aus der PDF extrahiert um weitere automatisierte Prozesse zu starten (z.B. Aktualisierung des Urlaubskontos oder initiieren von Verrechnungsprozessen).
7. Antrag mit allen Signaturen wird zurück an den Antragsteller geschickt.

3 SYSTEMVORRAUSSETZUNG

3.1 BETRIEBSSYSTEM

- Linux (getestet mit Ubuntu 9.10 32Bit)
- Windows XP (getestet mit SP3 32Bit)

3.2 SOFTWARE

- OpenSC Smartcard-Bundle
- OpenSC-Java (Version 0.2.2)
- Oracle min. JRE 6

3.3 HARDWARE

- Einen angeschlossenen und installierten Smartcard-Leser

3.4 NETZWERK

- Netzwerkzugriff auf die Down-, Uploadserver für PDF-Dokumente

4 SPEZIFIKATIONEN

Das Applet ist so konzipiert, dass die Workflowsteuerung alleine durch den Browser stattfindet. Dazu werden dem Applet sämtliche Informationen per Parameter übergeben.
In diesem Abschnitt wird beschrieben, über welche Schnittstellen und Parameter das Applet verfügt und wie diese genutzt werden können.

4.1 HARDWAREZUGRIFF (APPLET-SIGNATUR)

Um dem Java-Applet Hardwarezugriff zu ermöglichen (z.B. Smartcard-Leser, Festplattenzugriff oder das öffnen eigener Fenster unter Java innerhalb der Browser-Sandbox), muss dieses mit einem Zertifikat signiert sein. Die auf Github befindlichen Quellen beinhalten eine Maven – pom.xml, die dahin gehend konfiguriert ist, dass die FatJar *(jpdfsigner-1.2.0-Release-jar-with-dependencies.jar)* während des Maven-Kompilierprozesses mit einem Testzertifikat signiert wird. Sollten die Quellen mit einer veränderten Version dieser pom.xml oder gänzlich anders kompiliert werden, kann es sein dass die Jar nicht signiert wird und dadurch nicht im Internetbrowser startet!

4.2 PARAMETER

Der Parameter ***sourceContent*** ermöglicht die Übergabe eines oder mehrerer PDF-Dokumente als base64-kodierten String. Für jede PDF werden drei Informationen benötigt (Dreier-Touple):

- PDF bytes
- Dateiname
- pdfsignspec.xml (Signaturinformationen die für diese PDF gelten)

Alle Parameter müssen Base64-kodiert werden und werden durch ein „-" getrennt. Die PDF-dreier-Touple werden durch „--" getrennt.

Parameter	sourceContent
Modus	optional
Beispiel	Übergabe zweier PDF-Dokumente an das Applet <param name="sourceContent" value="base64pdf1bytes-base64Dateiname1-base64pdfsignspec1—base64pdf2bytes-base64Dateiname2-base64pdfsignspec2" />

Der Signaturgrund (Ist Teil der Signatur) kann über den Parameter ***signatureReason*** gesetzt werden. Dieser gilt für alle Dokumente die im Applet geladen wurden und dadurch signiert werden. Eigentlich sollte dieser Parameter Teil der SignSpecXML sein. Den Anforderungen nach war es jedoch notwendig, dass dieser Parameter für alle PDFs gilt.
Wird dieser Parameter nicht spezifiziert ist der Initialwert „" (Leer).

Parameter	signatureReason
Modus	Optional
Beispiel	Übergabe zweier PDF-Dokumente an das Applet <param name="signatureReason" value="Signatur eines Testdokumentes" />

Über den Parameter ***postDestination*** wird der Zielserver spezifiziert, an den das POST-Paket mit den signierten PDF-Dokumenten geschickt wird. Das Ziel ist üblicherweise ein Script, das den POST-Request entgegen nimmt, die signierten PDFs ausliest und weiter verarbeitet. Dieses Script muss einer gewissen Konformität zu Grunde liegen!
Näheres dazu unter Punkt *4.3 – Uploadscript*.

Parameter	postDestination
Modus	erforderlich
Beispiel	<param name="postDestination" value="http://localhost/sendpdf.jsp" />

Über den Parameter ***postDestination*** wird das Script definiert, an welches die signierten PDFs per POST gesendet werden. Diesem POST-request kann über den ***sessionID***-Parameter ein Cookie eingesetzt werden. Dieses Cookie ist dann Teil des Requests, der an das postDestination-Script (Uploadscript) gesendet wird. Das ist dann hilfreich, wenn im Laufe der Webanwendung Daten in Sessionvariablen gespeichert werden, die das Uploadscript zur Weiterverarbeitung der signierten PDFs benötigt.

Wichtig ist, dass dieser Wert folgender Konvention entspricht: key=value. Werden key und value nicht durch ein Leerzeichen getrennt, wird dieser Parameter ignoriert.

Parameter	sessionID
Modus	optional
Beispiel	<param name="sessionID" value="SessionID=54987ce654e111c32e1f " />

Nach dem Upload der Daten wird die in dem Parameter ***resultUrl*** spezifizierte URL an den Browser gesendet und geöffnet. Sollte dieser Parameter nicht übergeben werden, findet keine Weiterleitung im Browser statt. Die Weiterleitung zu dieser URL hat zur Folge, dass das Applet geschlossen wird.

Parameter	resultUrl
Modus	optional
Beispiel	<param name="resultUrl" value="http://localhost/result.jsp " />

Mit dem Parameter ***resultTarget*** legen Sie das Zielfenster fest, in dem die ***resultUrl*** geöffnet werden soll. Der Initialwert dieses Parameters ist *_self.* Wird die result-Url in einem neuen Fenster geöffnet (_blank) wird das Applet nicht beendet. Wird die result-Url im selben Fenster geöffnet (_self) wird das Applet beendet.

Parameter	resultTarget
Modus	optional
Beispiel	<param name="resultTarget" value="_blank " />

Das *JPDFSigner*-Applet wurde mit dem Logging-Framework *Log4J* ausgestattet. Standardmäßig werden die Logs in die Java-Konsole geschrieben. Über den Parameter ***logConfig*** kann eine base64-kodierte Log4J-Konfigurationsdatei (*.properties) mitgegeben werden, in der das Loggingverhalten seitens des Clients näher spezifiziert werden kann.

Weitere Informationen dazu unter *http://logging.apache.org/log4j/1.2/index.html*

Parameter	logConfig
Modus	Optional
Beispiel	<param name="logConfig" value="Base64-Kodierte Log4J-Konfigurationsdatei" />

Standardmäßig startet das Applet in einem externen Fenster. Dies kann unterbunden werden, indem der Parameter ***embedded*** auf *true* gesetzt wird. In diesem Fall startet das Applet nicht im externen Fenster sondern eingebettet im Browser. Dabei sollte darauf geachtet werden, dass das Applet eine angemessene Größe besitzt (width & height-Paramter des HTML-Applet bzw. –Object-Tags).

Parameter	embedded
Modus	Optional
Beispiel	<param name="embedded" value="true" />

Das Applet besitzt zwei Modi. Zum einen den *Signature_Mode(mode=0)* und zum Anderen den *Attach_User_Data_To_PDF_Mode (mode=1).* ***mode*** spezifiziert den Modus in dem das Applet gestartet wird. Standardmäßig startet das Applet im Signatur-Modus (mode = 0).

Der *Attach_User_Data_To_PDF_Mode* ist dazu gedacht XML-Nutzdaten zu erfassen und diese in die PDFs einzubetten. Diese Nutzdaten werden benötigt um die PDF-Dokumente automatisiert weiterverarbeiten zu können. Der gegenwärtige Use-Case an der RUB ist, dass gedruckte bzw. auf Papier geschriebene Dokumente eingescannt und als PDF gespeichert werden. Im nächsten Schritt werden diese PDFs in das Applet geladen und durch die XML-Nutzdaten erweitert (*Attach_User_Data_To_PDF_Mode*). Dann folgt die Signatur der PDFs und die automatische Weiterverarbeitung anhand des *Uploadscripts*.

Da die Implementierung zur Erfassung der Nutzdaten sehr eng an den Use-Case der Ruhr-Universität Bochum gebunden ist und weder ausreichend abstrakt noch programmtechnisch sehr aufschlussreich ist, wird auf die Dokumentation dieses Features komplett verzichtet. Das bedeutet gleichzeitig, dass dieser Parameter irrelevant ist und nur der Vollständigkeit halber erwähnt wird.

Parameter	mode
Modus	optional
Beispiel	<param name="mode" value="1" />

4.2.1 PDF-SIGNATURSPEZIFIKATIONEN

Um den Ablauf des Signaturprozesses genauer zu definieren, können Spezifizierungen an das Applet übergeben werden. Die Spezifikation kann jeder PDF beigefügt werden. Liegt keine Spezifikation für die entsprechende PDF vor, erfolgt eine nicht-sichtbare Signatur (kein visuellerSignaturstempel). Wie Spezifikationen hinzugefügt werden können kann der Parameterbeschreibung 4.2 – **sourceContent** entnommen werden. Signaturspezifikationen müssen in Form eines XML-Dokumentes vorliegen. Der Rumpf der XML muss wie folgt strukturiert sein:

```
<jpdfsigner_pdf_specs>
    <spec1></spec1>
    <spec2></spec2>
    .
    .
    .
</jpdfsigner_pdf_specs>
```

Abbildung 4.3-1

Beachten Sie bitte, dass das Dokument case-sensitive ist.
Innerhalb des Root-Tags „<jpdfsigner_pdf_specs>“ findet die Implementierung der Spezifikationen statt. Für jede vom Applet unterstützte Spezifizierung muss ein entsprechendes Kind-Element angelegt werden. Die Abbildung 4.3-1 besitzt beispielsweise zwei Spezifizierungen *(spec1 und spec2)*. Welche Spezifizierungen in der momentanen Version des Applets unterstützt werden, ist den Punkten 4.3.1 – 4.3.X zu entnehmen.

Sollte diese XML nicht wohl geformt sein, wird keine der Spezifikationen ausgeführt. Außerdem werden Spezifikationen, die nicht vom Applet unterstützt werden, sowie eine falsch implementierte/strukturierte XML, ignoriert *(korrekte Strukturierung - siehe Abbildung 4.3-1)*. Auch falsch implementierte Spezifikationen werden ignoriert. Alle Weiteren, korrekt implementierten, werden berücksichtigt.

Folgende Spezifikationen werden unterstützt:

4.2.2 PLATZIERUNG DES SIGNATURSTEMPELS

Kind-Elementenname: <stampPosition>

Definition
Diese Spezifizierung ermöglicht dem Applet, den visuellen Signaturstempel automatisch an einer vorgegebenen Position zu platzieren. Alternativ kann der Signaturstempel auch komplett entfernt werden.

Parameter

Mit dem Attribut ***visibleSignature*** kann die Sichtbarkeit des Signaturstempels definiert werden. Ist der Wert *true* wird der Stempel an den angegebenen Koordinaten platziert. Sollte der Wert *false* sein, wird auf dem Dokument kein Stempel aufgetragen und die angegebenen Koordinaten werden ignoriert.

Attribut	visibleSignature
Eltern-Element	<stampPosition>
Modus	erforderlich
Beispiel	<stampPosition visibleSignature=“false“>

Mit dem Element ***<x>*** wird die x-Koordinate des Punktes festgelegt, der die obere linke Ecke des Stempels auf der PDF bildet. Die Angabe erfolgt in Millimetern.

Element	<x>
Eltern-Element	<stampPosition>
Modus	erforderlich
Beispiel	<x>150</x>

Mit dem Element ***<y>*** wird die y-Koordinate des Punktes festgelegt, der die obere linke Ecke des Stempels

auf der PDF bildet. Die Angabe erfolgt in Millimetern.

Element	<y>
Eltern-Element	<stampPosition>
Modus	erforderlich
Beispiel	<y>250</y>

Mit dem Element ***<width>*** wird die Breite des Stempels spezifiziert. Die Angabe erfolgt in Millimetern.

Element	<width>
Eltern-Element	<stampPosition>
Modus	erforderlich
Beispiel	<width>50</width >

Mit dem Element ***<height>*** wird die Höhe des Stempels spezifiziert. Die Angabe erfolgt in Millimetern.

Element	<height>
Eltern-Element	<stampPosition>
Modus	erforderlich
Beispiel	<height>50</height>

Mit dem Element ***<page>*** wird die Seite definiert, auf welcher der Stempel platziert werden soll. Wird eine Seite angegeben die nicht existiert, wird der Stempel auf der nächst Möglichen platziert.
Beispiel:
Die Platzierung soll auf Seite 21 stattfinden. Das Dokument hat allerdings nur 19 Seiten. Somit erfolgt die Platzierung auf Seite 19.
Außerdem kann der Wert *last* übergeben werden. Somit wird der Stempel auf der letzten Seite des PDF-Dokumentes platziert.

Element	<page>
Eltern-Element	<stampPosition>
Modus	erforderlich
Beispiel	<page>2</page>

Beispiel

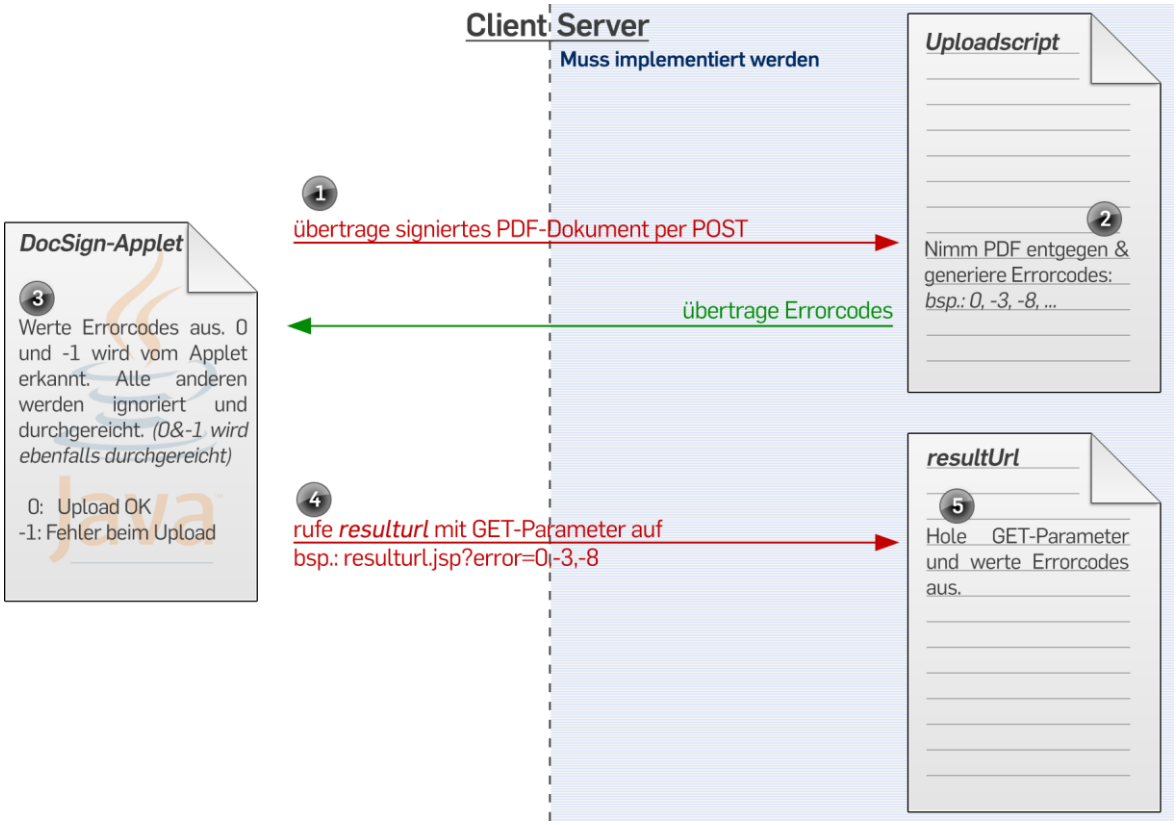
```
<!--
Im folgenden Beispiel wird ein sichtbarer Signaturstempel erzeugt. Dieser wird automatisch auf der letzten
Seite des PDF-Dokumentes  an der Position 150x250 mit dem Ausmaß 50x30 Millimeter platziert.
-->

<stampPosition visibleSignature="true">
    <x>150</x>
    <y>250</y>
    <width>50</width>
    <height>30</height>
    <page>last</page>
</stampPosition>
```

4.3 UPLOADSCRIPT

Das Uploadscript stellt dem *JPDFSigner*-Applet die Upload-Schnittstelle für die signierten PDF-Dokumente zur Verfügung. Diese werden in Form einer HTTP-POST-Anfrage an das Script gesendet. Wie anschließend mit den hochgeladenen Dokumenten weiter verfahren wird, hängt ganz von dem Betreiber des Dienstes und damit der Implementierung dieses Scriptes ab. Auch die Fehlerbehandlung ist dem Betreiber freigestellt. Somit können eigene Fehlercodes generiert und behandelt werden.
Beispiel: Eine im Uploadscript selbst implementierte „PDF-Validierungsfunktion" schlägt fehl. Diese Nachricht möchten Sie nun auf der Ergebnisseite ihrer Webanwendung anzeigen lassen. *(Ergebnisseite: Siehe Punkt 4.2 Parameter-resultUrl)*

Im Folgenden sehen sie eine Grafik, die diesen Workflow und den Informationsfluss skizziert:



1 Das Applet überträgt das signierte PDF-Dokument an das Uploadscript (das Uploadscript wird über den Parameter **postDestination** gesetzt. *Siehe Punkt 4.2.2 Antwort-Parameter*).

2 Das Uploadscript prüft ob das Dokument korrekt angekommen ist. Für diesen Fall wird der Fehlercode „0", andernfalls „-1" generieren. Einer dieser Fehlercodes sollte zurückgegeben werden, damit das Applet entsprechend reagieren kann *(Upload erfolgreich oder nicht erfolgreich)*. Andernfalls wird das Applet ohne eine Ergebnismeldung beendet. Nun können beliebig viele weitere Fehlercodes generiert werden. **Achtung: In der Releaseversion des Applets werden die Fehlercodes -9000 – -9999 reserviert sein. Das bedeutet, wenn das Applet unerwartet beendet werden muss, wird ein Fehlercode zwischen -9000 und -9999 generiert. Anschließend folgt ein Redirect auf die resultUrl mit dem entsprechenden Fehlercode. Somit können Appletfehler auf der resultUrl ausgewertet werden.** Bitte beachten Sie bei der Rückgabe der Fehlercodes auf folgende Konvention:

2.1 Die einzelnen Fehlercodes müssen kommagetrennt und als String zurückgegeben werden: 0,-3,-8...

2.2 Der Fehlercode-String muss direkt auf die HTML-Seite geschrieben werden, die das Script generiert – ohne HTML-Tags oder ähnlichem.

__Info:__ Das Applet funktioniert beim Upload der Daten wie ein herkömmlicher Internet-Browser. Es erhält also auch einen Response vom Server. Der Response beinhaltet den Inhalt der HTML-Seite, die aufgerufen wurde. In unserem Fall ist es das Uploadscript. Nach diesem Prinzip werden die Fehlercodes übertragen.

3 Das Applet erhält die kommagetrennten Fehlercodes und separiert diese. Wird der Errorcode „0“ ermittelt, erscheint eine Meldung, dass der Upload erfolgreich durchgeführt wurde. Wird der Errorcode „-1“ ermittelt erscheint eine Meldung, die auf einen Fehler beim Upload hinweist.

4 Anschließend wird die Ergebnisseite (**resultUrl** – *Siehe Punkt 4.2.2 Antwortparameter*) mit dem generierten Errorcode-String als GET-Parameter aufgerufen. Den Aufruf tätigt das Applet automatisch. (Bsp.: resulturl.jsp?error=0,-3,-8)

5 Die Ergebnisseite muss nun die über die GET-Methode übergebenen Fehlercodes, auswerten.

5 FUNKTIONSÜBERSICHT

„JPDFSigner“ wird über bestehende Webapplikationen aufgerufen und clientseitig ausgeführt. Sobald dies geschieht öffnet sich ein Fenster, welches die Bestätigung der vorgelegten Applet-Signatur erfordert (Das Applet enthält eine digitale Signatur). Wird diesem vertraut, startet die Software.

Sollten Fehler auftreten, werden diese in Form eines Dialoges mit einer entsprechenden Fehlerbeschreibung angezeigt. Eine Zusätzliche Fehlerprotokollierung ist von der übergebenen Log4J-Konfigurationsdatei abhängig. Weitere Informationen hierzu, können der Parameterbeschreibung ***logconfig*** unter Punkt *4.2* entnommen werden.

5.1 INSTALLATION

Damit „JPDFSigner“ korrekt funktioniert, müssen zunächst folgende Software-Pakete auf dem Client-PC installiert werden:

- **Smart Card Bundle von OpenSC:**
  - Windows: http://www.opensc-project.org/files/scb/scb-0.XX.exe (Vollinstallation)
  - Linux: Einige Linux-Softwarequellen stellen das Smart Card Bundle bereits zur Verfügung. Verwenden sie bitte vorzugsweise dieses. (Paketname unter Ubuntu: opensc)
    - **Einschränkungen unter Ubuntu 9.10 Karmic Koala**
      Die OpenSC-Pakete sind seit der Ubuntu Version 9.10 fehlerhaft, sodass diese nicht mehr genutzt werden können (OpenSC Version 0.11.8). Alternativ können die OpenSC-Pakete der Ubuntu Version 9.04 „Jaunty Jackalope“ verwendet werden (OpenSC Version 0.11.4) oder die 0.12er Version.
- **Java-Browser-Plugin**
  - Ausschließlich das von Oracle bereitgestellte Browser-Plugin der Java-Version 1.6 und aktueller.

Zusätzlich muss ein Smartcard-Leser angeschlossen und installiert sein. Dieser muss auf die RUBCards zugreifen können (z.B. der „SCR3310“ Smartcard-Leser).

Sobald die aufgeführten Komponenten korrekt installiert sind, sollte „JPDFSigner“ im vollen Umfang einsetzbar sein.

5.2 GUI

In der Anlage I befinden sich die Abbildungen 1-4 in denen die im Folgenden aufgelisteten GUI-Elemente eingezeichnet sind:

Anlage 1 - Abbildung 1:

#1. Expandieren/Kollabieren der Dokumentenansicht
#2. Vorherige PDF öffnen
#3. Nächste PDF öffnen
#4. PDF hinzufügen
#5. Markierte PDFs entfernen
#6. Zur vorherigen Seite blättern
#7. Anzeigen des PDF Viewers
#8. Anzeige der momentanen Seite & Seitenanzahl der geöffneten PDF
#9. Zur nächsten Seite blättern
#10. Anzeigen des XML Viewers
#11. Ansicht um 10% vergrößern
#12. Anzeige der momentanen Ansichtsgröße
#13. Ansicht um 10% verkleinern
#14. Ansicht auf die Seite automatisch anpassen
#15. Öffnen der Konfiguration
#16. Sprachauswahl
#17. Anzeigen der PDF-Attachments
#18. Anzeigen der Versions- & Lizenzinformationen
#19. Anzeigen der PDF-Signaturen (sofern vorhanden)
#20. PDF Viewer
#21. Signaturbutton (initiiert den Signatur- & Uploadvorgang)

Anlage 1 - Abbildung 2:

#22. Dokumentenansicht
#23. Geladenes Dokument
#24. Marker: markieren und „Markierte PDFs entfernen“ klicken um PDF aus der Liste zu entfernen
#25. Konfiguration
#26. Schließen der Konfiguration. Roter Button bedeutet, dass die Konfiguration geändert wurde und beim klicken gespeichert wird.
#27. Verwendeter Kartenleser
#28. Kartenleser-Auswahl. Das gelbe Icon indiziert, dass in diesem Lesegerät eine Karte steckt.
#29. Verwendetet Kryptobibliothek
#30. Anzeige der enthaltenen PDF-Attachments
#31. PDF-Attachment
#37. Dokumentenansicht-Slider (vergrößern/verkleinern der Dokumentenansicht)
#38. PDF-Dateiname

Anlage 1 – Abbildung 3:

#32. XML Viewer
#33. Pfad zur verwendeten Kryptobibliothek

Anlage 1 - Abbildung 4:

#34. PIN Pad zur Eingabe der Smartcard PIN
#35. Leer das PIN-Eingabe-Feld
#36. Startet den Signier- & Uploadprozess

5.2.1 KONFIGURATION

Nachdem alle Schritte aus Punkt 5.1 durchgeführt worden sind, muss das Applet beim ersten Start konfiguriert werden. Wenn das Smartcard-Bundle von OpenSC installiert und die opensc-pkcs11.so/dll im Pfad C:\windows\system32\opensc-pkcs11.dll (Windows) bzw. /usr/lib/opensc-pkcs11.so (Linux) gespeichert wurde, erfolgt die Konfiguration des Applets automatisch. Außerdem wird der zuerst gefundene Smartcard-Leser als Standardleser gespeichert, sofern dieser von der Kryptobibliothek unterstützt wird und korrekt installiert ist .

Anschließend wird im Benutzerverzeichnis unter jpdfsigner/jpdfsigner.cfg eine Konfigurationsdatei gespeichert, die alle relevanten Informationen enthält. Wird diese gelöscht, wird die automatische Konfiguration beim nächsten Start des Applets erneut durchgeführt.

Wurde die Konfiguration nicht automatisch durchgeführt, erhält der Benutzer den Hinweis, dass entweder Kryptobibliothek oder Smartcard-Leser nicht automatisch ermittelt werden konnte. Hierfür muss entweder die Kryptobibliothek und/oder der Smartcard-Leser manuell ausgewählt werden. Sollte das Konfigurationsfenster *(Anlage I, Abbildung 2 #25)* noch nicht geöffnet sein kann dies mit dem Button „Öffnen der Konfiguration“ *(Anlage I, Abbildung 1 #15)* geschehen. Nun kann über die Registerkarten „Kartenleser“ und „Bibliotheken“ *(Anlage I, Abbildung 2 #27 & #29)* der Kartenleser bzw. Pfad zur Kryptobibliothek ausgewählt werden.
Sobald Änderungen an der Konfiguration vorgenommen wurden, leuchtet der Button zum Schließen der Konfiguration rot *(Anlage I, Abbildung 2 #26)*. Schließt man die Konfiguration durch einen Klick auf diesen Button oder das gesamte Applet durch einen Klick auf das „X“ in der oberen rechten Ecke, wird die geänderte Konfiguration gespeichert.

5.2.2 PDF-DOKUMENT BETRACHTEN

Für die Betrachtung des Dokuments stehen dem PDF Viewer die Werkzeuge „Rein-, Raus zoomen“ und „Vor- und Zurückblättern“ zur Verfügung *(Anlage I, Abbildung 1 #11-#13)*. Außerdem kann eine Seite mit der linken Maustaste „gepackt“ und bewegt werden *(Anlage I, Abbildung #20)*. Somit kann auf nicht sichtbare Bereiche (wenn der Zoom zu groß ist als das die gesamte Seite auf den Bildschirm passt) oder auf die nächste Seite navigiert werden. Die Größe der Seite kann über den Button „Ansicht der Seite automatisch anpassen“ automatisch angepasst werden *(siehe Anlage I, Abbildung 1 #14)*.

5.2.3 ATTACHEMENTS ANZEIGEN

Der Attachment-Button blendet die Übersicht, der in der PDF eingebetteten Datei-Attachments ein *(Anlage I, Abbildung 1 #17)*. Hier können ausschließlich XML-Dokumente zur Ansicht gebracht werden, in dem auf das entsprechende Attachment mit der linken Maustaste geklickt wird. Das XML-Dokument wird dann im XML Viewer angezeigt *(Anlage I, Abbildung 3 #32)*. Andere Dateitypen werden zwar in der Übersicht angezeigt jedoch beim anklicken ignoriert. Außerdem können Attachments weder extrahiert noch verändert werden.

Der Grund hierfür ist, dass in einem speziellen Use-Case der RUB einige PDFs automatisch generiert und die Ursprungsdaten in ihrer rohen Form als XML-Attachment an die PDFs angeheftet werden. Somit besteht die PDF zum Einen aus einer visuellen und zum Anderen aus einer technischen Komponente, aus der die visuelle hervor geht. Der Signierende soll die Möglichkeit haben beide Komponenten einzusehen und dann erst zu entscheiden, ob er das Dokument signieren möchte.

5.2.4 SIGNATUREN ANZEIGEN

Der Button zur Anzeige der in der PDF enthaltenen Signaturen *(Anlage I, Abbildung 1 #19)* blendet alle vorhanden Signaturen und deren Trust-Chain ein. Dabei befinden sich Signaturen mit einer höheren Revisionsstufe weiter oben in der Liste. Die Revisionsstufe gibt Informationen darüber, welche Signatur zuerst erstellt wurde. Neuere Signaturen (weiter oben in der Liste) schließen alle vorhergehenden Signaturen ein. Außerdem beinhaltet jede Revision, zusätzlich zur Signatur, die PDF in dem Revisionszustand, in dem sie war als sie signiert wurde. JPDFSigner kann jedoch nur die aktuellste Revision des Dokumentes anzeigen.

5.2.5 SPRACHE ÄNDERN

Der Button mit der Landesflagge *(Anlage I, Abbildung 1 #16)* sollte das Ändern der Anzeigesprache ermöglichen. Gegenwärtig ist allerdings nur Deutsch möglich. Außerdem wurde dieses Feature nicht komplett implementiert, sodass einige GUI-Elemente die Text enthalten, nicht von der Sprachänderung betroffen sind.

5.2.6 DOKUMENTENANSICHT

Mit dem JPDFSigner ist es möglich mehrere PDFs gleichzeitig zu laden, anzusehen und zu signieren. Die Dokumentenansicht *(Anlage I, Abbildung 2 #22)* zeigt alle geladenen PDFs und deren erste Seite als Thumbnail an *(Anlage I, Abbildung 2 #23)*. Die Ansicht kann über den Slider *(Anlage I, Abbildung 1 #37)* vergrößert und verkleinert werden. Dadurch werden die Thumbnails ebenfalls größer und detaillierter oder kleiner. Sind die Thumbnails groß genug, wird zusätzlich der Name der PDF eingeblendet *(Anlage I, Abbildung 1 #38)*. Über die Buttons aus Anlage I, Abbildung 1 #1-#5 kann die Dokumentenansicht ein-/ausgeblendet, das vorherige/nächste Dokumente geladen und Dokumente hinzugefügt oder entfernt werden. Um ein Dokument im PDF Viewer zu öffnen, muss dessen Thumbnail mit der linken Maus taste angeklickt werden. Sämtliche Informationen wie Signaturen oder Attachments gelten dann dieser PDF. Um ein Dokument komplett zu entfernen muss dieses zunächst in der Dokumentenansicht markiert werden *(Anlage I, Abbildung 2 #24)*. Dann muss der „Markierte Dokumente entfernen"-Button geklickt werden *(Anlage I, Abbildung 1 #5)*. Anschließend werden alle markierten Dokumente entfernt und eventuell im PDF Viewer geöffnete PDFs geschlossen.

5.2.7 PDF-DOKUMENT SIGNIEREN

Sollten die unter Punkt 5.1 beschriebenen Installationen und die unter Punkt 5.2.1 beschriebenen Konfigurationen durchgeführt worden sein, kann das angezeigte PDF-Dokument signiert werden. Dazu muss auf den roten „Sign"-Button geklickt werden, der sich auf der rechten Seite der Hauptansicht befindet *(Anlage I, Abbildung 1 #21)*. Anschließend wird die Kryptobibliothek geladen und ein virtuelles PIN-Pad eingeblendet *(Anlage I, Abbildung 4 #34)*. Die PIN kann über das virtuelle PIN-Pad oder die Tastatur eingegeben und durch Enter oder dem Häckchen-Button unten links auf dem virtuellen PIN-Pad bestätigt werden. Dadurch startet der Signatur- und Uploadprozess.

TIP: Das virtuelle PIN-Pad wurde entwickelt, um eine sicherere PIN-Eingabe in nicht vertrauenswürdigen Umgebungen zu ermöglichen (z.B. öffentliche Terminals der Ruhr-Universität Bochum). Dadurch wird die eventuelle Verwendung von Keyloggern o.ä. wirkungslos.

Die eingegebene PIN muss sechsstellig sein. Die Eingabe von nicht-numerischen Zeichen wird unterbunden. Wurde die PIN falsch eingegeben, wird eine Fehlermeldung mit einem entsprechenden Fehlercode angezeigt. Anschließend kann die Eingabe wiederholt werden. Sollte jedoch festgestellt werden, dass die Karte aufgrund mehrmaliger Fehleingabe der PIN gesperrt wurde, wird das Programm mit einer entsprechenden Fehlermeldung beendet. Die Karte muss dann zunächst entsperrt werden.

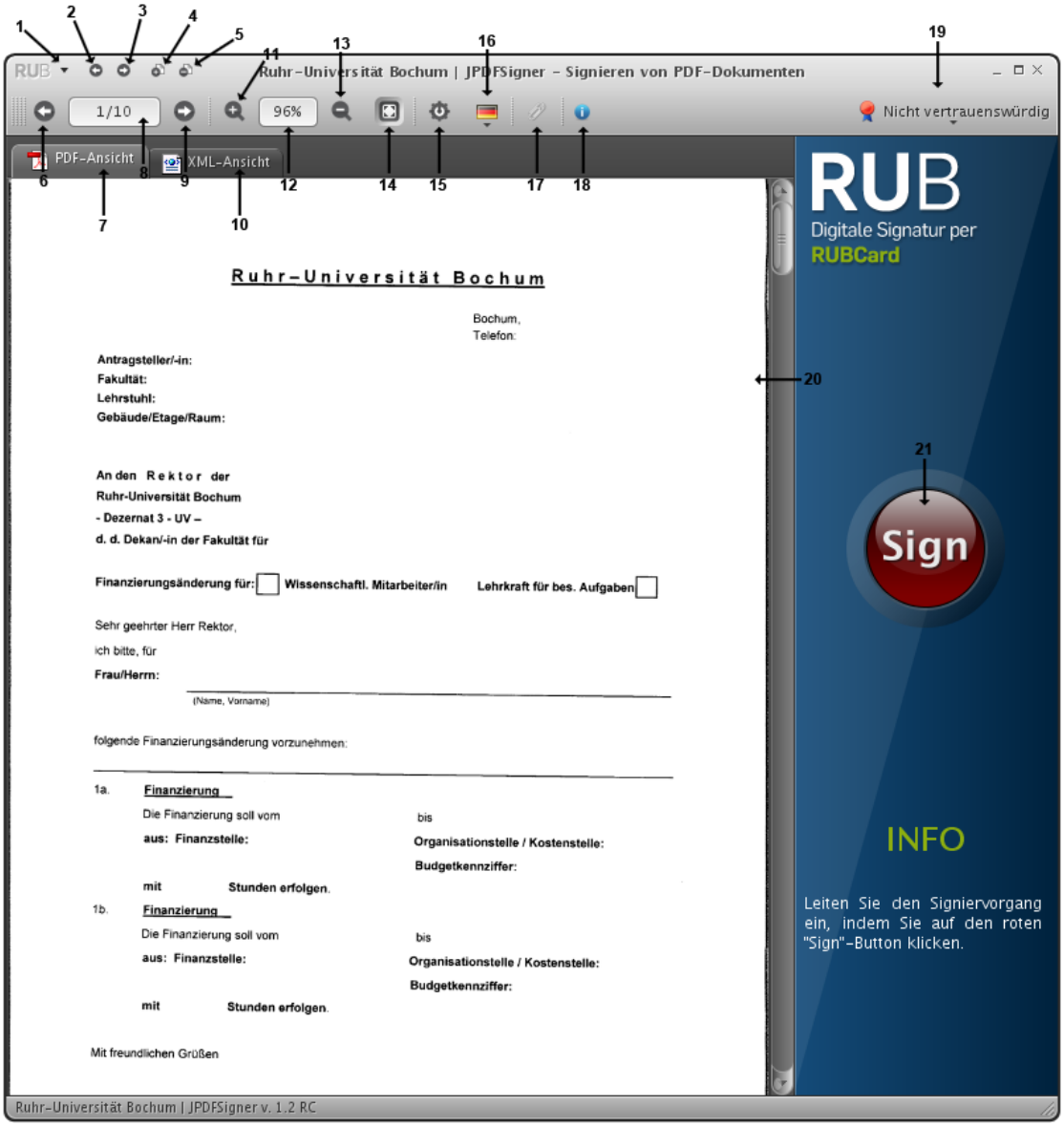
5.2.8 UPLOAD DES PDF-DOKUMENTS

Nachdem das PDF-Dokument erfolgreich signiert wurde *(Punkt 5.2.7)* erfolgt der Upload automatisch. Achten sie bitte darauf, dass ihre Firewall nicht die ausgehenden Verbindungen von „JPDFSigner" und die der Java-Runtime blockiert.

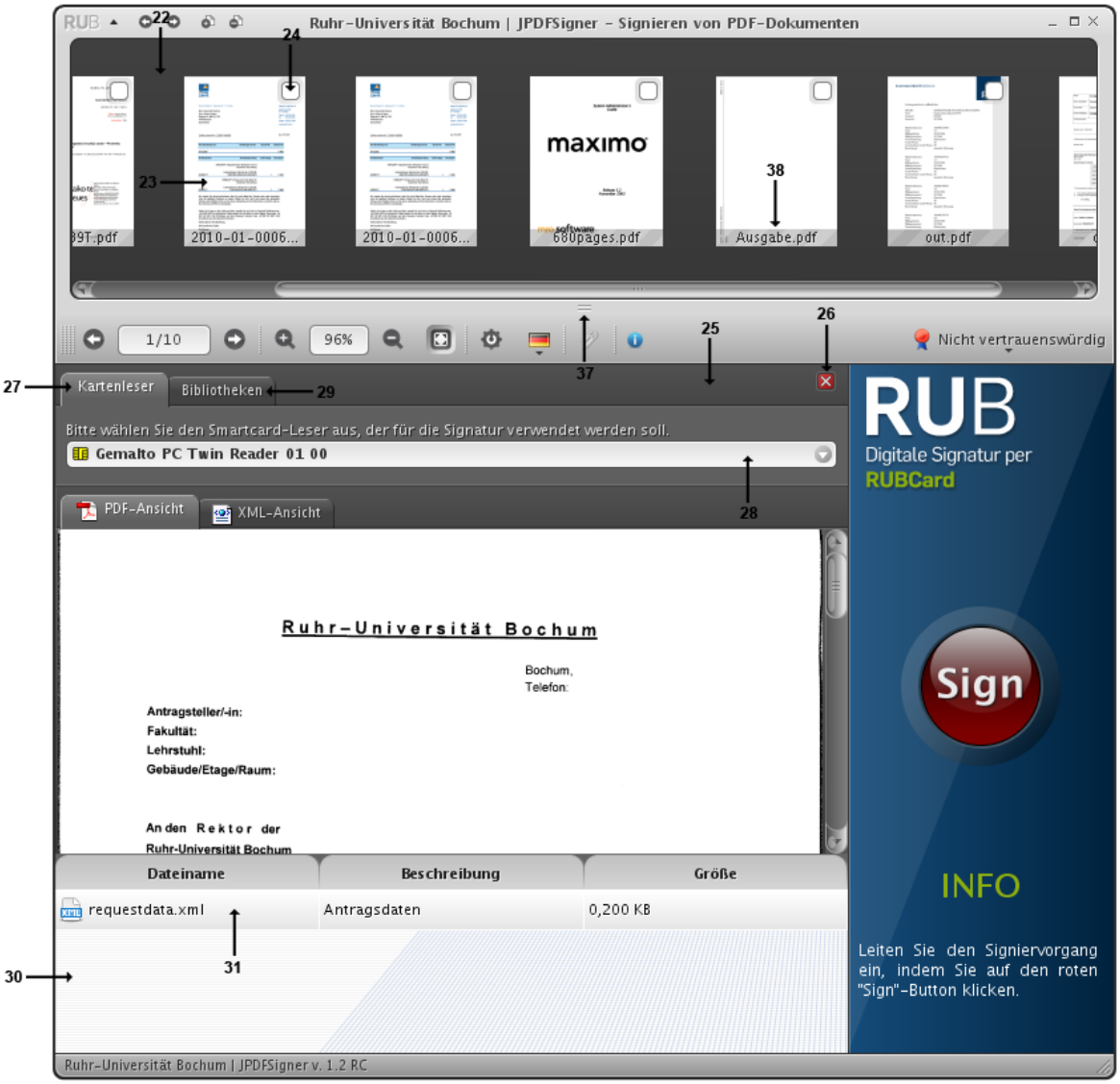
RUB

6 ANLAGE I

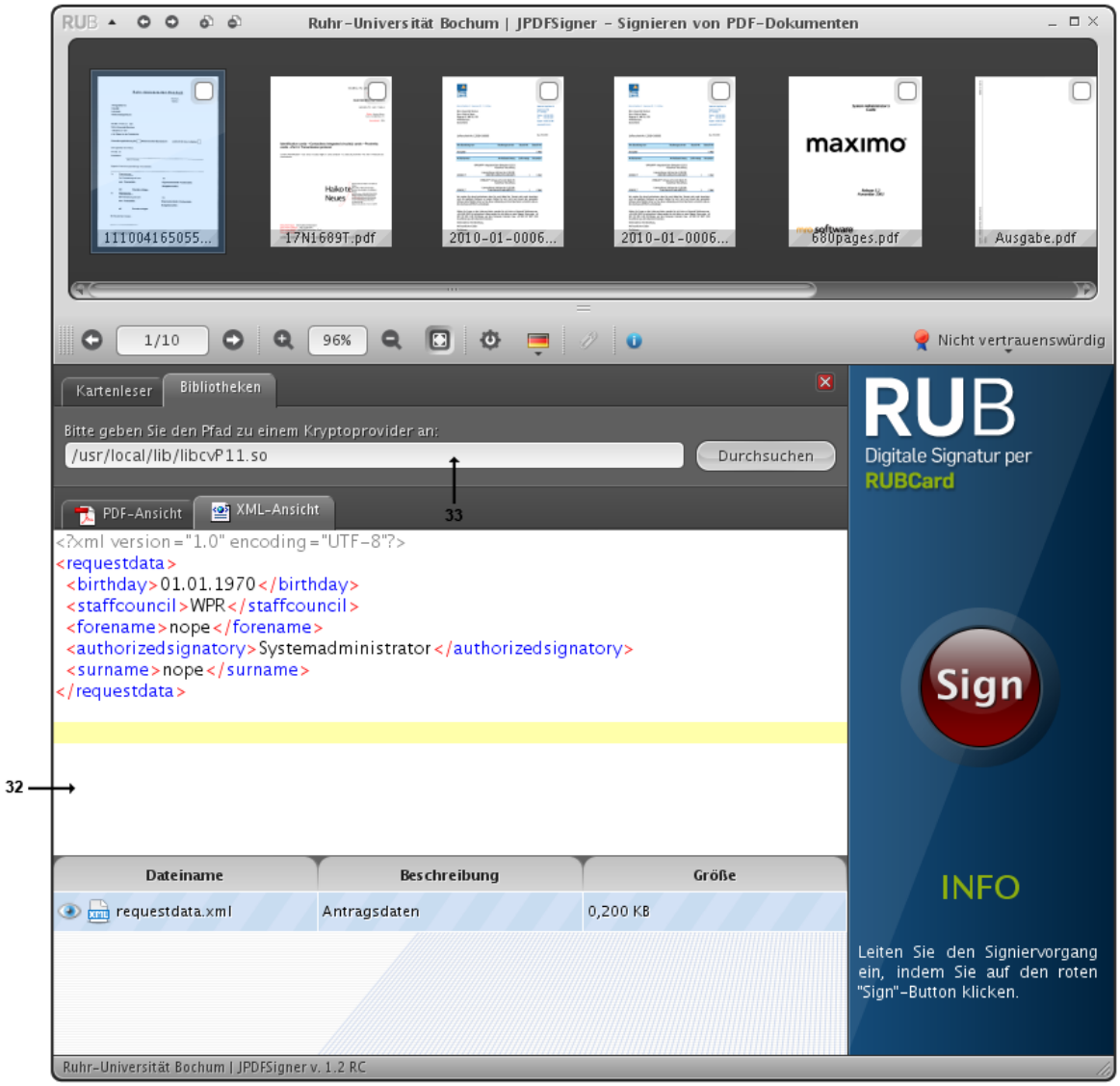
UI Screenshots


Ruhr-Universität Bochum | JPDFSigner - Signieren von PDF-Dokumenten
1/10
96%
Nicht vertrauenswürdig
PDF-Ansicht
XML-Ansicht
Ruhr-Universität Bochum
Bochum,
Telefon:
Antragsteller/-in:
Fakultät:
Lehrstuhl:
Gebäude/Etage/Raum:
An den Rektor der
Ruhr-Universität Bochum
- Dezernat 3 - UV -
d. d. Dekan/-in der Fakultät für
Finanzierungsänderung für: Wissenschaftl. Mitarbeiter/in Lehrkraft für bes. Aufgaben
Sehr geehrter Herr Rektor,
ich bitte, für
Frau/Herrn:
(Name, Vorname)
folgende Finanzierungsänderung vorzunehmen:
1a. Finanzierung
Die Finanzierung soll vom
bis
aus: Finanzstelle:
Organisationstelle / Kostenstelle:
Budgetkennziffer:
mit Stunden erfolgen.
1b. Finanzierung
Die Finanzierung soll vom
bis
aus: Finanzstelle:
Organisationstelle / Kostenstelle:
Budgetkennziffer:
mit Stunden erfolgen.
Mit freundlichen Grüßen
RUB
Digitale Signatur per
RUBCard
Sign
INFO
Leiten Sie den Signiervorgang ein, indem Sie auf den roten "Sign"-Button klicken.
Ruhr-Universität Bochum | JPDFSigner v. 1.2 RC

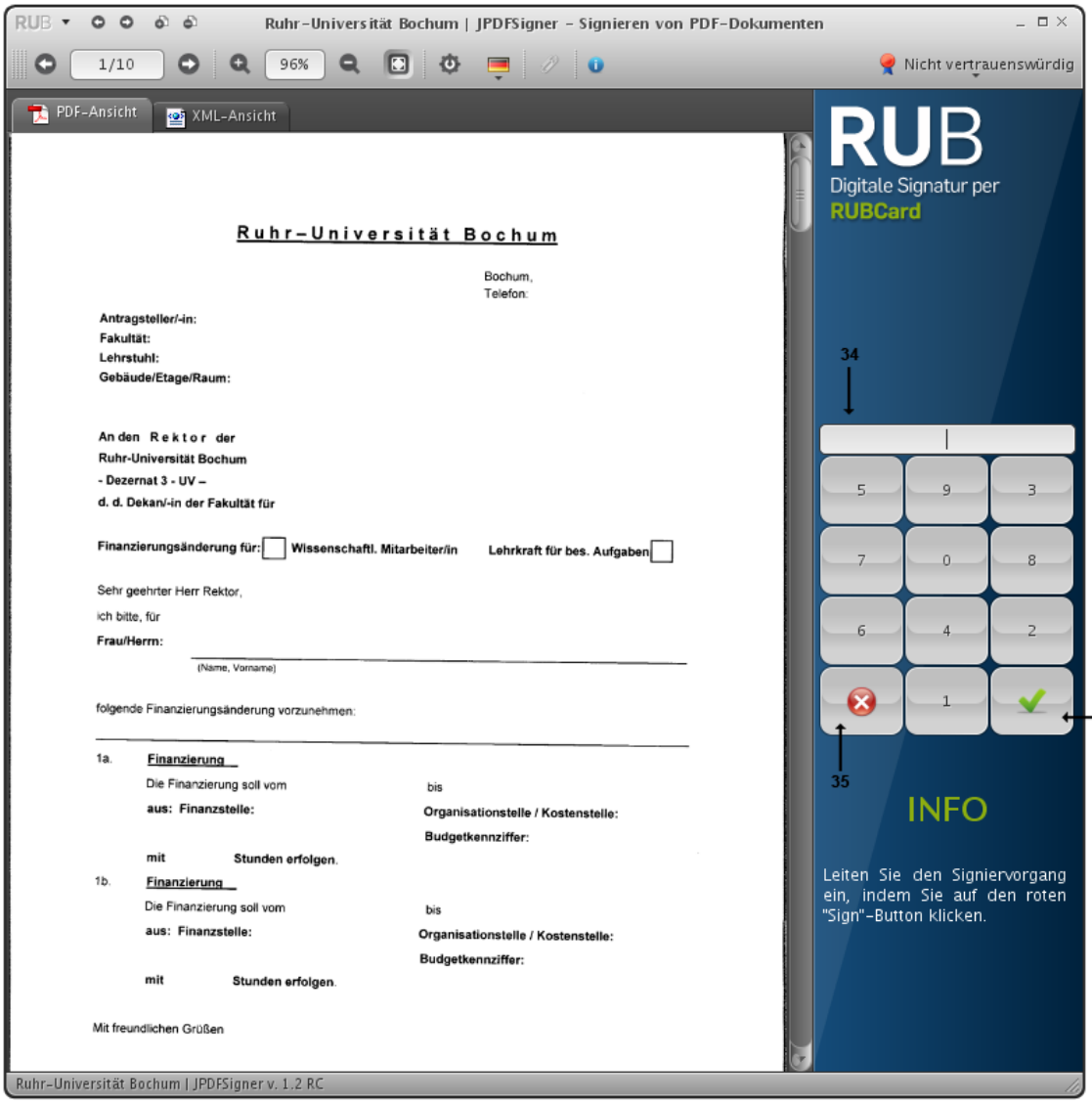
(Abbildung 1)


Ruhr-Universität Bochum | JPDFSigner - Signieren von PDF-Dokumenten
22
24
23
38
maximo
2010-01-0006...
2010-01-0006...
680pages.pdf
Ausgabe.pdf
out.pdf
1/10
96%
25
26
37
Nicht vertrauenswürdig
27
Kartenleser
Bibliotheken
29
Bitte wählen Sie den Smartcard-Leser aus, der für die Signatur verwendet werden soll.
Gemalto PC Twin Reader 01 00
28
PDF-Ansicht
XML-Ansicht
Ruhr-Universität Bochum
Bochum,
Telefon:
Antragsteller/-in:
Fakultät:
Lehrstuhl:
Gebäude/Etage/Raum:
An den Rektor der
Ruhr-Universität Bochum
Dateiname
Beschreibung
Größe
requestdata.xml
Antragsdaten
0,200 KB
30
31
RUB
Digitale Signatur per
RUBCard
Sign
INFO
Leiten Sie den Signiervorgang ein, indem Sie auf den roten "Sign"-Button klicken.
Ruhr-Universität Bochum | JPDFSigner v. 1.2 RC

(Abbildung 2)


Ruhr-Universität Bochum | JPDFSigner - Signieren von PDF-Dokumenten
111004165055...
17N1689T.pdf
2010-01-0006...
2010-01-0006...
680pages.pdf
Ausgabe.pdf
1/10
96%
Nicht vertrauenswürdig
Kartenleser
Bibliotheken
Bitte geben Sie den Pfad zu einem Kryptoprovider an:
/usr/local/lib/libcvP11.so
Durchsuchen
33
PDF-Ansicht
XML-Ansicht
<?xml version="1.0" encoding="UTF-8"?>
<requestdata>
<birthday>01.01.1970</birthday>
<staffcouncil>WPR</staffcouncil>
<forename>nope</forename>
<authorizedsignatory>Systemadministrator</authorizedsignatory>
<surname>nope</surname>
</requestdata>
32
Dateiname
Beschreibung
Größe
requestdata.xml
Antragsdaten
0,200 KB
RUB
Digitale Signatur per
RUBCard
Sign
INFO
Leiten Sie den Signiervorgang ein, indem Sie auf den roten "Sign"-Button klicken.
Ruhr-Universität Bochum | JPDFSigner v. 1.2 RC

(Abbildung 3)


RUB
Ruhr-Universität Bochum | JPDFSigner - Signieren von PDF-Dokumenten
1/10
96%
Nicht vertrauenswürdig
PDF-Ansicht
XML-Ansicht
Ruhr-Universität Bochum
Bochum,
Telefon:
Antragsteller/-in:
Fakultät:
Lehrstuhl:
Gebäude/Etage/Raum:
An den Rektor der
Ruhr-Universität Bochum
- Dezernat 3 - UV -
d. d. Dekan/-in der Fakultät für
Finanzierungsänderung für: Wissenschaftl. Mitarbeiter/in
Lehrkraft für bes. Aufgaben
Sehr geehrter Herr Rektor,
ich bitte, für
Frau/Herrn:
(Name, Vorname)
folgende Finanzierungsänderung vorzunehmen:
1a. Finanzierung
Die Finanzierung soll vom
bis
aus: Finanzstelle:
Organisationstelle / Kostenstelle:
Budgetkennziffer:
mit
Stunden erfolgen.
1b. Finanzierung
Die Finanzierung soll vom
bis
aus: Finanzstelle:
Organisationstelle / Kostenstelle:
Budgetkennziffer:
mit
Stunden erfolgen.
Mit freundlichen Grüßen
RUB
Digitale Signatur per
RUBCard
34
5
9
3
7
0
8
6
4
2
1
36
35
INFO
Leiten Sie den Signiervorgang ein, indem Sie auf den roten "Sign"-Button klicken.
Ruhr-Universität Bochum | JPDFSigner v. 1.2 RC

(Abbildung 4)