Governikus KG

Governikus
Signer WebEdition

Entwickler- und Administratorhandbuch
Governikus Signer WebEdition

Governikus Signer WebEdition, Version 2.9.0

Dokumentenversion: 2.9.0_0

Inhaltsverzeichnis

1 Einführung

Die Governikus Signer Web Edition ist eine Web-Anwendung zum Signieren und zum Ver- und Entschlüsseln von Dateien.

Hinweis: Der Governikus **WebSigner** heißt ab Release 2.9.0 Governikus Signer WebEdition oder kurz **WebEdition**.

Achtung: Durch die Umstellung der Auslieferung ist diese Version nicht mehr kompatibel zu Vorgängerversionen!

Signieren

Die WebEdition stellt dem Benutzer eine Oberfläche zum Signieren von Dateien zur Verfügung. Es wird die Erstellung von qualifizierten elektronischen Signaturen sowie von fortgeschrittenen Signaturen unterstützt. Die Erstellung von qualifizierten elektronischen Signaturen erfolgt über eine Signaturkarte in einem lokal bei dem Benutzer angeschlossenen Kartenleser. Für die Erstellung von fortgeschrittenen elektronischen Signaturen kann ein Software-Zertifikat genutzt werden.

Ver- und Entschlüsseln

Die WebEdition stellt dem Benutzer eine Oberfläche zum Verschlüsseln und eine Oberfläche zum Entschlüsseln von Dateien zur Verfügung. Die Ver- und Entschlüsselung kann über eine Signaturkarte in einem lokal bei dem Benutzer angeschlossenen Kartenleser erfolgen. Alternativ ist auch eine Verwendung eines Passwortes oder Software-Zertifikats möglich.

Um die Funktionalitäten der WebEdition nutzen zu können, wird ein Programm als Installationsdatei oder als ZIP-Archiv zur Verfügung gestellt. Der Aufruf erfolgt über eine Fachanwendung. Bei der Verwendung von Kartenlesern muss ein entsprechender Treiber installiert sein.

Die Bestimmung der zu verarbeitenden Datei erfolgt durch die Fachanwendung mit dem Aufruf der Anwendung. Es kann eine lokal vorliegende Datei bzw. ein lokales Verzeichnis bestimmt werden, oder es kann die zu verarbeitende Datei über eine URL referenziert werden. Die signierte Datei kann entweder an die Fachanwendung zurückgegeben oder lokal abgelegt werden.

Darüber hinaus kann die aufrufende Fachanwendung Einfluss auf den dem Benutzer zu Verfügung gestellten Funktionsumfang nehmen und die Anwendung dadurch präzise an den jeweiligen Workflow anpassen.

Achtung: Die Funktionen der WebEdition können nur durch eine Fachanwendung aufgerufen werden. Die Software kann nicht unabhängig von der Fachanwendung als Programm auf dem lokalen PC aufgerufen werden.

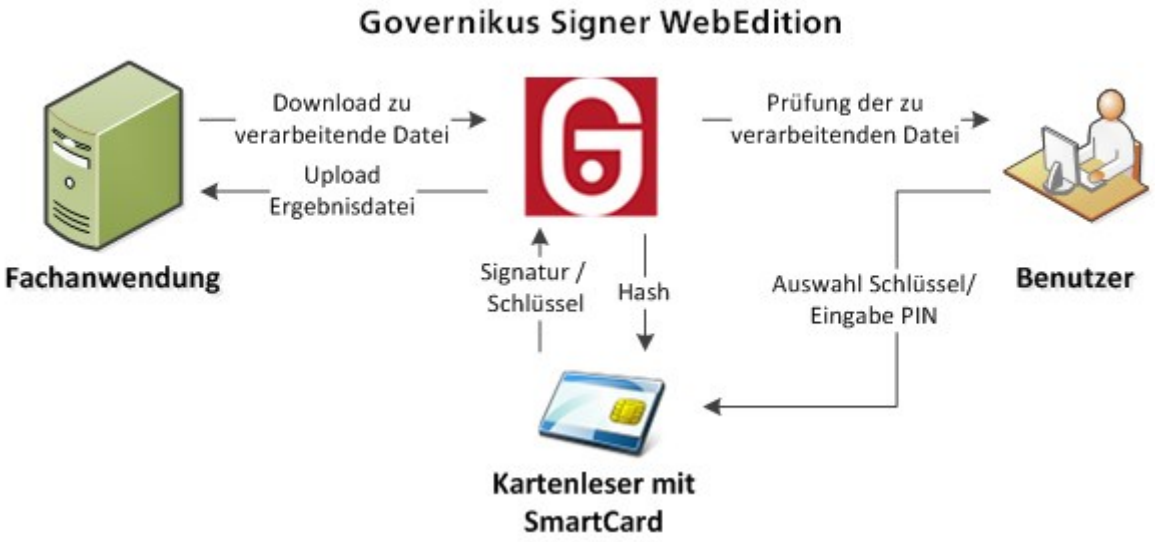

Abbildung 1: Ablaufdiagramm

Die Verarbeitung folgt diesem Ablauf:

- Nach dem Aufruf durch die Fachanwendung öffnet sich die Anwendung Governikus Signer WebEdition mit der Benutzeroberfläche. Die zu verarbeitenden Dateien werden ggf. auf den Benutzer-PC heruntergeladen.
- Der Benutzer wählt von einer Signaturkarte oder aus einer Datei den Signaturschlüssel, Entschlüsselungsschlüssel bzw. ein Verschlüsselungszertifikat aus.

Sofern es die Fachanwendung erlaubt, vergleiche Kapitel 2, kann der Benutzer folgende Optionen bestimmen:

- Signieren
  - Signaturformat (PKCS#7 detached, PKCS#7 enveloped, PDF-Inline)
  - Verhalten bei vorhandener Signatur (nur bei PDF-Inline-Signatur)
  - Bestimmung des Zielverzeichnisses
  - Eine lokale Kopie erstellen
  - Zeitstempel erzeugen
- Verschlüsseln
  - Entschlüsselungsmethode auswählen (Passwort oder Entschlüsselungsschlüssel von Signaturkarte oder Softwareschlüssel)
  - ZIP-Archive nach dem Entschlüsseln automatisch entpacken
  - Bestimmung des Zielverzeichnisses
  - Eine lokale Kopie erstellen
- Entschlüsseln
  - Verschlüsselungsmethode auswählen (Passwort oder Zertifikat)
  - Dateien automatisch vor dem Verschlüsseln in einem ZIP-Archiv zusammenfassen
  - Bestimmung des Zielverzeichnisses
  - Eine lokale Kopie erstellen

- Vor der Signaturerstellung bzw. dem Verschlüsseln hat der Benutzer die Möglichkeit, die zu signierenden Dateien einzusehen. Beim Entschlüsseln besteht diese Möglichkeit nicht.
- Nach dem Starten des Signatur- bzw. des Entschlüsselungsvorgangs durch den Benutzer, ist pro zu verarbeitender Datei die Eingabe der PIN erforderlich, es sei denn, es wird eine der unterstützten Stapelsignaturkarten oder ein Software-Schlüssel verwendet.
- Nach dem Abschluss der Dateiverarbeitung schließt sich die Anwendung und liefert einen `ReturnCode` an die aufrufende Anwendung zurück.

2 Governikus Signer WebEdition Lizenzdateien

Dieses Kapitel erklärt allgemeines, den grundlegenden Aufbau und den Inhalt der Lizenzdatei.

2.1 Allgemein

Die Governikus Signer WebEdition verfügt über zwei Lizenzdateien, die zur Konfiguration der Software genutzt werden können.

- Die Produktlizenzdatei wird mit der Governikus Signer WebEdition ausgeliefert und definiert den generellen Funktionsumfang der Software.
- Beim Aufruf der Governikus Signer WebEdition kann die aufrufende Anwendung eine weitere Lizenzdatei übergeben. Diese Benutzerlizenzdatei kann die Produktlizenz nur maskieren, d.h. es können Funktionen deaktiviert werden aber keine über die Produktlizenzdatei deaktivierten Funktionen wieder aktiviert werden.

2.2 Grundlegender Aufbau

Die Lizenzdatei ist ein XML-Dokument, das für jeden Unterpunkt einen Lizenztyp und ggf. Unterpunkte definiert. Jeder Lizenztyp hat die drei Eigenschaften "Editierbar", "im Prozess nutzen" und "Anzeigen", um den Funktionsumfang der Software zu konfigurieren.

- **Anzeigen**: Die entsprechenden grafischen Elemente werden auf der Benutzeroberfläche dargestellt.
- **Editierbar**: Der Benutzer hat die Möglichkeit, die Einstellung dieser Funktion über die Benutzeroberfläche zu ändern.
- **Im Prozess nutzen**: Die jeweilige Funktion steht grundsätzlich zur Nutzung zur Verfügung.

Von den acht möglichen Kombinationen dieser drei Eigenschaften, sind nur fünf Kombinationen in diesem Kontext nutzbar. In der nachfolgenden Tabelle sind die daraus resultierenden fünf Lizenztypen aufgeführt und die jeweils abgedeckten Eigenschaften durch ein "X" gekennzeichnet. Verfügbare Lizenztypen:

Lizenztyp	Editierbar	Im Prozess nutzen	Anzeigen
`enable`	X	X	X
`hide`		X	
`hide_disable`			
`disable`			X
`notEditable`		X	X

2.3 Inhalt der Lizenzdatei

Die Produktlizenzdatei der Governikus Signer WebEdition ist wie folgt aufgebaut:

```
<root displayName="Governikus Signer WebEdition">
  <general>enable
    <actions>
      <settings>
        <proxyserver>enable</proxyserver>
      </settings>
    </actions>
  </general>
  <process>
    <sign>
      <options>enable
       <visualization>enable</visualization>
        <signformat>enable
          <pkcs7detached>enable</pkcs7detached>
          <pdfinline>enable</pdfinline>
          <pkcs7enveloped>enable</pkcs7enveloped>
        </signformat>

        <extended_pdfsignature>enable
          <reason>enable</reason>enable
       </extended_pdfsignature>
        <timestampserver>enable</timestampserver>
        <timestamp>enable</timestamp>
        <trustedviewer>enable</trustedviewer>
      </options>
      <sourcefolder>enable
       <addfiles>enable</addfiles>
        <removefiles>enable</removefiles>
      </sourcefolder>enable
     <key>enable
        <software>enable</software>
      </key>
      <targetfolder>enable
        <targetfolder>enable</targetfolder>
        <localcopyfolder>enable</localcopyfolder>
      </targetfolder>
    </sign>
  </process>
</root>
```

Listing 1: Inhalt der WebEdition Produktlizenzdatei für das Signieren

Die Produktlizenzdatei für das Ver- und Entschlüsseln ist wie folgt aufgebaut:

```
<root displayName="Encrypt und Decrypt">
  <general>enable
    <actions>
      <settings>
        <proxyserver>enable</proxyserver>
      </settings>
    </actions>
  </general>
```

```
  <process>
    <decrypt>enable
      <targetfolder>enable
        <unzip>enable</unzip>
        <localcopyfolder>hide_disable</localcopyfolder>
        <targetfolder>enable</targetfolder>
      </targetfolder>
      <key>enable
        <software>enable</software>
        <password>enable</password>
      </key>
    </decrypt>
    <encrypt>enable
      <targetfolder>enable
        <targetfolder>enable</targetfolder>
        <createzip>enable</createzip>
        <localcopyfolder>hide_disable</localcopyfolder>
      </targetfolder>
      <key>enable
        <software>enable</software>
        <password>enable</password>
        <encryptmultiaddressee>enable</encryptmultiaddressee>
      </key>

    </encrypt>
  </process>
</root>
```

Listing 2: Inhalt der WebEdition Produktlizenzdatei für das Ver- und Entschlüsseln

Benutzerlizenzdatei

Die Benutzerlizenzdatei maskiert die Produktlizenzdatei, d.h. sie kann den Funktionsumfang weiter einschränken. Da alle relevanten Einstellungen der Produktlizenz den Lizenztyp `enable` tragen, entsprechen die Inhalte der Benutzerlizenzdatei direkt der "Gesamtlizenz".

Die nachfolgende Tabelle erläutert die einzelnen Elemente und deren möglichen Lizenztypen. Die detaillierte Beschreibung der einzelnen Funktionen entnehmen Sie bitte dem Benutzerhandbuch. Beachten Sie bitte, dass es nachfolgend nur darum geht, inwieweit die einzelnen Funktionen, Konfigurationsmöglichkeiten und Dialogschritte einem Benutzer zur Verfügung stehen. Eine Vorgabe der Konfiguration, z.B. welches Signaturformat oder welches Zielverzeichnis genutzt werden soll, erfolgt über Aufrufparameter.

Element	**Bedeutung**	**Mögliche Lizenztypen**				
		enable	**hide**	**notEditable**	**disable**	**hide_disable**
`sign`	Signaturprozess allgemein	X				
`sourcefolder`	Dialogschritt "Dateiauswahl"	X	X	X		
`addfiles`	Funktion "Dateien hinzufügen"	X	X	X	X	X

Element	Bedeutung	Mögliche Lizenztypen				
		enable	hide	notEditable	disable	hide_disable
removefiles	Funktion "Ausgewählte Dateien entfernen"	X	X	X	X	X
options	Dialogschritt "Optionen"	X	X	X		
signformat	Auswahl "Signaturformat wählen" im Dialogschritt "Optionen"	X	X	X		
pkcs7detached	Signaturformat PKCS#7 detached ("Signatur als gesonderte Datei beifügen")	X	X	X	X	X
pdfinline	Signaturformat PDF-Inline ("PDF-Signaturen stets einbetten")	X	X	X	X	X
pkcs7enveloped	Signaturformat PKCS#7 enveloped ("Dokument in Signaturdatei einbetten")	X	X	X	X	X
extended_pdfsignature	Erweiterte bzw. sichtbare PDF-Signatur ermöglichen ("Erweiterte PDF-Signatur einbetten")	X	X	X	X	X
reason	Angabe Signaturgrund bei erweiterter PDF-Signatur ("Grund der Unterschrift")	X	X	X	X	X
timestampserver	Konfiguration des Zeitstempelservers	X	X	X	X	X
timestamp	Erstellung von Zeitstempeln im Dialogschritt „Optionen“	X	X	X	X	X
visualization	Mindestanzahl der einzusehenden Dateien prüfen	X			X	
key	Dialogschritt "Schlüssel wählen"	X	X	X		
software	Funktion "Schlüssel aus Datei laden" (siehe auch nachfolgende Hinweisbox)	X	X	X	X	X
targetfolder	Dialogschritt "Zielverzeichnis wählen"	X	X	X		
targetfolder	Funktion "Zielverzeichnis wählen"	X	X	X		
localcopyfolder	Funktion "Lokale Kopie erstellen"	X				X
decrypt	Entschlüsselungsprozess allgemein	X				

Element	Bedeutung	Mögliche Lizenztypen				
		enable	hide	notEditable	disable	hide_disable
encrypt	Verschlüsselungsprozess allgemein	X				
unzip	Automatisches Entpacken von ZIP-Archiven nach dem Entschlüsseln	X	X	X	X	X
createzip	Automatisches Erstellen von ZIP-Archiven mit allen Dateien vor dem Verschlüsseln.	X	X	X	X	X
encryptmultiaddressee	Verschlüsseln für mehr als ein Zertifikat.	X	X	X	X	X
password	Ver-/Entschlüsselung über Passwort	X	X	X	X	X

Tabelle 1: Übersicht über die Parameter der Lizenzdatei

Darüber hinaus wird beim Einlesen der Benutzerlizenzdateien jeder Unterpunkt mit dessen Knotenpunkt maskiert. Somit kann kein Unterpunkt (z.B. Funktion) sichtbar und dessen Knotenpunkt (z.B. Dialogschritt) unsichtbar sein.

Das nachfolgende Beispiel zeigt eine Benutzerlizenz und die daraus resultierende Gesamtlizenz. Die Benutzerlizenz erlaubt nur Signaturen im PKCS#7-Format, aber keine PDF-Inline-Signaturen. Die Einstellungsmöglichkeit des Signaturformates im Dialogschritt "Optionen" soll nicht auf der Benutzeroberfläche dargestellt werden (d.h. welches Format verwendet werden soll, muss als Aufrufparameter übergeben werden). Es können nur Schlüssel von Signaturkarten verwendet werden. Beispiel:

```
<sign>enable
  <signformat>hide
    <pdfinline>disable</pdfinline>
  </signformat>
  <key>enable
    <software>hide_disable</software>
  </key>
  <targetfolder>
    <targetfolder>hide_disable</targetfolder>
  </targetfolder>
</sign>
```

Listing 3: Beispiel einer Benutzerlizenzdatei

```
<sign>enable
  <sourcefolder>enable
    <addfiles>enable</addfiles>
    <removefiles>enable</removefiles>
  </sourcefolder>
  <options>enable
    <signformat>hide
      <pkcs7detached>hide</pkcs7detached>
```

```
        <pdfinline>hide_disable</pdfinline>
        <pkcs7enveloped>hide</pkcs7enveloped>
      </signformat>
      <visualization>enable</visualization>
    </options>
    <key>enable
      <software>hide_disable</software>
    </key>
    <targetfolder>enable
      <targetfolder>hide_disable</targetfolder>
      <localcopyfolder>enable</localcopyfolder>
    </targetfolder>
  </sign>
```

Listing 4: Beispiel einer interpretierten Lizenzdatei ("Gesamtlizenz")

Standard-Benutzerlizenz

Ohne Angabe einer Benutzerlizenzdatei bietet die Governikus Signer WebEdition dem Benutzer den gesamten Funktionsumfang. I. d. R. sind Optionen wie Signaturformat sowie Zielverzeichnis durch den Einsatzzweck vorgegeben und sollen durch den Benutzer nicht geändert werden. Die zugehörigen Dialogschritte "Dateiauswahl", "Optionen" und "Zielverzeichnis wählen" können dann entfallen. Die nachfolgend abgebildete Lizenzdatei blendet die o. g. Dialogschritte aus.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <process>
    <sign>enable
      <sourcefolder>hide</sourcefolder>
      <options>hide</options>
      <targetfolder>hide</targetfolder>
    </sign>
  </process>
</root>
```

Listing 5: Inhalt der enthaltenen Standard-Benutzerlizenzdatei

Diese Standard-Lizenzdatei ist bereits in der Anwendung enthalten. Soll die Lizenzdatei verwendet werden, ist mit dem Aufruf der Anwendung der Parameter `licenceFile` (siehe folgende Kapitel) mit folgender URL anzugeben:

`http://<server>:<port>/GovernikusSignerWESample/dynamicSignLicence.xml`

Hinweis: Die mitgelieferte Online-Hilfe beschreibt den vollständigen Funktionsumfang. Es wird auch auf den Fall eingegangen, dass Einstellungsoptionen zwar sichtbar sind, aber durch den Benutzer nicht verändert werden können (Lizenztyp `disable`).

3 Funktionsübersicht

Nachfolgend werden die Funktionen anhand der Aufrufparameter beschrieben. Wie der Aufruf der Anwendung erfolgt, ist im Abschnitten 3.2 dargestellt. Im folgenden Abschnitt 3.1 sind zunächst die allgemeinen Aufrufparameter, die gleichermaßen für alle Funktionen gelten, und anschließen die funktionsspezifischen Parameter erläutert.

3.1 Aufrufparameter

Es ist zwischen Pflichtparametern, die bei jedem Aufruf übergeben werden müssen, und optionalen Parametern zu unterscheiden. Bei den optionalen Parametern ist das Zusammenspiel mit der GUI bzw. der dynamischen Lizenz zu beachten. Ist z.B. das Signaturformat über die GUI wählbar, wirkt der entsprechende Parameter nur als Vorbelegung, die vom Benutzer beliebig verändert werden kann. Soll der Benutzer dieses nicht ändern können, so ist diese Option über die Lizenzdatei (vgl. Kapitel 2) entweder zu deaktivieren `disable` oder zu verstecken `hide`.

3.1.1 Allgemeine Parameter

Pflichtparameter:

- **`signer.process=<Prozesstyp>`**: Geben Sie hier an, mit welcher Funktion die WebEdition gestartet werden soll. Ersetzen Sie `<Prozesstyp>` durch einen dieser Einträge `sign`, `encrypt` oder `decrypt`.
- **`signer.rcurl`**: URL (z.B. einer JSP), an die der Return-Code inkl. der Return-Code-Beschreibung über die `GET`-Parameter `returnCode` und `returnCodeDescription`, gesendet werden soll. Beispiel:
  - `http://<Servername>:8080/GovernikusSignerWESample/rcCode?transactionID=12345678`

Optionale Parameter

Abhängigkeiten einzelner Parameter untereinander sind zu beachten. Für nicht angegebene Parameter wird, wenn erforderlich, ein Default-Wert verwendet.

- **`signer.source`**: Dateiname oder URL der zur signierenden (Quell-) Datei. Alternativ kann auch ein Verzeichnis angegeben werden, aus dem dann alle vorhandenen Dateien ausgewählt werden. Datei- und Verzeichnisnamen müssen immer mit dem vollständigen Pfad angegeben werden. Im Fall einer URL wird von der zu signierenden Datei zunächst eine lokale Kopie angelegt. Der Name der lokalen Kopie kann der URL optional, durch ein Pipe-Zeichen getrennt, mitgegeben werden. Beispiel:
  - Einzeldatei lokal: `/home/governikus/temp/source/test.pdf`
  - Verzeichnis lokal: `/home/governikus/temp/source/`
  - Einzeldatei URL:
  - `https://<Server>:443/server_datei.pdf|lokale_datei.pdf|sha256-hash`
    - `lokale_datei.pdf`: Bei einer Quelle im Internet kann ein lokaler Dateiname mit angegeben werden, da nicht alle Internetquellen einen Dateinamen in der URL enthalten.

- **sha256-hash**: Zusätzlich kann ein Sha256-Hashwert der Datei im Aufruf mitgegeben werden. Wenn ein Hashwert mitgegeben wird, überprüft die WebEdition diesen mit der herunter geladenen Datei. Sollten die beiden Hashwerte nicht übereinstimmen, wird ein `ReturnCode 59 (HASHEDFILE_CHANGED)` an den Server gesendet und dem Benutzer im Fehlerdialog angezeigt. Mit dem Senden des ReturnCodes wird der lokal errechnete Hashwert mit der Datei verknüpft.

Hinweis: Sobald der Hashwert einer Serverdatei nicht mit dem lokal ermittelten Hashwert übereinstimmt und der Benutzer diesen Vorgang aber trotz der Fehlermeldung nicht abbricht, bekommt er keine weitere Fehlermeldung vor dem Signieren.

- Alle o. g. Quellen können auch mehrfach in beliebiger Kombination angegeben werden.

Hinweis: Sobald eine Quelldatei als URL übergeben wurde, sollte als `targetFolderTyp` immer `oneSpecial` übergeben werden, da die Ergebnisdatei einer URL-Quelldatei beim Parameter `sameAsSourceFolder` im Temp-Verzeichnis gespeichert werden würde.

- **signer.dest**: Lokales Zielverzeichnis oder Ziel-URL.

  Dieser Parameter wird nicht ausgewertet, wenn für folgenden Parameter gilt:

  - `signer.targetFolderType="sameAsSourceFolder"`

- **signer.targetFolderType**: Bestimmt die Art des Zielverzeichnisses:
  - **sameAsSourceFolder**: (Default) Verzeichnis der jeweiligen Quelldatei verwenden (nicht möglich, wenn der Parameter `source` eine URL enthält). Parameter `dest` wird nicht ausgewertet.
  - **oneSpecial**: separat angegebenes Zielverzeichnis (erfordert Parameter `dest`)
- **signer.processPolicy**: Steuert, mit welchem Dialogschritt die Anwendung startet. Welche Dialogschritte überhaupt möglich sind, wird über die Lizenz-Datei vorgegeben (vgl. Kapitel 2).
  - **skipCompletedStep**: (Default) Alle Dialogschritte, in denen die notwendigen Angaben bereits vorliegen (über Default-Einstellungen oder Parameter) werden übersprungen.
  - **firstStep**: Der Dialog beginnt mit dem ersten Schritt.
  - **lastStep**: Der Dialog startet mit dem Dialogschritt "Signieren".
- **signer.configurationFile**: Voll qualifizierter Dateiname einer lokalen Konfigurationsdatei oder URL zu einer Konfigurationsdatei. Über die Konfigurationsdatei können alle Einstellungen, die in der WebEdition über den Einstellungsdialog eingestellt werden, übergeben werden.
- **signer.licenceFile**: Voll qualifizierter Dateiname einer lokalen Lizenzdatei oder URL zu einer Lizenzdatei. Über die Lizenzdatei kann der Funktionsumfang der GUI eingeschränkt werden (vgl. Kapitel 2).
- **signer.redirectURL**: URL zu der nach Beendigung der Anwendung im Browser zurückgesprungen werden soll. Wenn keine `redirectURL` übergeben wurde, wird die Governikus Signer WebEdition geschlossen.

- **`signer.log4j.configuration`**: Log4J Konfigurationsdateiname. Folgende Konfigurationsdateien sind enthalten: Standard-Logging: `log4j-signer.xml`; Debug-Logging: `log4j-signer-debug.xml`.

3.1.2 Parameter für Signier-Funktion

Pflichtparameter:

- **`signer.sigFormat`**: Bestimmt das Signaturformat
  - **`detachedPKCS7`**: (Default) PKCS#7 Signaturdatei mit separater Inhaltsdatei. Die Inhaltsdatei wird immer mit an den Zielort übergeben.
  - **`envelopedPKCS7`**: PKCS#7 Signaturdatei mit eingebettetem signierten Inhalt

Optionale Parameter:

- **`signer.serialnumber`**: Seriennummer der zu verwendenden Signaturkarte oder des zu verwendenden Zertifikats.
  - Soll der Benutzer den Signaturschlüssel frei wählen, kann der Parameter `serialnumber` bzw. `keyStore` entfallen
  - **`automatic`**: Wird `automatic` angegeben, wird der Signaturschlüssel einer Signaturkarte automatisch ausgewählt, sofern die Karte beim Start der WebEdition bereits verfügbar ist und sich der Signaturschlüssel eindeutig bestimmen lässt. Zusammen mit dem Parameter `signer.processPolic == skipCompletedStep` kann so bei eingelegter Karte direkt auf den Dialogschritt "Signieren" gesprungen werden.
- **`signer.keyStore`**: Voll qualifizierter Dateiname des zu verwendenden Software-Signaturschlüssels (Dateiendung `.p12`).

Hinweis: Dieser Parameter wird nur ausgewertet, wenn der Parameter `signer.serialnumber` nicht übergeben wurde.

  - Soll der Benutzer den Signaturschlüssel frei wählen, kann der Parameter `keyStore` bzw. `serialnumber` entfallen.
- **`signer.pdfSignatureType`**: Steuert die Auswahl des Signaturformats für PDF-Dateien:

Hinweis: Nach der Erstellung einer PDF-Inline-Signatur enthält der Dateiname der signierten PDF-Datei immer den Zusatz `_signed`". Bsp.: `Rechnung_signed.pdf`

  - **`pdfInline`**: (Default) PDF-Dateien werden, unabhängig vom Parameter `signer.sigFormat`, im Format PDF-Inline, also einer in dem PDF eingebetteten PKCS#7-Signatur, signiert.

Hinweis: Die Einstellung, ob ein Signaturfeld verwendet wird, wird nur in der Konfigurationsdatei festgelegt (`signer.configurationFile`, siehe Kapitel 3.1.1). Die Beschreibung zur Konfigurationsdatei finden Sie in Kapitel 3.4.

  - **noPdfInline**: PDF-Dateien werden gemäß dem Parameter `signer.sigFormat` signiert.
  - **advancedPdfInline**: Diese Einstellung wird ab Version 2.7.6.0 nicht mehr unterstützt. Übergangsweise wird diese Einstellung wie die Einstellung **pdfInline** behandelt.
  - **automaticPdfInline**: Diese Einstellung wird ab Version 2.7.6.0 nicht mehr unterstützt. Übergangsweise wird diese Einstellung wie die Einstellung **pdfInline** behandelt.

- **signer.advancedPdfReason**: Signaturgrund für die erweiterte PDF-Signatur. Dieser Text wird in der PDF-Datei in den Unterschriftseigenschaften hinterlegt und in der sichtbaren PDF-Signatur dargestellt, sofern diese erstellt wird.
  - Der Signaturgrund wird nur bei erweiterten PDF-Inline-Signaturen (`signer.pdfSignatureType == advancedPdfInline`) ausgewertet.
- **signer.xmlSignatureType**: Steuert die Auswahl des Signaturformats für XML-Dateien:
  - **noXmlInline**: (Default) XML-Dateien werden gemäß dem Parameter `signer.sigFormat` signiert.
  - **xmlInline**: XML-Dateien werden, unabhängig von dem Parameter "`signer.sigFormat`", im Format XML-Inline, also einer in dem XML eingebetteten Signatur, signiert.
- **signer.createTimestamp**: Gibt an, ob ein Zeitstempel erzeugt werden soll.
  - **no**: (Default) Es wird kein Zeitstempel erzeugt.
  - **yes**: Ein Zeitstempel wird erzeugt. Es muss über den Parameter **signer.configurationFile** ein Zeitstempelserver in der Konfigurationsdatei mit übergeben werden.
- **signer.enableDuplicateSignatures**: Gibt an, ob eine zu signierende Datei vorher schon mit demselben Schlüssel signiert worden sein darf:
  - **yes**: (Default) Es wird keine Prüfung der Signaturen vor Anbringung der Signatur vorgenommen.
  - **no**: Vor Anbringung der Signatur wird geprüft, ob die zu signierende Datei schon eine Signatur mit demselben Schlüssel enthält. Sollte schon eine Signatur mit demselben Schlüssel vorhanden sein, so wird diese Datei nicht signiert und mit der Fehlermeldung `DUPLICATE_SIGNATURES_NOT_ALLOWED` übergangen.
- **signer.enablePseudonymSignatures**: Gibt an, ob für die Signaturerstellung eine Pseudonymkarte genutzt werden darf:
  - **yes**: (Default) Es wird keine Prüfung der Signaturschlüssel vor Anbringung der Signatur vorgenommen.
  - **no**: Vor Anbringung der Signatur wird geprüft, ob die ausgewählte Signaturkarte eine Pseudonymkarte ist. Sollte eine Pseudonymkarte ausgewählt worden sein, so werden die Schlüssel ausgegraut angezeigt und können vom Nutzer nicht ausgewählt werden.

3.1.3 Parameter Entschlüsselungs-Funktion

Pflichtparameter:

- **`signer.rcurl`**: siehe 3.1.1 Allgemeine Parameter

Optionale Parameter

- **`signer.serialnumber`**: Seriennummer der zu verwendenden Signaturkarte bzw. des Zertifikates oder der Dateiname eines Softwareschlüssels.
  - Soll der Benutzer den Signaturschlüssel frei wählen, kann der Parameter `serialnumber` entfallen
  - **`automatic`**: Wird `automatic` angegeben, wird der Entschlüsselungsschlüssel einer Signaturkarte automatisch ausgewählt, sofern die Karte beim Start der WebEdition bereits verfügbar ist und sich der Schlüssel eindeutig bestimmen lässt. Zusammen mit dem Parameter `signer.processPolic == skipCompletedStep` kann so bei eingelegter Karte direkt auf den Dialogschritt "Entschlüsseln" gesprungen werden.
  - Voll qualifizierter Dateiname des zu verwendenden Software-Entschlüsselungsschlüssels (Dateiendung `.p12`).
- **`signer.extractziparchiv`**: Gibt an, ob ein verschlüsseltes ZIP-Archiv nach der Entschlüsselung automatisch in ein Unterverzeichnis entpackt werden soll. Das Unterverzeichnis bekommt den Namen des ZIP-Archivs.
  - **`yes`**: (Default) Alle ZIP-Archive werden nach der Entschlüsselung automatisch in ein Unterverzeichnis entpackt.
  - **`no`**: Es erfolgt keine Prüfung ob es sich um ein ZIP-Archiv handelt. Je verschlüsselte Datei gibt es eine entschlüsselte Datei.
- **`signer.usepassword`:** Gibt an, ob eine verschlüsselte Datei mit einem Passwort entschlüsselt werden soll.
  - **`no`**: (Default) Die verschlüsselten Dateien werden mit einem Zertifikat entschlüsselt.
  - **`yes`**: Passwortverschlüsselte Dateien werden mit Passwort entschlüsselt.

3.1.4 Parameter Verschlüsselungs-Funktion

Pflichtparameter:

- **`signer.rcurl`**: siehe 3.1.1 Allgemeine Parameter
- **`signer.source`**: Dateiname oder URL der (Quell-) Datei. Alternativ kann auch ein Verzeichnis angegeben werden, aus dem alle vorhandenen Dateien (ohne Unterverzeichnisse) ausgewählt werden. Datei- und Verzeichnisnamen müssen immer voll qualifiziert angegeben werden. Im Fall einer URL wird von der zu signierenden Datei zunächst eine lokale Kopie angelegt. Der Name der lokalen Kopie kann der URL optional, durch ein Pipe-Zeichen getrennt, übergegeben werden. Beispiele:
  - Einzeldatei lokal: `/home/governikus/temp/source/test.pdf`
  - Verzeichnis lokal: `/home/governikus/temp/source/`
  - Einzeldatei URL: `<argument>signer.source</argument>`
  - **`lokaler_datei_name.pdf`**: Bei einer Quelle im Internet kann ein lokaler Dateiname mit angegeben werden, da nicht alle Internetquellen einen Dateinamen in der URL enthalten.

- **sha256-hash**: Zusätzlich kann ein Sha256-Hashwert der Datei im Aufruf mitgegeben werden. Wenn ein Hashwert mitgegeben wird, überprüft die WebEdition diesen mit der herunter geladenen Datei. Sollten beide Hashwerte nicht übereinstimmen, wird ein `ReturnCode 59 (HASHEDFILE_CHANGED)` an den Server gesendet und dem Benutzer im Fehlerdialog angezeigt. Mit dem Senden des Return-Codes wird der lokal errechnete Hashwert mit der Datei verknüpft.

Achtung: Die Übergabe eines Hashwerts ist nur bei einem Aufruf über eine URL möglich.

Hinweis: Sobald der Hashwert einer Serverdatei nicht mit dem lokal ermittelten Hashwert übereinstimmt, und der Benutzer diesen Vorgang aber trotz der Fehlermeldung nicht abbricht, wird keine weitere Fehlermeldung vor dem Signieren angezeigt.

Hinweis: Sobald eine Quelldatei als URL übergeben wurde, sollte als `targetFolderTyp` immer `oneSpecial` übergeben werden, da die Ergebnisdatei einer URL-Quelldatei beim Parameter `sameAsSource` im Temp-Verzeichnis gespeichert werden würde.

- **signer.serialnumber**: Seriennummer der zu verwendenden Signaturkarte oder des zu verwendenden Zertifikats.
  - Soll der Benutzer das Verschlüsselungszertifikat frei wählen, kann der Parameter `signer.serialnumber` bzw. `signer.certificate` entfallen.
  - **automatic**: Wird `automatic` angegeben, wird das Verschlüsselungszertifikat einer Signaturkarte automatisch ausgewählt, sofern die Karte beim Start der WebEdition bereits verfügbar ist und sich das Verschlüsselungszertifikat eindeutig bestimmen lässt. Zusammen mit dem Parameter `signer.processPolic == skipCompletedStep` kann so, bei eingelegter Karte, direkt auf den Dialogschritt "Verschlüsseln" gesprungen werden.
- **signer.certificate**: Voll qualifizierter Dateiname (URL oder Pfad) des zu verwendenden Software-Verschlüsselungszertifikates (`.cer` bzw. `.crt`-Datei).
- **signer.usepassword**: Gibt an, ob eine Datei mit einem Passwort verschlüsselt werden soll.
  - **no**: (Default) Die Dateien werden nicht mit einem Passwort verschlüsselt.
  - **yes**: Dateien werden mit Passwort verschlüsselt.
- **signer.createziparchivbeforeencrypt**: Gibt an, ob alle Dateien vor der Verschlüsselung automatisch in einem ZIP-Archiv zusammengefasst werden soll.
  - **yes**: (Default) Alle Dateien werden vor der Verschlüsselung in einem ZIP-Archiv zusammengefasst. Somit gibt es nur eine verschlüsselte Datei. Diese kann bei der Entschlüsselung auch wieder automatisch entpackt werden.
  - **no**: Alle Dateien werden einzeln verschlüsselt.

3.2 Aufruf der Governikus Signer Web Edition

Durch das Deployen der Datei `GovernikusSignerWebedition.war` in den Tomcat-Webserver wird für die Aufrufe der WebEdition die Datei `start.jsp` im Tomcat-`webapps`-Verzeichnis erstellt, die folgenden Inhalt hat.

```
<%@ page
info="WebEdition Config File"
%><%
String setupfile = request.getParameter("configfile");
response.reset();
response.setContentType("application/octet-stream");
response.setHeader("Content-Type", "application/force-download");
response.setHeader("Content-Transfer-Encoding", "binary");
response.setHeader("Content-Length", Integer.toString(setupfile.length()));
response.setHeader("Content-Disposition",                    "attachment;
filename=\"setup.gov-ws\"");%><%= setupfile%>
```

Listing 6: Inhalt der Datei `start.jsp`

Damit kann über eine URL die WebEdition mit der gewünschten Funktion aufgerufen werden.

```
http://<Servername>:<Port>/GovernikusSignerWebEdition/start.jsp?configfile=
http://<servername>:<Port>/GovernikusSignerWebEdition/testdata/sign.gov-ws
```

3.3 Steuerdatei der WebEdition

Für den Aufruf der Steuerdatei auf dem PC des Anwenders wird mit der aufrufenden URL oder über das Web-Portal eine Datei mit der Endung `.gov-ws` übergeben. Auf Windows Betriebssystemen wird die Dateiassoziation der Dateiendung `.gov-ws` mit der Steuerdatei automatisch bei der Installation erstellt.

Die Steuerdatei mit der Endung `.gov-ws`, die mit dem Aufruf der Steuerdatei übergeben wird, enthält die Parameter, die für die angeforderte Funktion (`sign`, `crypt` oder `decrypt`) benötigt werden. Beispiel:

```
signer.process=sign
signer.source = <Laufwerk/Serververzeichnis>:/ports.txt
#signer.configurationFile                                                    =
http://<Servername>:<Port>/GovernikusSignerWESample/configuration/configura
tion.xml
signer.dest = < Laufwerk/Serververzeichnis >:/
signer.targetFolderType = oneSpecial
signer.rcURL                                                                 =
http://<Servername>:<Port>/GovernikusSignerWESample/rcCode?transactionID=xx
.xx.xx.xx
signer.sigFormat = detachedPKCS7
signer.processPolicy = skipCompletedStep
signer.serialnumber = automatic
signer.pdfSignatureType = noPdfInline
signer.xmlSignatureType = noXmlInline
signer.enableDuplicateSignatures = yes
signer.enablePseudonymSignatures = yes
signer.createtimestamp = no
```

Listing 7: Beispiel für eine `.gov-ws` Datei

3.3.1 Java

Dadurch, dass die WebEdition als Programm auf dem Anwender-PC installiert ist, muss im Gegensatz zu früheren Versionen kein Java auf dem PC installiert sein oder aktualisiert werden.

3.4 Konfigurationsdatei

Der WebEdition wird eine Konfigurationsdatei übergeben. Diese Konfiguration enthält Einstellungen, die nicht als separate Parameter übergeben werden können:

- Mindestanzahl einzusehender Dateien
- Details zur erweiterten PDF-Signatur
- Zeitstempelserver

Ein kommentiertes Beispiel der Konfigurationsdatei ist weiter unter enthalten.

Hinweis: Die Einstellungen zu den Funktionen "Mindestanzahl einzusehender Dateien", "Erweiterte PDF-Signatur" und "Zeitstempelserver / Proxy" wirken nicht, wenn diese Funktionen über die Benutzerlizenzdatei deaktiviert sind (`disable` oder `hide_disable`).

3.4.1 Konfiguration für die Funktion Signieren

Mindestanzahl einzusehender Dateien

Es kann vorgegeben werden, ob das Einsehen von Dateien erforderlich ist und wie groß der Anteil der einzusehenden Dateien ist (Stichprobengröße). Das Signieren wird erst freigegeben, wenn der Benutzer den vorgegebenen Anteil an Dokumenten geöffnet hat.

Sichere Anzeige

Zu signierende Dateien können mit der sicheren Anzeige (siehe Benutzerhandbuch) oder mit der verknüpften Standardanwendung eingesehen werden. Die sichere Anzeige kann die Dateitypen Text (nur UTF8), TIFF, XML sowie PDF (nur Struktur inkl. Signaturen) vertrauenswürdig darstellen.

Spezielle Signaturformate

PAdES und CAdES sind die Weiterentwicklungen der bisherigen PDF-Inline bzw. PKCS#7 Signaturformate. Letztere sollte nur ausgewählt werden, wenn bei dem Signaturempfänger Kompatibilitätsprobleme bei mit den neueren Formaten existieren.

Details zur erweiterten PDF-Signatur

Hier kann die sichtbare PDF-Signatur aktiviert und konfiguriert werden. Details zu den Einstellungsmöglichkeiten entnehmen Sie bitte dem Benutzerhandbuch bzw. der kommentierten Beispiel-Konfigurationsdatei. Beachten Sie bitte folgende Einschränkungen:

- Damit eine sichtbare Signatur erzeugt wird, muss über die Einstellung `<visualisationTyp>` eine Visualisierungsart ausgewählt sein. Außerdem muss mindesten ein Textinhalt (Unterzeichnername, Ort, Datum oder Grund) vorhanden sein.

- Wenn die Größe der Textelemente die Größe des Visualisierungsbereichs überschreitet (vgl. Einstellung "Höhe", "Breite", "Grafik/Text" versus Schriftgröße), wird die Signaturerstellung abgebrochen. Aktivieren Sie zur Vermeidung von Abbrüchen bei z.B. langen Namen die Option `<useDynamicTextSize>`.
- Wird eine Grafik eingefügt, muss der Pfad, bzw. die URL, zur Grafik auch vom Client-PC des Benutzers erreichbar sein.

Signatur-Vorlagendateien können mit der Konfiguration nicht übergeben werden.

Zeitstempelserver

Als Zeitstempelserver kann nur ein Governikus System verwendet werden. Neben den typischen Angaben wie Host, Port, Pfad und einer ggf. erforderlichen Authentisierung, müssen die Angaben `systemID` und `opID` eingetragen werden. Diese Einstellungen werden durch den Administrator des Zeitstempelservers bereitgestellt.

3.4.2 Beispiel einer Konfigurationsdatei

Diese Konfigurationsdatei enthält sowohl einen Abschnitt für die Signaturerstellung, als auch für die Verschlüsselung. Die Datei ist auch in den Samples im Unterverzeichnis `configuration` der Datei `GovernikusSignerWESample.war` enthalten.

```
<?xml version="1.0" encoding="UTF-8"?><root>
  <process>
  <!-- Konfiguration Signaturerstellung (WebEdition) -->
    <sign>
      <!-- Einsehen von Dateien -->
      <fileVisualization>
        <!-- Einsehen von Dateien erzwingen ("true"|"false") -->
        <useVisualization>false</useVisualization>
        <!-- Anzahl einzusehender Dateien in Prozent ( 0 bis 100); nur
             wenn useVisualization=true -->
        <percent>0</percent>
      </fileVisualization>
      <!-- Sichere Anzeige zum Einsehen zu signierender Dateien verwenden
           (true|false) -->
      <useTrustedViewer>false</useTrustedViewer>
      <!-- Anstellen von "altem" PDF-Inline-Signaturformat PAdES
           verwenden ("true"|"false") (empfohlen) -->
      <usePAdES>true</usePAdES>
      <!-- Anstellen von "altem" CMS/PKCS#7-Signaturformat CAdES verwenden
           ("true"|"false") (empfohlen) -->
      <useCAdES>true</useCAdES>
      <!-- Einstellungen zur erweiterten bzw. sichtbaren PDF-Signatur -->
      <pdfSignatureExtention>
        <!-- Unterzeichnernamen aus Signaturzertifikat entnehmen
             ("true"|"false") -->
        <useSignatureIssuer>true</useSignatureIssuer>
        <!-- Unterzeichnernamen als Freitext oder leer, wenn kein Name
             angezeigt werden soll (nur wenn useSignatureIssuer=false) -->
        <signatureIssuer/>
        <!-- Ort als Freitext -->
        <signatureLocation/>
        <!-- Datum anzeigen ("true"|"false") und  Datumsformat (z.B.
             "dd.MM.yyyy", "MM/dd/yyyy", "EEEE, dd.MMMM.yyyy;
```

```
            EEEE=Wochentag, MMMM=Monat in Textform) -->
        <useDate>false</useDate>
        <dateFormat>dd.MM.yyyy kk:mm 'Uhr'</dateFormat>
        <!-- Dynamische("true") oder statische ("false") sichtbare Signatur
            erzeugen; nur wenn visualisationTyp!=NONE -->
        <useDynamicPdfSignature>true</useDynamicPdfSignature>
        <!-- Art der sichtb. Signatur: Nur Text ("TEXT"), Text und Grafik
            ("BOTH"), nur Grafik ("IMAGE"), keine ("NONE")  -->
        <visualisationTyp>TEXT</visualisationTyp>
        <!-- Position Grafik innerhalb der Signatur
            ("ImageLeft"|"ImageRight"|"ImageTop"|"ImageBottom") -->
        <visibleSignatureLayout>ImageLeft</visibleSignatureLayout>
        <!-- Signatur auf erster Seite ("1") oder letzter Seite ("-1") -->
        <page>1</page>

        <!-- Breite und Hoehe der Signatur in cm -->
        <visualWidth>5.0</visualWidth>
        <visualHeight>3.0</visualHeight>
        <!--  Signaturfeld für die sichtbare Signatur erstellen, falls
dieses noch nicht vorhanden  -->
        <!-- true | false  -->
        <createSignatureField>true</createSignatureField>
         <!-- Groessenverhaeltnis Grafik zu Text (1/2="0.33", 1/3="0.25",
            1/4="0.2" 2/1="0,66") -->
        <imageRatio>0.25</imageRatio>
        <!-- Position der Signatur, relativ zur Seitengroesse (0.0 bis 1.0,
            ausgehend von linker unterer Seitenecke)  -->
        <positionY>0.2</positionY>
        <positionX>0.4</positionX>
        <!-- PDF/A-Kompatibilitaet soll beibehalten werden ("true"|"false")
            -->
        <pdfACompatibility>true</pdfACompatibility>
        <!-- Schriftart ("COURIER"|"HELVETICA"|"TIMES"); wird ignoriert,
            wenn pdfACompatibility=true -->
        <textFont>TIMES</textFont>
        <!-- Schriftfarbe ("Black"|"Gray"|"Red"|"Green"|"Blue"|"Yellow");
            wird ignoriert, wenn pdfACompatibility=true -->
        <textColor>Black</textColor>
        <!-- Automatisches verkleinern der Schriftgroesse erlauben
            ("true"|"false") true empfohlen-->
        <useDynamicTextSize>true</useDynamicTextSize>
        <!-- Schriftgroesse -->
        <textSize>14</textSize>
        <!-- Textformatierung ("left"|"center"|"right") -->
        <textFormatting>center</textFormatting>
        <!-- Pfad oder URL zur Grafik (interne Grafiken:
            "certificate_128_white.gif" und "ballpen_128_white.gif" -->
        <imagefile>jar:http://localhost:8080/
        GovernikusSignerWEConfiguration/lib/SignProcess.jar!/resources
        /certificate_128_white.gif
        </imagefile>
      </pdfSignatureExtention>
      <!-- qualified | indetermined | advanced -->
      <signatureLevel>qualified</signatureLevel>
```

```
       <!-- Einstellungen Verbindung zum Zeitstempelserver -->
      <timestampserver>
        <!-- Vorgaben vom Governikus Zeitstempelserver -->
        <systemID/>
        <opID/>
        <!-- Hostname/IP-Adresse vom Zeitstempelserver -->
        <host/>
        <!-- Port vom Zeitstempelserver -->
        <port>80</port>
        <!-- Pfad uzm Zeitstempelservice (i.d.R.
             "/gov2core/services/Gov2CoreService") -->
        <path>/gov2core/services/Gov2CoreService</path>
      </timestampserver>
    </sign>
</process>
</root>
```

Listing 8: Konfigurationsdatei für die WebEdition

3.5 Return-Codes, Fehlerbehandlung Return-Codes, Fehlerbehandlung

Die WebEdition sendet jeden ReturnCode (RC), der während eines Prozessdurchlaufes durch einen Fehler bzw. durch das erfolgreiche verarbeiten einer Datei erstellt wird, an die als Aufrufparameter übergebene RC_URL. Damit kann die Fachanwendung jeden RC des Signiervorganges empfangen und auswerten.

Die Übergabe an die Fachanwendung erfolgt über einen GET und einen POST-Request. Die Übergabeparameter `returnCode`, `returnCodeDescription` und `param0` sind im POST-Request enthalten. Die Übergabe erfolgt zum Zeitpunkt des Auftretens des jeweiligen Ereignisses. Details entnehmen Sie bitte dem Quellcode des Beispiel-RCservlets aus den Samples.

Einige RCs können bei der Auswertung außer Acht gelassen werden. Wird z.B ein RC 50 `INVALID_PIN` gefolgt von einem RC 3 `FILE_PROCESSED` gesendet, so wurde die PIN beim zweiten Versuch richtig eingegeben und die Datei wurde erfolgreich signiert.

Wird jedoch der RC 50 von einem RC 54 `PIN_INPUT_CANCEL` gefolgt, so hat der Nutzer die erneute PIN-Eingabe abgebrochen und somit wurde die Datei nicht erfolgreich signiert. Es folgt auch kein RC 3 für diese und die folgenden Dateien. Für alle folgenden Dateien wird ein RC 5 `FILE_CANCELLED` gesendet.

Das Ende des gesamten Signiervorgangs (interessant bei der Verarbeitung von mehreren Dateien in einem Vorgang) wird immer durch Return-Code "0" oder "1" gekennzeichnet.

3.5.1 Standard ReturnCodes

Die nachfolgende Tabelle enthält alle im Produkt enthaltenen ReturnCodes. Dies schließt auch einige ReturnCodes mit ein, die projektbezogen nicht von Bedeutung sind (z. B. wenn lediglich Softwarezertifikate zur Verwendung kommen).

Return-Code	Belegung	Bedeutung	Error Level
0	`OK`	Der Vorgang {0} wurde erfolgreich	1

Return -Code	Belegung	Bedeutung	Error Level
		durchgeführt	
1	ABORTED	Der Vorgang wurde abgebrochen	4
3	FILE_PROCESSED	Die Datei {0} wurde erfolgreich verarbeitet	1
5	FILE_CANCELLED	Die Verarbeitung der Datei {0} wurde abgebrochen	1
10	MISSING_PARAM	Fehlender Parameter {0}	4
11	ILLEGAL_PARAM	Parameterfehler {0}	4
12	MISSING_UNLIMITED_STRENGTH_POLICY	Die Aktion kann nicht durchgeführt werden. Bitte beenden Sie die Anwendung und installieren Sie die Java-Erweiterung für starke Verschlüsselung (Java Cryptographic Extension).	2
20	FILE_NOT_FOUND	Die Datei {0} wurde nicht gefunden	3
21	FILE_IN_USE	Die Datei {0} kann nicht gespeichert werden, da sie von einer anderen Anwendung benutzt wird	4
22	SOURCE_DIR_EMPTY	Das Verzeichnis {0} ist leer	4
24	CONNECTION_ERROR	Fehler beim Verbinden zur Adresse: {0}	2
25	CANNOT_SAVE_OUTPUT	Die Ausgabedatei {0} kann nicht gespeichert werden unter {1}	3
26	CANNOT_WRITE_TO_FOLDER	Die Datei {0} kann nicht im Zielverzeichnis gespeichert werden. Bitte überprüfen Sie ihre Sicherheitseinstellungen!	3
27	CANNOT_UNZIP	Das ZIP-Archiv konnte nicht entpackt werden	3
28	CANNOT_CREATE_ZIP	Das ZIP-Archiv konnte nicht erstellt werden	3
29	FILE_IS_EMPTY	Die Datei {0} ist leer	3
30	CANNOT_READ_FOLDER	Die Ordner {0} hat keine Leseberechtigung	4
31	CANNOT_OPEN_SOURCE	Die Datei {0} konnte weder geöffnet noch gelesen werden	4
40	KEY_NOT_FOUND	Signaturschlüssel (SW-Zertifikat) {0} nicht vorhanden	4
41	KEY_EXPIRED	Der Gültigkeitszeitraum des ausgewählten Signaturzertifikats {0} ist überschritten. Das Signieren von Dateien ist nur mit einem gültigen Zertifikat möglich. Bitte wählen Sie ein gültiges Zertifikat aus.	2
42	KEY_NOT_YET_VALID	Der gewählte Signaturschlüssel {0} ist	2

Return -Code	Belegung	Bedeutung	Error Level
		noch nicht gültig Der Gültigkeitszeitraum des ausgewählten Signaturzertifikats {0} ist noch nicht erreicht Das Signieren von Dateien ist nur mit einem gültigen Zertifikat möglich. Bitte wählen Sie ein gültiges Zertifikat aus.	
43	CANNOT_LOAD_KEYSTORE	Der Signaturschlüssel konnte nicht geladen werden. Bitte prüfen Sie Ihre Einstellungen.	4
45	CARD_NOT_YET_INITIALIZED	Der gewählte Schlüssel wurde noch nicht freigeschaltet.	4
46	CARD_REMOVED	Während des Vorganges wurde entweder die Signaturkarte aus dem Kartenleser genommen oder der Kartenleser wurde vom Rechner getrennt.	4
47	WRONG_KEY	Die Datei konnte mit dem gewählten Schlüssel nicht entschlüsselt werden	4
50	INVALID_PIN	Die eingegebene PIN ist falsch. {0}.	3
51	TOO_MANY_WRONG_PINS	Der Fehlbedienungszähler der Signaturkarte ist abgelaufen	4
52	UNKNOWN_SIG_TYPE	Signaturformat wird nicht unterstützt	4
53	UNKNOWN_SIG_ALGORITHM	Nicht unterstützter Signaturalgorithmus {0}	4
54	PIN_INPUT_CANCEL	Die PIN-Eingabe wurde abgebrochen	4
55	SHOWNFILE_CHANGED	Die Datei {0} wurde nach dem letzten Zugriff in Ihrem Dateisystem verändert	4
57	UNSUPPORTED_SIG_ALGORITHM	Der von Ihnen gewählte Algorithmus wird durch die gewählte Signaturkarte nicht unterstützt. Bitte wählen Sie unter 'Einstellungen' einen anderen Algorithmus.	4
58	SUSPENDED	Die Karte ist im Suspended-Modus. Grund: Sie haben ihre PIN zweimal falsch eingegeben! Bevor Sie die PIN erneut eingeben müssen Sie vorher die CAN der Karte eingeben.	4
59	HASHEDFILE_CHANGED	Die Datei {0} wurde nach dem herunterladen verändert	4
76	NETSIGNER_SHA1_CHECK_ERROR	Das Zertifikat zur Authentisierung gegenüber dem Governikus NetSigner wurde nicht gefunden. Bitte prüfen Sie Ihre Einstellungen.	1

Return -Code	Belegung	Bedeutung	Error Level
81	UNSUPPORTED_PDF_DIALECT	Die PDF-Datei {0} enthält Elemente, die von der verwendeten Library "iText" (www.lowagie.com) nicht verarbeitet werden können.	3
82	PDF_SIGNATURE_VISUALISATION_TOO_BIG	Die sichtbare PDF-Signatur konnte nicht erzeugt werden. Der Text ist für das vorgegebene Textfeld zu groß. Sie sollten den Text verkürzen, eine kleine Schriftgröße wählen oder die Höhe bzw. Breite der Visualisierung vergrößern.	2
83	PDF_SIGNATURE_VISUALISATION_WIDTH_TOO_BIG	Die sichtbare PDF-Signatur konnte nicht erzeugt werden. Die Breite der Signatur soll nicht größer sein, als die Breite der PDF-Seite. Sie sollten eine kleine Signaturgröße wählen.	2
89	DUPLICATE_SIGNATURES_NOT_ALLOWED	Die Datei wurde bereits mit dem gleichen Signaturzertifikat signiert. Eine doppelte Signatur kann nicht durchgeführt werden.	3
99	UNKNOWN	Ein unbekannter Fehler ist aufgetreten. Der Vorgang wurde abgebrochen. Bitte wiederholen Sie den aktuellen Vorgang oder informieren Sie ihren Systemadministrator	3

Tabelle 2: Übersicht über die Return-Codes

4 Administration

In diesem Kapitel finden Sie Erklärungen zu den Themen SSL Verbindung, Server SSL-Zertifikate, SSL-Client-Authentication, Online-Hilfe, Anpassung der `provider.properties`, JAR-Datei erstellen und signieren und WebEdition deployen.

4.1 Systemanforderungen

Application-Server (Governikus Signer WebEdition)

Für den Application-Server, der die Governikus Signer WebEdition bereitstellt, gelten keine besonderen Auflagen hinsichtlich der Hard- und Software. Die Governikus KG als Application-Server den Einsatz eines Apache Tomcat Webservers in 8.5.

Application-Server (Fachanwendung)

Für den Application-Server, der die Kommunikation mit der WebEdition übernimmt, müssen folgende Punkte beachtet werden:

- Das URI-Encoding muss auf "UTF-8" eingestellt werden, falls Dateien mit Umlauten der WebEdition zum Download übergeben werden, siehe dazu 4.2 URI-Encoding.
- Das Dateisystem muss mit dem Zeichensatz "ISO-8859-1" eingestellt sein, damit das `FileSaveServlet` aus `GovernikusWebSigmerSample.war` die Dateien mit Umlauten richtig speichern kann.
- Für eine SSL-Verbindung mit dem Server siehe 4.3.

4.2 URI-Encoding

Um das URI-Encoding des Webservers einzustellen, muss in der Datei `conf/server.xml` des Webservers der HTTP- bzw. SSL-Connector um den Parameter `URIEncoding="UTF-8"` erweitert werden.

Tomcat conf/server.xml

```
<Connector port="8080" maxHttpHeaderSize="8192"  maxThreads="150"
minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
redirectPort="8443" acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true"  URIEncoding="UTF-8"/>
```

Hinweis: Diese Änderung beeinträchtigt den Betrieb eines auf demselben Server betriebenen Governikus Systems nicht. Um Auswirkungen auf andere Web-Anwendungen auszuschließen, kann auch, anstelle der Anpassung des bestehenden Connectors, ein weiterer Connector mit einem anderen Port (und einem anderem URI-Encoding) angelegt werden.

4.3 SSL Verbindung

Die WebEdition kann sich auch über SSL-Client-Authentication mit dem Server verbinden.

Achtung: Die SSL-Client-Authentication gegenüber dem Server muss mit offiziellen Zertifikat gesichert werden, z.B. vom Anbieter Verisign Inc. – also nur offizielle Zertifikate die von Java erkannt werden.

4.4 Governikus Signer WebEdition deployen

Die WAR-Datei `GovernikusSignerWebEdition.war` muss in das Deploy-Verzeichnis des Webservers kopiert werden. Das Deploy-Verzeichnis ist bei den unterstützten Webservern in den folgenden Unterverzeichnissen zu finden.

JBoss

`server/standalone/deployments/`

Tomcat

`webapps/`

Sobald der Webserver gestartet wurde, wird der Governikus Signer WebEdition unter der Adresse `http://hostname:Portnummer/GovernikusSignerEdition` zur Verfügung gestellt.

Um die Beispiele zu nutzen, muss die WAR-Datei `GovernikusSignerWESamples.war` ebenfalls deployed werden. Der Vorgang ist analog zu oben. Unter der Adresse `http://hostname:Portnummer/GovernikusSignerWESample` steht dann die Startseite zur Verfügung.

Hinweis: Wenn Sie eine bereits installierte Version aktualisieren möchten, löschen Sie die alte Version vom Webserver, bevor Sie die neue Version aufspielen. Achten Sie darauf, dass beim Entfernen der .war-Datei die zugehörige Webanwendung komplett vom Webserver gelöscht wurde.

5 Auflagen für den Betrieb gemäß Signaturgesetz

Um auch in der praktischen Verwendung der Software Governikus Signer WebEdition den Ansprüchen des Signaturgesetzes gerecht zu werden, sind hinsichtlich des Betriebes besondere Anforderungen zu gewährleisten. Der vorliegende Abschnitt dieses Handbuchs erläutert Ihnen die spezifischen Anforderungen und relevanten Sicherheitsfunktionen.

Die Anforderungen nehmen u. a. Bezug auf die zugrundeliegenden Rollen beim Betrieb der Software. Nachfolgend wird daher zunächst noch einmal das Rollenmodell erläutert.

	**Hinweis:** Weitere Informationen zum Signaturgesetz, qualifizierten elektronischen Signaturen und den zugrunde liegenden kryptografischen Hintergründen finden Sie auch auf den Seiten der Bundesnetzagentur.

5.1 Sicherheitsfunktionen

Das Produkt stellt die Sicherheitsfunktionen "Unterstützung bei der Erzeugung von qualifizierten elektronischen Signaturen" zur Verfügung.

5.2 Rollen

Zur Umsetzung der Anforderungen sind folgende Rollen im Rahmen der WebEdition definiert:

- **Administrator:** Ein Administrator ist für die Verwaltung und Organisation der grundlegenden IT-Infrastruktur zuständig, die für die WebEdition benötigt wird. Der System-Administrator kann beispielsweise ein Administrator bei einem Dienstanbieter sein, der die Systeme hostet. Typische Aktivitäten - mit dedizierter Rechtebeschränkung und Protokollierung - des System-Administrators sind:
  - Konfiguration, Betriebsüberwachung und Sicherung von Servern und Betriebssystem
  - Konfiguration und Betriebsüberwachung der Netzwerkkomponenten
- **Nutzer/Benutzer:** Anwender der Software von einem Arbeitsplatz aus.

Diese Nutzer entsprechen im SigG-konformen Betrieb natürlichen Personen.

5.3 Technische Anforderungen

Für den signaturgesetz-konformen Betrieb benötigen Sie folgende technische Ausstattung:

- Hard- und Software: Die für den SigG-konformen Betrieb geeigneten Umgebungen sind entsprechend gekennzeichnet. Die für den Betrieb der WebEdition unterstützte Hard- und Software ist in diesem im Handbuch im Kapitel "Systemanforderungen" beschrieben.

5.4 Anforderungen an die Einsatzumgebung

Folgende räumlichen Gegebenheiten für den Server sind zu gewährleisten:

- Die Anforderungen an einen "geschützten Einsatzbereich (Regelfall/Standardlösung)" werden umgesetzt, um "potentielle Angriffen über das Internet, ein angeschlossenes Intranet, einen manuellen Zugriff Unbefugter und einen Datenaustausch per Datenträger [...] durch eine Kombination von Sicherheitsvorkehrungen in der Signaturanwendungskomponente selbst und der Einsatzumgebung mit hoher Sicherheit" [BNetzA2005] hinsichtlich eines hohen Angriffspotenzials abzuwehren:
  - Auflagen zur Anbindung an ein Netzwerk:

    Netzwerkverbindungen sind so abzusichern, dass Angriffe erkannt bzw. unterbunden werden - z. B. durch eine geeignet konfigurierte Firewall, geeignete Absicherung der Kommunikationsstrecke zwischen Client und Server und durch die Verwendung geeigneter Anti-Viren-Programme.
  - Die Verbindung zwischen dem Client (PC des Benutzers) und Server ist über das Protokoll HTTPS zu realisieren. Wenn die Anwendung im Internet zur Verfügung gestellt wird, muss auch die Web-Seite, von der die Anwendung durch den Benutzer aufgerufen wird, ausschließlich über HTTPS bereitgestellt werden, um die Vertrauenswürdigkeit der Anwendung gegenüber dem Benutzer zu gewährleistet.
  - Auflagen zur Sicherheit der IT-Plattform und Programme:

    Es ist zu gewährleisten, dass von der Hardware, auf der die WebEdition bereitgestellt wird, keine Angriffe ausgehen. Insbesondere ist sicherzustellen, dass die auf dem eingesetzten Computer installierte Software nicht böswillig manipuliert oder verändert werden kann, auf dem Computer keine Viren oder trojanischen Pferde eingespielt werden können, die Hardware des Computers nicht unzulässig verändert werden kann.
- Auflagen zum Schutz vor manuellem Zugriff Unbefugter und Datenaustausch per Datenträger: Es werden die folgenden baulichen, personellen und organisatorischen Anforderungen umgesetzt:
  - Der Server, auf dem die WebEdition bereitgestellt wird, befindet sich in einem Betriebsraum. Es muss Sorge dafür getragen werden, dass ein Zugriff Unbefugter ausgeschlossen ist oder zumindest mit hoher Sicherheit erkennbar wird - beispielsweise durch ein Sperren des Bildschirms oder Verschließen des Raumes bei Abwesenheit
  - Beim Übertragen von Daten, die auf Datenträgern vorliegen muss - z. B. durch die Verwendung geeigneter Anti-Viren-Programme - sichergestellt werden, dass keine Viren oder trojanische Pferde übertragen werden können

5.5 Anforderungen an den sicheren Betrieb

Auflagen an alle hinsichtlich der Güte der Passwörter:

- Bitte nutzen Sie gute Passworte, d. h.
  - kein Trivialpasswort (z. B. "BBBBBBBB" oder "12345678"),
  - Passwort mit mindestens einem Zeichen pro Passwort, das kein Buchstabe ist (Sonderzeichen oder Zahl),
  - Passwort, das mindestens 8 Zeichen lang ist.

Das System unterstützt Sie hierbei.

- Passwörter sind geheim zu halten: Stellen Sie sicher, dass niemand Ihr Passwort entdeckt.
- Für die Administratoren müssen Vertreterregelungen für Krankheit und Urlaub bestehen.
- Die Konfiguration der Software erfolgt über ein 4-Augen-Prinzip unter Aufsicht des IT-Sicherheits- oder Datenschutzbeauftragten, der jede Konfiguration explizit autorisieren muss, was dokumentiert wird. Der IT-Sicherheits- oder Datenschutzbeauftragte führt regelmäßige Prüfungen der Konfiguration durch.
- Vor der Installation der Software ist die Integrität des Installationspakets über einen Vergleich eines vor Ort erstellten Hashwerts mit dem durch die Governikus KG veröffentlichten Hashwert zu prüfen.

6 Verzeichnisse

Abbildungen

Listings

Tabellen