Einrichtungsfragen beantwortet Ihnen gerne unser Service-Team unter 01806 – Signatur (74462887)*

* 0,20 € pro Anruf aus dem deutschen Festnetz - Mobilfunkpreise können abweichen

Mo.- Fr. 08.00 Uhr bis 17.00 Uhr
E- Mail: support@mentana.de

Vielen Dank, dass Sie sich für die Lösungen auf www.signaturportal.de interessieren.

Dieses Dokument beschreibt die technischen Möglichkeiten des Zugriffs auf die Services von www.signaturportal.de. In diesem PDF- Dokument finden Sie auch alle verfügbaren Links auf Beispiele und freie Sourcen die wir bereitstellen um den Dienst zu nutzen.

Die Nutzung der Funktionen rund um die qualifizierte elektronische Signatur kann von OEM- Partnern der Mentana-Claimsoft GmbH kostenlos genutzt werden, um eigene Applikationen mit entsprechenden Funktionen zu ergänzen oder eigene ASP- Dienste anzubieten. Um OEM Partner von www.signaturportal.de zu werden, ist lediglich ein kostenloser und formloser Antrag per E-Mail erforderlich (support@mentana.de). Als OEM Partner erhalten Sie spezielle ePorto- Konditionen für Ihre Kunden und eine Handelspanne auf die ePorto-Umsätze Ihrer Kunden.

Inhaltsverzeichnis

1 Welche Services gibt es?

Es gibt 2 technisch unterschiedliche Lösungsmöglichkeiten die auf signaturportal.de angeboten werden.

Entweder Sie nutzen einen **SOAP Webservices** oder Sie verwenden die **SMTP- Diensteplattform** „sigmail“

1.1 Welche Funktionen bieten die SOAP Webservices?

	Funktion	Kurzbeschreibung
1.	**Signatur**	Anfordern von Signaturen gemäß § 2 Nr. 3 SigG, § 14 Abs. 3 UStG
2.	**Verifikation**	Verifikationsanzeige für Signaturen mit XML- Ergebnis Liste gemäß § 15 UstG
3.	**Qual. Verifikation**	Qualifizierte Verifikation mit Ergebnisprotokoll in signiertem PDF Dokument incl. Zeitstempel gemäß § 2 Nr. 14 SigG (GDPdU/GoBS)
4.	**Zeitstempel**	Anfordern von Zeitstempeln gemäß § 2 Nr. 14 SigG für beliebige Dateien (Archiv).
5.	**Überweisung**	Überweisen von ePorto zwischen Konten (z.B. um 2 Tochtergesellschaften mit verschiedenen Konten über 1 ePorto Kontingent zu versorgen)
6.	**Registrierung**	Registrierung eines Accounts auf Sinaturportal.de aus der eigenen Applikation heraus. Eine Kundenanmeldung über das WebFrontend von signaturportal.de kann entfallen.

1.2 Welche Funktionen bieten die SMTP- Diensteplattform?

Die SMTP- Diensteplattform „sigmail“ ermöglicht den Versand von E-Mails über SMTP unter Nutzung spezieller Funktionen wie Signatur oder Verifikation von Attachments nach deutschem Recht (SigG) oder dem Recht für 27 europäische Staaten und der Schweiz.

2 II. Beschreibung SMTP- Diensteplattform

2.1 Signaturservice

So kann ein PDF- Dokument welches einer E- Mail als Attachement beigefügt wurde (Rechnung), direkt vom SMTP- Server beim **Versand** an den Empfänger gemäß § 14 Abs. 3 UStG, § 2 Nr.3 SigG signiert werden. Über die Optionseinstellungen auf www.signaturportal.de können Sie zusätzlich folgende Funktionen für diesen Service veranlassen:

a) **Kopie der signierten Rechnung in ein IMAP- Verzeichnis legen**

b) **Kopie der signierten Rechnung an Referenzadresse senden**

c) **Ihre gewohnte E-Mailadresse als „reply to“ eintragen (Antwort-E-Mailadresse)**

d) Verifikationsprotokoll für Empfänger erzeugen und Mitversenden (1 ePorto)

e) Einen Proxy einzurichten um die eigene Absender- Domaine zu nutzen.

f) Ablage der signierten Belege (Ausgangsrechnungen) in einem Archiv

2.2 Verifikationsservice

Auf der SMTP- Diensteplattform „sigmail“ werden POP3/IMAP Postfächer angelegt über die ebenfalls spezielle Funktionen für den Posteingang genutzt werden können.

So kann ein PDF- Dokument welches einer E- Mail als Attachement beigefügt wurde (Rechnung), direkt vom SMTP- Server beim **Empfang** auf dem Postfach „Kunde@sigmail.de“ gemäß § 15 UStG verifiziert werden und es wird ein GDPdU/GOBS- konformes Protokoll unter Anforderung eines Zeitstempels gemäß § 2 Nr.14 SigG erzeugt und als **2** Attachment der E-Mail hinzugefügt, sodass beim Empfang auf dem Mail- Client bereits alle steuerrechtlichen Verpflichtungen erfüllt sind.

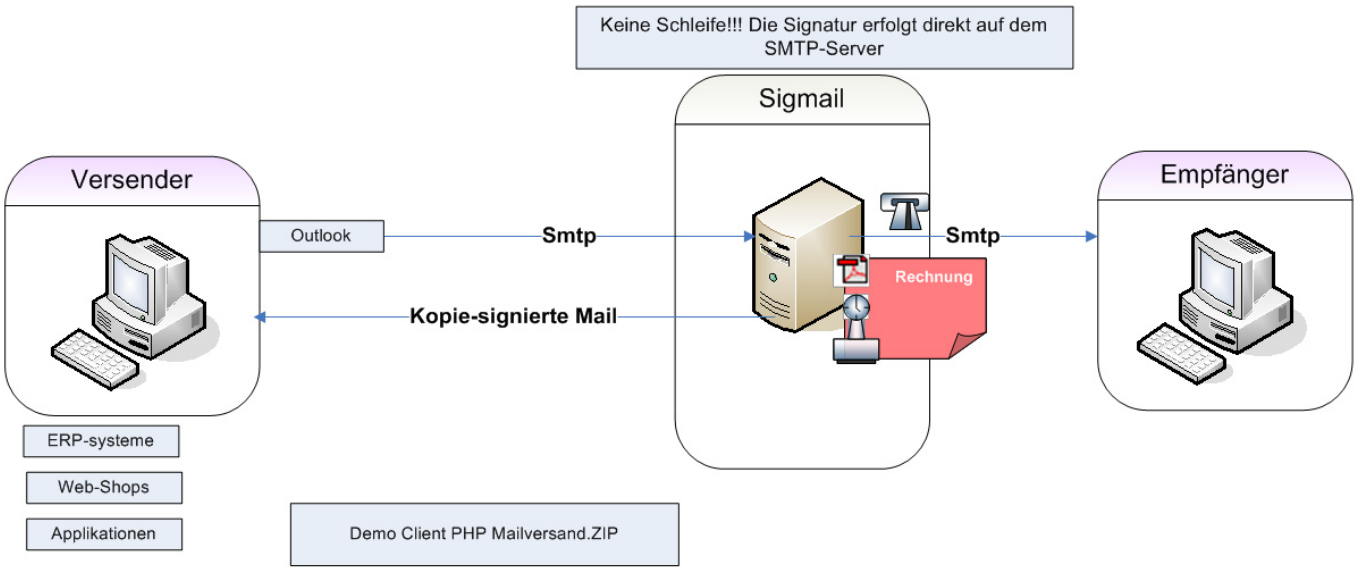
Über die Optionseinstellungen auf www.signaturportal.de können Sie zusätzlich folgende Funktionen für diesen Service veranlassen:

a) Verarbeitung von ZIP- Archiven (z.B. Telekom, 1&1 etc.)

b) Forwarding des Posteingangs auf Referenzadresse oder eine Alternativadresse

c) Ablage einer Kopie der Verifizierten Rechnungen in ein Archiv

2.3 Ablaufschema für SMTP- Diensteplattform „sigmail“ (Signatur/Verifikation/Archiv)



2.4 Technische Daten SMTP- Diensteplattform „sigmail“

Sofern Ihre Applikation bereits die Einrichtung eines SMTP E-Mail Kontos unterstützt geben Sie bitte folgende Daten ein:

Sofern Sie zunächst die **Erzeugung** einer E-Mail aus der Applikation umsetzen müssen gehen Sie wie folgt vor:
Um den SMTP- Signaturserver zu nutzen, müssen Sie eine SMTP Nachricht erstellen, die bestimmte Einträge enthalten muss.
Nutzen Sie unsere Beispiele:
www.signaturportal.de/freware/ebilling-signature-php-mail-script.zip

Bitte beachten Sie, dass wir **nur noch SSL gesicherte SMTP- Verbindungen zulassen.**
Sie müssen sich zunächst auf www.signaturportal.de anmelden und die sigmail Benutzerdaten in das Skript übernehmen! Richten Sie bitte zunächst einen Standard-Client ein um Fehlerquellen bei der Account-Einrichtung auszuschließen.

3 Beschreibung SOAP Webservices

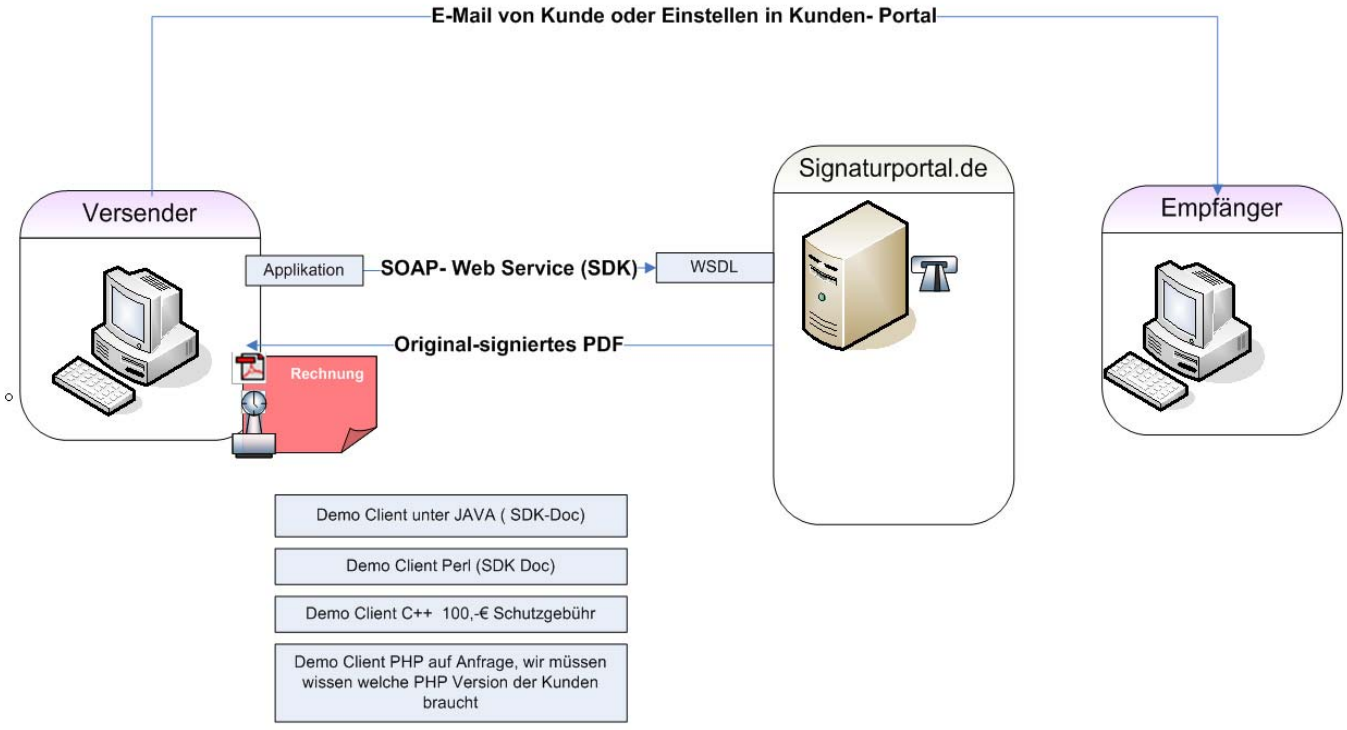
3.1 Allgemeine Spezifikation der SOAP-Webservices

Die Services werden auf Basis des offenen Standards Simple Object Access Protocol – „SOAP“ (siehe http://www.w3.org/TR/SOAP/), sowie Web-Service-Description-Language – „WSDL“ (siehe http://www.w3.org/TR/wsdl) organisiert. SOAP ist ein Transport-Protokoll, das beschreibt, wie Web-Services ihre Nachrichten über das Internet verschicken. Das Protokoll ist unabhängig von den Inhalten der Nachrichten oder der genutzten Entwicklungsumgebung und Programmiersprache. Der WSDL-Standard ist für die Beschreibung der Inhalte der Nachrichten verantwortlich. Diese Definition wird in einer XML-Datei beschrieben, die für die Implementierung in den Client-Anwendungen herangezogen wird.

Die WSDL-Datei für den Signatur-Service nach deutschem Recht (SigG) oder dem Recht für 27 europäische Staaten und der Schweiz. **(Funktionen 1 bis 12)** erreichen Sie unter:
https://www.signaturportal.de/wsdl/smmi_basic.wsdl

Die Beschreibung und die URL für den **Registrierungs-Service (Funktion 6)** senden wir Ihnen nach telefonischer Kontaktaufnahmen und Registrierung zu. Bitte melden Sie sich **unter 01806-Signatur (74462887)**

3.2 Ablaufschema für SOAP Webservices



	Methode	Kurzbeschreibung
1.1	test	Methode zur Durchführung eines Verbindungstests mit dem Service von Client aus
1.2	test_param	Methode zur Durchführung eines Verbindungstests mit dem Anmeldedaten zum Service von Client aus
1.3	sign	Anfordern einer qualifizierten Signatur nach §2 Nr. 3 SigG als PDF- embedded signature **Eingangsformat:** PDF 1.4 -1.6 **Antwort:** PDF 1.4 -1.6 mit embedded signature
1.4	sign_extern	Anfordern einer qualifizierten Signatur nach §2 Nr. 3 SigG als CMS/ PKC#7 Struktur **Eingangsformat:** beliebiges File **Antwort:** CMS/ PKC#7 Struktur
1.5	verify	Anfordern eines Verifikationsprotokolls: **Eingangsformat:** PDF 1.4 -1.6 mit embedded signature **Antwort:** Verifikationsprotokoll als PDF-File
1.6	verify_xml	Anfordern eines Verifikationsprotokolls: **Eingangsformat:** PDF 1.4 -1.6 mit embedded signature **Antwort:** Verifikationsprotokoll als XML-Dokument
1.7	verify_extern	Anfordern eines Verifikationsprotokolls: **Eingangsformat:** Beliebiges File + CMS/ PKC#7 Struktur **Antwort:** Verifikationsprotokoll als PDF-File.
1.8	verify_extern_xml	Anfordern eines Verifikationsprotokolls: **Eingangsformat:** Beliebiges File + CMS/ PKC#7 Struktur **Antwort:** Verifikationsprotokoll als XML-File.
1.9	balance	Methode zum abfragen des ePorto- Kontostandes
2.0	transfer	Methode zum Überweisen von ePorto zwischen Accounts
2.1	timestamp	Methode zum Anfordern eines Zeitstempels §2 Nr. 14 SigG,mit SHA 256/ mind. RSA 2048 durch Übergabe der kompletten Datei. **Eingangsformat:** Beliebiges File **Antwort:** Timestamp.tsr
2.2	sha256_timestamp	Methode zum Anfordern eines Zeitstempels §2 Nr. 14 SigG,mit SHA 256/ mind. RSA 2048 durch Übergabe des **Hashwerts** einer Datei. Eingangsformat: SHA 256 String Antwort: Timestamp.tsr.

3.3.1 Die Methode „test“ hat folgenden Aufbau:

```
<wsdl:message name="testRequest" />
 <wsdl:message name="testResponse">
 <wsdl:part name="testReturn" type="xsd:string">
 <wsdl:documentation>Returns a string</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
```

Der Methoden-Aufruf **„testRequest“** erfolgt ohne Angabe weiterer Parameter.
Der Response **<part name="testReturn" type="xsd:string"/>** enthält einen xsd- string mit folgendem Inhalt:

```
" Test funktioniert ~:)"
```

3.3.2 Die Methode „test_param“ hat folgenden Aufbau:

```
<wsdl:message name="test_paramRequest">
- <wsdl:part name="text" type="xsd:string">
 <wsdl:documentation>Test text you send to the service.</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
- <wsdl:message name="test_paramResponse">
- <wsdl:part name="test_paramReturn" type="xsd:string">
 <wsdl:documentation>Returns some response text</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
```

Der Methoden-Aufruf **„test_param“** erfolgt unter Angabe folgender Parameter und enthält:
$text :: = Hier kann ein Freier Text eingegeben werden.
Der Response **<part name="test_paramReturn" type="xsd:string"/>** enthält ein xsd- string mit folgendem Aufbau:

```
<enthält den Frei-Text aus dem Request>
```

3.3.3 Die Methode „sign“ hat folgenden Aufbau:

```
 <wsdl:message name="signRequest">
- <wsdl:part name="username" type="xsd:string">
  <wsdl:documentation>Your UserName</wsdl:documentation>
  </wsdl:part>
- <wsdl:part name="passwort" type="xsd:string">
  <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
  </wsdl:part>
- - <wsdl:part name="kontoNr" type="xsd:int">
  <wsdl:documentation>Your Konto Number</wsdl:documentation>
  </wsdl:part>
- <wsdl:part name="dateiname" type="xsd:string">
  <wsdl:documentation>The name of the file you like to put the signature on.</wsdl:documentation>
  </wsdl:part>
- <wsdl:part name="data" type="xsd:base64Binary">
  <wsdl:documentation>Your file as a binary encoded with base64#</wsdl:documentation>
  </wsdl:part>
-<wsdl:part name="region" type="xsd:string">
–<wsdl:documentation>Optional your signature card region (two-digit ISO Code)</wsdl:documentation>
  </wsdl:part>
–<wsdl:part name="lang" type="xsd:string">
–<wsdl:documentation>Optional the language for your signature description (two-digit ISO Code)</wsdl:documentation>
  </wsdl:part>
  </wsdl:message>
– <wsdl:message name="signResponse">
– <wsdl:part name="signReturn" type="xsd:base64Binary">
  <wsdl:documentation>Returns your file (in binary mode encoded with base64) with the signature</wsdl:documentation>
  </wsdl:part>
</wsdl:message>
```

Der Methoden-Aufruf **„signRequest“** erfolgt unter Angabe folgender Parameter und enthält:

$username :: = der Username auf Signaturportal.de
$passwort :: = das Gateway-Passwort (nicht das Login-Passwort!)
$kontoNr :: = die Signaturportal.de Kontonummer (z. B. 1010101010)
$dateiname :: = Dateiname der zu signierenden Datei unter Angabe der Datei- Endung „.pdf“
$data :: = die PDF-Datei als base64Binary
$region :: = der 2 Zeichen ISO-Code des Landes

$lang :: = der 2 Zeichen ISO-Code der Sprache
Der Response **<part name=" signReturn" Type="xsd:base64Binary"/>** enthält entweder die Rückgabe response = 200 dann Fehler. Prozess nicht erfolgreich
oder es wird ein **xsd:base64Binary** zurück geliefert der das PDF File („xxxx.pdf“) enthält.

3.3.4 Die Methode „sign_extern“ hat folgenden Aufbau:

```
<wsdl:message name="sign_externRequest">
−<wsdl:part name="username" type="xsd:string">
 <wsdl:documentation>Your UserName</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="passwort" type="xsd:string">
 <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="kontoNr" type="xsd:int">
 <wsdl:documentation>Your Konto Number</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="dateiname" type="xsd:string">
−<wsdl:documentation>The name of the file you like to put the signature on.</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="data" type="xsd:base64Binary">
 <wsdl:documentation>Your file as a binary encoded with base64#</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="region" type="xsd:string">
−<wsdl:documentation>Optional your signature card region (two-digit ISO Code)</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="lang" type="xsd:string">
−<wsdl:documentation>Optional the language for your signature description (two-digit ISO Code)</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
−<wsdl:message name="sign_externResponse">
−<wsdl:part name="sign_externReturn" type="xsd:base64Binary">
−<wsdl:documentation>Returns the extern signature of your file (in binary mode encoded with
base64)</wsdl:documentation>
 </wsdl:part>
</wsdl:message>
```

Der Methoden-Aufruf **„signRequest“** erfolgt unter Angabe folgender Parameter und enthält:
$username :: = der Username auf Signaturportal.de
$passwort :: = das Gateway-Passwort (nicht das Login-Passwort!)
$kontoNr :: = die Kontonummer von signaturportal.de (z. B. 1010101010)
$dateiname :: = Dateiname der zu signierenden Datei

$data :: = die Signatur-Datei als base64Binary
$region :: = der 2 Zeichen ISO-Code des Landes
$lang :: = der 2 Zeichen ISO-Code der Sprache

Der Response **<part name=" signReturn" Type="xsd:base64Binary"/>** enthält entweder die Rückgabe response = 200 dann Fehler. Prozess nicht erfolgreich
oder es wird ein **xsd:base64Binary** zurück geliefert der das PKC#7 File („xxxx.p7s/PkcS/etc.“) enthält.

3.3.5 Die Methode „verify“ hat folgenden Aufbau:

```
<wsdl:message name="verifyRequest">
- <wsdl:part name="username" type="xsd:string">
 <wsdl:documentation>Your UserName</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="passwort" type="xsd:string">
 <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="kontoNr" type="xsd:int">
 <wsdl:documentation>Your Konto Number</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="dateiname" type="xsd:string">
 <wsdl:documentation>The name of the file you like to verify.</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="data" type="xsd:base64Binary">
 <wsdl:documentation>Your file as a binary encoded with base64</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="region" type="xsd:string">
−<wsdl:documentation>Optional your signature card region (two-digit ISO Code)</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="lang" type="xsd:string">
−<wsdl:documentation>Optional the language for your signature description (two-digit ISO
Code)</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
- <wsdl:message name="verifyResponse">
- <wsdl:part name="verifyReturn" type="xsd:base64Binary">
 <wsdl:documentation>Returns a file (in binary mode encoded with base64) with the result of the
verification.</wsdl:documentation>
 </wsdl:part>
```

Der Methoden-Aufruf **„verifyRequest“** erfolgt unter Angabe folgender Parameter und enthält:

$username :: = der Username auf Signaturportal.de
$passwort :: = das Gatewaypaswort (nicht das Loginpasswort!)
$kontoNr :: = die Signaturportal.de Kontonummer (z. B. 1010101010)
$dateiname :: = Dateiname der zu signierenden Datei unter Angabe der Datei- Endung “.pdf“
$data :: = die PDF-Datei als base64Binary
$region :: = der 2 Zeichen ISO-Code des Landes
$lang :: = der 2 Zeichen ISO-Code der Sprache

Der Response **<part name=" verifyReturn" Type="xsd:base64Binary"/>** enthält entweder die Rückgabe response = 200 dann Fehler. Prozess nicht erfolgreich
oder es wird ein **xsd:base64Binary** zurück geliefert der das PDF File („xxxx_verifikationsprotokoll.pdf“) enthält.

3.3.6 Die Methode „verify_xml“ hat folgenden Aufbau:

Siehe 2.5. Response liefert im fehlerfreien Fall ein **xsd:base64Binary** zurück, welches ein XML File („xxxx_verifikationsprotokoll.xml“) enthält.

3.3.7 Die Methode „verify_extern“ hat folgenden Aufbau:

```
<wsdl:message name="verify_externRequest">
−<wsdl:part name="username" type="xsd:string">
 <wsdl:documentation>Your UserName</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="passwort" type="xsd:string">
 <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="kontoNr" type="xsd:int">
 <wsdl:documentation>Your Konto Number</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="dateiname" type="xsd:string">
 <wsdl:documentation>The name of the file you like to verify.</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="data" type="xsd:base64Binary">
 <wsdl:documentation>Your file as a binary encoded with base64</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="sigdateiname" type="xsd:string">
 <wsdl:documentation>The name of the extern signature file.</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="sigdata" type="xsd:base64Binary">
−<wsdl:documentation> Your extern signature file as a binary encoded with base64 </wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="region" type="xsd:string">
−<wsdl:documentation>Optional your signature card region (two-digit ISO Code)</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="lang" type="xsd:string">
−<wsdl:documentation>Optional the language for your signature description (two-digit ISO
Code)</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
−<wsdl:message name="verify_externResponse">
−<wsdl:part name="verify_externReturn" type="xsd:base64Binary">
−<wsdl:documentation> Returns a file (in binary mode encoded with base64) with the result of the verification.
</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
```

Der Methoden-Aufruf „verify_externRequest“ erfolgt unter Angabe folgender Parameter und enthält:

$username :: = der Username auf Signaturportal.de
$passwort :: = das Gateway-Passwort (nicht das Login-Passwort!)
$kontoNr :: = die Signaturportal.de Kontonummer (z. B. 1010101010)
$dateiname :: = Dateiname der zu signierenden Datei
$data :: = die PDF-Datei als base64Binary
$sigdateiname :: = Dateiname der Signatur-Datei
$sigdata :: = die Signatur-Datei als base64Binary
$region :: = der 2 Zeichen ISO-Code des Landes
$lang :: = der 2 Zeichen ISO-Code der Sprache

Der Response <part name=" verifyReturn" Type="xsd:base64Binary"/> enthält entweder die Rückgabe response = 200 dann Fehler. Prozess nicht erfolgreich
oder es wird ein xsd:base64Binary zurück geliefert der das PDF File („xxxx_verifikationsprotokoll.pdf“) enthält.

3.3.7 *Die Methode „verify_extern_xml“ hat folgenden Aufbau:*

Siehe 1.7, Response liefert im fehlerfreien Fall ein **xsd:base64Binary** zurück, welches ein XML File („xxxx_verifikationsprotokoll.xml“) enthält.

3.3.8 *Die Methode „balance“ hat folgenden Aufbau:*

```
<wsdl:message name="balanceRequest">
- <wsdl:part name="username" type="xsd:string">
 <wsdl:documentation>Your UserName</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="passwort" type="xsd:string">
 <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="kontoNr" type="xsd:int">
 <wsdl:documentation>Your Konto Number</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
- <wsdl:message name="balanceResponse">
- <wsdl:part name="balanceReturn" type="xsd:int">
 <wsdl:documentation>Returns your current amount of eporto on your account.</wsdl:documentation>
 </wsdl:part>
```

Der Methoden-Aufruf **„balanceRequest“** erfolgt unter Angabe folgender Parameter und enthält:

$username :: = der Username auf Signaturportal.de
$passwort :: = das Gateway-Passwort (nicht das Login-Passwort!)
$kontoNr :: = die Signaturportal.de Kontonummer (z. B. 1010101010)

Der Response **<part name=" balanceReturn" Type="xsd:int"/>** enthält den Kontostand als positiven oder negativen Wert

3.3.9 Die Methode „transfer“ hat folgenden Aufbau:

```
<wsdl:message name="transferRequest">
- <wsdl:part name="username" type="xsd:string">
 <wsdl:documentation>Your UserName</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="passwort" type="xsd:string">
 <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="kontoNr" type="xsd:int">
 <wsdl:documentation>Your Konto Number</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="targetkontoNr" type="xsd:int">
 <wsdl:documentation>The kontoNr of the account you like to transfer eporto. Only positive amount is
allowed.</wsdl:documentation>
 </wsdl:part>
- <wsdl:part name="amount" type="xsd:unsignedInt">
 <wsdl:documentation>The amount you like to transfer.</wsdl:documentation>
 </wsdl:part>
 </wsdl:message>
- <wsdl:message name="transferResponse">
- <wsdl:part name="transferReturn" type="xsd:int">
 <wsdl:documentation>Returns the amount that was successful transferd.</wsdl:documentation>
 </wsdl:part>
```

Der Methoden-Aufruf **„transferRequest“** erfolgt unter Angabe folgender Parameter und enthält:

$username	:: = der Username auf Signaturportal.de
$passwort	:: = das Gateway-Passwort (nicht das Login-Passwort!)
$kontoNr	:: = die Signaturportal.de Kontonummer (z. B. 1010101010)
$targetkontoNr	:: = die Signaturportal.de Kontonummer an welche die Transaktion in ePorto überwiesen werden soll.
$amount	:: = die Anzahl der zu überweisenden ePortos

Der Response **<part name=" transferReturn" Type="xsd:int"/>** enthält
„100“ für OK oder „ xxx“ für einen Errorcode

3.3.10 Die Methode „timestamp“ hat folgenden Aufbau:

```
<wsdl:message name="timestampRequest">
−<wsdl:part name="username" type="xsd:string">
 <wsdl:documentation>Your UserName</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="passwort" type="xsd:string">
 <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="kontoNr" type="xsd:int">
 <wsdl:documentation>Your Konto Number</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="dateiname" type="xsd:string">
−<wsdl:documentation>The name of the file you like to get the timestamp for.</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="data" type="xsd:base64Binary">
 <wsdl:documentation>Your file as a binary encoded with base64#</wsdl:documentation>
 </wsdl:part>
</wsdl:message>
−<wsdl:message name="timestampResponse">
−<wsdl:part name="timestampReturn" type="xsd:base64Binary">
−<wsdl:documentation>Returns the timestamp of your file (in binary mode encoded with base64)</wsdl:documentation>
 </wsdl:part>
</wsdl:message>
```

Der Methoden-Aufruf **„timestampRequest“** erfolgt unter Angabe folgender Parameter und enthält:

$username	:: = der Username auf Signaturportal.de
$passwort	:: = das Gateway-Passwort (nicht das Login-Passwort!)
$kontoNr	:: = die Signaturportal.de Kontonummer (z. B. 1010101010)
$dateiname	:: = Dateiname der zu signierenden Datei
$data	:: = die Datei als base64Binary

Der Response **<part name=" timestampReturn" Type=" xsd:base64Binary"/>** enthält entweder die Rückgabe response = 200 dann Fehler. Prozess nicht erfolgreich
oder es wird ein **xsd:base64Binary** zurück geliefert der das .tsr File („xxxx_zeitstempel.tsr“) enthält.

3.3.11 Die Methode „sha256_timestamp“ hat folgenden Aufbau:

```
<wsdl:message name="timestampRequest">
−<wsdl:part name="username" type="xsd:string">
 <wsdl:documentation>Your UserName</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="passwort" type="xsd:string">
 <wsdl:documentation>Your GatewayPassword</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="kontoNr" type="xsd:int">
 <wsdl:documentation>Your Konto Number</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="dateiname" type="xsd:string">
−<wsdl:documentation>The name of the file you like to get the timestamp for.</wsdl:documentation>
 </wsdl:part>
−<wsdl:part name="data" type="xsd:base64Binary">
 <wsdl:documentation>Your file as a binary encoded with base64#</wsdl:documentation>
 </wsdl:part>
</wsdl:message>
−<wsdl:message name="timestampResponse">
−<wsdl:part name="timestampReturn" type="xsd:base64Binary">
−<wsdl:documentation>Returns the timestamp of your file (in binary mode encoded with base64)</wsdl:documentation>
 </wsdl:part>
</wsdl:message>
```

Der Methoden-Aufruf **„timestampRequest“** erfolgt unter Angabe folgender Parameter und enthält:

$username :: = der Username auf Signaturportal.de
$passwort :: = das Gateway-Passwort (nicht das Login-Passwort!)
$kontoNr :: = die Signaturportal.de Kontonummer (z. B. 1010101010)
$dateiname :: = Dateiname der zu signierenden Datei
$data :: = der SHA256-Hash-Wert als base64Binary

Der Response **<part name=" timestampReturn" Type=" xsd:base64Binary"/>** enthält entweder die Rückgabe response = 200 dann Fehler. Prozess nicht erfolgreich
oder es wird ein **xsd:base64Binary** zurück geliefert der das .tsr File („xxxx_zeitstempel.tsr“) enthält.

3.3.12 Beschreibung der Fehlerrückgaben (ErrorCodes):

100 - allgemeiner Fehler (nicht näher Spezifizier barer Verbindungsabbau)

200 - Userdaten fehlerhaft
210 – Account-Daten stimmen nicht -> Account-Nummer prüfen (soap:Client.Userdata)
220 - Username - Passwort -> Username, Gateway-Passwort? (soap:Client.Authentication)
300 - Account weißt nicht die nötigen Voraussetzungen (fehlende Userdaten/Berechtigungen)
310 - Vollmacht wurde noch nicht erteilt / entzogen (soap:Client.UserPermissions)
311 - PostIdent ist noch nicht erfolgt (soap:Client.UserPermissions)
312 - Vollmachtdatei ist nicht vorhanden (soap:Client.UserPermissions)
313 - Firmeninformationen sind unvollständig hinterlegt (soap:Client.UserPermissions)
320 - Konto weißt kein ausreichendes Guthaben auf (soap:Client.Solvent)
-> Konto gesperrt, Tageskontingent überschritten, Kontostand 0 und kein Flatrate-Kunde

400 - Verarbeitungsfehler
410 - Vorverarbeitung fehlgeschlagen
411 - Einbinden des Vollmachtdokuments fehlgeschlagen (soap:Server)
420 - Hauptverarbeitung fehlgeschlagen (PDF-dll)
421 – Signatur Fehlgeschlagen (soap:Server)
422 - Verifikation fehlgeschlagen (soap:Server)
423 - Zeitstempel fehlgeschlagen (soap:Server)
424 - Ermitteln des Kontostands fehlgeschlagen (soap:Server)

500 - Zugriff auf Postfach fehlgeschlagen
501 - Username - Passwort -> Username darf nicht leer sein.
502 - Username - Passwort -> Passwort darf nicht leer sein.
503 - Targethost wurde falsch angegeben.
504 - Mail wurde nicht übergeben (ist null)

4 Beispielclients für die Funktionen 1 bis 12

4.1 Sourcen JAVA

Sie finden die aktuelle Version unter folgenden URLs:

https://www.signaturportal.de/freeware/ebilling-signature-java.zip

https://www.signaturportal.de/freeware/ebilling-signature-java-commandline.zip

https://www.signaturportal.de/freeware/ebilling-signature-java.rar

https://www.signaturportal.de/freeware/ebilling-signature-java-commandline.rar

Das SDK enthält ein funktionsfähiges JAVA Projekt für die Erzeugung und Prüfung elektronischer Rechnungen.

sigportalrunner2.0_java.1.5.tgz

Java SDK 1.5.0, neuestes Update u beziehen unter http://java.sun.com/javase/downloads/index_jdk5.jsp

Ein Update auf das neueste Java-Release wird empfohlen.

sigportalrunner2.0_java.1.6.tgz

Java SDK 1.6.0, neuestes Update zu beziehen unter http://java.sun.com/javase/downloads/index.jsp

Unter Linux lassen sich die Pakete über das Paketmanagement installieren (z.B. YAST unter Suse) oder mittels des heruntergeladenen Binaries. Eine Installation unter Windows erfolgt prinzipiell mittels des Java-Installers für die passende Version. Sonstige Pakete/Programme sind nicht notwendig.

Installation sigportalrunner:

Je nach Java-Version auf dem Zielsystem wird die passende Distribution mittels eines Archivprogrammes im Zielverzeichnis (vom Benutzer anzulegen) entpackt.

Unter Linux würde ein Aufruf für GNU-tar wie folgt aussehen:

'cd /...../meinZielverzeichnis/; tar xzf sigportalrunner2.0_java.1.6.tgz'

Aufruf:

Der sigportalrunner wird mittels Kommandozeilenparametern gesteuert. Dazu wechselt der Benutzer in das Installationsverzeichnis. Vor dem erstmaligen Aufruf sind folgende Benutzerdaten „username“, „passwort“, „region“, „lang“ und „accountID“ in der Datei sigportal.properties vom Benutzer einzutragen.

Kommandozeilenaufruf: 'sigportal <method> <inputfile> [signaturefile] [outputfile]

Folgende Parameter werden unterstützt:

* method = [sign|sign_extern|verify|verify_extern|verify_xml|verify_extern_xml|timestamp|sha256_timestamp|test|test_param]

* inputfile - die Eingabedatei

* signaturefile - die Datei, die eine Signatur enthält, nur bei method=verify_extern

* outputfile - die Ergebnisdatei, falls Parameter nicht vorhanden wird Eingabedatei überschrieben

Nach Durchführung der aufgerufenen Methode signalisiert der sigportalrunner das

Ergebnis mit folgenden Rückgabewerten:

* Rückgabewerte: 0 - erfolgreich

* Rückgabewerte: 1 - Konfigurationsfehler
* Rückgabewerte: 2 - Servicefehler
* Rückgabewerte: 3 - IO-Fehler
* Rückgabewerte: 4 - Kontostand nicht ausreichend
* Rückgabewerte: 5 - falsche oder fehlende Parameter

4.2 Sourcen Perl

Sie finden die aktuelle Version unter folgenden URLs:

https://www.signaturportal.de/freeware/ebilling-signature-perl.zip

https://www.signaturportal.de/freeware/ebilling-signature-perl-commandline.zip

https://www.signaturportal.de/freeware/ebilling-signature-perl.rar

https://www.signaturportal.de/freeware/ebilling-signature-perl-commandline.rar

4.3 Sourcen .Net

Sie finden die aktuelle Version unter folgenden URLs:

https://www.signaturportal.de/freeware/ebilling-signature-csharp.zip

4.4 Sourcen C++

Sie finden die aktuelle Version unter folgenden URL:

https://www.signaturportal.de/freeware/ebilling-signaturportal-cpp.zip

4.5 Sourcen PHP

Sie finden die aktuelle Version unter folgenden URLs:

https://www.signaturportal.de/freeware/ebilling-signature-php.zip

https://www.signaturportal.de/freeware/ebilling-signature-php.rar

Haben Sie weitere Fragen?

Nutzen Sie einen der nachfolgenden Kontakte:

Mo.- Fr. 08.00 Uhr bis 17.00 Uhr

01806 – Signatur (74462887)*
(* 0,20 € pro Anruf aus dem deutschen Festnetz - Mobilfunkpreise können abweichen)

E- Mail: support@mentana.de