BMF

BUNDESMINISTERIUM
FÜR FINANZEN

Version 1.07
Wien, 14. Oktober 2010

Zugang
Webservice

e-zoll

Inhalt

Impressum

Für den Inhalt verantwortlich (organisatorisch)

BUNDESMINISTERIUM FÜR FINANZEN

ABTEILUNG: **VI/5**

APPLIKATION: **Zoll Europa (ZE)**

ADRESSE: **Hintere Zollamtsstraße 4**
A - 1030 Wien

ANSPRECHPARTNER: **Amtsdirektor Christian Rimser**

TELEFON: **+43 (0)1-514 33 / 505428**

FAX: **+43 (0)1-514 33 / 5905428**

E-MAIL: **christian.rimser@bmf.gv.at**

Für den Inhalt verantwortlich (technisch)

BUNDESRECHENZENTRUM GESELLSCHAFT MBH

ABTEILUNG: **Bürgerkonten Zoll**

ADRESSE: **Hintere Zollamtsstraße 4**
A - 1030 Wien

ANSPRECHPARTNER: **Mag. Franz Führer**

TELEFON: **+43 (0)1-711 23 / 2142**

FAX: **+43 (0)1-711 23 / 2152**

E-MAIL: **franz.fuehrer@brz.gv.at**

Vorwort

Dieses Dokument beschreibt den Aufbau und den Ablauf der Webservices von 'e-zoll.at' und den Zugang zu diesen.

Änderungsübersicht

Änderungen gegenüber der Version 1.06

Prüfungen im Rahmen der PreValidation werden nur mehr im aktuellen Prüfungsdokument taxativ aufgeführt.

Änderungen gegenüber der Version 1.05

Auf Grund gestiegener Sicherheitsanforderungen und zum Schutz Ihrer Daten wird der unverschlüsselte Datenaustausch (http) eingestellt.

Änderungen gegenüber der Version 1.04

Operatordefinition bei Lizenzwebservice und Fehlermeldung (LIC-005)
transitRequestBean um attachment erweitert

Änderungen gegenüber der Version 1.03

Änderung im SOAP-Envelope (username-token)
Eigenes Testwebservice

Änderungen gegenüber der Version 1.02

Diverse Änderungen in Zusammenhang mit Data Center Zoll
Appendix für Lizenzen hinzugefügt

Änderungen gegenüber der Version 1.01

Die Methode 'Duplicate LRN' wird entfernt.
Der Fehlercode zu 'User is not permitted' wird geändert.
Die Bedeutung der Werte des Transit Response Bean wird geändert.
Die SSL-Adresse wird aufgenommen.
Ein Beispiel für Content Type wird aufgenommen

Änderungen gegenüber der Version 1.00

Die Methoden 'User is not permitted' und 'Dublicate LRN' werden neu aufgenommen.

Zugang

Allgemeines

Portal

Der Aufruf des Webservices erfolgt derzeit über https und das Portal Austria. Die derzeitig gültige Adresse lautet:

https://txm.portal.at:443/ezoll/transit

Portaluser

Für die Authentifizierung muss für einen Wirtschaftsbeteiligten ein Portaluser angelegt sein. Dieser und das dazugehörige Passwort müssen im SOAP-SecurityHeader im sogenannten <UsernameToken> angeführt werden.

Zum Testen wurde folgender user angelegt:

name: ezolluser@portal.at
pass: ezoll
(operator: test)

UsernameToken

Der <UsernameToken> entspricht diesem Aufbau:

```
<env:Envelope xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ns0="urn:http://brz.gv.at/ezoll/V01"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <env:Header>
            <wsse:Security>
                  <wsse:UsernameToken>
                        <wsse:Username>ezolluser@portal.at</wsse:Username>
                        <wsse:Password>ezoll</wsse:Password>
                  </wsse:UsernameToken>
            </wsse:Security>
      </env:Header>
      <env:Body>
                  ... SOAP-Body ...
      </env:Body>
</env:Envelope>
```

Die fettgedruckten Teile stellen das UsernameToken bzw. nötige Ergänzungen dar. Der Namensraum und die Groß- und Kleinschreibung ist genau einzuhalten. Im

<wsse:Username> erfolgt die Angabe des Portalusers und im <wsse:Password> dessen Passwort. Dies erfolgt in Klarschrift, da bereits eine verschlüsselte Verbindung benützt wird.

Operator

In der Folge wird häufiger der Begriff Operator verwendet, dazu soll hier eine kurze Erklärung bzw. Abgrenzung erfolgen.

Wie bereits erwähnt gibt es für das Portal pro Wirtschaftsbeteiligtem genau einen User. Es kann sein, dass ein Wirtschaftsbeteiligter auf mehrere Standorte verteilt ist und auch seine EDV dementsprechend dezentral läuft. Würde nun jeder Standort alle Nachrichten für den Portaluser abholen können, so würde jeder andere Standort das Nachsehen haben, sprich seine Daten nicht erhalten. Um dies zu verhindern wurde der ‚Operator' eingeführt.

Dieser stellt einen logischen Ordnungsbegriff dar, der von uns nicht gewartet wird, d.h. die Verantwortung der Verwendung liegt beim Wirtschaftsbeteiligten.

Bei jeder Nachricht muss ein Operator angegeben werden, dies kann eine Person, eine Abteilung oder ein Standort sein. Mit der Anmeldung eines Versandvorganges ist dieser an den anmeldenen Operator gebunden, d.h. kein anderer Operator dieses Wirtschaftsbeteiligten kann Nachrichten zu diesem Versand schicken bzw. abholen.

Beispiel:

Spediteuer S hat zwei Standorte A1 und A2 und eine dezentrale EDV. Er führt zwei Operatoren ein (entsprechend den Standorten) A1 und A2.

Ein Mitarbeiter von Standort A1 eröffnet einen Versandfall V1 mit einer TR100. Ab diesem Zeitpunkt ist V1 an A1 gebunden, alle Nachrichten zu diesem Fall können nur von dem Operator V1 verschickt und abgeholt werden.

Sollte nun ein Mitarbeiter des Standorts A2 die Stornierung von V1 veranlassen, so wird dieser als nicht berechtigt zurückgewiesen. Sollte eine Nachricht zu V1 zum Abholen bereitstehen, so kann diese nur vom Operator A1 (= ein Mitarbeiter von Standort A1) abgeholt werden.

Ein Wirtschaftsbeteiligter kann so viele Operator einführen wie er organisatorisch bzw. technisch für nötig hält, aber mindestens einen muss er führen.

Will ein Wirtschaftsbeteiligter dieses Service nicht in Anspruch nehmen, so er hat einen fixen Operator der immer angeführt wird.

Webservice

Es werden vier Methoden angeboten, zum Testen der Verbindung, zum Einsenden und zum Abholen von Daten. Die Parameterübergabe und –rückgabe erfolgt per Beans bzw. Structs.

sendMessages

Dieses Webservice dient zum Abgeben von Nachrichten/Daten. Dies erfolgt mit Hilfe eines Arrays von Beans/Structs (=TransitRequestBean). Jedes Bean stellt den Container einer Nachricht dar, d.h. es können mehrere Nachrichten pro Webservice-Aufruf übergeben werden.

SendMessages liefert ein Array von Beans (= TransitResponseBean) dessen Größe der Größe des Inputarrays entspricht. Für jedes eingehende Bean, gibt es ein korrespondierendes Antwortbean. Die Zuordnung erfolgt mittels einer eindeutigen ID des TransitRequestBeans. Das Antwortbean gibt an, ob die Nachricht syntaktisch fehlerfrei war bzw. ob die Nachricht entgegengenommen wurde.

Sollten technische Probleme, wie Datenbank nicht erreichbar, oder organisatorische Probleme, wie CRN unbekannt, auftreten, so werden auch diese im TransitResponseBean vermerkt.

Die genaue Beschreibung der Beans bzw. Datenstrukturen erfolgt im Anschluß.

getMessages

Dieses Webservice dient zur Abfrage nach neuen Nachrichten für einen angegeben Operator. Beim Aufruf wird ein String übergeben der den Operator angibt dessen Nachrichten abgeholt werden sollen.

Dieses Service gibt ein Array von TransitResponseBeans zurück. Um das Rückgabearray nicht zu groß werden zu lassen gibt es gewisse Einschränkungen auf die Anzahl der Nachrichten bzw. Attachments die auf einmal verschickt werden. Anhand des ContentTypes des letzten ResponseBeans wird mitgeteilt ob es noch Nachrichten gibt; mehr zum BeansHandling im Anschluß.

getDCZ

Mit dieser Methode können Daten (Updates) von DataCenterZoll abgeholt werden. Beim Aufruf muss ein TransitRequestBean übergeben werden. Dieses enthält eine Anfrage um Daten (DC100).
Konnte die Anfrage erfolgreich behandelt werden, so wird ein ResponseBean mit einer DC101 zurückgesendet. Sollten die gewünschten Updates vorhanden sein, so werden diese Daten gezippt ins Attachment des ResponseBeans geschrieben.

testMessage

Diese Nachricht dient zum Testen und verwendet daher keine Beans, der Aufruf erfolgt ohne Parameter und die Rückgabe erfolgt als String.

Testbetrieb

Das Testwebservice ist unter folgender Adresse erreichbar:

https://txm.portal.at:443/ezollTest/transit

Nachrichten die an an das TestWebservice übergeben werden, müssen den Testindikator (<Msg><Test>1</Test></Msg>) auf 1 gesetzt haben.

Zum Testen wurde der Portaluser ezolluser@portal.at angelegt. Für diesen User und den Operator ‚test' gibt es zwei Testnachrichten, davon eine mit Attachment, die jederzeit per getMessages(„test") abgeholt werden können.

Beans

Es werden zwei Typen von Beans verwendet. Für den Aufruf ein Array von TransitRequestBeans und als Rückgabe ein Array von TransitResponseBeans.

TransitRequestBean

Ein Array von TransitRequestBeans wird vom Wirtschaftsbeteiligten als Parameter beim Webservice „sendMessages" mitgegeben.

Das TransitRequestBean hat folgenden Aufbau:

id (long) – Pflicht
message (String) – Pflicht
operatorId (String) – Pflicht
attachment (base64Binary) – optional

id: eine eindeutige ID innerhalb des Arrays. Dient zum Zuordnen des korrespondierenden TransitResponseBean

message: eigentliche XML-Nachricht

operatorId: der Operator der diese Nachricht erstellt/versendet hat

attachment: Bytearray dass ein File (bspw. pdf oder zip) beinhaltet.

TransitResponseBean

Ein Array von TransitResponseBean ist der Rückgabewert der beiden Weberservice-Methoden „sendMessages" und „getMessages"

Das TransitResponseBean hat folgenden Aufbau:

id (long) – Pflicht
message (String) – optional
attachment (base64Binary) – optional
operatorId (String) – Pflicht
contentType (int) – Pflicht

id: eine eindeutige ID innerhalb des Arrays. Bei „sendMessages" entspricht diese ID genau jener des zugehörigen geschickten TransitRequestBeans.

message: XML-Nachricht oder Fehlermeldung (je nach contentType)

attachment: Bytearray dass ein File (bspw. pdf oder zip) beinhaltet.

operatorId: Empfänger der Nachricht der diese Nachricht direkt oder indirekt veranlaßte.

contentType: gibt an um welche Meldung/Nachricht es sich handelt, auch als eine Art Status zu verstehen. *contentType* kann folgende Werte haben:

Wert	Bedeutung
1	Message: der Inhalt ist eine Transitnachricht, diese steht in message, sollte es ein Attachment geben so wird dieses mit attachment mitgegeben und es gibt noch weitere Nachrichten die zum Abholen bereitliegen.
2	Error: Fehlermeldung, der genaue Fehlertext steht im message.
3	ACK: Empfangsbestätigung bei „sendMessages", d.h. Nachricht ist syntaktisch richtig und wurde entgegengenommen.
4	NO_MESSAGES: bei „getMessages", wenn es zur Zeit keine neuen Nachrichten für den Operator gibt.
5	LAST_MESSAGE: kennzeichnet bei „getMessages" die letzte Nachricht die zum abholen bereit liegt.,

Beispiel für contentType

Dieses Beispiel soll das System des ContentTypes näher bringen.
Unter der Annahme daß max. 8 Nachrichten pro Request abgeholt werden können und 10 Nachrichten zur Abholung bereit stehen, erhält der WB folgende Beans mit folgenden ContentTypes per Request:

Beim 1. Request (getMessages)

1. TransitResponseBean: ContentType = 1
2. TransitResponseBean: ContentType = 1
3. TransitResponseBean: ContentType = 1
4. TransitResponseBean: ContentType = 1
5. TransitResponseBean: ContentType = 1
6. TransitResponseBean: ContentType = 1
7. TransitResponseBean: ContentType = 1
8. TransitResponseBean: ContentType = 1

2. Request:

1. TransitResponseBean: ContentType = 1
2. TransitResponseBean: ContentType = 5

3. Request:

1. TransitResponseBean: ContentType = 4

Solange ContentType 4 bis wieder mindestens eine Nachricht zum Abholen bereitliegt.

Fehlernachricht

Wird ein TransitResponseBean mit dem ContentType = 2 („Error") zurückgegeben, so enthält der message-String eine Fehlernachricht. Diese ist in xml-Struktur aufgebaut – in einer abgespeckten Variante der TR101.

```
<Msg>
   <FuncErr>
      <ETy />     ... ErrorTyp = ErrorCode
      <Point />   ... Errorpointer, wo der Fehler innerhalb der eingehenden Nachricht auftrat
      <EReas /> ... ErrorReason = Schema oder Rule
      <OrigVal /> ... OriginalValue, der Wert der die Regel verletzte
     </FuncErr>
</Msg>
```

Beispiel für eine Schema-Fehlernachricht:

```
<?xml version="1.0" encoding="UTF-8"?>
<Msg>
	<FuncErr>
		<ETy>15</ETy>
		<Point>line='5' column='9' - Invalid content was found starting with element 'SdrID'. One of
'{"http://brz.gv.at/ezoll/V01":MsgRcp}' is expected.</Point>
		<EReas>9904</EReas>
	</FuncErr>
	<FuncErr>
		<ETy>15</ETy>
		<Point>line='171' column='34' - Value 'P - an..4' with length = '9' is not facet-valid with respect to maxLength '4'
for type 'an..4'. The value 'P - an..4' of element 'DocCd' is not valid.</Point>
		<EReas>9904</EReas>
	</FuncErr>
	<FuncErr>
		<ETy>15</ETy>
		<Point>line='197' column='10' - The content of element 'Seals' is not complete. One of
'{"http://brz.gv.at/ezoll/V01":ID}' is expected.</Point>
		<EReas>9904</EReas>
	</FuncErr>
</Msg>
```

Beispiel für „CRN unknown":

```
<?xml version="1.0" encoding="UTF-8"?>
<Msg>
	<FuncErr>
		<ETy>15</ETy>
		<Point>Msg.Refs.CRN</Point>
		<EReas>99006</EReas>
		<OrigVal>04AT0000000000</OrigVal>
	</FuncErr>
</Msg>
```

PreValidation

Wird eine Nachricht per sendMessages übermittelt so wird diese zuerst gegen ein Schema geprüft.

Entspricht die Nachricht dem Schema, so werden einige Regeln vorab überprüft. Wird eine dieser verletzt, so wird auch hier die Nachricht mit einer Fehlernachricht abgewiesen.

Wenn das XML dem Schema entspricht und alle Regeln erfüllt werden, wird die Nachricht angenommen und mit einem Bean mit dem Content-Type 3 bzw. ‚ACK' beantwortet.

Diese Regeln und die entsprechenden Fehlermeldungen sind dem jeweils gültigen Prüfungsdokument (Abschnitt: Webservice Prüfungen) zu entnehmen.

Webservice für Lizenzen (AMA, BML, PAWA)

Zur Übermittlung der Lizenzdaten über das e-Zoll-Webservice wird dieses um eine neue Sendemethode erweitert, die sich von der bereits vorhandenen sendMessages nach außen kaum unterscheidet. Zum Abholen der Abschreibungen (= <LicenseUsages>) wird die bereits implementierte getMessages verwendet.

Operator

Beim Aufruf der Lizenzwebservices (sendLicenses bzw. getMessages) muss der Operator der ausstellenden Behörde der Lizenzen entsprechen. Es werden drei Operator unterstützt: AMA, BML, PAWA. Die austellenden Behörden der einzelnen Lizenzen (<Admin> bei LicenseData.xml) müssen dem Operator entsprechen, ansonsten wird eine Fehlermeldung zurückgeliefert (LIC-005).

sendLicenses

Diese Methode dient zum Abgeben von Lizenzen bzw. Änderungen von Lizenzen (= <LicenseData>. Dies erfolgt mit Hilfe eines Arrays von Beans/Structs (=TransitRequestBean). Jedes Bean stellt den Container einer Nachricht dar, d.h. es können mehrere Nachrichten pro Webservice-Aufruf übergeben werden.

Das Handling erfolgt wie bei der bereits vorhandenen Methode sendMessages (siehe e-Zoll Zugangsdokumentation), wobei hier die Verarbeitung synchron erfolgt und auch applikationsabhängige Fehler retourniert werden können (siehe weitere Fehlermeldungen im Anschluss).

PreValidation

Bei der Methode sendLicenses werden folgende PreValidations durchgeführt:

- Technical Error (99000)
- No Operator (99001)
- Another request of same user/operator (99002)
- No Message (99003)
- Invalid XML (99004)
- Wrong Testindicator (99005)
- User is not permitted (99010)

Nähere Informationen entnehmen Sie den vorhergehenden Beschreibungen.

Spezifische Fehlermeldungen

Nach der erfolgreichen Prevalidation einer Nachricht erfolgt sofort die Verarbeitung der gesandten Daten. Bei der Verarbeitung können zusätzliche Fehler auftreten, in diesem Fall wird die Nachricht, wie bei einer Verletzung der PreValidation, mit einer entsprechenden Fehlermeldung abgewiesen. Folgende Fehler können auftreten:

Gap detected

Es wurde die Lückenlosigkeit bei der Paketnummer <PkgNr> verletzt.

mit ErrorCode = 15
ErrorReason = LIC-001
ErrorPointer = LicenseData.PkgNr
OrigVal = "Uebermittelte PackageNr: # / Erwartete PackageNr: #"

Unknown Administration

Die überlieferte Behörde <Admin> ist nicht bekannt.

mit ErrorCode = 15
ErrorReason = LIC-002
ErrorPointer = LicenseData.<...>.Admin
OrigVal = Originalwert von <Admin>

License number is not unique

Es wurde eine neue Lizenez geschickt, die Lizenznummer <Licnr> wurde aber bereits verwendet.

mit ErrorCode = 15
ErrorReason = LIC-003
ErrorPointer = LicenseData.<...>.LicNr
OrigVal = Originalwert von <LicNr>

License number unknown

Bei einer Änderung/Stornierung wurde eine unbekannte Lizenznummer angegeben.

mit ErrorCode = 15
ErrorReason = LIC-004
ErrorPointer = LiceseData.<...>.LicNr
OrigVal = Originalwert von <LicNr>

Wrong Adminstration

Die ausstellende Behörde einer Lizenz (Neuanlage oder Änderung) entspricht nicht dem Operator der sendLicenses aufgerufen hat.

mit ErrorCode = 15
ErrorReason = LIC-005
ErrorPointer = LiceseData.<...>.Admin
OrigVal = Originalwert von <Admin>

XML/XSD-Definition

LicenseData

Beinhaltet Lizenzdaten und deren Änderungen die an das BRZ via sendLicenses versendet werden.

LicenseUsages

Beinhaltet die Abschreibungen durch die Zollbehörde. Diese werden von der Behörde (AMA, BML, PAWA) beim BRZ via getLicenses abgeholt.

WSDL

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions targetNamespace="urn:http://brz.gv.at/ezoll/V01" name="EzollWebService"
xmlns:tns="urn:http://brz.gv.at/ezoll/V01" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/">
 <types>
  <xsd:schema>
   <xsd:import namespace="urn:http://brz.gv.at/ezoll/V01" schemaLocation="EzollWebService_schema1.xsd"/>
  </xsd:schema>
 </types>
 <message name="getMessages">
  <part name="parameters" element="tns:getMessages"/>
 </message>
 <message name="getMessagesResponse">
  <part name="parameters" element="tns:getMessagesResponse"/>
 </message>
 <message name="sendLicenses">
  <part name="parameters" element="tns:sendLicenses"/>
 </message>
 <message name="sendLicensesResponse">
  <part name="parameters" element="tns:sendLicensesResponse"/>
 </message>
 <message name="sendMessages">
  <part name="parameters" element="tns:sendMessages"/>
 </message>
 <message name="sendMessagesResponse">
  <part name="parameters" element="tns:sendMessagesResponse"/>
 </message>
 <message name="testMessage">
  <part name="parameters" element="tns:testMessage"/>
 </message>
 <message name="testMessageResponse">
  <part name="parameters" element="tns:testMessageResponse"/>
 </message>
 <message name="getDCZ">
  <part name="parameters" element="tns:getDCZ"/>
 </message>
 <message name="getDCZResponse">
  <part name="parameters" element="tns:getDCZResponse"/>
 </message>
 <message name="getPostDeclarations">
  <part name="parameters" element="tns:getPostDeclarations"/>
 </message>
 <message name="getPostDeclarationsResponse">
  <part name="parameters" element="tns:getPostDeclarationsResponse"/>
 </message>
 <portType name="WebService">
  <operation name="getMessages">
```

```
    <input message="tns:getMessages"/>
    <output message="tns:getMessagesResponse"/>
  </operation>
  <operation name="sendLicenses">
    <input message="tns:sendLicenses"/>
    <output message="tns:sendLicensesResponse"/>
  </operation>
  <operation name="sendMessages">
    <input message="tns:sendMessages"/>
    <output message="tns:sendMessagesResponse"/>
  </operation>
  <operation name="testMessage">
    <input message="tns:testMessage"/>
    <output message="tns:testMessageResponse"/>
  </operation>
  <operation name="getDCZ">
    <input message="tns:getDCZ"/>
    <output message="tns:getDCZResponse"/>
  </operation>
  <operation name="getPostDeclarations">
    <input message="tns:getPostDeclarations"/>
    <output message="tns:getPostDeclarationsResponse"/>
  </operation>
</portType>
<binding name="WebServicePortBinding" type="tns:WebService">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="getMessages">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="sendLicenses">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="sendMessages">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
```

```
    </output>
   </operation>
   <operation name="testMessage">
    <soap:operation soapAction=""/>
    <input>
     <soap:body use="literal"/>
    </input>
    <output>
     <soap:body use="literal"/>
    </output>
   </operation>
   <operation name="getDCZ">
    <soap:operation soapAction=""/>
    <input>
     <soap:body use="literal"/>
    </input>
    <output>
     <soap:body use="literal"/>
    </output>
   </operation>
   <operation name="getPostDeclarations">
    <soap:operation soapAction=""/>
    <input>
     <soap:body use="literal"/>
    </input>
    <output>
     <soap:body use="literal"/>
    </output>
   </operation>
  </binding>
  <service name="EzollWebService">
   <port name="WebServicePort" binding="tns:WebServicePortBinding">
    <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
   </port>
  </service>
</definitions>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" targetNamespace="urn:http://brz.gv.at/ezoll/V01" xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="sendLicenses" type="ns1:sendLicenses" xmlns:ns1="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="sendLicenses">
  <xs:sequence>
   <xs:element name="arrayOfTransitRequestBean_1" type="ns2:transitRequestBean" maxOccurs="unbounded"
minOccurs="0" xmlns:ns2="urn:http://brz.gv.at/ezoll/V01"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="transitRequestBean">
  <xs:sequence>
   <xs:element name="id" type="xs:long"/>
   <xs:element name="message" type="xs:string" nillable="true"/>
   <xs:element name="operatorId" type="xs:string" nillable="true"/>
   <xs:element name="attachment" type="xs:base64Binary" nillable="true"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="sendLicensesResponse" type="ns3:sendLicensesResponse" xmlns:ns3="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="sendLicensesResponse">
  <xs:sequence>
   <xs:element name="result" type="ns4:transitResponseBean" maxOccurs="unbounded" minOccurs="0"
xmlns:ns4="urn:http://brz.gv.at/ezoll/V01"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="transitResponseBean">
  <xs:sequence>
   <xs:element name="attachment" type="xs:base64Binary" nillable="true"/>
   <xs:element name="contentType" type="xs:int"/>
   <xs:element name="id" type="xs:long"/>
   <xs:element name="message" type="xs:string" nillable="true"/>
   <xs:element name="operatorId" type="xs:string" nillable="true"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="testMessage" type="ns5:testMessage" xmlns:ns5="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:element name="testMessageResponse" type="ns6:testMessageResponse" xmlns:ns6="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="testMessageResponse">
  <xs:sequence>
   <xs:element name="result" type="xs:string" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="sendMessages" type="ns7:sendMessages" xmlns:ns7="urn:http://brz.gv.at/ezoll/V01"/>
```

```
 <xs:complexType name="sendMessages">
  <xs:sequence>
   <xs:element name="arrayOfTransitRequestBean_1" type="ns8:transitRequestBean" maxOccurs="unbounded"
minOccurs="0" xmlns:ns8="urn:http://brz.gv.at/ezoll/V01"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="sendMessagesResponse" type="ns9:sendMessagesResponse"
xmlns:ns9="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="sendMessagesResponse">
  <xs:sequence>
   <xs:element name="result" type="ns10:transitResponseBean" maxOccurs="unbounded" minOccurs="0"
xmlns:ns10="urn:http://brz.gv.at/ezoll/V01"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="getMessages" type="ns11:getMessages" xmlns:ns11="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="getMessages">
  <xs:sequence>
   <xs:element name="String_1" type="xs:string" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="getMessagesResponse" type="ns12:getMessagesResponse"
xmlns:ns12="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="getMessagesResponse">
  <xs:sequence>
   <xs:element name="result" type="ns13:transitResponseBean" maxOccurs="unbounded" minOccurs="0"
xmlns:ns13="urn:http://brz.gv.at/ezoll/V01"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="getPostDeclarations" type="ns14:getPostDeclarations" xmlns:ns14="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="getPostDeclarations">
  <xs:sequence>
   <xs:element name="String_1" type="xs:string" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:element name="getPostDeclarationsResponse" type="ns15:getPostDeclarationsResponse"
xmlns:ns15="urn:http://brz.gv.at/ezoll/V01"/>

 <xs:complexType name="getPostDeclarationsResponse">
  <xs:sequence>
   <xs:element name="result" type="ns16:transitResponseBean" maxOccurs="unbounded" minOccurs="0"
xmlns:ns16="urn:http://brz.gv.at/ezoll/V01"/>
```

```
    </xs:sequence>
  </xs:complexType>


  <xs:element name="getDCZ" type="ns19:getDCZ" xmlns:ns19="urn:http://brz.gv.at/ezoll/V01"/>

  <xs:complexType name="getDCZ">
    <xs:sequence>
      <xs:element name="transitRequestBean" type="ns20:transitRequestBean" minOccurs="0"
xmlns:ns20="urn:http://brz.gv.at/ezoll/V01"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="getDCZResponse" type="ns21:getDCZResponse" xmlns:ns21="urn:http://brz.gv.at/ezoll/V01"/>

  <xs:complexType name="getDCZResponse">
    <xs:sequence>
      <xs:element name="result" type="ns22:transitResponseBean" minOccurs="0" xmlns:ns22="urn:http://brz.gv.at/ezoll/V01"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```