Bundesministerium
Finanzen

Technische Aufbau/Ablauf-Beschreibung CTW-Webservice Zugang

Webservice für Wirtschaftsbeteiligte

Wien, 01. Oktober 2024

Inhalt

1 Vorwort

Dieses Dokument dient als Leitfaden für den Aufbau, Ablauf und die Nutzung der Webservices in Bezug auf elektronische Zollanwendungen. Dabei wird nicht nur die technische Architektur dargestellt, sondern auch der sicherheitsrelevante Zugang zu diesen Anwendungen.

2 Änderungsübersicht

1. 08.02.2024: Änderung **CTW-Out-Services** Link bei 3.1 und 3.2
2. 21.03.2024: Erweiterung URLs für WB-Umgebung (Links bei 3.2)

3 Allgemeines

3.1 Zugänge

Der Zugriff auf den Webservice wird über bestimmte Webserviceschnittstellen realisiert, die das HTTPS-Protokoll nutzen. Die relevanten Adressen des Webservices sind:

- **CTW-In-Services: https://txm.portal.at/at.gv.bmf-ef.ctw.in-p/soap/ezoll**
- **CTW-Out-Services: https://txm.portal.at/at.gv.bmf-ef.ctw.out-p/soap/ezoll/out**

3.2 Testbetrieb

Das Testwebservice ist unter folgenden Adressen erreichbar:

- **CTW-In-Services: https://txm.portal.at/at.gv.bmf-ef.ctw.in-t/soap/ezoll**
- **CTW-Out-Services: https://txm.portal.at/at.gv.bmf-ef.ctw.out-t/soap/ezoll/out**

Info: Nachrichten die an das TestWebservice übergeben werden, müssen den Testindikator (<Msg><Test>1</Test></Msg>) auf ‚1' gesetzt haben. Um Testnachrichten übermitteln zu können, muss Ihrem Webservicekonto im Unternehmensserviceportal das Recht "Testwebservice: eZoll" zugewiesen werden. Ein Anleitung dazu finden Sie auf der BMF Homepage.
Mit dem Operator ‚test' können mehrere Testnachrichten, davon eine mit Attachment, jederzeit per getMessages(‚test') abgeholt werden.

Das neue Testwebservice für das **ACCS-Projekt** für die Verfahren Import, Export und Transit ist unter folgenden Adressen erreichbar:

- **CTW-In-Services: https://txm.portal.at/at.gv.bmf-ef.ctw.in-wb/soap/ezoll**
- **CTW-Out-Services: https://txm.portal.at/at.gv.bmf-ef.ctw.out-wb/soap/ezoll/out**

Info: Im Header des Requests ist der Content-Type zwingend auf „text/xml" zu setzen. Als Testnachricht kann kann die SOAP Methode „<testMessage> genutzt werden.

3.3 Webservicekonto

Zur Authentifizierung eines Wirtschaftsbeteiligten ist die Einrichtung eines Webservicekontos im Unternehmensserviceportal erforderlich, wobei die entsprechenden Webservicerechte zuzuweisen sind. Der dazugehörige Benutzername (z. B.: s0XXXX..) und das Passwort sollten im SOAP-SecurityHeader unter <UsernameToken> (WSS-Username-Token-Profile-V1.1.1) angegeben werden.

Info: Ein Anleitung dazu finden Sie auf der BMF Homepage.

3.4 Anwendungen

Aktuell nutzen folgende Zoll-Anwendungen den neue CTW-Webservice Zugang:

- ICS2
- Zoll Korridormodul (ab 2024)
- ACCS (ab 2024)

3.5 UsernameToken

Der <UsernameToken> (WSS-Username-Token-Profile-V1.1.1) 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 (WSS-Username-Token-Profile-V1.1.1) bzw. nötige Ergänzungen dar. Der Namensraum und die Groß- und Kleinschreibung sind 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.

3.6 Operator

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

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 dezentral läuft. Würde nun jeder Standort alle Nachrichten für den Portaluser abholen können, so hätte jeder andere Standort das Nachsehen, 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 anmeldenden Operator gebunden, d.h. kein anderer Operator dieses Wirtschaftsbeteiligten kann Nachrichten zu diesem Versand schicken bzw. abholen.

Beispiel:

Spediteur 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.

4 Webservices

4.1 CTW-In-Services

CTW-In-Services stellt ein SOAP-Webservice mit mehreren Operationen bereit. CTW-In-Services empfängt XML-Nachrichten für mehrere IT-Anwendungen. Nach allgemeiner Gültigkeitsprüfung (z.B. XML-Schema Prüfung) werden gültige Nachrichten zur weiteren Bearbeitung, je nach Nachrichtentyp, jeweils an eine bestimmte IT-Anwendung weitergeleitet.

4.2 CTW-Out-Services

CTW-Out-Services stellt ein SOAP-Webservice mit mehreren Operationen bereit. Ein zugelassener Wirtschaftsbeteiligter ruft dieses SOAP-Service auf, um vorhandene XML-Nachrichten abzuholen.

4.3 Methoden

CTW-In-Services und CTW-Out-Services erhalten die gleichen Methoden, aber nicht alle Methoden werden von beiden Webservices („CTW-In-Services“ und „CTW-Out-Services“) unterstützt.

4.3.1 sendMessages

CTW-In-Services

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 (TransitResponseBean). 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 ‚ICRef nicht eindeutig', auftreten, so werden diese im TransitResponseBean vermerkt.

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

CTW-Out-Services

Diese Methode wird von CTW-Out-Services nicht unterstützt.

4.3.2 getMessages

CTW-In-Services

Diese Methode wird von CTW-In-Services nicht unterstützt.

CTW-Out-Services

Dieses Webservice dient zur Abfrage nach neuen Nachrichten für einen angegebenen 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 Einschränkungen auf die Anzahl der Nachrichten bzw. Attachments, die auf einmal verschickt werden. Anhand des ContentTypes des letzten ResponseBeans wird mitgeteilt, ob es weitere Nachrichten gibt; mehr zum BeansHandling im Anschluss.

4.3.3 getDCZ

CTW-In-Services / CTW-Out-Services

Diese Methode wird von CTW-In-Services/CTW-Out-Services nicht unterstützt.

4.3.4 getPostDeclarations

CTW-In-Services / CTW-Out-Services

Diese Methode wird von CTW-In-Services/CTW-Out-Services nicht unterstützt.

4.3.5 sendLicenses

CTW-In-Services / CTW-Out-Services

Diese Methode wird von CTW-In-Services/CTW-Out-Services nicht unterstützt.

4.3.6 testMessage

CTW-In-Services / CTW-Out-Services

Diese Nachricht dient zum Testen und verwendet daher keine Beans, der Aufruf erfolgt ohne Parameter und die Rückgabe erfolgt als String (z.B ‚Hallo user! Erfolgreich bei CTW InServices angekommen.').

4.4 Beans

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

4.4.1 TransitRequestBean

Ein Array von TransitRequestBeans wird vom Wirtschaftsbeteiligten als Eingabeparameter 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:	base64Binary, das ein File (bspw. im PDF-Format oder gezippt) beinhaltet. Attachement wird zurzeit nicht unterstützt und darf nicht angegeben werden.

4.4.2 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:	base64Binary, das ein File (bspw. im PDF-Format oder gezippt) beinhaltet.
operatorId:	Empfänger der Nachricht der diese Nachricht direkt oder indirekt veranlasste.
contentType:	gibt an um welche Meldung/Nachricht es sich handelt und ist auch als Status zu interpretieren.

Tabelle 1 Werte für den contentType im TransitResponseBean

Wert	Bedeutung
1	Message: der Inhalt ist eine Nachricht, diese steht in message,. Es können auch noch weitere Nachrichten zur Abholung bereitliegen.
2	Error: Fehlermeldung, der genaue Fehlertext steht in 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 näherbringen.

Unter der Annahme, dass max. 8 Nachrichten pro Request abgeholt werden können und 10 Nachrichten zur Abholung bereitstehen, erhält der Wirtschaftsbeteiligte bei drei aufeinanderfolgenden Requests folgende Beans mit folgenden ContentTypes:

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.

4.5 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 reduzierten Variante der TR101 (Vergl. BMF, Codelisten: Codeliste NC_01000 für Nachrichtenartencodes).

<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:

```
<Msg>
  <FuncErr>
  <ETy>15</ETy>
    <Point>cvc-complex-type.2.4.a: Invalid content was found starting with element '{"http://brz.gv.at/ezoll/V01":declarationType}'. One of '{"http://brz.gv.at/ezoll/V01":LRN}' is expected.</Point>
    <EReas>W00001</EReas>
  </FuncErr>
  <FuncErr>
    <ETy>15</ETy>
    <Point>cvc-pattern-valid: Value 'FCAG' is not facet-valid with respect to pattern '[A-Za-z]{3}' for type 'a3'.</Point>
    <EReas>W00001</EReas>
  </FuncErr>
  <FuncErr>
    <ETy>15</ETy>
    <Point>cvc-type.3.1.3: The value 'FCAG' of element 'incotermCode' is not valid.</Point>
    <EReas>W00001</EReas>
  </FuncErr>
```

```
</Msg>
```

Beispiel für ‚ICRef nicht eindeutig ':

```
<Msg>
  <FuncErr>
   <ETy>26</ETy>
    <Point>Msg.ICRef</Point>
    <EReas>W00004</EReas>
    <OrigVal>779</OrigVal>
  </FuncErr>
</Msg>
```

4.6 Prevalidation

Wird eine Nachricht per sendMessages an CTW-In-Services übermittelt, wird diese zuerst auf formale Korrektheit geprüft.

Entspricht die Nachricht dem Schema, so werden einige Regeln[1] vorab überprüft. Wird eine dieser verletzt, so wird 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 - ‚ACK' beantwortet.

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

5 WSDL

CTW-WebService

```
<?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>
```

CTW-WebService-schema1

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema targetNamespace="urn:http://brz.gv.at/ezoll/V01"
    version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ns1="urn:http://brz.gv.at/ezoll/V01">
    <xs:element name="sendLicenses" type="ns1:sendLicenses" />
    <xs:complexType name="sendLicenses">
        <xs:sequence>
            <xs:element name="arrayOfTransitRequestBean_1"
                type="ns1:transitRequestBean" minOccurs="0"
maxOccurs="unbounded" />
        </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" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
    <xs:element name="sendLicensesResponse"
        type="ns1:sendLicensesResponse" />
    <xs:complexType name="sendLicensesResponse">
        <xs:sequence>
            <xs:element name="result" type="ns1:transitResponseBean"
                minOccurs="0" maxOccurs="unbounded" />
        </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="ns1:testMessage" />
        <xs:complexType name="testMessage">
                <xs:sequence />
        </xs:complexType>
        <xs:element name="testMessageResponse"
                type="ns1:testMessageResponse" />
        <xs:complexType name="testMessageResponse">
                <xs:sequence>
                        <xs:element name="result" type="xs:string" minOccurs="0" />
                </xs:sequence>
        </xs:complexType>
        <xs:element name="sendMessages" type="ns1:sendMessages" />
        <xs:complexType name="sendMessages">
                <xs:sequence>
                        <xs:element name="arrayOfTransitRequestBean_1"
                                type="ns1:transitRequestBean" minOccurs="0"
maxOccurs="unbounded" />
                </xs:sequence>
        </xs:complexType>
        <xs:element name="sendMessagesResponse"
                type="ns1:sendMessagesResponse" />
        <xs:complexType name="sendMessagesResponse">
                <xs:sequence>
                        <xs:element name="result" type="ns1:transitResponseBean"
                                minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
        </xs:complexType>
        <xs:element name="getMessages" type="ns1:getMessages" />
        <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="ns1:getMessagesResponse" />
        <xs:complexType name="getMessagesResponse">
                <xs:sequence>
                        <xs:element name="result" type="ns1:transitResponseBean"
                                minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
        </xs:complexType>
```

```
        <xs:element name="getPostDeclarations"
                type="ns1:getPostDeclarations" />
        <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="ns1:getPostDeclarationsResponse" />
        <xs:complexType name="getPostDeclarationsResponse">
                <xs:sequence>
                        <xs:element name="result" type="ns1:transitResponseBean"
                                minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
        </xs:complexType>
        <xs:element name="getDCZ" type="ns1:getDCZ" />
        <xs:complexType name="getDCZ">
                <xs:sequence>
                        <xs:element name="transitRequestBean"
                                type="ns1:transitRequestBean" minOccurs="0" />
                </xs:sequence>
        </xs:complexType>
        <xs:element name="getDCZResponse" type="ns1:getDCZResponse" />
        <xs:complexType name="getDCZResponse">
                <xs:sequence>
                        <xs:element name="result" type="ns1:transitResponseBean"
                                minOccurs="0" />
                </xs:sequence>
        </xs:complexType>
</xs:schema>
```

Tabellenverzeichnis

Literaturverzeichnis

Bundesministerium für Finanzen: Codelisten. Aktuelle Fassung unter: https://www.bmf.gv.at/zoll/e-zoll/technische-informationen.html

Bundesministerium für Finanzen: EzollWebService. EzollWebService.xml. Aktuelle Fassung unter: https://www.bmf.gv.at/zoll/e-zoll/technische-informationen.html

Bundesministerium für Finanzen: EzollWebService-schema1. EzollWebService_schema1.xml. Aktuelle Fassung unter: https://www.bmf.gv.at/zoll/e-zoll/technische-informationen.html

Bundesministerium für Finanzen: Prüfungen. Aktuelle Fassung unter: https://www.bmf.gv.at/zoll/e-zoll/technische-informationen.html

OASIS Open: Web Services Security Username Token Profile Version 1.1.1. 18 May 2012. OASIS Standard. http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-UsernameTokenProfile-v1.1.1-os.html

Impressum

Medieninhaber, Verleger und Herausgeber:
Bundesministerium für Finanzen, Johannesgasse 5, 1010 Wien
Wien, 21.März 2024
Version: 1.2

Für den Inhalt verantwortlich (organisatorisch):
Bundesministerium für Finanzen
Abteilung: I/11-ZE, IT Zoll
Adresse: Hintere Zollamtsstraße 2b, 1030 Wien
E-Mail: post.i-11-ze@bmf.gv.at

Für den Inhalt verantwortlich (technisch):
Bundesrechenzentrum Gesellschaft mbH
Abteilung: I-SP-SO, Zoll & Weitere Applikationen
Adresse: Hintere Zollamtsstraße 4, 1030 Wien
E-Mail: post.i-sc-betrieb-ze@brz.gv.at

Bundesministerium für Finanzen
Johannesgasse 5, 1010 Wien
+43 1 514 33-0
bmf.gv.at