BMF

BUNDESMINISTERIUM
FÜR FINANZEN

Version 1.04
Wien, 28. April 2016

VIP Webservice

Spezifikation

VIP

VerbrauchsteuerInternetPlattform

Inhalt

Impressum

BUNDESMINISTERIUM FÜR FINANZEN

ABTEILUNG: **V/5**

APPLIKATION: **Zoll Österreich (ZO)**

ADRESSE: **Hintere Zollamtsstraße 2b**
A - 1030 Wien

ANSPRECHPARTNER: **Günter Decker / Manfred Dibon**

TELEFON: **+43 (0)1-514 33 / 505443 oder 505433**

E-MAIL: **POST.V-5-ZO@BMF.GV.AT**

Für den Inhalt verantwortlich (technisch)

BUNDESRECHENZENTRUM GESELLSCHAFT MBH

ABTEILUNG: **Finanzanwendungen Zoll**

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

ANSPRECHPARTNER: **DI Hans-Jürgen Kozik**

TELEFON: **+43 664 88956162**

Vorwort

Dieses Dokument spezifisiert die elektronische Schnittstelle zur Übermittlung von Daten an Verbrauchsteuerapplikationen, wie EMCS (Excise Movement Control System) oder EVA (Elektronische Verbrauchsteueranmeldung). Es wird der Aufbau und der Ablauf des VIP Webservices und der Zugang zu diesem definiert und beschrieben.

Änderungsübersicht

Version	Datum	Beschreibung	Autor
1.00	23.10.2008	Erstellung des Dokuments	Petioky
1.01	19.01.2009	Definition des Aufbaus der PortaluserID	Petioky
1.02	07.03.2013	Neue Webservice Methoden	Kozik
1.03	04.10.2013	Neue Fehlermeldung „Method not supported fo this client"	Petioky
1.04	28.04.2016	Klarstellung betreffend Operator in EMCS Neue Webservice Methoden: getMessagesForVIDManualAcknowledgement, acknowledgeMessages, setWaiting	Kozik

Zugang

Der Aufruf des VIP Webservices erfolgt über das Portal Austria. Dieses führt auch die Authentifizierung und Autorisierung mit Hilfe des so genannten UsernameTokens durch. Jeder Teilnehmer hat einen eigenen Account, dies ist ein Portaluser in Form einer eindeutigen E-Mailadresse und eines Passworts.
Anträge auf Einrichtung eines Portalusers können beim BMF Abteilung V/5-ZO schriftlich beantragt werden (siehe Impressum).

Portaluser

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

Die Portaluser ID entspricht folgendem Format *<Firmenname>*@vst.bmf.gv.at, wobei Firmenname dem Wirtschaftsbeteiligten entsprechen sollte und nur aus Buchstaben, Ziffern, Unterstrich und Bindestrichen bestehen darf, d.h. wie eine herkömmliche E-Mail-Adresse.

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://vst.bmf.gv.at/vip/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>user@vst.bmf.gv.at</wsse:Username>
                        <wsse:Password>pw1234!</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.

Endpointadresse

Der Aufruf des Webservices erfolgt über folgende Adresse lautet

https://txm.portal.at:443/vip/webservice

Operator, VID

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. Ein Wirtschaftsbeteiligter kann nun mehrere Standorte und/oder Bewilligungen haben. Diese Standorte könnten auch über eigene unabhängige EDV-Systeme verfügen. Um den Wirtschaftsbeteiligten bei der organisatorischen und technischen Verteilung der Nachrichten zu unterstützen, wurde der „Operator" eingeführt.

Bei jeder gesendeten 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 angemeldenen Operator gebunden, d.h. kein anderer Operator dieses Wirtschaftsbeteiligten kann Nachrichten zu diesem Versand schicken bzw. abholen.

Durch die Einführung der nationalen Prüfung N008[*)] wurde festgelegt, dass in EMCS als Operator stets die VID-Nummer (Verbrauchsteuer Identifikationsnummer) der betroffenen Bewilligung angegeben werden muss. Bei jeder gesendeten Nachricht muss als Operator die VID angegeben werden zu der die Nachricht gehört, bspw. eine Verbrauchsteueranmeldung. Werden Nachrichten abgeholt, bspw. eine Empfangsbestätigung bei EMCS, muss auch immer die gewünschte VID des Nachrichtenempfängers angegeben werden.

*) *Die nationale Prüfung N008 lautet:*
"Geprüft wird, ob der Versender der Nachricht berechtigt ist die Nachricht zu übermitteln. Bei den Nachrichten EM812, EM813, EM814, EM815, FB801 wird geprüft, ob der Versender der Nachricht dem Versender laut e-VD entspricht.
Bei der Nachricht EM818 wird geprüft, ob der Versender der Nachricht dem Empfänger laut e-VD entspricht.
Bei der Nachricht EM840 wird geprüft, ob der Versender der Nachricht entweder dem Versender oder dem Empfänger laut e-VD entspricht."

Testbetrieb

Nachrichten die an das TestWebservice übergeben werden, müssen den Systemindikator auf „t" und den Testflag der Nachricht (<Header><TestFlag>1</TestFlag></Header>) auf „1" gesetzt haben.

Das Testwebservice ist voraussichtlich ab 15. Jänner 2009 unter folgender Adresse erreichbar:

https://txm.portal.at:443/vipTest/webservice

Testuser

Die Portaluser ID für das Testwebservice entspricht folgendem Format *<Firmenname>*@vst-test.bmf.gv.at.

Testuser können im BMF Abteilung V/5-ZO per e-Mail (post.v-5-zo@bmf.gv.at) beantragt werden. Die erforderlichen Daten werden dem Antragsteller umgehend formlos mitgeteilt.

Webservice Methoden

Es werden vier Methoden angeboten, zum Testen der Verbindung, zum Einsenden und zum Abholen von Daten. Die Parameterübergabe und –rückgabe erfolgt per VipWebserviceBean (ist ein JavaBean bzw. Struct), dieses wird im anschließenden Kapitel beschrieben.

sendMessage

Dieses Webservice dient zum Abgeben von Nachrichten/Daten. Dies erfolgt mit Hilfe eines VipWebserviceBean. Jedes Bean stellt den Container einer Nachricht dar, d.h. es kann genau eine Nachricht übergeben werden.

sendMessage liefert als Ergebnis ein VipWebserviceBean. 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 VID unbekannt, auftreten, so werden diese als Fehlernachricht verpackt im VipWebserviceBean geliefert.

Input: operator, system, contentType, messageID, messageType, message
Result: VipWebserviceBean

getMessagesForVID

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

Dieses Service gibt ein Array von VipWebserviceBeans 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 Beans wird mitgeteilt ob es noch Nachrichten gibt; mehr zum BeansHandling im Anschluß.

Input: operator, system, contentType, messageID (in diesem Fall aber eine call_uuid)
Result: VipWebserviceBean

- ➔ Um den Webserver zu entlasten, wird ersucht die Methode getMessagesForVID nach erfolglosen Abfragen, d.h. es wurden keine neue Nachrichten zurückgeliefert, nach zwei Minuten oder mehr erneut aufzurufen.

testService

Diese Nachricht dient zum Testen der Verbindung und verwendet daher keine Beans, der Aufruf erfolgt ohne Parameter und die Rückgabe erfolgt als String mit Angabe der Serveruhrzeit und der Versionsnummer des Webservices.

Input: keiner
Result: String

verifyMessage

Dieses Webservice funktioniert wie sendMessage, mit dem Unterschied, dass die Nachricht nur geprüft wird und es zu keiner Verarbeitung kommt.

Input: operator, system, contentType, messageID, messageType, message
Result: VipWebserviceBean

getTypeForVID

Dieses Webservice liefert die Klassifizierung einer Bewilligung zB. Steuerlager, Registrierter Empfänger.

Input: VID
Result: VipWebserviceBean

getProductCategoryForVID

Dieses Webservice liefert die Produktkategorien, die für eine Bewilligung bewilligt wurden zB. W200, T400

Input: VID
Result: VipWebserviceBean

recover

Dieses Webservice funktioniert wie sendMessage, mit dem Unterschied, dass nur Nachrichten, die im Fallbackverfahren erstellt wurden, gesendet werden dürfen.

Input: operator, system, contentType, messageID, messageType, message
Result: VipWebserviceBean

setWaiting

Dieses Webservice dient dazu Nachrichten wieder in die Queue zur Abholung zu stellen.

Input: call_uuid, operator, system, messageIDs
Result: VipWebserviceBean

getMessagesForVIDManualAcknowledgement

Dieses Webservice funktioniert wie die Service getMessagesForVID. Der Unterschied besteht darin, dass man den Erhalt der Nachricht mit dem Service acknowledgeMessages bestätigen muss. Wird eine Nachricht nicht innerhalb von 6 Minuten bestätigt, wird diese automatisch wieder in die Queue zur Abholung gestellt.

Input: call_uuid, operator, system
Result: VipWebserviceBean

acknowledgeMessages

Dieses Webservice dient dazu, den Erhalt der Nachrichten aus dem Service getMessagesForVIDManualAcknowledgement zu bestätigen.

Input: call_uuid, operator, system, messageIDs
Result: VipWebserviceBean

Webservice Parameter

Für das Handling der Datenübergabe bei Request und Response wird ein einheitliches Datenformat (unter Java auch Bean genannt) verwendet. Das VipWebserviceBean besteht aus folgenden sieben Attributen.

operator

Versender bzw. Empfänger der Nachricht bzw. Nummer der Bewilligung (VID).

Typ: String

Pflicht: immer

system

Dieser Indikator gibt an ob die Nachricht für das Entwicklungs-, Test- oder Produktivsystem gedacht ist bzw. in diesem produziert wurde.

Typ: Indikator/Charakter

Wert	Bedeutung
e	Entwicklungssystem
t	Testsystem
p	Produktivsystem

Pflicht: immer

contentType

gibt an um welche Meldung/Nachricht es sich handelt, auch als eine Art Status zu verstehen.

Typ: Indikator

Wert	Bedeutung
1	Message: der Inhalt ist eine Nachricht, diese steht in message, sollte es ein Attachment geben so wird dieses mit attachment mitgegeben. Bei „*getMessagesForVID*“ heißt es zusätzlich dass noch weitere Nachrichten zum Abholen bereitliegen.
2	Error: Fehlermeldung, die genaue Fehlerbeschreibung steht im message.

3	ACK: Empfangsbestätigung bei „*sendMessage*“, d.h. Nachricht ist syntaktisch richtig und wurde entgegengenommen.
4	NO_MESSAGES: bei „*getMessagesForVID*“ wenn es zur Zeit keine neuen Nachrichten für den Operator gibt oder bei „*getMessageByID*“ dass es für die gewählte Kombination MessageID/Operator keine Nachricht gibt.
5	LAST_MESSAGE: kennzeichnet bei „*getMessagesForVID*“ die letzte Nachricht die zum abholen bereit liegt. Wurde bei „*getMessageByID*“ eine Nachricht gefunden so wird diese mit diesem ContentType geliefert.

Pflicht: immer

Beispiel für contentType:

Dieses Beispiel soll das System des ContentTypes näher bringen.
Unter der Annahme daß max. 6 Nachrichten pro Request abgeholt werden können und 8 Nachrichten zur Abholung bereit stehen, erhält der WB folgende Beans mit folgenden ContentTypes per Request:

Beim 1. Request (getMessagesForVID):

1. VipWebserviceBean: ContentType = 1
2. VipWebserviceBean: ContentType = 1
3. VipWebserviceBean: ContentType = 1
4. VipWebserviceBean: ContentType = 1
5. VipWebserviceBean: ContentType = 1
6. VipWebserviceBean: ContentType = 1

2. Request (getMessagesForVID) gleich im Anschluss, da noch Nachrichten vorhanden sind:

1. VipWebserviceBean: ContentType = 1
2. VipWebserviceBean: ContentType = 5

3. Request (getMessagesForVID) nach mindestens zwei Minuten:

1. VipWebserviceBean: ContentType = 4

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

messageType

Gibt den Typ der Nachricht (bspw. EM801A) an die unter message zu finden ist.

Typ: String

Pflicht: wenn message befüllt ist

messageID

Diese wird nicht vom Ersteller der Nachricht generiert, sondern ist der MessageIdentifier aus dem Header der XML-Nachricht.

Für das Service setWaiting muss der MessageIdentifier der XML-Nachricht, die über das Servcie sendMessage übermittelt wurde, verwendet werden.

Für das Service acknowledgeMessages muss der MessageIdentifier der XML-Nachricht, die über das Servcie getMessagesForVIDManualAcknowledgement empfangen wurde, verwendet werden.

Für das Service getMessagesForVID wird als messageID eine call_uuid erwartet.

Typ: String

Pflicht: als Inputparameter bei setWaiting und acknowledgeMessages und wenn das Feld message befüllt ist.

message

XML-Nachricht oder Fehlermeldung (je nach contentType).

Typ: String

Pflicht: als Inputparameter bei sendMessage

attachment

Bytearray dass ein File (bspw. pdf) beinhaltet.

Typ: binary (base 64)

Pflicht: nie

call_uuid

Eindeutige Identifikation der Nachricht. Diese wird vom Ersteller der Nachricht generiert und muss eindeutig sein.

Typ: String

Pflicht: immer

responseMessageLimit

Maximale Anzahl der Nachrichten, die pro Request abgeholt werden können.

Typ: Zahl zwischen 5 und 20

Pflicht: optional

Fehlernachricht

Wird ein VipWebserviceBean mit dem ContentType = 2 („Error“) zurückgegeben, so enthält der message-String eine Fehlernachricht, die diesem Format entspricht. (zugehöriges Schema ist im Anhang ersichtlich)

```
<VipWebserviceError>
   <Error>
        <Code />      ... Code (WS##), siehe nächsten Abschnitt
        <Descr />     ... Description, Fehlerbeschreibung, siehe nächsten Abschnitt
        <Point />     ... Point, wo der Fehler auftrat
        <OrigVal />   ... OriginalValue, der Wert der die Regel verletzte
   </Error>
</VipWebserviceError>
```

Beispiel für einen falschen Nachrichtentyp

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:VipWebserviceError
xmlns:tns="urn:http://vst.bmf.gv.at/vip/v01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <tns:Error>
        <Code>WS04</Code>
        <Descr>Unknown messageType</Descr>
        <Point>messageType<//Point>
        <OrigVal>EM80</OrigVal>
    </tns:Error>
</tns:VipWebserviceError>
```

Beispiel für einen Fehler bei der Schemavalidierung

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:VipWebserviceError
xmlns:tns="urn:http://vst.bmf.gv.at/vip/v01"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <tns:Error>
        <Code>WS08</Code>
        <Descr>Invalid message</Descr>
        <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>
    </tns:Error>
    <tns:Error>
        <Code>WS08</Code>
        <Descr>Invalid message</Descr>
        <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>
    </tns:Error>
    <tns:Error>
        <Code>WS08</Code>
        <Descr>Invalid message</Descr>
        <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>
    </tns:Error>
</tns:VipWebserviceError>
```

Rules

Wird eine Nachricht per sendMessage übermittelt so wird diese zuerst gegen einige allgemeine Regeln und abschliessend gegen das zugehörige Schema geprüft. Wird eine der Rules verletzt, so wird die eingehende Nachricht mit einer Fehlernachricht abgewiesen, d.h. das retournierte VipWebserviceBean hat den ContentType 2 bzw. „Error".

Wenn das XML dem Schema entspricht und alle Regeln erfüllt werden, wird die Nachricht angenommen und mit einem Bean mit dem ContentType 3 bzw. „ACK" beantwortet.

Diese Regeln und die entsprechenden Fehlermeldungen werden in Folge vorgestellt und beschrieben.

WS00 – Technical error

Sollte ein technischer Fehler (bspw. Datenbank nicht gefunden) und daher auch eine korrekte Nachricht nicht angenommen werden, so erfolgt folgende Fehlermeldung

Code = WS00
Descr = Technical error
Point = BRZ
OrigVal = null

WS01 – Missing data

Wenn bei einem Bean ein Pflichtfeld oder ein Aufrufparameter nicht befüllt ist, dann wird der Webserviceaufruf nicht bearbeitet und mit folgendem Fehler abgewiesen.

Code = WS01
Descr = Missing data
Point = *bspw. VipWebserviceBean, vid, operator, system, msgType, messageID, message*
OrigVal = null

WS02 – Wrong system

Wenn eine Nachricht versendet wird, so muss der Systemindikator dem des Zielsystems sein (siehe Beschreibung des VipWebserviceBeans). Ist dies nicht der Fall so wird die Nachricht mit „Wrong System" abgelehnt.

Code = WS02
Descr = Wrong system
Point = system
OrigVal = *original Systemindikator*

WS03 – Another request of same user/operator

Pro User und Operator kann nur ein getMessages-Request ausgeführt werden. Wird bereits ein Request behandelt, so wird der zweite mit dieser Fehlermeldung abgelehnt.

Code = WS03
Descr = Another request of same user/operator
Point = Bean
OrigVal = *original user and operator*

WS04 – Unknown messageType

Ist der mitgesandte Nachrichtentyp nicht bekannt, so wird die Nachricht abgelehnt.

Code = WS04
Descr = Unknown messageType
Point = messageType
OrigVal = *original messageType*

WS05 – Duplicate messageID

Die MessageID muss pro Wirtschaftsbeteiligten und Operator eindeutig sein. Ist dies nicht der Fall, so wird ein Fehler „Duplicate MessageID" zurückgeliefert.

Code = WS05
Descr = Duplicate messageID
Point = messageID
OrigVal = *original messageID*

WS06 – User not permissible for operator

Ein Portaluser darf nur die ihm zugewiesenen VIDs benutzen, sendet zum Beispiel ein User eine Verbrauchsteueranmeldung für eine nicht zugewiesene VID, so wird diese abgelehnt. Es kann auch sein dass die VID dem System überhaupt nicht bekannt ist, dies wird aber aus Gründen der Sicherheit nicht unterschieden

Code = WS06
Descr = User not permissible for operator
Point = operator
OrigVal = *original user and operator*

WS07 – Operator is not permitted for the procedure

Jede gesendete Nachricht ist einem Verfahren (EMCS, EVA, …) zugeordnet und die Bewilligung bestimmt welche Verfahren erlaubt sind. Wird zum Beispiel mit einer nationalen Bewilligung eine EMCS-Meldung so wird diese abgewiesen.

Code = WS07
Descr = Operator is not permitted for the procedure
Point = MsgType
OrigVal = *original msgType*

WS08 – Invalid message

Tritt eine Verletzung des Schemas ein, so wird das negative Ergebnis der Validierung in einer Fehlernachricht verpackt zurückgegeben.

Code = WS08
Descr = Invalid message
Point = *Zeile und Spalte des OriginalXML + SchemaFehlermeldung*
OrigVal = null

WS09 – Method not supported for this client

Das VipWebservice wird für verschiedene Mandanten und Systeme eingesetzt. Nicht bei allen Mandanten/Clients sind alle Webservice Methoden erlaubt. Sollte eine für das jeweilige System nicht erlaubte Webservice Methode aufgerufen werden, so wird dieser Aufruf mit dieser Fehlermeldung abgewiesen.

Code = WS09
Descr = Method not supported for this client
Point = Bean
OrigVal = *ClientID bzw. Mandantenkennung des aufgerufenen VipWebservices*

Anhang

WSDL der Schnittstelle (VipWebservice.wsdl)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<definitions
        targetNamespace="urn:http://vst.bmf.gv.at/vip/v01"
        name="VipWebservice" xmlns:tns="urn:http://vst.bmf.gv.at/vip/v01"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:wsu=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
        xmlns="http://schemas.xmlsoap.org/wsdl/">

 <types>
  <xsd:schema>
   <xsd:import namespace="urn:http://vst.bmf.gv.at/vip/v01" schemaLocation="VipWebservice_schema1.xsd"/>
  </xsd:schema>
 </types>

 <message name="getMessagesForVID">
  <part name="parameters" element="tns:getMessagesForVID"/>
 </message>

 <message name="getMessagesForVIDResponse">
  <part name="parameters" element="tns:getMessagesForVIDResponse"/>
 </message>

 <message name="testService">
  <part name="parameters" element="tns:testService"/>
 </message>

 <message name="testServiceResponse">
  <part name="parameters" element="tns:testServiceResponse"/>
 </message>

 <message name="sendMessage">
  <part name="parameters" element="tns:sendMessage"/>
 </message>

 <message name="sendMessageResponse">
  <part name="parameters" element="tns:sendMessageResponse"/>
 </message>

 <portType name="Webservice">
  <operation name="getMessagesForVID">
   <input message="tns:getMessagesForVID"/>
   <output message="tns:getMessagesForVIDResponse"/>
  </operation>
```

```
    <operation name="testService">
      <input message="tns:testService"/>
      <output message="tns:testServiceResponse"/>
    </operation>

    <operation name="sendMessage">
      <input message="tns:sendMessage"/>
      <output message="tns:sendMessageResponse"/>
    </operation>
  </portType>

  <binding name="VipWebservicePortBinding" type="tns:Webservice">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="getMessagesForVID">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>

    <operation name="testService">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>

    <operation name="sendMessage">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

  <service name="VipWebservice">
    <port name="VipWebservicePort" binding="tns:VipWebservicePortBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
    </port>
  </service>
</definitions>
```

XSD der Schnittstelle (VipWebservice_schema1.xsd)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema
        version="1.0"
        targetNamespace="urn:http://vst.bmf.gv.at/vip/v01"
        xmlns:tns="urn:http://vst.bmf.gv.at/vip/v01"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="getMessagesForVID" type="tns:getMessagesForVID"/>

 <xs:element name="getMessagesForVIDResponse" type="tns:getMessagesForVIDResponse"/>

 <xs:element name="sendMessage" type="tns:sendMessage"/>

 <xs:element name="sendMessageResponse" type="tns:sendMessageResponse"/>

 <xs:element name="testService" type="tns:testService"/>

 <xs:element name="testServiceResponse" type="tns:testServiceResponse"/>

 <xs:complexType name="sendMessage">
  <xs:sequence>
   <xs:element name="input" type="tns:VipWebserviceBean" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="VipWebserviceBean">
  <xs:sequence>
   <xs:element name="operator" type="xs:string"/>
   <xs:element name="system" type="xs:string"/>
   <xs:element name="contentType" type="xs:int"/>
   <xs:element name="messageType" type="xs:string" minOccurs="0"/>
   <xs:element name="messageID" type="xs:string" minOccurs="0"/>
   <xs:element name="message" type="xs:string" minOccurs="0"/>
   <xs:element name="attachment" type="xs:base64Binary" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="sendMessageResponse">
  <xs:sequence>
   <xs:element name="response" type="tns:VipWebserviceBean" form="qualified" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="testService">
  <xs:sequence/>
 </xs:complexType>
```

```
 <xs:complexType name="testServiceResponse">
  <xs:sequence>
   <xs:element name="response" type="xs:string" form="qualified" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="getMessagesForVID">
  <xs:sequence>
   <xs:element name="vid" type="xs:string" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="getMessagesForVIDResponse">
  <xs:sequence>
   <xs:element name="response" type="tns:VipWebserviceBean" form="qualified" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

XSD der Fehlernachricht (VipWebserviceError.xsd)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema
	targetNamespace="urn:http://vst.bmf.gv.at/vip/v01"
	version="1.0"
	 xmlns:tns="urn:http://vst.bmf.gv.at/vip/v01"
	xmlns:xs="http://www.w3.org/2001/XMLSchema">

	<xs:element name="VipWebserviceError" type="tns:VipWebserviceErrorType"/>

	<xs:complexType name="ErrorType">
		<xs:sequence>
			<xs:element name="Code" type="xs:string" />
			<xs:element name="Descr" type="xs:string" />
			<xs:element name="Point" type="xs:string" />
			<xs:element name="OrigVal" type="xs:string" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="VipWebserviceErrorType">
		<xs:sequence>
			<xs:element name="Error" type="tns:ErrorType" form="qualified" nillable="false" minOccurs="1"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```