AIS – Electronic Import Customs Procedures
KKK2 subproject
Interface specification

DISTRIBUTION LIST

NAME	TITLE	A/C/I
Brigadier general Dr. Elek Mózs, director general of revenue	Project owner	I
Colonel Zsolt Szabó, customs director	Business project manager	I
Colonel Vilmos Vályi-Nagy, head of department	IT project manager	A
Zoltán Ancsin	Operative project manager	I
László Tomsics	Quality manager	V
János Orsolya	KKK2 subproject manager	R
Major Tibor Buzási	CDPS Work group leader	I
Captain Szabolcs Kajtár	External relations work group manager	I
Lieutenant colonel Ferenc Sebők	e-service project manager	V
Attila Szigyártó	Operations manager	V

(R: Responsible, A = Approval, V = Verification, I = Information)

STATUS DATA SHEET

Issue	Received	Status	Date	Name & signature
1.12	-	Based on the approved quality criteria, version 1.12 of the product reported as ready.	06-08-2007	János Orsolya
1.12	06-08-2007	Product verified for IT purposes.	___-08-2007	Ferenc Sebők
1.12	06-08-2007	Product verified for IT purposes.	___-08-2007	Attila Szigyártó
1.12	06-08-2007	Product verified; the opinion of the above officers entitled for verification summarized in the proposal for decision.	___-08-2007	László Tomsics

DOCUMENT HISTORY

Issue	Date	Description	Changed parts
1.0	30-01-2007	Version proposed by the subproject for approval.	Overall document
1.1	05-03-2007	Version adjusted and corrected according to the Decision Report including the changes deemed necessary in the course of implementation.	Overall document
1.11	20-04-2007	Clarified version	4.1.2.2, 5.1.2.2.1, 5.1.2.3.2, 5.1.3.1
1.12	01-08-2007	Web service Download() changed.	5.1.3.1 Download 7.1.2 Business faults

1.14	10-10-2007	Experiences gathered in the meantime documented and clarified	3.1.2, 4.1.3.1.1, 5.1.2.3.1, 5.1.3.1, 5.1.3.2, 5.1.5, 7.
1.15	26-11-2007	Clarified version	5.1.1.1, 7.1.2
1.16	11-12-2007	Web service changed: HTTP User Agent, SOAP version, Connection log,	5.1.1, 5.1.4.1
1.17	30-07-2008	Attachment Envelope introduced	3.1.2, 4.2, 4.6, 12.3
1.18	14-10-2008	Example messages updated	4.6.2
1.19	07-09-2009	EORI identifier defined in KKK2	4.1.2.1, 4.1.3.1.7

Table of Contents

1. Purpose and scope

The purpose is to precisely define the interface for the connected customs programs of the Clients.

2. Purpose of message exchange

The exchange of messages is aimed at providing an option for the Clients to present in electronic format data to the customs office concerning specific procedures prescribed by laws and regulations. If data are presented by electronic means the customs offices shall send electronic responses (notices) to the Clients.

3. Components of the message exchange

3.1 Concepts

3.1.1 VP Envelope

Messages posted within the system are packed in XML envelopes. The envelope contains the address and every other information required for successful delivery and processing. All the participants involved in mailing must support the use of the envelope since this is the only way to ensure a consistent operation.

3.1.2 Attachment Envelope

In certain cases the enveloped original message is attached one or more other messages that need to be forwarded together with the original message. Both the further messages belonging to the original message and the original message itself may be embedded in VPEnvelope using (AttachmentEnvelope).

3.1.3 Channel

A channel is a logical means where the KKK user may send messages to others and receive messages sent to him. Typically a separate channel is created for every transaction type (e.g. “Filing the Single Administrative Document”). Each channel shall be used only for the traffic of a specific message type.

3.1.4 Client

For the purposes of this document “Client” shall mean any legal person, entity with no legal personality or any natural person having a regulated relation with the Hungarian Customs and Finance Guard and conducting electronic data exchange with the Hungarian Customs and Finance Guard through the KKK2 system (in this sense for instance banks and relevant authorities are included).

In the course of the registration process the Client or his representative must show up at the customs office where the customs office user helps to clearly match the Client with the relevant element of the Standard Client Records (EÜC) and make sure that the Client has adequate privileges for electronic message exchange.

3.1.5 KKK user

The KKK user is a real or technical user registered through the web-based system, provided with adequate identifiers, who is capable of conducting specific electronic data exchange through the KKK2 system.

There are three basic levels of users:

1. The *basic KKK user* has registered on the web-based platform, provided some personal data (name, e-mail address, etc.), but is not related or assigned to the Client (although may become a Primary KKK user assigned to the Client – see next section).
2. The Primary KKK user assigned to the Client is the primary equivalent of a real Client appearing in the KKK2 electronic system that has been registered as a basic KKK user and has presented the activation code received in the customs office in the course of the Client registration and has received thus all the channel rights or privileges assigned to the Client and is able to transfer such privileges to the secondary users created by him and may access the messages sent to his own secondary users.
3. *The secondary KKK users assigned to the Client* are users that may be linked directly to the Client that send and receive the messages of the Client. They are created and deleted basically by the primary user.

3.1.6 KKK-Web

The outer part of the KKK2 system is the KKK-Web, which in essence resembles a mailbox system. The main task of KKK-Web is to temporarily store the messages received from a KKK user or sent to a KKK user, provide data connection options for the KKK user, as well as receive and forward messages to KKK-Közvetítő (KKK-sender).

The data link platform may be accessed through a browser or the web service.

The primary data exchange platform is the web service: through it the user programs operated by the Client and linked to the KKK2 (prepared by software houses) may exchange data.

On the web platform – using a browser – the user can administer the data exchange and the related services.

3.1.7 KKK-Közvetítő

The task of the KKK-Közvetítő (sender) is to route messages between the KKK-Web and the other systems. It takes messages from the KKK-Web and forwards each of them to the relevant business system and vice versa, forwards the messages created in the business system to the KKK-Web user.

The KKK-Közvetítő archives the messages arriving from the KKK-Web, checks the address, determines on the basis of EÜC data whether the sender was authorized to send the message or not, checks the content for format, sends a receipt and conveys it to the relevant business system.

When response messages are sent from the business system, the KKK-Közvetítő checks the address, archives and then conveys the message to KKK-Web.

3.1.8 EÜC

The purpose for creating a Unique Client Address Register (EÜC) is to keep record of the entities entitled to perform electronic data exchange through KKK2, which means a unique input for the Clients. It supports registration and creates a registry that describes all the managed channels, Clients and users, as well as their interrelations.

3.1.9 VP Receipt

The receipt message is a notice created within the KKK2 system that informs the user – for instance – that his message has been received, what was the result of the checks performed on the message, and whether or not the message was conveyed to the business system.

3.1.10 VP Fault

The fault message is a notice generated within the KKK2 system that informs the user that a fault was found by KKK2 when the message was processed.

3.1.11 Business systems

These include systems that are currently operating or being developed and that process the message presented by the user and received through the KKK-Közvetítő system.

3.1.12 Message type

The message types identify the XML schemas to be used for the conveyance of specific messages. Each specific message type may be sent only on the related channel(s).

3.1.13 Business error

In the course of operation of the KKK-Web some of the activities performed by the user may lead to errors: the user may try to execute in inadequate operation or enter an improper piece of data in the system.

The KKK-Web has been prepared in advance to handle these errors and reacts in a programmed, synchronous way to inform the user about the reason of the error.

3.1.14 Exception

When unexpected errors occur the KKK-Web generates an Exception, which is logged. For safety reasons the user is not informed about the reason of the error and the way to solve it, only about the origination of an error.

3.2 Procedures

3.2.1 Sending in a message

Client:

- Prepares a message, packs it in an envelope and sends it to KKK-Web

KKK-Web:

- Receives the message

- Performs initial checks to verify the existence of the minimum conditions required for reception
- If an error is found, reports the error to the caller in a synchronous way and stores not the message. The error is reported:
    - On the web platform: by a fault message
    - In the web service: by a status-type parameter including the descriptor of the business error
- If no error is found, it stores the message and generates a receipt, which is placed in the appropriate user channel.
  In the receipt message the Event tag will be “Receive”.

KKK-Közvetítő:

- Receives the message from KKK-Web
- Performs comprehensive checks
- Determines the business system and sends the message to it.
- If an error is found, a fault message is prepared and stored in the relevant user channel.
- If no error is found, a receipt is prepared and stored in the appropriate user channel. In the receipt message the Event tag will be “Delivery”.
- Delivers the generated receipt or fault message to the KKK-Web.

Business system:

- Receives the message and takes it out from the envelope and begins processing
- At specific processing points generates its own receipts and sends them through KKK-Közvetítő.

3.2.2 Sending out a message

Business system:

- Prepares a message (it may be its own receipt or a business message), puts it into an envelope, places the address and sends it to KKK-Közvetítő.

KKK-Közvetítő:

- Receives the message
- Performs comprehensive checks
- Delivers the message to KKK-Web

KKK-Web:

- Receives the message from KKK-Közvetítő

Client:

- Downloads the enveloped message from KKK-Web
- Deletes the messages in KKK-Web

4. KKK2 messages

4.1 VP Envelope

4.1.1 Review

In the course of data transfer between the KKK, the business systems and the Client the messages originating from the Client or a business system are placed in an envelope, which may attach specific descriptive (meta) data to the message.

Specific properties may be completed in the envelope by message type and directions. The sender (conveyor) of the message is responsible for delivering a kind of message to the recipient whose envelope features all the relevant properties.

4.1.2 Schema

The envelope consists of two major parts: the Header and the Body. The header includes the descriptive parts while the body includes the embedded message. Both parts must be specified.

4.1.2.1 Header

Properties of the header:

Description in Hungarian	Name of Tag	Compulsory or Not	Remark
Message ID	MessageID	+	Unique message ID
Hivatkozott üzenet azonosítója	RelatesTo	-	The message ID value for the relevant message
Message Type	MessageType	+	Type of the message root element
Küldő	From	+	Technical name of sending user or channel
Címzett	To	-	The addressee of the message (technical name of the channel or user). The Client must specify the technical name of the channel, but it is not compulsory if it is sent as a response of the business system. In this case the relevant message ID must be stated.
Válaszcím	ReplyTo	-	The reply must be sent here. If not specified, the response goes to the sender identified in the From field.
Képviselet	OnBehalfOf	-	The Client on behalf of whom the Sender has

			sent the message
Létrehozás dátuma	Created	+	The date the message was created
Feltöltés dátuma	Uploaded	-	The date the message was uploaded. It is not attached to the message by the KKK rather than by the Client.
Egyéb tulajdonságok	Properties	-	Specific message properties by name and value

Example XML fragment:

```
<vpe:Header>
        <vpe:MessageID>uuid:7fc16c00-ecb1-11da-921d-0002a5d5c51b</vpe:MessageID>
        <vpe:RelatesTo>uuid:12234554-e341-1522-95e4-0d45f2d5467b</vpe:RelatesTo>
        <vpe:MessageType>http://schemas.vam.gov.hu/TC32/1.0#CD225A</vpe:MessageType>
        <vpe:From>http://vam.gov.hu/TC32</vpe:From>
        <vpe:To>user:123984</vpe:To>
        <vpe:ReplyTo></vpe:ReplyTo>
        <vpe:OnBehalfOf>vpid:12345</vpe:OnBehalfOf>
        <vpe:Created>1999-05-31T13:20:00.000-05:00</vpe:Created>
        <vpe:Properties >
                <vpe:Property name="name_0" >Property_0</vpe:Property>
        </vpe:Properties>
</vpe:Header>
```

Message exchange requiring Eori identification:

```
<vpe:Header>
        <vpe:MessageID>uuid:7fc16c00-ecb1-11da-921d-0002a5d5c51b</vpe:MessageID>
        <vpe:RelatesTo>uuid:12234554-e341-1522-95e4-0d45f2d5467b</vpe:RelatesTo>
        <vpe:MessageType>http://schemas.vam.gov.hu/TC32/1.0#CD225A</vpe:MessageType>
        <vpe:From>http://vam.gov.hu/TC32</vpe:From>
        <vpe:To>user:123984</vpe:To>
        <vpe:ReplyTo></vpe:ReplyTo>
        <vpe:OnBehalfOf>eori:AT1234</vpe:OnBehalfOf>
        <vpe:Created>1999-05-31T13:20:00.000-05:00</vpe:Created>
        <vpe:Properties >
                <vpe:Property name="name_0" >Property_0</vpe:Property>
        </vpe:Properties>
</vpe:Header>
```

4.1.2.2 Body

The message is embedded within the Body tag and may be:

- business message
- receipt message
- fault message

Example XML fragment:

```
<vpe:Body>
        <CD225A xmlns="http:"//schemas.vam.gov.hu/TC32/1.0
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

        <filenev>xyz001.xml</filenev>
        <MesSenMES3>abcde</MesSenMES3>
        <grnszam>adat</grnszam>
        …
        …
        </CD225A>
</vpe:Body>
```

In this example a business message including the CD225 root element has been embedded into the envelope.

4.1.3 Titles and marks used in Envelopes

4.1.3.1.1 Message ID

It is unique message ID that clearly identifies the envelopes message. The ID must be generated by the maker of the envelope.

Clear identifications means that each message ID value may be assigned only to one enveloped message within the KKK system and no further messages may be sent using the same identifier.

The value of the identifier must be generated in accordance with the UUID (Universally Unique Identifier - http://www.ietf.org/rfc/rfc4122.txt) standard.

Type: anyURI (http://www.w3.org/TR/xmlschema-2/#anyURI)

Creation: ”uuid:” + GENERÁLT_UUID_ÉRTÉK_STRING_FORMÁBAN

Example value:

```
uuid:59efb860-ecb1-11da-9ad0-0002a5d5c51b
```

4.1.3.1.2 Relates To

When a response message is sent, the RelatesTo field indicates the message that is being answered. The type and creation are as for the MessageID field.

4.1.3.1.3 Message Type

The message type identifies the type of the message embedded in the body.

Type: anyURI (http://www.w3.org/TR/xmlschema-2/#anyURI)

Creation:

- If the embedded message has a name space:
  name space of embedded message + “#” + name of root element of the embedded message
- If the embedded message has no name space:
  name of the root element of the embedded message

Example: If the embedded message has a name space

Embedded message:

```
<CD225A xmlns="http:"//schemas.vam.gov.hu/TC32/1.0 >
…
</CD225A>
```

Example value:

```
http://schemas.vam.gov.hu/TC32/1.0#CD225A
```

Example: If the embedded message has no name space

Embedded message:

```
<CD225A >
        …
</CD225A>
```

Example value:

```
CD225A
```

4.1.3.1.4 From

It is the identifier of the sender or the system sending the message.

Type: anyURI (http://www.w3.org/TR/xmlschema-2/#anyURI)

Creation:

- By the Client: “user:” + KKK2 user ID
- By the business system: technical name of channel

4.1.3.1.5 To

The To field identifies the addressee of the message; it may be a business system or a user .

Type: anyURI (http://www.w3.org/TR/xmlschema-2/#anyURI)

Creation:

- By the Client: technical name of channel
- By the business system: completion not obligatory.
  - If completed: “user:” + KKK2 user identifier.
  - If not completed, the addressee should be searched using RelatesTo based on the incoming messages of KKK-Közvetítő. The data generated this way would be incorporated in the above format into the message.

4.1.3.1.6 Reply To

The ReplyTo field identifies the addressee of the reply message, which may be a business system or a user.

Type: anyURI (http://www.w3.org/TR/xmlschema-2/#anyURI)

Creation:

- By the Client: completion not obligatory, if it is completed: “user:” + KKK2 user identifier.
- By the business system: completion not required.

4.1.3.1.7 On Behalf Of

The OnBehalfOf field shows the representative: the sender that sends the message on behalf of the Client.

Type: anyURI (http://www.w3.org/TR/xmlschema-2/#anyURI)

Creation:

- Client ID Type + “:” + value of the identifier
- Client ID types that may be used:
  - vpid – “GTR” Client identifier, domestic identifier for operators
  - eori – “GTR” Client identifier, operator identifier used in the EU
  - adoig – Tax ID (VAT ID)
  - adoazon – Tax ID code
  - egyebazon – Other identifier

Example value:

```
vpid:HU12345
```

```
eori:AT1234
```

4.1.3.1.8 Created

It is the date when the message is created.

Type: xs:dateTime (http://www.w3.org/TR/xmlschema-2/#dateTime)

Example value:

```
1999-05-31T13:20:00.000-05:00
```

4.1.3.1.9 Uploaded

It is the date when the message is uploaded. It is assigned to the message by KKK and not by the Client.

Type: xs:dateTime (http://www.w3.org/TR/xmlschema-2/#dateTime)

Example value:

```
1999-05-31T13:20:00.000-05:00
```

4.1.3.1.10 Properties

Under Properties, it is possible to take out individual properties from the message into the header of the envelope. There may be 0, 1 or more children property elements under the Properties element.

Each Property element means one property. Each Property element has the following features:

Name in Hungarian	Name in XML	XML type	Compulsory?	Data type
Név	Name	Attribute	+	xs:string
Érték	-	Data content of Property tag	+	xs:string

Example XML excerpt:

```
<vpe:Properties xmlns:vpe="http://schemas.vam.gov.hu/VPEnvelope/1.0">
        <vpe:Property name="name_0" >Property_0</vpe:Property>
        <vpe:Property name="name_1" >Property_1</vpe:Property>
        <vpe:Property name="name_2">Property_2</vpe:Property>
</vpe:Properties>
```

4.2 Attachment Envelope

4.2.1 Description

The message coming from the Client or originating from a business system in the course of data exchange between KKK, the business systems and the Client is inserted into an envelope message (VPEnvelope), which may be used to link descriptive (meta) data to the message. In certain cases the enveloped original message is attached one or more other messages that need to be sent together with the original message. These further messages related to the original message and the original message itself may be embedded in VPEnvelope using the attachment envelope (AttachmentEnvelope).

Different properties may be completed in the attachment envelope by message type and direction. The person handing over the message is responsible for making sure that the attachment envelope includes all the properties properly completed.

The need for using the AttachmentEnvelope as well as the number of attachments and their contents must be defined by the Interface Specification of the business system.

4.2.1.1 Attachment Envelope (root) element

The attachment envelope consists of three major elements: AttachmentHeaders, Body and AttachmentContents. The headers contain the data describing the attached messages, the body contains the original message, and the attached messages contain the messages attached to the business message.

All the three parts must be defined.

4.2.1.2 Headers of attached messages (AttachmentHeaders)

The header includes zero or more header elements. Each header element describes an attached message, whose data content is attached at the appropriate AttachmentContent element of the AttachmentContents.

4.2.1.3 Attachment Header

The header describes the properties of each attached message. The content of the header is as follows:

Name in Hungarian	Name of tag	Compulsory?	Remark
Csatolásazonosító	AttachmentID	+	Unique identifier within AttachmentEnvelope
Csatolt üzenet MIME típus	MimeType	+	
Csatolt üzenet tárolásának módja	Format	+	The value can be Binary, XML
Csatolt üzenet neve	Name	-	In general it is file name
Csatolt üzenethez tartozó megjegyzés vagy leírás	Comment	-	
Egyéb tulajdonságok	Properties	-	Unique message properties in a set of name/value.

4.2.1.4 Body

The original message is embedded in the body tag and has attachments.

4.2.1.5 Attachment Contents

The attached message includes zero or more attached messages.

4.2.1.6 Attachment Content

The AttachmentContent includes the data content of each attachment. The attached message has a specific AttachmentID property, which coincides with the value of the AttachmentID element of the AttachmentHeader belonging to the message.

The data content may be stored in a binary or XML format. The storage method is set in the Format of the AttachmentHeader element belonging to the message.

If the value of Format is Binary the data content is stored within AttachmentContent in a BinaryData element featuring BASE64 coding.

If the value of Format is XML the data content is directly embedded in the XmlData element within the AttachmentContent.

4.2.2 Joint use of VPEnvelope and AttachmentEnvelope

The AttachmentEnvelope must be embedded within the Body of the VPEnvelope.

4.2.2.1 Message Type

In a conventional case, when the VPEnvelope includes the original message, the message type should be created on the basis of the original message included in the Body tag of the VPEnvelope.

Due to the AttachmentEnvelope embedding, the message type should contain the name space and root element name of the AttachmentEnvelope in the VPEnvelope, which would mask the original message. In message type, information concerning the original message must be provided in VPEnvelope, so in case of AttachmentEnvelope embedding the message type must be created from the name space and root element of the message contained in the AttachmentEnvelope.

4.3 KKK2 VPReceipt

4.3.1 Review

In the case of a receipt message the Receipt type defined in the "http://schemas.vam.gov.hu/VPReceipt/1.0" name space is enveloped and sent back.

The RelatesTo value in the envelope will be set with reference to the MessageID value of the incoming message.

The event type is denoted by the Event tag, whose value can be one of the values enumerated in ReceiptEventEnum.

No attachment belongs to the receipt message.

4.3.2 Scheme

4.3.2.1 Receipt

The Receipt tag is the root element of the receipt message. It may contain the following children elements:

Name in Hungarian	Name of tag	Compulsory?	Remark
Nyugta típus	Event	+	Informs concerning the preparation of the receipt.
Részletes információ	Detail	-	It is not currently defined.

4.3.2.2 Event

The Event tag takes a value from the ReceiptEventEnum list. Currently the following elements are defined for ReceiptEventEnum type:

Name of value	Meaning
Receive	KKK2 has received the Client's message.
Delivery	KKK2 has delivered the Client's message.

4.3.2.3 Detail

The Detail tag may be used later on to indicate further information pieces in the receipt message.

4.4 KKK2 error message (VPFault)

4.4.1 Review

In the case of a Fault message the Fault type defined in the "http://schemas.vam.gov.hu/VPFault/1.0" name space is enveloped and sent back.

In the envelope of the fault message the RelatesTo value is referenced to the value of the MessageID of the incoming message.

The category of fault is indicated by the Code tag, whose value must be one of the values listed in FaultCodeEnum.

The fault can be identified with the Subcode tag: the Value tag contains the fault code while the Text tag includes a textual description of the fault.

No attachment belongs to the fault message.

4.4.2 Scheme

4.4.2.1 Fault

The Fault tag is the root element of the fault message. It may have the following children elements:

Name in Hungarian	Name of tag	Compulsory?	Remark
Hibakategória	Code	+	
Hibaleíró	Subcode	-	
Részletes információ	Detail	-	Not currently defined.

4.4.2.2 Code

The Code tag takes a value from to FaultCodeEnum list. Currently the following values are defined for the FaultCodeEnum type:

Name of value	Meaning
InvalidXml	Invalid XML
SenderMismatch	Invalid sender
MessageTypeMismatch	Invalid message type

RoutingDenied	Routing denied
InvalidDelegation	Invalid delegation
VersionMismatch	Mismatched message version
DuplicateGuid	The message ID is already present in the system
OtherFault	Other fault

4.4.2.3 Subcode

The Subcode element is the description of the fault message. It may include the following children elements:

Name in Hungarian	Name of tag	Compulsory?	Remark
Hibakód	Value	+	Allows identification of the fault from the program.
Szöveges leírás	Text	+	A message intelligible for the user.
Beágyazott hibaleíró	Subcode	-	

4.4.2.4 Detail

The Detail tag may be used later on to indicate further information pieces in the fault message.

4.5 Business message

See the Interface Specification of the relevant business system.

4.6 KKK2 message examples

Messages are always enveloped so that a message sent to KKK2 must comply with the schema of both the envelope and the embedded message(s). The KKK2 scheme performs checks and when the message fails to comply with the relevant schemes, it sends back a fault message.

4.6.1 Envelope message

4.6.1.1 Receipt message

Example for an enveloped receipt message sent by KKK-Web to the Client when receives the message sent by the Client:

```
<?xml version="1.0" encoding="utf-8"?>
<vp:VPEnvelope xmlns:vpr="http://schemas.vam.gov.hu/VPReceipt/1.0"
xmlns:vp="http://schemas.vam.gov.hu/VPEnvelope/1.0">
	<vp:Header>
		<vp:MessageID>uuid:d0b24e0e-f454-4656-9fdf-054a241ab81e</vp:MessageID>
		<vp:RelatesTo>uuid:2a9c439d-8530-178d-e040-000ad8e80bf1</vp:RelatesTo>
	<vp:MessageType>http://schemas.vam.gov.hu/VPReceipt/1.0#Receipt</vp:MessageType>
		<vp:From>http://vam.gov.hu/KKK_WEB</vp:From>
		<vp:To>userid:10000045</vp:To>
		<vp:Created>2007-03-01T13:52:40.3825253+01:00</vp:Created>
	</vp:Header>
	<vp:Body>
		<vpr:Receipt xmlns:vpr="http://schemas.vam.gov.hu/VPReceipt/1.0">
```

```
                        <vpr:Event>Receive</vpr:Event>
                </vpr:Receipt>
        </vp:Body>
</vp:VPEnvelope>
```

4.6.1.2 Business message

The example presents the technical use of VPEnvelope - business message (in other words, it not necessarily contains real business data).

In this example the "http://schemas.vam.gov.hu/CDPS/ERT/1.0#ERT" type business message may be seen in an envelope.

The content of the business message and the parameters of the envelope are defined by the Interface Specification of the relevant business system.

```
<?xml version="1.0" encoding="utf-8" ?>
<vp:VPEnvelope xmlns:vp="http://schemas.vam.gov.hu/VPEnvelope/1.0">
        <vp:Header>
                <vp:MessageID>uuid:5312D58B-2CBC-88E1-E040-000A23E81401</vp:MessageID>
                <vp:RelatesTo>uuid:59efb860-ecb2-11da-9ad1-0002a5d52295</vp:RelatesTo>
                <vp:MessageType>http://schemas.vam.gov.hu/CDPS/ERT/1.0#ERT</vp:MessageType>
                <vp:From>http://vam.gov.hu/CDPS</vp:From>
                <vp:To>userid:10000045</vp:To>
                <vp:Created>2008-07-28T12:17:43.861+02:00</vp:Created>
        </vp:Header>
        <vp:Body>
                <ERT xmlns="http://schemas.vam.gov.hu/CDPS/ERT/1.0" VPID="HU0000005582"
KOT_AZON="20050709102255" DATUM="20080728121741" CEL_SYSTEM ="CDPSERT" VHKOD="HU100000"
uzenetkuldo="123456789012345678901234">
                        <ERTESITES>
                                <CDPS_ID>HU100000242007A01436</CDPS_ID>
                                <UZENET>E20: Manuális feldolgozás  miatt a vámeljárás
lefolytatása érdekében haladéktalanul vegye fel a kapcsolatot a fő/vámhivatallal!</UZENET>
                                <DATUM>20080728121740</DATUM>
                        </ERTESITES>
                </ERT>
        </vp:Body>
</vp:VPEnvelope>
```

4.6.2 Envelope message with attachment

The example presents the technical use of VPEnvelope - AttachmentEnvelope - business message (in other words, it not necessarily contains real business data).

The need for using the AttachmentEnvelope, the number and content of the attachments, the content of the business message and the parameters of the envelope are defined by the Interface Specification of the relevant business system.

In the example, the "http://schemas.vam.gov.hu/CDPS/HAT/1.0#HAT" type business message has two attachments. The first attachment is a PDF file and the second one is an XML file (which in reality would be nonsense for this business message).

```
<?xml version="1.0" encoding="utf-8" ?>
<vp:VPEnvelope xmlns:vp="http://schemas.vam.gov.hu/VPEnvelope/1.0">
 <vp:Header>
  <vp:MessageID>uuid:59315217-632D-3DBF-E040-11AC4C0222F4</vp:MessageID>
  <vp:RelatesTo>uuid:ec1adba8-d4e5-4101-b41a-afb9256b57a3</vp:RelatesTo>
  <vp:MessageType>http://schemas.vam.gov.hu/CDPS/HAT/1.0#HAT</vp:MessageType>
  <vp:From>http://vam.gov.hu/CDPS</vp:From>
  <vp:Created>2008-10-14T07:59:37.340+02:00</vp:Created>
  <vp:Properties>  </vp:Properties>
 </vp:Header>
 <vp:Body>
  <ae:AttachmentEnvelope xmlns:ae="http://schemas.vam.gov.hu/AttachmentEnvelope/1.0">
   <ae:AttachmentHeaders>
    <ae:AttachmentHeader>
     <ae:AttachmentID>1</ae:AttachmentID>
```

```
      <ae:MimeType>application/pdf</ae:MimeType>
      <ae:Format>Binary</ae:Format>
      <ae:Name>E0150047A023282</ae:Name>
      <ae:Comment>Elektronikus határozat állomány</ae:Comment>
     </ae:AttachmentHeader>
    </ae:AttachmentHeaders>
    <ae:Body>
     <HAT xmlns="http://schemas.vam.gov.hu/CDPS/HAT/1.0" VPID="HU0000015413"
KOT_AZON="AIS_0020_agrim 20071005133310198" DATUM="20081014075936" CEL_SYSTEM ="CDPSHAT"
VHKOD="HU015000" uzenetkuldo="Kaiser">
      <ERTESITES>
       <CDPS_ID>HU015000242007E11128</CDPS_ID>
       <UZENET>E23:  A HU015000242007E11128  CDPS_ID-hez tartozó elektronikus határozat
elkészült, jelen értesítésem mellékleteként letölthető!</UZENET>
       <DATUM>20081014075936</DATUM>
      </ERTESITES>
     </HAT>
    </ae:Body>
    <ae:AttachmentContents>
     <ae:AttachmentContent attachmentID="1" >
      <ae:BinaryData>
JVBERi0xLjQKJeLjz9MKMiAwIG9iaiA8PC9MZW5ndGggNjQvRmlsdGVyL0ZsYXRlRGVjb2RlPj5z
dHJlYW0KeJwr5HIK4TI2U7AwMFMISeEyUNA1tAAx9N0MFQyNFELSuDQ8UnNy8hXC84tyUjRDsoBK
DEAKXEO4ArkAugIOjwplbmRzdHJlYW0KZW5kb2JqCjQgMCBvYmo8PC9QYXJlbnQgMyAwIFIvQ29u
dGVudHMgMiAwIFIvVHlwZS9QYWdlL1Jlc291cmNlcz8L1Byb2NTZXQgWy9QREYgL1RleHQgL0lt
YWdlQiAvSW1hZ2VDIC9JbWFnZUldL0ZvbnQ8PC9GMSAxIDAgUj4+Pj4vTWVkaWFCb3hbMCAwIDU5
NSA4NDJdPj4KZW5kb2JqCjEgMCBvYmo8PC9CYXNlRm9udC9IZWx2ZXRpY2EvVHlwZS9Gb250L0Vu
Y29kaW5nL1dpbkFuc2lFbmNvZGluZy9TdWJ0eXBlL1R5cGUxPj4KZW5kb2JqCjMgMCBvYmo8PC9U
eXBlL1BhZ2VzL0NvdW50IDEvS2lkc1s0IDAgUl0+PgplbmRvYmoKNSAwIG9iajw8L1R5cGUvQ2F0
YWxvZy9QYWdlcyAzIDAgUj4+CmVuZG9iago2IDAgb2JqPDwvUHJvZHVjZXIoaVRleHQgMi4xLjAt
U05BUFNIT1QgXChieSBsb3dhZ2llLmNvbVwpKS9Nb2REYXRlKEQ6MjAwODAzMjcxNzI3NDYrMDEn
MDAnKS9DcmVhdGlvbkRhdGUoRDoyMDA4MDMyNzE3Mjc0NiswMScwMCcpPj4KZW5kb2JqCnhyZWYK
MCA3CjAwMDAwMDAwMDAgNjU1MzUgZiAKMDAwMDAwMDMwMSAwMDAwMCBuIAowMDAwMDE1IDAw
MDAwIG4gCjAwMDAwMDAzODggMDAwMDAgbiAKMDAwMDE0NSAwMDAwMCBuIAowMDAwMDAwNDM4
IDAwMDAwIG4gCjAwMDAwMDA0ODIgMDAwMDAgbiAKdHJhaWxlcgo8PC9Sb290IDUgMCBSL0lEIFs8
ZWNmNjg1NmZiNDUxYzAzYjdiYWNiNThhZTNlN2ZlYmQ+PGNkZjJkZDIwODUzMWY2NTFiMzEyNTk0
MzQ5ZjdmZjA2Pl0vSW5mbyA2IDAgUi9TaXplIDc+PgpzdGFydHhyZWYKNjIyCiUlRU9GCg==
      </ae:BinaryData>
     </ae:AttachmentContent>
    </ae:AttachmentContents>
   </ae:AttachmentEnvelope>
  </vp:Body>
</vp:VPEnvelope>
```

4.6.3 KKK2 receipt message

Sample XML fragment without envelope:

```
<vpr:Receipt xmlns:vpr="http://schemas.vam.gov.hu/VPReceipt/1.0">
        <vpr:Event>Receive</vpr:Event>
        <vpr:Detail/>
</vpr:Receipt>
```

4.6.4 KKK2 fault message

Sample XML fragment without envelope:

```
<vpf:Fault xmlns:vpf="http://schemas.vam.gov.hu/VPFault/1.0">
        <vpf:Code>InvalidXml</vpf:Code>
        <vpf:Subcode>
                <vpf:Value>Value</vpf:Value>
                <vpf:Text>Text_0</vpf:Text>
        </vpf:Subcode>
</vpf:Fault>
```

5. Message exchange platforms of the KKK-Web application

5.1 KKK-Web web service platform

The application of the Client may upload and download messages on the web service platform.

5.1.1 Definition of a message protocol

The web service complies with the WSI Basic Profile version 1.1 standard, so the SOAP 1.1 protocol is supported.

The identification is made in the HTTP layer according to the BASIC authentication. For a successful identification the ID parameters should be input on the Client side:

- User identifier
- User password

The web service is only available through the SSL channel.

In order to identify the Client's software, when the web service is called, the text value of the HTTP User Agent should be set as the following string: software name; version; date of issue; manufacturer;

5.1.1.1 Server certificate

The SSL certificate of the web server backing the KKK2 web service comes from an official certificate authority.

The Client's software can trust in this certificate in two ways:

- The current SSL certificate may be made reliable through the interface of the programming frame system so that the program can contact the web server. When a certificate is substituted on the web server, another certificate should be trusted, which means that the program should be changed or configured on the Client's side.
- The programming framework system is related to the local system that stores the reliable certificate authorities in whose certificates the programming framework system has automatic trust.
  Under Java such storage is available in the "Java Control Panel" tools while the .NET framework system uses the Windows certificate warehouse.
  This solution has the advantage that in case of a change only the storage system needs to be administered: if the certificate authority is not yet included, it needs to be added. If the certificate authority has not changed or is already included in the storage facility, the system will automatically have trust in the new certificate.

Official certificates are normally valid for one or two years, and in certain cases may be extended once more. If the extension is not available, the Hungarian Customs and Finance Guard needs to buy new reliable certificates and must replace the old one. The developers must prepare for the change and make sure that such change produces the lowest deal of troubles to the Clients possible.

In function of the solution used for trusting in the certificate the Client's software may require program alteration or configuration.

5.1.2 API information

5.1.2.1 Introduction

The web service offers an option for uploading to the appropriate channel the enveloped messages of the KKK user and downloading the enveloped messages sent to the user.

To reduce the number of web service calls between the Client and the KKK2, messages are arranged in packets for the purposes of downloading. Each such packet may include one or more enveloped messages. In practice the packet means a message block including enveloped messages.

Subject to certain limitations, the Client may decide on the server the number of messages he wants to receive in a packet. When downloading, the packet features the messages that were uploaded earlier and have been not deleted yet by the Client (FIFO). After downloading the Client must confirm that the relevant messages have been successfully processed and thus may be deleted. As long as the Client fails to give a feedback about successful downloading the message will appear in the packet every time a download is made, and since the sizes of the packet are pre-determined, the Client may download further files only if reports that the downloading was successful. API provides the Client with a Queue-type management.

5.1.2.2 Data structures

5.1.2.2.1 Message

The enveloped XML messages may be uploaded or downloaded in a Message-type structure.

Its fields are as follows:

Name	Description
string ID	Message identifier
DateTime CreatedAt	Date of creation of message
byte[] Content	Message XML in binary format

Remark:

The Content field includes the content of the XML file (stream). The ID field is the unique identifier of the message that must coincide with the MessageID field featured in the header of the envelope (except the lead-in “uuid:” qualifier).

5.1.2.2.2 Status (Description of business faults)

The Status structure is used for storing business faults.

Its fields are as follows:

Name	Description
int ID	Business fault identifier. Faults may be identified in the program using the code.
string Message	Text of business fault.

Remark:

Business faults and technological faults are separated in the web service. A technological fault appears in the form of an Exception, while the business fault is defined in each case as a return value.

5.1.2.3 Methods

5.1.2.3.1 ConnectionTest

The connection test is to enable the Client to test – when the software is implemented or configured – whether the given authentication data and address are appropriate to reach the web service. At this point the Client may realize that the user or password or url are not adequate.

It is prohibited to use this function in operation or as a systematic check (every minute) since it may generate unnecessary traffic and payload.

Definition in language C#:

```
void ConnectionTest(out Status status);
```

Parameters:

Name	Type	Way	Description
Status	Status	Out	Business fault descriptor

Definition in Java:

```
public Status connectionTest() throws java.rmi.RemoteException;
```

Parameters:

Name	Type	Way	Description
	Status	Return value	Business fault descriptor

Remark:

The transaction is logged on the KKK Web page.

5.1.2.3.2 Delete – Deletion of downloaded messages

The Client reports that the relevant messages have been successfully downloaded.

Definition in language C#:

```
void Delete(string[] messageIDs, out Status[] statuses);
```

Parameters:

Name	Type	Way	Description
messageIDs	string[]	In	Block of message IDs. The Client receives the value of the message ID as the ID field of the message block when Download() is called.
Statuses	Status []	Out	Business fault descriptors. A return value is prepared for each message ID that is sent.

Definition in Java:

```
public Status[] delete(java.lang.String[] messageIDs) throws
java.rmi.RemoteException;
```

Parameters:

Name	Type	Way	Description

messageIDs	String[]	In	Message IDs
	Status []	Return value	Business fault descriptors. A return value is prepared for each message ID that is sent.

Remark:

With this transaction the relevant messages is marked that on the Client side it has been successfully downloaded and processed. After this point the message is not included anymore in the packet prepared at Download() and it is deleted on the KKK2 page.

5.1.2.3.3 Download – Downloading a message packet

Downloading of a message packet from the channel.

Definition in language C#:

```
void Download(string channelName, int maxMessageCount, out Message[] messages, out
Status status);
```

Parameters:

Name	Type	Way	Description
channelName	string	In	Technical name of channel
maxMessageCount	Int	In	The Client may receive up to this number of messages in a packet
Messages	Message[]	Out	Message packet Completed fields: ID, CreatedAt, Content
Status	Status	Out	Business fault descriptor.

Definition in Java:

```
public DownloadResponse download(java.lang.String channelName, int maxMessageCount)
throws java.rmi.RemoteException;
```

Parameters:

Name	Type	Way	Description
channelName	String	In	Technical name of channel
maxMessageCount	Int	In	The Client may receive up to this number of messages in a packet
	DownloadResponse	Return value	Class including the outgoing messages and status parameters referred to in the C# definition.

Remark:

The procedure returns a packet from the non-deleted messages available in the relevant channel and includes the oldest messages (FIFO). The maximum count of the packet components is the lower of the KKK2 configuration and the value specified by the Client. If there is no downloadable message, the message packet

will be empty. After downloading, the Client must report - using the Delete() procedure - what messages were successfully deleted.

5.1.2.3.4 Upload – Uploading of a message

Uploading of a message.

Definition in language C#:

```
void Upload(Message message, out Status status);
```

Parameters:

Name	Type	Way	Description
message	Message	In	Message
status	Status	Out	Business fault descriptor.

Definition in Java:

```
public Status upload(Message message) throws java.rmi.RemoteException;
```

Parameters:

Name	Type	Way	Description
message	Message	In	Upload message. Fields to be completed: ID, Content, CreatedAt
	Status	Return value	Business fault descriptor.

Remark:

The uploaded message will be assigned to a channel, and the channel is defined by the data included in the header of the envelope (To field).

5.1.3 Procedures

5.1.3.1 Downloading

As a first step on the Client side, the connection of the web service client must be set. The authentication data, the web service URL and possibly the web service client proxy settings must be provided.

After defining the connection settings and calling Download() it is possible to download the first message packet from the parametered channel of the user identified by the KKK-Web web service. After downloading the Client must securely save the messages on his side.

Once the downloaded messages have been processed, the Client must send feedback calling Delete() and report that the messages may be deleted on the KKK-Web side and that it is ready to download the next packet. As along as there is no request for deleting the message this specific message will continue appearing among the messages of the Download() call. As soon as the Client finishes processing the first packet a newer Download() call may be made to inquire the next packet. As regards the functions of the web service the UUID-type identifier of the message is a separate parameter (equal to the value of the MessageID field in the envelope minus the lead-in “uuid:”), so that after downloading the Client does not need to check or read the message in order to call the deletion function. The KKK-Web will delete

the messages of the user only if this is requested by the user by calling the Delete() function.

After each call of the KKK-Web web service the business fault descriptor must be reviewed and relying on it a decision needs to be made concerning fault management or any intervention by the user.

If the Client wants to download messages one by one rather than in batch or bundled mode, the maxMessageCount parameter in the Download() call should be set to one.

When downloading messages it is to take into account that due to the queue-type operation a KKK user may download from one channel on one single thread.

When the Client has successfully downloaded every message from the channel using this procedure, at least 60 seconds should be waited and the downloading process may restart only after. This is needed to ensure that the Client is not enquiring the KKK-Web on a continuous or cyclic way and allow message exchange using larger packets.

When the Client finishes successfully downloading the messages (that means that the last Download call returned 0 messages) and makes a new attempt to inquire within 60 seconds, business fault number 506 is returned.

It may happen that due to a network problem the Client becomes disconnected during downloading. It may also happen that the Client has already downloaded the message and saved it into the backup system but the deletion function cannot be called or the connection breaks during deletion. In such cases, at the time of the next downloading the Client will receive again the already downloaded and saved messages or if retries an interrupted deletion receives the business fault “10506 – This message has been already deleted”.

Pseudo code:

```
// Maximum number of messages the client whishes to upload simultaneously
VAR maxMessageCount = 50

// Channel’s technical name
VAR channelName = ’AIS’

// KKK2 web service’s client-side proxy object
VAR KKKWebservice proxy

// client’s method: web service authentication, web service url,
// setting HTTP User Agent on proxy
CALL ClientInitWebservice(proxy)

// Donwloading the first message packet from KKK-WEB
CALL proxy.Download(channelName, maxMessageCount, out messages, out status)
IF status <> 0 THEN GOTO ERROR

VAR messageCount = messages.Length;

// until there are packets containing messages
WHILE messageCount > 0

      VAR string [] messageIDs;

      // processing of messages found in the packet
      FOR EACH message IN messages
      // client saves the message to its backend system
            CALL ClientSaveMessageToBackend(message, out result)
```

```
                // if saved successfully, the message may be deleted
                IF result <> 0 THEN
                        messageIDs.Add (message.ID)
                END IF

        END FOR

        // deletion of processed messages on KKK-WEB
        CALL proxy.Delete(messageIDs, out statuses);
        FOR EACH status IN statuses
                IF status <> 0 THEN GOTO ERROR
        END FOR

        // Downloading of the next message packet
        CALL proxy.Download(channelName, maxMessageCount, out messages, out status);
        IF status <> 0 THEN GOTO ERROR

        // updating messageCount
        messageCount = messages.Length;

ENDWHILE
```

5.1.3.2 Uploading

The first step is to set the client connection of the web service on the client page. It requires entering the authentication data, the URL of the web service and eventually the proxy settings of the web service client.

After setting the connection, messages can be uploaded to the appropriate channel identified in the parameters of the user approved by the KKK-Web web service, by calling Upload().

After each KKK-Web web service call the business fault descriptor needs to be checked and a decision needs to be made concerning fault management or intervention by the user.

There is no restriction on parallel uploading.

It may happen that for network problem the uploading by Client becomes disrupted. It may also happen that the Client has already uploaded the message and the uploading has been administered by KKK Web but the Client does not receive a reply (for instance, there is timeout during an Upload() call). At this point the Client tries to upload again, but the KKK Web will realize the duplicated upload only if the message ID (MessagedID) of the Client is consistently assigned to the message and is not generated again every time an upload is made. This message ID is unique within KKK and the same identifier cannot be entered again, so on the second attempt the Client will receive the business fault “10507 – A message with this identifier already exists in the system”.

Pseudo code:

```
// Channel’s technical name
VAR channelName = ’AIS’

// KKK2 web service’s client-side proxy object
VAR KKKWebservice proxy

// client’s method: web service authentication, web service url,
// setting HTTP User Agent on proxy
CALL ClientInitWebservice(proxy)
```

```
VAR Message message = new Message ()
// Loading message content
CALL ClientLoadMessageFromBackend (message)

// Uploading to KKK-Web
CALL proxy.Upload(message, out status);
IF status <> 0 THEN GOTO ERROR
```

5.1.4 Requirements on the Client side

KKK-Web provides a mailbox-system service on the data exchange platform. This data exchange platform is loosely linked and the KKK-WEB is available through the Internet so it is not guaranteed that the connection between the Client and KKK-WEB works always. For the above reasons the Client must be prepared for asynchronous operation.

In the application of the Client when a message is uploaded, it has to be put in a queue and the processing system (working in asynchronous way) takes charge of sending it. Also when a message is downloaded a separate processing system is used for downloading and saving messages. The processing of messages on the Client side should be independent from the downloading of messages from KKK-WEB.

The activities of the Client are logged on KKK-WEB and every step made either in good or bad faith can be traced.

The Client needs to make sure that in his application every event related to data exchange may be monitored and in case of fault or error he should be able to trace it back on his side. The application must be able to detect when KKK2 sends a business fault, an exception or a fault message and must be ready to handle and control these faults.

5.1.4.1 KKK2 connection log

The application of the Client must keep a log on the client system as described here and must keep the messages of the KKK2 connection log at least for the last 12 hours.

The log must be in the form of a text file where each event takes a new line.

For each event the log must include the following:

- Date and time of event.
  Format:_YYYY.MM.DD. HH:mm:SS (local time)
- Request ID or Thread ID
  If the SOAP call is made on multiple threads, this identifier may connect the XX begin of call/ XX end of call/ Exception events.
  This column is not needed if calls are made on one single thread, in serial (asynchronous) mode,

The following events should be logged:

- Start of Client's application. Parameters:
  - Client software name, version, date issued, manufacturer
- Halt of Client's application
- SOAP connection parameters. Parameters:
  - Web service URL address
  - User identifier

    - Authentication Type (Basic or X509)
    - Client machine IP address
    - Firewall/proxy IP address (if any and if configured in the application)
- An exception occurred during a SOAP call. Parameters:
    - Content of exception (not only the fault text but also every data piece of the exception object, if possible)
    - HTTP status code (if any)
- Start of SOAP ConnectionTest call. Parameters:
    - None
- End of SOAP ConnectionTest call. Parameters:
    - status.ID,
    - status.Message
- Start of SOAP Delete call. Parameters:
    - string[] messageIDs block: values by element
- End of SOAP Delete call. Parameters:
    - Status[] block: value of ID and Message properties by element
- Start of SOAP Download call. Parameters:
    - string channelName
    - int maxMessageCount
- End of SOAP Download call. Parameters:
    - status.ID,
    - status.Message
    - Message[] block: value of ID property by element
- Start of SOAP Upload call. Parameters:
    - message.ID
- End of SOAP Upload call. Parameters:
    - status.ID,
    - status.Message

5.1.5 Basic authentication under the .NET platform

Using basic authentication with the default settings of .NET the connection with the KKK-Web web service would not efficient so some fine-tuning on .NET-based clients could be necessary.

The traditional ASP.NET web service client proxy is generated by .NET from the SoapHttpClientProtocol class, whose ancestor is the WebClientProtocol. The WebClientProtocol has a PreAuthenticate property that should be used for the basic authentication.

If PreAuthenticate = false (default), then a matching is made before each web service call between the server and the client concerning identification: the client sends a request without authentication and if authentication is compulsory, the a server sends back an “HTTP Error 401.2 - Unauthorized: Access is denied due to server configuration” error message to indicate that it wants to identify the client by all means.

Only at this point the client answers with the name/password combination in the Authorization field, which means that each web service call will originate two HTTP calls.

In the case of KKK-Web web service no matching is needed since it allows only authenticated requests.

If PreAuthenticate = true, the matching is made only for the first call and it is not necessary afterwards. In practice this means that the client proxy object should not be created and set again before each call, only once (and PreAuthenticate = true), and then the server should be called on a set object.

In order to cancel or ban the very first matching the generated proxy needs to be modified: the GetWebRequest() method should be overwritten and the value of the Authorization header should be set. The example below is under .NET 2.0 and with the use of partial class it modifies the proxy class; under 1.0 the function should be inserted into the generated proxy code.

```
using System;
using System.Net;
using System.Text;
using System.Web.Services;

namespace proxy_osztaly_nevtere
{

    public partial class MessageHandler
    {
        protected override System.Net.WebRequest GetWebRequest(Uri uri)
        {
            HttpWebRequest request;
            request = (HttpWebRequest)base.GetWebRequest(uri);

            if (PreAuthenticate)
            {
                NetworkCredential networkCredentials =
                    Credentials.GetCredential(uri, "Basic");

                if (networkCredentials != null)
                {
                    byte[] credentialBuffer = new UTF8Encoding().GetBytes(
                        networkCredentials.UserName + ":" +
                        networkCredentials.Password);

                    request.Headers["Authorization"] =
                        "Basic " +
                        Convert.ToBase64String(credentialBuffer);
                }
                else
                {
                    throw new ApplicationException("No network
credentials");
                }
            }
            return request;

        }
    }
}
```

6. Business system-specific elements

The business-system-specific elements on the envelope are as follows:

- (MessageType)
- (To)
- (OnBehalfOf)
- (Properties)

See the description of how these elements should be handled in the Interface Specification of the relevant business system.

7. Description of fault handling

7.1 Synchronous feedback from the KKK-Web web service

The web service returns synchronous exceptions (SOAP Fault message) and business fault descriptors (Status parameter for web service parameters). Technological faults appear as exceptions, while business faults are defined as return values for every case.

7.1.1 Exceptions

An exception is only sent to the web service if any of the resources is not available (e.g. database) or other basic error has occurred.

The authentication is performed by the web server of the web service so a wrong authentication appears as an exception on the Client's side.

7.1.2 Business faults

Most of the business faults indicate problems where human intervention and adjustment or fixing are needed, which means that the Client could only proceed if the fault is eliminated with reliance on the fault text.

Expected business faults:

ID	ConnectionTest	Delete	Download	Upload	Fault text	Explanation
0	x	x	x	x	Everything OK.	
502		x			At least one of the transferred message IDs is not in GUID format.	For a Delete() call the values included in the string[] messageIDs parameter are not in the UUID format
503		x			The transferred ID block has zero value.	For a Delete() call the string[] messageIDs parameter has zero value,
504			x		The transferred channel ID has zero or empty string value.	For a Download() call the channelName parameter is empty

505			x		The MaxMessageCount parameter should be greater than zero.	For a Download() call the value of maxMessageCount must be larger than zero.
506			x		Too frequent inquiry	After successful download of a message the Client did not wait at least 60 seconds.
507	x	x	x	x	The user is not a member of the safety group.	Possible reasons: attempt is made to use a test system with a live user or vice versa.
508	x	x	x	x	Authentication unsuccessful; the user could not be identified in the LDAP directory (Anonymous user)	The authentication certificate is not assigned to a user or the BASIC authentication is not successful.
9501				x	Could not read the From field in the envelope.	The value fails to have the user:FELH_AZONOSÍTÓ format.
9502				x	Could not read the MessageId field in the envelope.	The value fails to have the uuid:UUID_ÉRTÉK format.
9503				x	Could not read the MessageType field in the envelope.	The field is not completed.
9504				x	Could not read the RelatesTo field in the envelope.	The value fails to have the uuid:UUID_ÉRTÉK format.
9505				x	Could not read the To field in the envelope.	The field is not completed.
9506				x	The message ID given as parameter when uploading fails to match the one found in the MessageId field of the envelope.	At Upload() the UUID value given in the message.ID fails to match the UUID value found in the MessageId field of the envelope.
9507				x	The message ID given as parameter when uploading could not be interpreted.	At Upload()the value given in the message.ID is not in the UUID format
9508				x	The uploading user does not match the one included in the From field of the envelope.	The identifier of the user defined with the authentication of the web service fails to match the one found in the From field of the envelope.
9509				x	Could not read the envelope header in XML.	Could not interpret the content of the Header tag. A detailed fault message may be requested from the HelpDesk. In general the xs:DateTime type values are not standard.
9510				x	Could not find the envelope header in XML.	Could not find the Header tag.
9511				x	The uploaded XML is not properly formatted.	Self-understanding.
10501			x	x	The technical channel name fails to exist.	The channel defined in the To field of the envelope is not included in the EÜC.
10502			x	x	The technical channel name is not authorized on this day.	The channel defined in the To field of the envelope is not valid: has expired or will

						become active later.
10506		x			You have already deleted this message.	The Client has already requested the deletion of the message featuring this identifier. This can be granted if the connection was broken while requesting the previous deletion.
10507				x	There is already a message with this identifier in the system.	The Client has already uploaded a message with this message ID (MessageID). It may be granted if the connection was broken during the first uploading and the confirmation of successful uploading did not reach the Client.
10508		x			No message with this identification exists.	Attempt is made to delete a non-existing message: messageID has wrong value.
10509		x			This message has been not read yet.	The deletion of a message not yet downloaded is requested. There are weak chances for that.
10510				x	N message type with this identification exists.	The channel/message type couple has been not configured in EÜC.
10511				x	This message type is not authorized on this day.	The channel/message type couple is not valid: either expired or will become active in the future.
10512		x			This message does not belong to the user.	You want to delete a message belonging to other user. There are weak chances for that.
10513				x	No scheme with this identification exists.	This message type is not configured in EÜC.
10514				x	The scheme is not authorized on this day.	This message type is not valid: either expired or will become active in the future.
10515				x	The uploading of messages is not authorized for users for this message type.	You want to upload a message type that could be only downloaded.
10516				x	The uploading of messages is not authorized for users on this channel.	This channel is only used for downloading.
10517		x	x	x	The user has been banned.	The user is in banned status.
10518		x	x	x	The user has not been activated.	Has no valid e-mail address or it is being confirmed now.
10519		x	x	x	The user is not authorized on this day.	This user is not valid: either expired or will become active in the future.

7.2 KKK-Web – receipt message

KKK-Web prepares a receipt when the message is received from the Client.

7.3 KKK-Közvetítő – delivery message

KKK-Közvetítő prepares a receipt when the message is delivered to the business system.

7.4 KKK-Közvetítő – fault message

KKK-Közvetítő prepares a KKK2 fault message about the faults originated in the course of message processing and sends it back to the uploader (sender).

7.5 Fault handling in message sending

In section **Hiba! A hivatkozási forrás nem található. Hiba! A hivatkozási forrás nem található.** we noted that uploading by the Client is fault-tolerant only if the message ID (MessagID) is formally and consistently assigned to the message in the background system of the Client and it does not generate again and again when uploading is made.

Another important point is that the Client can be sure that his message has reached the business system only if has received both receipts (Receive, Delivery). If KKK Közvetítő sends a fault message instead of the receipt message, then the message could not be delivered to the business system. To fix the fault a correction is required on the Client side and then the message should be resent – but now with a newly generated message ID.

8. Documents used and referenced

In the preparation of the product the following documents have been used:

Code	Reference number	Title	Version
1.	AIS/DOC/PID/V1.3/2006.11.06.	Project Initiation Document	1.3
2.	PQP	Quality Assurance Plan	2.0
3.	KKK2/DOC/RENKOV/V2.0/2006.12.01.	KKK2 System requirements	2.0
4.	KKK2/DOC/ARCH/V2.1/2007.01.08.	KKK2 Architecture plan	2.1
5.	AIS/DOC/CDPS/ISP/V1.0/2006.11.29	CDPS_I Interface Specification	1.0
6.	VPRK/KKK2/DOC/TECHSPEC/V1.0/2007.01.22.	KKK2 Technical specification	1.0

9. Acronyms and abbreviations

In this documentation the following acronyms and abbreviations have been used:

Acronym	Description
AIS	Electronic Import Customs Procedures
API	Application Programming Interface
CDPS_I	Hungarian Customs Declaration Processing System
EÜC	Standard Register of Client Addresses
KKK	External Communication Centre
PID	Project Initiation Document

Acronym	Description
PQP	Project Quality Plan
SSL	Secure Sockets Layer – a safe data exchange protocol encrypted with "RSA" procedure for the communication between web servers and their clients
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
WSDLA	Web Services Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

10. Quality criteria

The KKK2 Interface Specification document has been prepared considering the following quality criteria:

- Must be understandable, clear and precise.
- Must harmonize with the products used in its preparation.

11. Quality control

The quality assurance activities related to the KKK2 Interface Specification product have been made in the frame of a quality control and quality inspection exercise.

12. Annexes

12.1 WSDL description of the KKK-Web service

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://soap.vam.gov.hu/KKK/messagehandler/1.0"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
targetNamespace="http://soap.vam.gov.hu/KKK/messagehandler/1.0"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
      <wsdl:types>
            <s:schema elementFormDefault="qualified"
targetNamespace="http://soap.vam.gov.hu/KKK/messagehandler/1.0">
                  <s:element name="ConnectionTest">
                        <s:complexType />
                  </s:element>
                  <s:element name="ConnectionTestResponse">
                        <s:complexType>
                              <s:sequence>
                                    <s:element minOccurs="1" maxOccurs="1"
name="status" type="tns:Status" />
                              </s:sequence>
                        </s:complexType>
                  </s:element>
                  <s:complexType name="Status">
```

```
                        <s:sequence>
                              <s:element minOccurs="1" maxOccurs="1"
name="ID" type="s:int" />
                              <s:element minOccurs="1" maxOccurs="1"
name="Message" type="s:string" />
                        </s:sequence>
                  </s:complexType>
                  <s:element name="Delete">
                        <s:complexType>
                              <s:sequence>
                                    <s:element minOccurs="1" maxOccurs="1"
name="messageIDs" type="tns:ArrayOfString" />
                              </s:sequence>
                        </s:complexType>
                  </s:element>
                  <s:complexType name="ArrayOfString">
                        <s:sequence>
                              <s:element minOccurs="0"
maxOccurs="unbounded" name="string" nillable="true" type="s:string" />
                        </s:sequence>
                  </s:complexType>
                  <s:element name="DeleteResponse">
                        <s:complexType>
                              <s:sequence>
                                    <s:element minOccurs="1" maxOccurs="1"
name="statuses" type="tns:ArrayOfStatus" />
                              </s:sequence>
                        </s:complexType>
                  </s:element>
                  <s:complexType name="ArrayOfStatus">
                        <s:sequence>
                              <s:element minOccurs="0"
maxOccurs="unbounded" name="Status" nillable="true" type="tns:Status" />
                        </s:sequence>
                  </s:complexType>
                  <s:element name="Download">
                        <s:complexType>
                              <s:sequence>
                                    <s:element minOccurs="1" maxOccurs="1"
name="channelName" type="s:string" />
                                    <s:element minOccurs="1" maxOccurs="1"
name="maxMessageCount" type="s:int" />
                              </s:sequence>
                        </s:complexType>
                  </s:element>
                  <s:element name="DownloadResponse">
                        <s:complexType>
                              <s:sequence>
                                    <s:element minOccurs="1" maxOccurs="1"
name="messages" type="tns:ArrayOfMessage" />
                                    <s:element minOccurs="1" maxOccurs="1"
name="status" type="tns:Status" />
                              </s:sequence>
                        </s:complexType>
                  </s:element>
                  <s:complexType name="ArrayOfMessage">
                        <s:sequence>
                              <s:element minOccurs="0"
maxOccurs="unbounded" name="Message" nillable="true" type="tns:Message" />
                        </s:sequence>
                  </s:complexType>
```

```
                  <s:complexType name="Message">
                        <s:sequence>
                              <s:element minOccurs="1" maxOccurs="1"
name="ID" type="s:string" />
                              <s:element minOccurs="1" maxOccurs="1"
name="CreatedAt" type="s:dateTime" />
                              <s:element minOccurs="1" maxOccurs="1"
name="Content" type="s:base64Binary" />
                        </s:sequence>
                  </s:complexType>
                  <s:element name="Upload">
                        <s:complexType>
                              <s:sequence>
                                    <s:element minOccurs="1" maxOccurs="1"
name="message" type="tns:Message" />
                              </s:sequence>
                        </s:complexType>
                  </s:element>
                  <s:element name="UploadResponse">
                        <s:complexType>
                              <s:sequence>
                                    <s:element minOccurs="1" maxOccurs="1"
name="status" type="tns:Status" />
                              </s:sequence>
                        </s:complexType>
                  </s:element>
            </s:schema>
      </wsdl:types>
      <wsdl:message name="ConnectionTestSoapIn">
            <wsdl:part name="parameters" element="tns:ConnectionTest" />
      </wsdl:message>
      <wsdl:message name="ConnectionTestSoapOut">
            <wsdl:part name="parameters"
element="tns:ConnectionTestResponse" />
      </wsdl:message>
      <wsdl:message name="DeleteSoapIn">
            <wsdl:part name="parameters" element="tns:Delete" />
      </wsdl:message>
      <wsdl:message name="DeleteSoapOut">
            <wsdl:part name="parameters" element="tns:DeleteResponse" />
      </wsdl:message>
      <wsdl:message name="DownloadSoapIn">
            <wsdl:part name="parameters" element="tns:Download" />
      </wsdl:message>
      <wsdl:message name="DownloadSoapOut">
            <wsdl:part name="parameters" element="tns:DownloadResponse" />
      </wsdl:message>
      <wsdl:message name="UploadSoapIn">
            <wsdl:part name="parameters" element="tns:Upload" />
      </wsdl:message>
      <wsdl:message name="UploadSoapOut">
            <wsdl:part name="parameters" element="tns:UploadResponse" />
      </wsdl:message>
      <wsdl:portType name="MessageHandlerSoap">
            <wsdl:operation name="ConnectionTest">
                  <wsdl:input message="tns:ConnectionTestSoapIn" />
                  <wsdl:output message="tns:ConnectionTestSoapOut" />
            </wsdl:operation>
            <wsdl:operation name="Delete">
                  <wsdl:input message="tns:DeleteSoapIn" />
                  <wsdl:output message="tns:DeleteSoapOut" />
```

```
            </wsdl:operation>
            <wsdl:operation name="Download">
                  <wsdl:input message="tns:DownloadSoapIn" />
                  <wsdl:output message="tns:DownloadSoapOut" />
            </wsdl:operation>
            <wsdl:operation name="Upload">
                  <wsdl:input message="tns:UploadSoapIn" />
                  <wsdl:output message="tns:UploadSoapOut" />
            </wsdl:operation>
      </wsdl:portType>
      <wsdl:binding name="MessageHandlerSoap"
type="tns:MessageHandlerSoap">
            <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
/>
            <wsdl:operation name="ConnectionTest">
                  <soap:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/ConnectionTest"
style="document" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="Delete">
                  <soap:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/Delete"
style="document" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="Download">
                  <soap:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/Download"
style="document" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="Upload">
                  <soap:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/Upload"
style="document" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
      </wsdl:binding>
      <wsdl:binding name="MessageHandlerSoap12"
type="tns:MessageHandlerSoap">
```

```
            <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http" />
            <wsdl:operation name="ConnectionTest">
                  <soap12:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/ConnectionTest"
style="document" />
                  <wsdl:input>
                        <soap12:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap12:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="Delete">
                  <soap12:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/Delete"
style="document" />
                  <wsdl:input>
                        <soap12:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap12:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="Download">
                  <soap12:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/Download"
style="document" />
                  <wsdl:input>
                        <soap12:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap12:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
            <wsdl:operation name="Upload">
                  <soap12:operation
soapAction="http://soap.vam.gov.hu/KKK/messagehandler/1.0/Upload"
style="document" />
                  <wsdl:input>
                        <soap12:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap12:body use="literal" />
                  </wsdl:output>
            </wsdl:operation>
      </wsdl:binding>
      <wsdl:service name="MessageHandler">
            <wsdl:port name="MessageHandlerSoap"
binding="tns:MessageHandlerSoap">
                  <soap:address
location="http://localhost:91/Users/MessageHandler.asmx" />
            </wsdl:port>
            <wsdl:port name="MessageHandlerSoap12"
binding="tns:MessageHandlerSoap12">
                  <soap12:address
location="http://localhost:91/Users/MessageHandler.asmx" />
            </wsdl:port>
      </wsdl:service>
       </wsdl:definitions>
```

12.2 KKK2 VPEnvelope

```
<?xml version="1.0" encoding="utf-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://schemas.vam.gov.hu/VPEnvelope/1.0"
      targetNamespace="http://schemas.vam.gov.hu/VPEnvelope/1.0"
      elementFormDefault="qualified">


      <!-- VPEnvelope -->

      <xs:element name="VPEnvelope" type="tns:VPEnvelope" />
      <xs:complexType name="VPEnvelope">
            <xs:annotation>
                  <xs:documentation>
                        VPEnvelope is the envelope schema for the messages
                        sent to or from the Hungarian Customs and Finance
Guard.
                  </xs:documentation>
            </xs:annotation>
            <xs:sequence>
                  <xs:element ref="tns:Header" />
                  <xs:element ref="tns:Body" />
            </xs:sequence>
      </xs:complexType>


      <!-- Header -->

      <xs:element name="Header" type="tns:Header" />
      <xs:complexType name="Header">
            <xs:annotation>
                  <xs:documentation>
                        Header contains the meta-information fields of the
envelope.
                  </xs:documentation>
            </xs:annotation>
            <xs:sequence>
                  <xs:element name="MessageID"
type="tns:AttributedURIType" />
                  <xs:element name="RelatesTo"
type="tns:AttributedURIType" minOccurs="0" />
                  <xs:element name="MessageType"
type="tns:AttributedURIType" />
                  <xs:element name="From" type="tns:EndPointReferenceType"
/>
                  <xs:element name="To" type="tns:EndPointReferenceType"
minOccurs="0" />
                  <xs:element name="ReplyTo"
type="tns:EndPointReferenceType" minOccurs="0" />
                  <xs:element name="OnBehalfOf"
type="tns:EndPointReferenceType" minOccurs="0" />
                  <xs:element name="Created" type="xs:dateTime" />
                  <xs:element name="Uploaded" type="xs:dateTime"
minOccurs="0"  />
                  <xs:element name="Properties" type="tns:PropertiesType"
minOccurs="0" />
            </xs:sequence>
      </xs:complexType>
```

```
	<!-- Body -->

	<xs:element name="Body" type="tns:Body" />
	<xs:complexType name="Body">
		<xs:annotation>
			<xs:documentation>
				The Body element contains the payload of the
message.
			</xs:documentation>
		</xs:annotation>
		<xs:sequence>
			<xs:any namespace="##any" maxOccurs="unbounded"
processContents="lax" />
		</xs:sequence>
	</xs:complexType>


	<!-- Basic Types -->

	<xs:complexType name="PropertiesType">
		<xs:sequence>
			<xs:element name="Property" minOccurs="0"
maxOccurs="unbounded" type="tns:PropertyType" />
		</xs:sequence>
		<xs:anyAttribute namespace="##other" processContents="lax" />
	</xs:complexType>

	<xs:complexType name="PropertyType">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="name" type="xs:string"
use="required" />
				<xs:anyAttribute namespace="##other"
processContents="lax" />
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>

	<xs:complexType name="EndPointReferenceType">
		<xs:simpleContent>
			<xs:extension base="tns:AttributedURIType">
				<xs:anyAttribute namespace="##other"
processContents="lax" />
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>

	<xs:complexType name="AttributedURIType">
		<xs:simpleContent>
			<xs:extension base="xs:anyURI">
				<xs:anyAttribute namespace="##other"
processContents="lax" />
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>

</xs:schema>
```

12.3 KKK2 AttachmentEnvelope

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://schemas.vam.gov.hu/AttachmentEnvelope/1.0"
        targetNamespace="http://schemas.vam.gov.hu/AttachmentEnvelope/1.0"
        elementFormDefault="qualified">

        <!-- AttachmentEnvelopeType -->

        <xs:element name="AttachmentEnvelope" type="tns:AttachmentEnvelopeType"/>
        <xs:complexType name="AttachmentEnvelopeType">
                <xs:sequence>
                        <xs:element ref="tns:AttachmentHeaders" />
                        <xs:element ref="tns:Body" />
                        <xs:element ref="tns:AttachmentContents" />
                </xs:sequence>
        </xs:complexType>

        <!-- AttachmentHeaders -->

        <xs:element name="AttachmentHeaders" type="tns:AttachmentHeadersType"/>
        <xs:complexType name="AttachmentHeadersType">
                <xs:sequence>
                        <xs:element ref="tns:AttachmentHeader" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>

        <xs:element name="AttachmentHeader" type="tns:AttachmentHeaderType"/>
        <xs:complexType name="AttachmentHeaderType">
                <xs:sequence>
                        <xs:element name="AttachmentID" type="xs:string"/>
                        <xs:element name="MimeType" type="xs:string" />
                        <xs:element name="Format" type="tns:FormatEnum" />
                        <xs:element name="Name" type="xs:string" minOccurs="0"/>
                        <xs:element name="Comment" type="xs:string" minOccurs="0"/>
                        <xs:element name="Properties" type="tns:PropertiesType" minOccurs="0"
/>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="PropertiesType">
                <xs:sequence>
                        <xs:element name="Property" minOccurs="0" maxOccurs="unbounded"
type="tns:PropertyType" />
                </xs:sequence>
                <xs:anyAttribute namespace="##other" processContents="lax" />
        </xs:complexType>

        <xs:complexType name="PropertyType">
                <xs:simpleContent>
                        <xs:extension base="xs:string">
                                <xs:attribute name="name" type="xs:string" use="required" />
                                <xs:anyAttribute namespace="##other" processContents="lax" />
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>

        <xs:simpleType name="FormatEnum">
                <xs:restriction base="xs:QName">
                        <xs:enumeration value="Binary" />
                        <xs:enumeration value="Xml" />
                </xs:restriction>
        </xs:simpleType>

        <!-- Body -->

        <xs:element name="Body" type="tns:BodyType"/>
        <xs:complexType name="BodyType">
                <xs:sequence>
                        <xs:any namespace="##any" maxOccurs="unbounded" processContents="lax"
/>
                </xs:sequence>
        </xs:complexType>
```

```
	<!-- AttachmentContents -->

	<xs:element name="AttachmentContents" type="tns:AttachmentContentsType"/>
	<xs:complexType name="AttachmentContentsType">
		<xs:sequence>
			<xs:element ref="tns:AttachmentContent" minOccurs="0"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>

	<xs:element name="AttachmentContent" type="tns:AttachmentContentType"/>
	<xs:complexType name="AttachmentContentType">
		<xs:sequence>
			<xs:element ref="tns:BinaryData" minOccurs="0" />
			<xs:element ref="tns:XmlData" minOccurs="0" />
		</xs:sequence>
		<xs:attribute name="attachmentID" type="xs:string" />
	</xs:complexType>

	<xs:element name="BinaryData" type="xs:base64Binary" nillable="true"/>
	<xs:element name="XmlData" type="tns:XmlDataType" />

	<xs:complexType name="XmlDataType">
		<xs:sequence>
			<xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
processContents="lax" />
		</xs:sequence>
	</xs:complexType>

</xs:schema>
```

12.4 KKK2 VPReceipt

```
<?xml version="1.0" encoding="utf-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
	xmlns:tns="http://schemas.vam.gov.hu/VPReceipt/1.0"
	targetNamespace="http://schemas.vam.gov.hu/VPReceipt/1.0"
	elementFormDefault="qualified">

	<!-- Receipt and its types -->

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

	<xs:complexType name="Receipt">
		<xs:sequence>
			<xs:element name="Event" type="tns:ReceiptEventEnum" />
			<xs:element name="Detail" type="tns:Detail"
minOccurs="0" />
		</xs:sequence>
		<xs:anyAttribute namespace="##other" processContents="lax" />
	</xs:complexType>

	<xs:simpleType name="ReceiptEventEnum">
		<xs:restriction base="xs:QName">
			<xs:enumeration value="Receive" />
			<xs:enumeration value="Delivery" />
		</xs:restriction>
	</xs:simpleType>

	<xs:complexType name="Detail">
		<xs:sequence>
			<xs:any namespace="##any" processContents="lax"
minOccurs="0" maxOccurs="unbounded"  />
		</xs:sequence>
		<xs:anyAttribute namespace="##other" processContents="lax" />
```

```
	</xs:complexType>

</xs:schema>
```

12.5 KKK2 VPFault

```
<?xml version="1.0" encoding="utf-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
	xmlns:tns="http://schemas.vam.gov.hu/VPFault/1.0"
	targetNamespace="http://schemas.vam.gov.hu/VPFault/1.0"
	elementFormDefault="qualified">

	<!-- Fault and its types -->

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

	<xs:complexType name="Fault">
		<xs:sequence>
			<xs:element name="Code" type="tns:FaultCodeEnum" />
			<xs:element name="Subcode" type="tns:Subcode"
minOccurs="0" />
			<xs:element name="Detail" type="tns:Detail"
minOccurs="0" />
		</xs:sequence>
		<xs:anyAttribute namespace="##other" processContents="lax" />
	</xs:complexType>

	<xs:simpleType name="FaultCodeEnum">
		<xs:restriction base="xs:QName">
			<xs:enumeration value="InvalidXml" />
			<xs:enumeration value="SenderMismatch" />
			<xs:enumeration value="MessageTypeMismatch" />
			<xs:enumeration value="RoutingDenied" />
			<xs:enumeration value="InvalidDelegation" />
			<xs:enumeration value="VersionMismatch" />
			<xs:enumeration value="DuplicateGuid" />
			<xs:enumeration value="OtherFault" />
		</xs:restriction>
	</xs:simpleType>

	<xs:complexType name="Subcode">
		<xs:sequence>
			<xs:element name="Value" type="xs:QName" />
			<xs:element name="Text" type="xs:string" />
			<xs:element name="Subcode" type="tns:Subcode"
minOccurs="0" />
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="Detail">
		<xs:sequence>
			<xs:any namespace="##any" processContents="lax"
minOccurs="0" maxOccurs="unbounded"  />
		</xs:sequence>
		<xs:anyAttribute namespace="##other" processContents="lax" />
	</xs:complexType>

</xs:schema>
```