GME's Web Services

Date: **27 Feb. 2004**

TABLE OF CONTENTS

1. Introduction

With a view to establishing an system-to-system interconnection between the Market Participant's information system and the Electricity Market information system, the latter system exposes a programmatic interface.
The interface is implemented via the GMEWS Web Service.

1.1 Purpose

This document describes the operation and use of the GMEWS version 1.0.4.3 Web Service.

1.2 DEFINITIONS & ACRONYMS

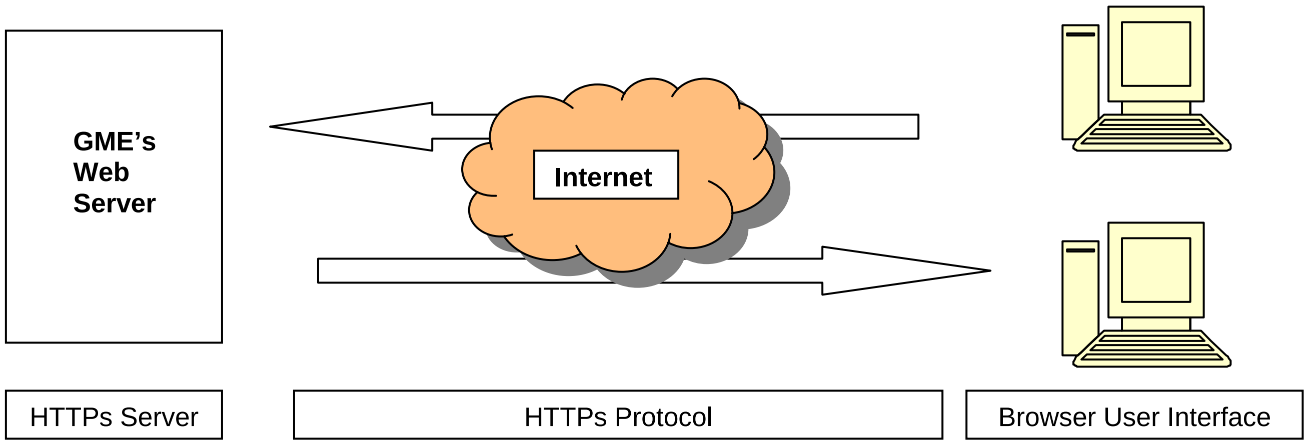
GME	Gestore del Mercato Elettrico
GRTN	Gestore della Rete di Trasmissione Nazionale
DB	Data Base
GUI	Graphical User Interface
BUI	Browser User Interface
SOAP	Simple Object Access Protocol
WSDL	Web Service Description Language

1.3 Reference Documents

[1] RSA Laboratories. *PKCS #7: Cryptographic Message Syntax Standard.* Version 1.5, Nov. 1993
[2] "XML Implementation Guide for Market Participant", document posted by GME at http://www.mercatoelettrico.org/Biblioteca/BiblioDefault.aspx in the Technical Documents - Information System section)

2. Reference Context

The following is the scenario of interaction with the Electricity Market information system, installed at GME:

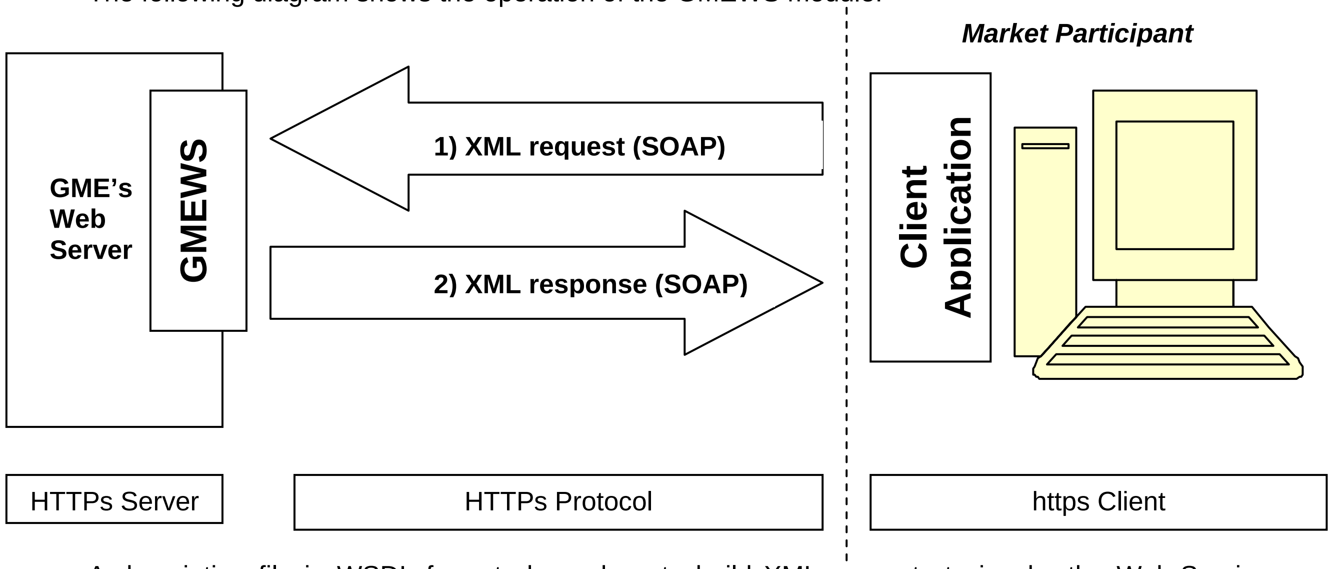


Through the Internet, Market Participants access the Web Server, where the portal application (BUI) is installed. The application enables Market Participants to submit queries for market data, as well as bids/supply offers. Authentication to GME's Web Server takes place by using a digital authentication certificate. The certificate is associated with a given user authorised to perform transactions on behalf of the Market Participant.

Offers/bids may be submitted both on line via the web form and by uploading appropriately structured XML files. To ensure the non-repudiation of offers/bids submitted to GME, the Market Participant signs the digital offers and includes its certificate used for signing.

The GMEWS was developed to provide Market Participant applications with a programming interface to communicate with the GME's system. The module implements a web-service type interface based on SOAP messages and https transport.

The following diagram shows the operation of the GMEWS module:



A description file in WSDL format shows how to build XML requests to invoke the Web-Service-supported methods and how to read the related responses.

Appendix A describes the WSDL of GMEWS.

The client application should invoke the Web-Service-supported methods by sending XML requests, which are structured as described in the WSDL.

The Web Server translates the received XML request into a call to the requested function and awaits its completion. After completion of the invoked method, the Web Server translates the result into an XML response and resends it to the client application.

By using the description file again, the client application can read the XML response and obtain the result of the transaction.

XML requests and responses are SOAP messages to be sent over an https connection.

Thus, to interact with GMEWS, a client application should:
- establish an https connection with GME's site (hosting GMEWS), including SSL authentication (based on SSL client certificates);
- build up SOAP request messages and read the corresponding response messages, as described in the GMEWS WSDL.

When sending a document GME (e.g. submission of offers/bids or margins), GMEWS expects to receive a PKCS #7 signed document (cfr. [1]), including a legally valid signature.
So, the client application should prepare a signed document before including it into a SOAP request message[1].

Thus, a client application interacting with GMEWS should have:
- a certificate for SSL client authentication (to manage SSL authentication to GMEWS);
- a signature device (with associated drivers) and a certificate for signing, compliant with cnipa (formerly AIPA) standards (in order to affix the signature onto the documents to be sent)
- SW tools or libraries enabling:
    - http + SSL communication (https);
    - SOAP messaging according to the WSDL of GMEWS;
    - preparation of signed messages in PKCS #7 format (interworking with the drivers of the signature device).

For that purpose, on a Microsoft platform the following application development tools can be used:
- SOAP toolkit 2.0 SP2 or 3.0 and .NET Framework 1.1 (https communication + SOAP messaging);
- CryptoAPI (preparation of documents in PKCS #7 format)

Those tools have been tested compatibility and interoperability with the GMEWS.

Appendix C provides hints and suggestions on the settings to be used for using the certificates in the Microsoft environment.

[1] The signed document should not be encrypted, because encryption is obtained at the TCP/IP transport level via the SSL connection to GMEWS.

3. The GMEWS interface

3.1 Description

GMEWS Web Service allows programmatic access to a subset of functions that the user may select by connecting to GME's system web portal.

As for the portal access, the interaction via GMEWS is via the HTTPS protocol - to protect the transmitted data - and is subject to SSL client authentication on the basis of a digital certificate - to check the identity of the user[2]: to be able to invoke the functions of GMEWS, the client application should authenticate itself to GME's web server (hosting GMEWS) by sending an SSL client digital certificate and setting up an SSL connection to it.

3.2 GMEWS methods

GMEWS exposes the following functions:

- **Login**
  It initiates a work session with GME's system.
  It is the first function that the client application should invoke in order to use the functionality exposed by GMEWS. Invoking this funnction, the Market Participant identifies itself to GME's system.
  In its response, GMEWS returns a session identifier that the client application should specify whenever invoking a function of GMEWS.

- **Logout**
  It terminates a work session with GME's system. The user application should invoke this function in order to release the resources being used, before breaking up the https connection

- **UploadMessage**
  It sends an XML document to GME's system. The document should have an XML format accepted by the Electricity Market system (see "XML Implementation Guide for Market Participant" (cfr. [2]).
  GMEWS expects a signed document in PKCS#7 format (cfr. [1])[3], including the digital signature certificate.
  If the document is accepted, GMEWS returns a response with the identifier that has been assigned to the document, as well as the date and time of receipt; if the document is not accepted, GMEWS responds by sending an XML document showing the reason for the transaction failure.

- **DownloadMessage**
  It downloads messages, if any, concerning the offer/bid documents that have been submitted into the market (e.g notifications, schedules).
  This function checks whether there are notification messages to be downloaded (i.e. unread) and returns the first unread message, if there are any. The returned message is in the XML

[2] As can be seen in the WSDL, the Web Service is designed to support authentication via login and password; however, for actual participation in the market, this basic authentication mode is not accepted.

[3] MIME Base 64 encoded.

format adopted by the Electricity Market system (see “XML Implementation Guide for Market Participant” (cfr. [2]).
GME’s system marks the returned message as “read”; if the user subsequently invokes the Download Message method, that message will not be returned again. To download an already read message, the ForceDownloadMessage method is available.

- **GetNextMessage**
  It retrieves the list of messages still to be downloaded (unread). It returns a list in XML format with the name and identifier of each message available for downloading. The maximum number of messages included in the list may be specified upon the invocation.

- **ForceDownloadMessage**
  This function downloads a specific message, which is identified via its own identifier; the message can be downloaded even if it was downloaded already.
  GME’s system marks the returned message as “read”.

4. Use of GMEWS

The following paragraphs describe how to use the GMEWS, as indicated in the WSDL.

4.1 Login

It is the GMEWS method that should be called first.
It establishes a work session with GME's system.

The method has two input parameters: UserName and Password.

Login(UserName, Password)

The following is an example of SOAP message showing the login method invocation:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
             <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
                 <SessionId></SessionId>
           </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
            <Login>
                 <UserName>aaaaa</UserName>
                 <Password>bbbbb</Password>
            </Login>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note the following:

- the SOAP Header must contain an empty SessionId element (SOAP header is defined in the WSDL with the attribute required='true');

- the Username and Password parameters may have any value: as SSL client authentication is required, GMEWS ignores their value and identifies the user according to the SubjectName of the digital certificate used for SSL client authentication.

Appendix D and Appendix E give some guidelines on how to invoke the login method and establish a session with GMEWS.

4.1.1 Login response

After invoking the login method, GMEWS returns a Login response, confirming that a work session has been set up.
The following is an example of SOAP message showing a login response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
      <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
            <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
       </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
   <SOAP-ENV:Body>
        <LoginResponse>
              <Result>true</Result>
         </LoginResponse>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- if the response is positive (Result = TRUE), then the SessionId in the SOAP header will show the identifier that GME’s system assigned to that user session. In following invocations of GMEWS methods (Upload, Download, Logout), the client application should include this SessionId value;

- if the response is negative, a message of SOAP Fault type is sent; this message may have the following script:

```
    <?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
     <SOAP-ENV:Body>
           <SOAP-ENV:Fault>
               <faultcode>SOAP-ENV:Server</faultcode>
               <faultstring>Error executing ExactManager.Login. 0000000019 IComManager::Login - Unable to login, invalid user id or
password.</faultstring>
            </SOAP-ENV:Fault>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

Note

In GME’s system, the session expires after a given time of inactivity.
GMEWS rejects invocations referring to an expired session. In that case GMEWS returns a message of SOAP Fault type (see below).

4.2 DownloadMessage

The DownloadMessage method is invoked by entering one input parameter (**MPNumber)** and two empty output parameters (**MessageName, MessageContent)**:

DownloadMessage(MPNumber, MessageName, MessageContent)

The following is an example of SOAP message with the DownloadMessage method invocation:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
        <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
            <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
        </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
         <DownloadMessage>
             <MPNumber>IDOPERATORE</MPNumber>
             <MessageName></MessageName>
             <MessageContent></MessageContent>
         </DownloadMessage>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- in the SOAP header, the SessionId must be filled with the value returned by the Login method;
- **MPNumber** is the alphanumerical identifier (in GME’s system) of the Market Participant to which the messeages to be downloaded are addressed.

4.2.1 DownloadMessage response

After the invocation of the DownloadMessage method, GMEWS returns a DownloadMessage response. The following is an example of SOAP message showing a DownloadMessage response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
     <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
          <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
      </SOAPSDK1:SessionInfo>
    </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <DownloadMessageResponse>
      <Result>true</Result>
      <MessageName>FA_FileName0001.33390899166024.out.xml</MessageName>
      <MessageContent>&lt;?xml version=&apos;1.0&apos; encoding=&apos;ISO-8859-1&apos; ?&gt;
&lt;PIPEFunctionalAcknowledgement xmlns = &quot;urn:XML-PIPE&quot;
  xmlns:xsi = &quot;http://www.w3.org/2001/XMLSchema-instance&quot;
  xsi:schemaLocation = &quot;urn:XML-PIPE PIPEFunctionalAcknowledgementv1_0.xsd&quot;
  ReferenceNumber=&quot;33390000491957&quot;                    OriginalReferenceNumber=&quot;ANSW001274358&quot;
Status=&quot;Accept&quot; CreationDate=&quot;20031205135028&quot; Version=&quot;1.0&quot;&gt;
            &lt;TradingPartnerDirectory&gt;
                        &lt;Sender&gt;
                                    &lt;TradingPartner PartnerType = &quot;Operator&quot;&gt;
                                                &lt;CompanyName&gt;GME&lt;/CompanyName&gt;
                                                &lt;CompanyIdentifier&gt;IDGME&lt;/CompanyIdentifier&gt;
                                    &lt;/TradingPartner&gt;
                        &lt;/Sender&gt;
                        &lt;Recipient&gt;
                                    &lt;TradingPartner PartnerType = &quot;Market Participant&quot;&gt;
                                                &lt;CompanyName&gt;OPERATORE&lt;/CompanyName&gt;
                                                &lt;CompanyIdentifier&gt;IDOPERATORE&lt;/CompanyIdentifier&gt;
                                    &lt;/TradingPartner&gt;
                        &lt;/Recipient&gt;
            &lt;/TradingPartnerDirectory&gt;

            &lt;TransactionAcknowledgement  Status  =  &quot;Accept&quot;  PIPTransactionType  =  &quot;BidAwardResponse&quot;
OriginalReferenceNumber = &quot;33390002948441&quot;/&gt;
&lt;/PIPEFunctionalAcknowledgement&gt;
      </MessageContent>
    </DownloadMessageResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- if the response contains a message with tag Result equal to ’true’, then the **MessageName** tag shows the name of the message and the XML (HTML encoded) content of the message is available in **MessageContent.** The example refers to a Functional Acknowledgement message. In general, it may also be message among those specified in the “XML Implementation Guide for Market Participant” (cfr. [2] – e.g. a bid notification or unit schedule;
- If the response has a message with a tag Result equal to ’false’, then there are no messages to be downloaded and the response does not contain any message.

If the user is not authorised to download messages for the specified Market Participant, or if the session has expired or for any other reason, then a message of SOAP fault type is sent in place of the previous message; such message may have the following structure (expired session):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>
            <faultstring> 1002 Access denied. You must login to access this service. </faultstring>
         </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.3 UploadMessage

The UploadMessage method is invoked by entering three input parameters (**MPNumber, MessageName, MessageContent)**:

UploadMessage(MPNumber, MessageName, MessageContent)

The following is an example of SOAP message showing the UploadMessage method invocation:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
      <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
         <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
      </SOAPSDK1:SessionInfo>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      < UploadMessage>
         <MPNumber>IDOPERATORE</MPNumber>
         <MessageName>FileName0001.xml</MessageName>
         <MessageContent>.....PKCS#7 Signed Document ..... </MessageContent>
      </UploadMessage>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope
```

Note that:

- in the SOAP header, the SessionId must be filled with the value returned by the Login method;
- **MPNumber** is the alphanumerical identifier (in GME's system) of the Market Participant on whose behalf the XML document is being sent;
- **Message Content** is the signed XML document in ***PKCS#7***[4] format (indicated in the example above above as "**.....PKCS#7 Signed Document .....**"), which is being transmitted (e.g. submission of offers/bids or margins).
  The PKCS #7 message should be PKCS 7 ASN encoded and must include:
  - the XML document to be sent to GME; this document should be compliant with the XML documents described defined in the "XML Implementation Guide for Market Participant" (cfr. [2])
  - the document signature;
  - the digital signature certificate of the signer, with X.509 ASN encoding.

  No other certificates, lists of revocations or qualified attributes should be added.
  The signature should be produced with the algorithms:
  - sha-1 (hashing)[5] + rsa (encryption)

  Appendix F gives some guidelines on how to obtain a signed message in PKCS #7 format using Microsoft CryptoAPI
- **Message Name** is the name the user gives to the message being transmitted.

The method uploads the XML document into GME's system, on behalkf of the Market Participant specified in **MPNumber**.

[4] MIME Base 64 encoded.

[5] At present, GME's system also supports signatures with hashing md5; however, within a short time, this digest algorithm will no longer be accepted and the documents signed with it will be rejected. Therefore, its use is not recommended.

4.3.1 UploadMessage response

After the invocation of the UploadMessage method, GMEWS returns an UploadMessage response.
The following is an example of SOAP message showing an UploadMessage response when the upload transaction has been successfully completed:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
        <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
            <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
        </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
      <UploadMessageResponse>
        <Result>&lt;?xml version=&quot;1.0&quot;?&gt;
               &lt;UPLOAD_RESPONSE&gt;
               &lt;REQUEST_STATUS&gt;COMPLETED&lt;/REQUEST_STATUS&gt;
               &lt;MESSAGE_NAME&gt;FileName001.xml&lt;/MESSAGE_NAME&gt;
               &lt;MESSAGE_ID&gt;33450899166608&lt;/MESSAGE_ID&gt;
               &lt;TIMESTAMP&gt;11/12/2003 09.12.12.275 (GMT+01)&lt;/TIMESTAMP&gt;
               &lt;DATE&gt;20031211&lt;/DATE&gt;
               &lt;TIME&gt;091212&lt;/TIME&gt;
               &lt;/UPLOAD_RESPONSE&gt;
        </Result>
        </UploadMessageResponse>
      </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```

If the tag Result is successful, hereafter is an example of the XML message (with HTML encoding) that is returned:

```
 <?xml version="1.0">
<UPLOAD_RESPONSE>
        <REQUEST_STATUS>COMPLETED</REQUEST_STATUS>
        <MESSAGE_NAME>FileName001.xml</MESSAGE_NAME>
        <MESSAGE_ID>33450899166608</MESSAGE_ID>
        <TIMESTAMP>11/12/2003 09.12.12.275 (GMT+01)</TIMESTAMP>
        <DATE>20031211</DATE>
        <TIME>091212</TIME>
</UPLOAD_RESPONSE>
```

Note that:

- if the <**REQUEST_STATUS>** field shows the value COMPLETED, then the upload transaction has been successfully completed;
- **<MESSAGE_NAME>** displays the name the user gave to the message;
- **<MESSAGE_ID>** displays the identifier that GME's system has assigned to the message (the message can be retrieved from the GME's web portal by entering this identifier in the XML Transactions/Messages Menu);
- **<DATE>** e **<TIME>** show the date and time (timestamp) of receipt of the message. The Electricity market system assigns priorities to received messages according to that date/time).

Instead of the previous message, if the user is not authorised to download messages on behalf of the selected Market Participant, or if the session has expired or for any other reason, a message of SOAP fault type is returned; such message may have the following structure (the example message concerns an expired session):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>
            <faultstring> 1002 Access denied. You must login to access this service. </faultstring>
         </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.4 GetNextMessage

The GetNextMessage method is invoked by entering two input parameters (MPNumber and MaxNumberOfMessages) and two empty output parameters (NumberOfMessages, MessageList):

GetNextMessage(MPNumber, MaxNumberOfMessages, NumberOfMessages, MessageList)

The following is an example of SOAP message invoking the GetNextMessage method:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header>
       <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
             <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
       </SOAPSDK1:SessionInfo>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
       <GetNextMessage>
            <MPNumber>IDOPERATORE</MPNumber>
            <MaxNumberOfMessages>10</MaxNumberOfMessages>
            <NumberOfMessages></NumberOfMessages>
            <MessageList></MessageList>
        </GetNextMessage>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- the SessionId in the SOAP header must be filled with the value returned by the Login method;
- **MPNumber** shows the alphanumerical identifier (in GME’s system) of the Market Participant for which the user is checking for any messages;
- **MaxNumberOfMessages** is the maximum number of messages to be obtained in the response.

4.4.1 GetNextMessage response

After the GetNextMessage method invocation, GMEWS returns a GetNextMessage response. The following is an example of SOAP message showing a GetNextMessage response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
       <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
            <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
       </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
       <GetNextMessageResponse>
          <Result>True</Result>
          <NumberOfMessages>3</NumberOfMessages>
          <MessageList>&lt;MessageList&gt;
 &lt;Message&gt;
  &lt;MessageId&gt;33390899166026&lt;/MessageId&gt;
  &lt;MessageName&gt;FA_FileName001.33390899166026.out.xml&lt;/MessageName&gt;
 &lt;/Message&gt;
 &lt;Message&gt;
  &lt;MessageId&gt;33390899166028&lt;/MessageId&gt;
  &lt;MessageName&gt;FA_ FileName002.33390899166028.out.xml&lt;/MessageName&gt;
 &lt;/Message&gt;
  &lt;Message&gt;
  &lt;MessageId&gt;33390899166096&lt;/MessageId&gt;
  &lt;MessageName&gt;FA_ FileName004.33390899166096.out.xml&lt;/MessageName&gt;
 &lt;/Message&gt;
&lt;/MessageList&gt;
           </MessageList>
        </GetNextMessageResponse>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- the **<MessageList>** tag contains a list of messages <Message>; for each message in the list, the message identifier (MessageId tag) is shown along with the name the GME's system has assigned to the message (MessageName tag);
- if there are no messages to be downloaded (empty list), then a message as below is obtained:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
      <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
          <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
      </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
         <GetNextMessageResponse>
              <Result>False</Result>
              <NumberOfMessages>0</NumberOfMessages>
              <MessageList></MessageList>
         </GetNextMessageResponse>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If the user is not authorised to download messages on behalf of the specified Market Participant, or if the session has expired or for any other reason, a message of SOAP fault type is obtained in resposnse; such message may have the following structure (the example message below concerns an expired session):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>
            <faultstring> 1002 Access denied. You must login to access this service. </faultstring>
         </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.5 ForceDownloadMessage

The ForceDownloadMessage method is invoked by entering two input parameters (MPNumber and MessageId) and two empty parameters (MessageName, MessageContent):

ForceDownloadMessage(MPNumber, MessageId, MessageName, MessageContent)

With this method, the user may download a specific message. In order to do that you need to know the Message identifier the GME's system assigned to that message.

The following is an example of SOAP message with the ForceDownloadMessage invocation:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
      <SOAP-ENV:Header>
           <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
                      <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
           </SOAPSDK1:SessionInfo>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
             <ForceDownloadMessage>
                      <MPNumber>IDOPERATORE</MPNumber>
                      <MessageId>33390899166028</MessageId>
                      <MessageName></MessageName>
                      <MessageContent></MessageContent>
             </ForceDownloadMessage>
         </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- the SessionId in the SOAP header should specify the value that has been returned by the Login method;
- **MPNumber** shows the alphanumerical identifier (in GME's system) of the Market Participant whom the message is addressed to;
- **MessageId** is the identifier that GME's system has assigned to the message to be downloaded.

4.5.1 ForceDownloadMessage response

After the ForceDownloadMessage method invocation, GMEWS returns a ForceDownloadMessage response. The following is an example of SOAP message with a ForceDownloadMessage response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <SOAP-ENV:Header>
            <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
                 <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
            </SOAPSDK1:SessionInfo>
         </SOAP-ENV:Header>
         <SOAP-ENV:Body>
             <ForceDownloadMessageResponse>
               <Result>True</Result>
               <MessageName>FA_FileName001.33390899166028.out.xml</MessageName>
               <MessageContent> …..see similar content in DownloadMessage…..
                 </MessageContent>
                 </ForceDownloadMessageResponse>
           </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- if the requested message exists (**Result** = TRUE), then the **MessageName** tag shows the name of the notification message and the content of the message is in **MessageContent,**.

In this version of GMEWS (1.0.4.3), if the message does not exist, a message of SOAP Fault type is sent in place of the previous message; such message may have the following script:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
          <SOAP-ENV:Fault>
              <faultcode>SOAP-ENV:Server</faultcode>
              <faultstring>[oUIMgr.GetMessages()] This array is fixed or temporarily locked</faultstring>
           </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If the user is not authorised to download messages on behalf of the specified Market Participant, or if the session has expired or for any other reason, a message of SOAP fault type is sent in place of the previous message; such message may have the following structure (the example message below concerns an expired session):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
          <SOAP-ENV:Fault>
              <faultcode>SOAP-ENV:Server</faultcode>
              <faultstring> 1002 Access denied. You must login to access this service. </faultstring>
           </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.6 Logout

The Logout method is invoked without entering any parameter:

Logout()

This method terminates the work session.

The following is an example of SOAP message showing the Logout method invocation:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
      <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
         <SessionId>787F67A9-C240-4EBB-BFB5-FCE253123F05</SessionId>
      </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
        <Logout></Logout>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Note that:

- the SessionId in the SOAP header should specify the value that was returned by the Login method; such value identifies the session to be terminated.

4.6.1 Logout response

After the Logout method invocation, GMEWS returns a Logout response confirming the closing of the session.
The following is an example of SOAP message showing a Logout response:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header>
      <SOAPSDK1:SessionInfo xmlns:SOAPSDK1="http://www.ipex.it/gmews/">
           <SessionId></SessionId>
      </SOAPSDK1:SessionInfo>
     </SOAP-ENV:Header>
     <SOAP-ENV:Body>
        <LogoutResponse>
           <Result>true</Result>
      </LogoutResponse>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

If the session has expired or for any other reason, a message of SOAP Fault type is sent in place of the previous message; such message may have the following structure (the message concerns an expired session):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Server</faultcode>
         <faultstring> 1002 Access denied. You must login to access this service. </faultstring>
       </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Appendix A : WSDL of GMEWS

The following is a description of the Web Service WSDL, which may be obtained at the following URLs:

http://www.ipex.it/gmews/gmews.wsdl
https://www.ipex.it/gmews/gmews.wsdl (requires SSL client certificate authentication)

################################ gmews.wsdl ##

```
<?xml version='1.0' encoding='UTF-8' ?>
 <!-- Generated 07/26/02 by Microsoft SOAP Toolkit WSDL File Generator, Version 1.02.813.0 -->
<definitions  name ='gmews'   targetNamespace = 'http://www.ipex.it/gmews/'
                 xmlns:wsdlns='http://www.ipex.it/gmews/'
                 xmlns:typens='http://www.ipex.it/gmews/'
                 xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
                 xmlns:xsd='http://www.w3.org/2001/XMLSchema'
                 xmlns:stk='http://schemas.microsoft.com/soap-toolkit/wsdl-extension'
                 xmlns='http://schemas.xmlsoap.org/wsdl/'>
  <types>
    <schema targetNamespace='http://www.ipex.it/gmews/'
      xmlns='http://www.w3.org/2001/XMLSchema'
      xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
      xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
      elementFormDefault='qualified'>
    </schema>
  </types>
  <message name='Login'>
    <part name='UserName' type='xsd:string'/>
    <part name='Password' type='xsd:string'/>
  </message>
  <message name='LoginResponse'>
    <part name='Result' type='xsd:boolean'/>
  </message>
  <message name='Logout'>
  </message>
  <message name='LogoutResponse'>
    <part name='Result' type='xsd:boolean'/>
  </message>
  <message name='UploadMessage'>
    <part name='MPNumber' type='xsd:string'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
  <message name='UploadMessageResponse'>
    <part name='Result' type='xsd:string'/>
  </message>
  <message name='GetNextMessage'>
    <part name='MPNumber' type='xsd:string'/>
    <part name='MaxNumberOfMessages' type='xsd:string'/>
    <part name='NumberOfMessages' type='xsd:string'/>
    <part name='MessageList' type='xsd:string'/>
  </message>
  <message name='GetNextMessageResponse'>
    <part name='Result' type='xsd:string'/>
    <part name='NumberOfMessages' type='xsd:string'/>
    <part name='MessageList' type='xsd:string'/>
  </message>
  <message name='ForceDownloadMessage'>
    <part name='MPNumber' type='xsd:string'/>
    <part name='MessageId' type='xsd:string'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
  <message name='ForceDownloadMessageResponse'>
    <part name='Result' type='xsd:string'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
```

```
<message name='DownloadMessage'>
  <part name='MPNumber' type='xsd:string'/>
  <part name='MessageName' type='xsd:string'/>
  <part name='MessageContent' type='xsd:string'/>
</message>
<message name='DownloadMessageResponse'>
  <part name='Result' type='xsd:boolean'/>
  <part name='MessageName' type='xsd:string'/>
  <part name='MessageContent' type='xsd:string'/>
</message>
<message name='SessionInfo'>
  <part name='SessionId' type='xsd:string'/>
</message>
<portType name='ManagerSoapPort'>
  <operation name='Login' parameterOrder='UserName Password'>
    <input message='wsdlns:Login' />
    <output message='wsdlns:LoginResponse' />
  </operation>
  <operation name='Logout' parameterOrder=''>
    <input message='wsdlns:Logout' />
    <output message='wsdlns:LogoutResponse' />
  </operation>
  <operation name='UploadMessage' parameterOrder='MPNumber MessageName MessageContent'>
    <input message='wsdlns:UploadMessage' />
    <output message='wsdlns:UploadMessageResponse' />
  </operation>
  <operation name='GetNextMessage' parameterOrder='MPNumber MaxNumberOfMessages NumberOfMessages MessageList'>
    <input message='wsdlns:GetNextMessage' />
    <output message='wsdlns:GetNextMessageResponse' />
  </operation>
  <operation name='ForceDownloadMessage' parameterOrder='MPNumber MessageId MessageName MessageContent'>
    <input message='wsdlns:ForceDownloadMessage' />
    <output message='wsdlns:ForceDownloadMessageResponse' />
  </operation>
  <operation name='DownloadMessage' parameterOrder='MPNumber MessageName MessageContent'>
    <input message='wsdlns:DownloadMessage' />
    <output message='wsdlns:DownloadMessageResponse' />
  </operation>
</portType>
<binding name='ManagerSoapBinding' type='wsdlns:ManagerSoapPort' >
  <stk:binding preferredEncoding='UTF-8'/>
  <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http' />
  <operation name='Login' >
    <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
    <input>
      <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                        use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </input>
    <output>
      <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                        use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </output>
  </operation>
  <operation name='Logout' >
    <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
    <input>
      <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                        use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </input>
    <output>
      <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                        use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    </output>
  </operation>
  <operation name='UploadMessage' >
    <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
    <input>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </input>
   <output>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </output>
  </operation>
  <operation name='GetNextMessage' >
   <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
   <input>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </input>
   <output>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </output>
  </operation>
  <operation name='ForceDownloadMessage' >
   <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
   <input>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </input>
   <output>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </output>
  </operation>
  <operation name='DownloadMessage' >
   <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
   <input>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </input>
   <output>
    <soap:header required='true' message='wsdlns:SessionInfo' part='SessionId'
                          use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
    <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                          encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
   </output>
  </operation>

 </binding>
 <service name='gmews' >
  <port name='SoapPort' binding='wsdlns:ManagerSoapBinding' >
   <soap:address location='https://www.ipex.it/gmews/wslistener.asp' />
  </port>
 </service>
</definitions>
```

Appendix B : WSDL of GMEWS, compatible with .NET Framework 1.1

The following is the Web Service WSDL, which is .NET Framework 1.1 compatible and which may be obtained at the following URLs:

http://www.ipex.it/gmews/gmews_NET.wsdl
https://www.ipex.it/gmews/gmews_NET.wsdl

Given the distinctive features of .NET Framework, a compatible specific WSDL was developed.

This WSDL addresses compatibility aspects that are not supported by the WSDL described in Appendix A, which was originally tested for compatibility against the SOAP Toolkit.
In particular, this WSDL uses complex types in the SOAP header and automatic generation of .NET code for a Web Service client application was proven.

```
###########################   gmews_NET.wsdl  ########################################################

<?xml version='1.0' encoding='UTF-8' ?>
 <!-- Generated 07/26/02 by Microsoft SOAP Toolkit WSDL File Generator, Version 1.02.813.0 -->
<definitions  name ='gmews'   targetNamespace = 'http://www.ipex.it/gmews/'
                 xmlns:wsdlns='http://www.ipex.it/gmews/'
                 xmlns:typens='http://www.ipex.it/gmews/'
                 xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
                 xmlns:xsd='http://www.w3.org/2001/XMLSchema'
                 xmlns:stk='http://schemas.microsoft.com/soap-toolkit/wsdl-extension'
                 xmlns='http://schemas.xmlsoap.org/wsdl/'>
  <types>
    <schema targetNamespace='http://www.ipex.it/gmews/'
      xmlns='http://www.w3.org/2001/XMLSchema'
      xmlns:SOAP-ENC='http://schemas.xmlsoap.org/soap/encoding/'
      xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
      elementFormDefault='qualified'>

      <xsd:complexType name="SessionInfo">
        <xsd:sequence>
          <xsd:element minOccurs="0" maxOccurs="1" name="SessionId" type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
     </schema>
  </types>
  <message name='Login'>
    <part name='UserName' type='xsd:string'/>
    <part name='Password' type='xsd:string'/>
  </message>
  <message name='LoginResponse'>
    <part name='Result' type='xsd:boolean'/>
  </message>
  <message name='Logout'>
  </message>
  <message name='LogoutResponse'>
    <part name='Result' type='xsd:boolean'/>
  </message>
  <message name='UploadMessage'>
    <part name='MPNumber' type='xsd:string'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
  <message name='UploadMessageResponse'>
    <part name='Result' type='xsd:string'/>
  </message>
  <message name='GetNextMessage'>
    <part name='MPNumber' type='xsd:string'/>
    <part name='MaxNumberOfMessages' type='xsd:string'/>
    <part name='NumberOfMessages' type='xsd:string'/>
    <part name='MessageList' type='xsd:string'/>
  </message>
  <message name='GetNextMessageResponse'>
    <part name='Result' type='xsd:string'/>
    <part name='NumberOfMessages' type='xsd:string'/>
    <part name='MessageList' type='xsd:string'/>
  </message>
  <message name='ForceDownloadMessage'>
    <part name='MPNumber' type='xsd:string'/>
    <part name='MessageId' type='xsd:string'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
  <message name='ForceDownloadMessageResponse'>
    <part name='Result' type='xsd:string'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
  <message name='DownloadMessage'>
    <part name='MPNumber' type='xsd:string'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
  <message name='DownloadMessageResponse'>
    <part name='Result' type='xsd:boolean'/>
    <part name='MessageName' type='xsd:string'/>
    <part name='MessageContent' type='xsd:string'/>
  </message>
  <message name='SessionInfoHeader'>
    <part name='SessionInfo' type='wsdlns:SessionInfo'/>
  </message>
  <portType name='ManagerSoapPort'>
    <operation name='Login' parameterOrder='UserName Password'>
      <input message='wsdlns:Login' />
      <output message='wsdlns:LoginResponse' />
    </operation>
    <operation name='Logout' parameterOrder=''>
      <input message='wsdlns:Logout' />
      <output message='wsdlns:LogoutResponse' />
    </operation>
    <operation name='UploadMessage' parameterOrder='MPNumber MessageName MessageContent'>
      <input message='wsdlns:UploadMessage' />
      <output message='wsdlns:UploadMessageResponse' />
    </operation>
    <operation name='GetNextMessage' parameterOrder='MPNumber MaxNumberOfMessages NumberOfMessages MessageList'>
      <input message='wsdlns:GetNextMessage' />
      <output message='wsdlns:GetNextMessageResponse' />
    </operation>
    <operation name='ForceDownloadMessage' parameterOrder='MPNumber MessageId MessageName MessageContent'>
      <input message='wsdlns:ForceDownloadMessage' />
      <output message='wsdlns:ForceDownloadMessageResponse' />
    </operation>
    <operation name='DownloadMessage' parameterOrder='MPNumber MessageName MessageContent'>
      <input message='wsdlns:DownloadMessage' />
      <output message='wsdlns:DownloadMessageResponse' />
    </operation>
  </portType>
  <binding name='ManagerSoapBinding' type='wsdlns:ManagerSoapPort' >
    <stk:binding preferredEncoding='UTF-8'/>
    <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http' />
    <operation name='Login' >
      <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
      <input>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                      use='encoded' namespace='http://www.ipex.it/gmews/'
                      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                      use='encoded' namespace='http://www.ipex.it/gmews/'
                      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                      encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </output>
    </operation>
    <operation name='Logout' >
      <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
      <input>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </output>
    </operation>
    <operation name='UploadMessage' >
      <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
      <input>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </output>
    </operation>
    <operation name='GetNextMessage' >
      <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
      <input>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </output>
    </operation>
    <operation name='ForceDownloadMessage' >
      <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
      <input>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </output>
    </operation>
    <operation name='DownloadMessage' >
      <soap:operation soapAction='https://www.ipex.it/gmews/wslistener.asp' />
      <input>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </input>
      <output>
        <soap:header required='true' message='wsdlns:SessionInfoHeader' part='SessionInfo'
                             use='encoded' namespace='http://www.ipex.it/gmews/'
                             encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
```

```
        <soap:body use='encoded' namespace='http://www.ipex.it/gmews/'
                              encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' />
      </output>
    </operation>
  </binding>
  <service name='gmews' >
    <port name='SoapPort' binding='wsdlns:ManagerSoapBinding' >
      <soap:address location='https://www.ipex.it/gmews/wslistener.asp' />
    </port>
  </service>
</definitions>
```

Appendix C : Suggestions for the use of digital certificates in the Windows environment

1) The client application will require two certificates: one SSL client authentication certificate and one certificate for signing purpose. The Windows environment with Internet Explorer 6.0.28 or higher rejects authentication via a certificate intended for signing purpose.
   Make sure to select certificate intended for SSL client authentication when managing authentication and setting up the https session.

2) To use certificates in the Windows environment, certificates need to be imported into the Windows key stores. In order to import a certificate that is located on a hardware encryption device (SmartCard or token), you need to use the software tools supplied with the device. By default, these tools usually import the certificates into the CURRENT_USER Windows store. If the client application is a multi-user one and the certificates are to be accessed independently of the type of connected user, the certificates should be physically moved into the Windows LOCAL_MACHINE store.

3) To allow a client application on a Windows environment to successfully use digital certificates, those certificates need to be trusted on the client machine.
   A non-trusted certificate on the client machine gives errors (cross on red background) or warnings (yellow question points): Windows always tries to recover the complete trust chain before using a certificate and this may cause malfunctions in the client application.
   Thus, in addition to user certificates, also the certificate of the Certification Authority (CA) that issued the certificate should be imported into the CURRENT_USER (or LOCAL_MACHINE) stores. If the CA is a subordinate CA, then the whole chain of CA certificates leading from such CA to a root CA should be imported.
   CA root certificates should be imported into “Trusted Root Certification Authorities” store.
   Subordinate CA certificates should be imported into “Intermediate Certification Authorities” store.
   User certificates should be imported into the Personal store.

4) Problems were encountered upon downloading the WSDL in HTTPS, when using the SOAP Toolkit. The WDSL may be downloaded also over http (no SSL) at the address specified in Appendix A.

5) .NET Framework version 1.1, has no native classes for direct access to certificates in the Windows key stores. Possible alternatives approaches to overcome this limitation are:

   A) Access the Windows key stores via the platform invoke (Pinvoke) of the corresponding Microsoft CryptoAPI primitives (they can be found in crypt32.dll)

   B) Use the Microsoft CAPICOM component which supports access to the certificate stores (release 2.0.0.3 is currently downloadable from the Microsoft site and also provides examples in the C# language)

   C) Install the additional Web Service Extension (WSE) package; however, version 2.0 (compatible with .NET Framework 1.1) is still unofficial and not integrated with Framework.

   The next version of .NET Framework, named “Longhorn” (still in the alpha version at present, see http://longhorn.msdn.microsoft.com/lhsdk/ndp/ovrwhatsnewinwhidbeyalpha.aspx), is expected to integrate some functionality of CAPICOM for accessing certificate stores.

Appendix D : example of invocation of the Login method with SOAP Toolkit 3.0

The following is an excerpt of code in the Visual Basic 6.0 language, exemplifying how the Login method may be invoked by using SOAP Toolkit 3.0:

ClientHeaderHandler Class (used for managing the SOAP header in the SOAP Toolkit):

```
############################## ClientHeaderHandler.cls ######################################

Option Explicit
Implements IHeaderHandler

Public m_SessionId As String
Private Const HEADER_ELEMENT_NAMESPACE As String = _
"http://p3.elsag.it/gmews/"
Private Const HEADER_ELEMENT_NAME As String = "SessionInfo"
Private Const USER_ELEMENT_NAME As String = "SessionId"

Private Sub Class_Initialize()
  m_SessionId = ""
End Sub

Private Function IHeaderHandler_readHeader( _
    ByVal pReader As SoapReader30, _
    ByVal pHeaderNode As MSXML2.IXMLDOMNode, _
    ByVal pObject As Object) _
    As Boolean

    Dim SessionId
    SessionId = pHeaderNode.selectSingleNode("SessionId").Text
    If (SessionId <> "") Then
      m_SessionId = SessionId
    End If
    IHeaderHandler_readHeader = True
End Function

Private Function IHeaderHandler_willWriteHeaders() _
    As Boolean
    IHeaderHandler_willWriteHeaders = True
End Function

Private Sub IHeaderHandler_writeHeaders( _
    ByVal pSerializer As MSSOAPLib30.ISoapSerializer, _
    ByVal pObject As Object)

    pSerializer.StartHeaderElement HEADER_ELEMENT_NAME, HEADER_ELEMENT_NAMESPACE
    pSerializer.startElement USER_ELEMENT_NAME
    pSerializer.WriteString m_SessionId
    pSerializer.endElement
    pSerializer.EndHeaderElement

End Sub

############################## ClientHeaderHandler.cls ######################################
```

Login method implementation:

m_Info.WSDLPath="http://www.ipex.it/gmews/gmews.wsdl"
m_Info.EndPointUrl = "https://www.ipex.it/gmews/wsListener.asp"
m_Info.SSLEnable = True
m_Info.Version = ""
m_Info.SSLCertificate = Common Name (? proprieta' CN) of the certificate used

The method:

```
 SetProxy oSoapClient
```

may be used to set the parameters of the proxy, if the user relies on a proxy server for connecting to the Internet (see "ProxyServer", "ProxyUser", "ProxyPassword" properties of the HttpConnector30 object).

```
############################  Login (LoginId, Password) #######################################

  Dim bRes As Boolean
  Dim oSoapClient As MSSOAPLib30.SoapClient30
  Dim oHeaderHandler As ClientHeaderHandler
  Dim number As Long


  Set oSoapClient = New MSSOAPLib30.SoapClient30
  oSoapClient.MSSoapInit m_Info.WSDLPath

  oSoapClient.ConnectorProperty("EndPointURL") = m_Info.EndPointUrl

  If (m_Info.SSLEnable) Then
     oSoapClient.ConnectorProperty("SSLClientCertificateName") = m_Info.SSLCertificate
  End If

  Set oHeaderHandler = New ClientHeaderHandler
  oHeaderHandler.m_SessionId = m_Info.Version
  oHeaderHandler.m_SessionId = ""
  Set oSoapClient.HeaderHandler = oHeaderHandler

  SetProxy oSoapClient

 '
```

If everything has been correctly installed/configured, a dialog box is shown when running this method, as an intermediate step in the SSL handshaking: the user is asked to enter the required PIN into the dialog box to access the private key of the certificate that is resident on the hardware encryption device

```
 '
  bRes = oSoapClient.Login(LoginId, Password)

  If (bRes) Then
    m_SessionId = oHeaderHandler.m_SessionId
  End If

############################  End of Login method #######################################
```

Appendix E : example of invocation of the Login/Download methods with .NET Framework 1.1

The following is an example of VB.NET code for invoking the Login/Download methods in the .NET Framework 1.1.
The example assumes that the certificate was exported into a file.

For details on how to access the digital certificates on the Windows key store in the .NET Framework, refer to Appendix C – par. 5.

The **WebReference** proxy class is automatically generated in Visual Studio .NET 2003 through the function “Add Web Reference”, when selecting the WSDL **gmews_NET.wsdl** which is defined in Appendix B.

```
#############################        ClientWS.vb        ##########################################

Imports System
Imports System.Security.Cryptography.X509Certificates
Imports System.Security.Cryptography
Imports System.Net
Imports System.Text

Module ClientWS
   Sub Main()

      Dim SessionId As String
      Dim MPNumber As String = "IDOP"
      Dim MessageName As String = ""
      Dim MessageContent As String = ""
      SessionId = ""
      Login(SessionId)
      If SessionId <> "" Then
         Download(SessionId, MPNumber, MessageName, MessageContent)
      End If
   End Sub

   Sub Login(ByRef SessionId As String)
      Dim VSNETClient As New WebReference.gmews
      Dim strMessage As String

      '
      ' Proxy setting used for connection
      '
      SetProxy  “proxyuser”, “proxypassword”, “proxydomain”

      ‘
      ‘  Loading of X509 certificate: the SSL client authentication certificate is available on a file.
      ‘
      Dim x509 As X509Certificate = X509Certificate.CreateFromCertFile("D:\PublicAuthenticationCertificate.cer")
      VSNETClient.ClientCertificates.Add(x509)

      Dim bLoginResult As Boolean
      Dim szLogin As String = ""
      Dim szPassword As String = ""
      Try
         VSNETClient.SessionInfoValue = New WebReference.SessionInfo
         VSNETClient.SessionInfoValue.SessionId = ""

‘ If everything has been correctly installed/configured, a dialog box is shown when running this method, as an intermediate step in the SSL handshaking; the user is asked to enter the required PIN to unlock access to the private key of the certificate that is stored on the on the hardware encryption device

        bLoginResult = VSNETClient.Login(szLogin, szPassword)
      Catch err As Exception
         strMessage = err.Message
      Finally
```

```
            SessionId = VSNETClient.SessionInfoValue.SessionId
        End Try
    End Sub

    Sub Download(ByVal SessionId As String, ByVal MPNumber As String, ByRef MessageName As String, ByRef MessageContent As
String)
        Dim VSNETClient As New WebReference.gmews
        Dim strMessage As String
        '
        ' Proxy setting used for connection
        '
        SetProxy  “proxyuser”, “proxypassword”, “proxydomain”
        ‘
        ‘  Loading of X509 certificate: the SSL client authentication certificate is available on a file.
        ‘
        Dim x509 As X509Certificate = X509Certificate.CreateFromCertFile("D:\PublicAuthenticationCertificate.cer")
        VSNETClient.ClientCertificates.Add(x509)
        Dim bDownloadResult As Boolean

        Try
            VSNETClient.SessionInfoValue = New WebReference.SessionInfo
            VSNETClient.SessionInfoValue.SessionId = SessionId
            bDownloadResult = VSNETClient.DownloadMessage(MPNumber, MessageName, MessageContent)
        Catch err As Exception
            strMessage = err.Message
        End Try
    End Sub
   End Sub
End Module
```

##########################   End of ClientWS.vb   ######################################

Appendix F : preparation of a PKCS#7 signed message

The following is an excerpt of C++ code showing how a PKCS #7 signed message can be prepared created using the Microsoft CryptoAPI:

```
//
// Initially, the user must acquire the context of the certificate for signing and assign it to pCertContext
//
pCertContext : context of the selected signature certificate
//
// Definition of hashing algorithm:  SHA-1
//
#define HASH_ALGORITHM              szOID_OIWSEC_sha1
// #define HASH_ALGORITHM             szOID_RSA_MD5
//
// definition of the encoding types to be used:  PKCS#7 ASN encoding for the signed message and of X.509 ASN for the certificate
//
#define MY_ENCODING_TYPE  (PKCS_7_ASN_ENCODING | X509_ASN_ENCODING)

#####  Sign( BYTE *InData, long InDataLen, BYTE **OutData, long *OutDataLen) ########

//  the CRYPT_SIGN_MESSAGE_PARA structure must be prepared; it gives the parameters that are to be used
//  for signing the message
    memset(&SignMessagePara, 0, sizeof(CRYPT_SIGN_MESSAGE_PARA));
    SignMessagePara.cbSize = sizeof(CRYPT_SIGN_MESSAGE_PARA);
    SignMessagePara.HashAlgorithm.pszObjId = HASH_ALGORITHM;  //sets the hashing algorithm(digest algorithm)
    SignMessagePara.pSigningCert = pCertContext;              //sets the certificate to be used for signing
    SignMessagePara.dwMsgEncodingType = MY_ENCODING_TYPE; //sets the PKCS#7 message and certificate encoding
    SignMessagePara.cMsgCert = 1;                             //sets that one certificate is included in the PKCS #7 message
    SignMessagePara.rgpMsgCert = &pCertContext;               //selects the certificate to be included into the PKCS #7 message
                                                               //NOTE: the hash encryption (RSA) algorithm for signing
                                                               signature is derived from the algorithm specified in the public
                                                               key certificate
    rgcbToBeSigned[0] = InDataLen;
    rgpbToBeSigned[0] = (BYTE *)InData;
    // obtains the maximum size of the buffer holding the final document with the signature
    //
    rc=CryptSignMessage(&SignMessagePara,
                    FALSE,          // undetached signature
                    1,
                    rgpbToBeSigned,
                    rgcbToBeSigned,
                    NULL,
                    &OutDataLenSign);
    //
    // produces the signature of the document
    //
    *OutData = (char *)malloc(OutDataLenSign);
     rc=CryptSignMessage(&SignMessagePara,
                    FALSE,
                    1,
                    rgpbToBeSigned,
                    rgcbToBeSigned,
                    (BYTE *)(*OutData),
                    &OutDataLenSign);
    *OutDataLen=OutDataLenSign;
#####  Sign( BYTE *InData, long InDataLen, BYTE **OutData, long *OutDataLen) ########
```

After, buffer *OutData has to be encoded as MIME Base 64.

Guidelines on how to produce a PKCS#7 signed message in Framework .NET 1.1

Framework .NET version 1.1 does not directly support the production of signed messages in PKCS#7 format: it supports the SHA1 hashing but not the encryption that is required to obtain the signature of the PKCS#7 message content.
Next version of the .NET Framework, called "Longhorn", (still in the alpha stage, see http://longhorn.msdn.microsoft.com/lhsdk/ndp/ovrwhatsnewinwhidbeyalpha.aspx) is expected to support PKCS#7 signed documents.

Possible alternative approaches to overcome this limitation and produce a PKCS#7 signed message using Framework .NET 1.1. are:

A) Use the platform invoke (Pinvoke) of the corresponding functions of Microsoft CryptoAPI Microsoft (residing on DLL crypt32.dll)

B) Use the CAPICOM component of Microsoft to sign the message (in the .NET Framework 1.1 environment, but also from VB6): by default, this component produces a PKCS#7 signed message with SHA1 hashing).

Note

The CAPICOM component internally converts the text to be signed into Unicode 2 encoding (2 bytes per character). GME's system, instead, expects that the XML document to be received (with "iso-8859-1 or iso Latin-1 encoding – cfr. [2]) is transmitted with ANSI encoding (1 byte per character).
The following are two workarounds (for VB6 and for C# of Framework .NET 1.1) to bypass this problem.

B1) In VB6, force a conversion from Unicode to ByteString on the content to be signed, by calling a routine such as the one described below:

```
'******************************* Example  VB6  **********************************

Content: content of the document to be signed
SignedData: CAPICOM.SignedData object
Signer: CAPICOM.Signer object

'*****************************************************************************
' Subroutine: Ustr2Bstr
' Synopsis  : Unicode to ByteString conversion
' Parameter :
'           Ustr - Unicode String
'*****************************************************************************
function Ustr2Bstr(Ustr)
  'unicode string to byte string conversion
  dim lngLoop
  dim strChar
  Ustr2Bstr = ""
  for lngLoop = 1 to Len(Ustr)
        strChar = Mid(Ustr, lngLoop, 1)
        Ustr2Bstr = Ustr2Bstr & ChrB(AscB(strChar))
  next
end function

….
SignedData.Content = Ustr2Bstr(Content)
Message = SignedData.Sign(Signer, false)
….

'******************************* End of VB6 example  **********************************
```

B2) In .NET 1.1, replace the “string” type with “native int” and redefine the property content type from string to native int, thereby overriding the automatic conversion of strings into Unicode. The process is shown hereunder:

In Framework .NET, the string datatype (string) is a string with Unicode characters.
Import CAPICOM into Framework .NET 1.1 with the following steps:

1) tlbimp capicom.dll /out=Interop.CAPICOM.dll /namespace=CAPICOM

2) ildasm Interop.CAPICOM.dll /out=capicom.il

3) replace "string " with "native int" in the following rows of the capicom.il file:

```
property string Content()
get instance string CAPICOM.SignedDataClass::get_Content()
set instance void CAPICOM.SignedDataClass::set_Content(string)
get instance string CAPICOM.ISignedData::get_Content()
set instance void CAPICOM.ISignedData::set_Content(string)
get instance string CAPICOM.EnvelopedDataClass::get_Content()
set instance void CAPICOM.EnvelopedDataClass::set_Content(string)
set instance void CAPICOM.IEnvelopedData::set_Content(string)
get instance string CAPICOM.IEnvelopedData::get_Content()
get instance string CAPICOM.EncryptedDataClass::get_Content()
set instance void CAPICOM.EncryptedDataClass::set_Content(string)
get instance string CAPICOM.IEncryptedData::get_Content()
set instance void CAPICOM.IEncryptedData::set_Content(string)
```

4) replace "string marshal(bstr)" with "native int" in the following rows of the capicom.il file:

```
instance void set_Content([in] string marshal( bstr) pVal) runtime
managed internalcall
instance string marshal( bstr) get_Content() runtime managed
internalcall
```

After redefining the property Content type from the string type of the .NET environment to native int, you may override the automatic conversion of strings into Unicode.
Then, to produce the digital signature, encode a portion of “unsafe” code in C#, as shown hereunder:

```
'******************************* Example C#  *****************************************************

byteArrayContent: byte array of the document to be signed
signedData: CAPICOM.SignedDataClass object imported into Framework .NET (tlbimp.exe)
signer: CAPICOM.SignerClass object imported into Framework .NET (tlbimp.exe)
....
unsafe
{
    fixed (byte * bstrPtr= &byteArrayContent[0])
    {
        signedData.Content = new IntPtr (bstrPtr);
    }
}
signedContent = signedData.Sign(signer, false,CAPICOM_ENCODING_TYPE.CAPICOM_ENCODE_BASE64);

'******************************* End of Example C#  ***************************************************
```