Web Services GME

data: **27-feb-04**

INDICE

1. Introduzione

Allo scopo di permettere una connessione applicativa tra sistema informatico dell'Operatore di Mercato ed il sistema informatico del Mercato Elettrico, quest'ultimo espone un'interfaccia di tipo programmatico.

Questa interfaccia è realizzata tramite il Web Service GMEWS.

1.1 Scopo

Scopo di questo documento è di fornire una descrizione del funzionamento e delle modalità di utilizzo del Web Service GMEWS versione 1.0.4.3.

1.2 DEFINIZIONI & ACRONIMI

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

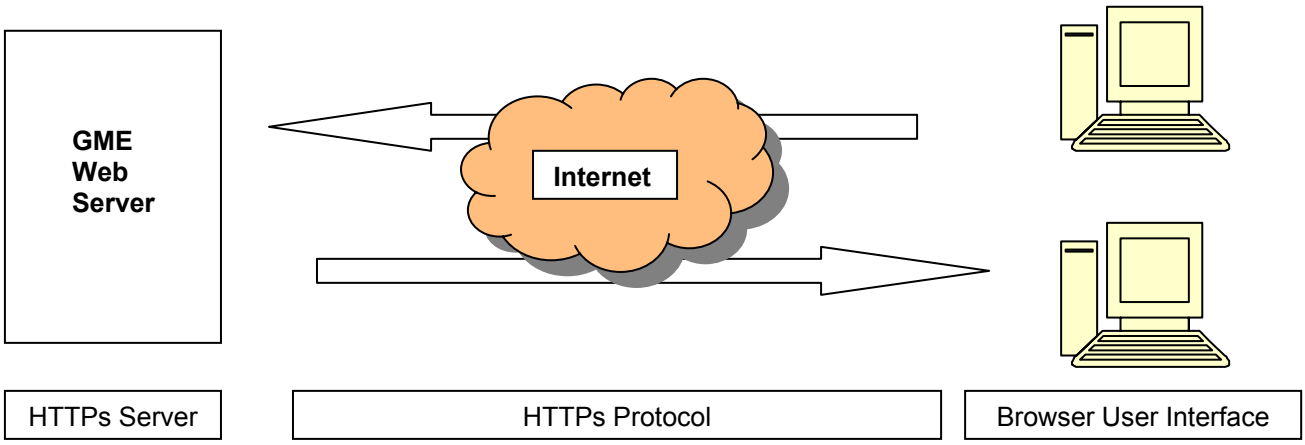
1.3 Documenti di riferimento

[1] RSA Laboratories. *PKCS #7: Cryptographic Message Syntax Standard.* Version 1.5, Nov. 1993

[2] "XML Implementation Guide for Market Participant", documento messo a disposizione dal Gestore del Mercato Elettrico sulla pagina Web http://www.mercatoelettrico.org/Biblioteca/BiblioDefault.aspx nella parte Documenti Tecnici - Sistema Informatico)

2. Il contesto di riferimento

Lo scenario di interazione con il sistema del Mercato Elettrico, installato presso il GME, è schematizzato nella figura seguente:

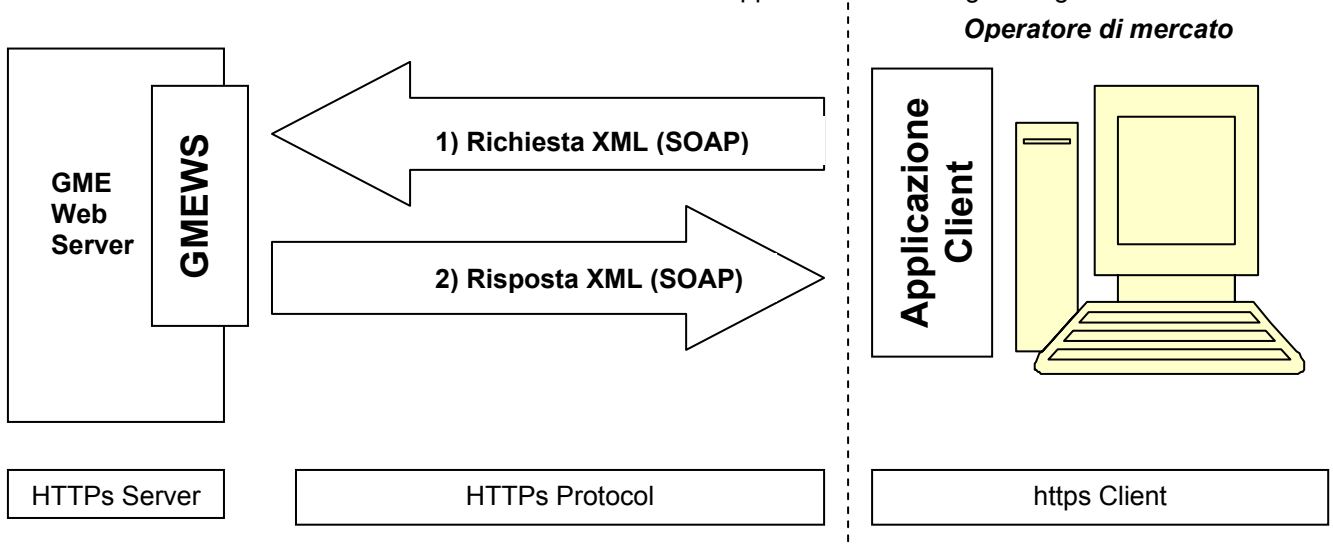


Gli operatori di Mercato, attraverso la rete Internet, accedono al Web Server dove risiede l'applicazione portale, di tipo BUI, che permette di effettuare le operazioni di consultazione del mercato e la sottomissione di offerte di acquisto/vendita. L'autenticazione nei confronti del Web Server GME avviene utilizzando un certificato digitale di autenticazione associato ad un determinato utente abilitato a compiere operazioni per conto dell'Operatore di Mercato.

La sottomissione delle offerte puó avvenire sia online, tramite web form, che tramite upload di file XML opportunamente strutturati. Per garantire la non ripudiabilità delle offerte presentate al GME, l'Operatore di Mercato vi appone la firma digitale e allega il proprio certificato di firma.

Allo scopo di consentire l'interazione con il sistema del GME attraverso un'interfaccia programmatica è stato predisposto il modulo GMEWS, che implementa una interfaccia di tipo web service, basata su messaggi SOAP e trasporto https.

Lo schema di funzionamento del modulo GMEWS é rappresentato dalla figura seguente:



Il modulo GMEWS espone le sue funzionalità su Internet tramite un file di descrizione in formato WSDL, che descrive come strutturare le richieste XML per invocare i metodi supportati dal Web Service e come vanno interpretate le relative risposte.

In Appendice A é riportato il WSDL del GME WS.

L'applicazione client deve invocare i metodi supportati dal Web Service inviando richieste XML strutturate come descritto nel WSDL.

Il Web Server traduce la richiesta XML ricevuta nella relativa chiamata alla funzionalità richiesta ed attende la sua esecuzione. Al termine dell'esecuzione del metodo richiamato, traduce il risultato in una risposta XML e la rispedisce all'applicazione client.

L'applicazione client, utilizzando ancora il file di descrizione, é in grado di interpretare la risposta XML e ricavare il risultato dell'operazione.

Le richieste e le risposte XML sono messaggi SOAP che devono essere trasmessi all'interno di connessione https

Un'applicazione client per potere interagire con il GMEWS deve quindi:
- gestire la comunicazione via https con il sito web del GME che ospita il GMEWS, inclusa la fase di autenticazione SSL, basata sull'uso di certificati client SSL
- preparare messaggi SOAP di richiesta come descritto dal WSDL e interpretare i corrispondenti messaggi di risposta

Nel caso di invio di un documento al GME (sottomissione offerte, sottomissione margini) il messaggio SOAP inviato al GMEWS deve contenere un documento firmato in formato PKCS #7 (cfr. [1]), con firma a validità legale.
L'applicazione client deve quindi firmare il documento prima di inserirlo in un messaggio SOAP di richiesta[1]

Sono quindi necessari per una applicazione client che deve interagire con il GMEWS:
- un certificato client di autenticazione SSL (per gestire l'autenticazione SSL con il GMEWS)
- un dispositivo di firma (con i driver associati) e un certificato di firma qualificata a norme AIPA (per apporre la firma sui documenti da inviare)
- tools o librerie SW in grado di gestire:
    - comunicazione http + SSL (https)
    - SOAP messaging secondo il WSDL del GMEWS
    - preparazione di messaggi firmati in formato PKCS #7 (interagendo con i driver del dispositivo di firma)

Su una piattaforma Microsoft per lo sviluppo dell'applicazione si possono sfruttare:
- Microsoft SOAP toolkit 2.0 SP2 o 3.0 e .NET Framework 1.1 (comunicazione https + SOAP messaging)
- Microsoft CryptoAPI (preparazione di documenti in formato PKCS #7)

rispetto ai quale è stata verificata la compatibilità e l'interoperabilità.

In Appendice C sono forniti alcuni suggerimenti sulle impostazioni da effettuare in ambiente Microsoft per l'uso dei certificati.

[1] Il documento firmato non deve essere cifrato perche' la cifratura e' effettuata a livello di trasporto TCP/IP attraverso la connessione SSL con il GMEWS.

3. Funzioni esposte dal GMEWS

3.1 Descrizione

Il Web Service GMEWS fornisce l'accesso programmatico ad un sottoinsieme delle funzionalità disponibili all'utente collegandosi al portale (BUI) del sistema GME.

Come l'accesso BUI anche l'interazione tramite il GMEWS sfrutta il protocollo HTTPS - a protezione dei dati trasmessi - ed è soggetta ad autenticazione SSL client, basata su certificato digitale, per verificare l'identità di chi si collega.[2]: per potere invocare le funzioni del GMEWS l'applicazione client deve autenticarsi con web server del GME che ospita il GMEWS (tramite invio di certificati digitale SSL client) e stabilire con esso una connessione SSL.

3.2 Funzionalità

Il GMEWS espone la seguenti funzioni :

- **Login**
  Consente di stabilire una sessione di lavoro con il sistema del GME.
  È la prima funzione che l'applicativo client deve invocare per potere usufruire delle funzionalità esposte dal GMEWS. Con questa operazione l'Operatore di Mercato si identifica nei confronti del sistema GME.
  Il GMEWS restituisce nella risposta un identificativo di sessione che l'applicazione client deve indicare in tutte le successive invocazioni di funzioni del GMEWS.

- **Logout**
  Termina una sessione di lavoro con il sistema GME. Quando ha terminato la sessione di lavoro, l'applicativo dell'utente deve invocare questa funzione al fine di rilasciare le risorse utilizzate.

- **UploadMessage**
  Consente di inviare un documento XML al sistema GME. Il documento deve essere nel formato accettato dal sistema del Mercato Elettrico (vedi "XML Implementation Guide for Market Participant" (cfr. [2]).
  Il GMEWS si attende un documento firmato in formato PKCS#7 (cfr [1])[3], comprendente il certificato di firma digitale impiegato.
  Se il documento è accettato il GMEWS restituisce una risposta che contiene l'identificativo attribuito al documento inviato, la data e l'ora di ricezione; se non è accettato restituisce in risposta un documento XML contenente la motivazione del fallimento dell'operazione.

- **DownloadMessage**
  Consente di scaricare eventuali messaggi di notifica relativi ai documenti di offerta presentati sul mercato.
  Questa funzione controlla se ci sono messaggi di notifica ancora da scaricare (cioè non letti) e, se ce ne sono, restituisce il primo messaggio scaricabile Il messaggio di notifica è nel formato accettato dal sistema del Mercato Elettrico (vedi "XML Implementation Guide for Market Participant" (cfr. [2]).

[2] Come si evince dal WSDL il web service è predisposto per supportare l'autenticazione tramite login e password, ma per la partecipazione effettiva al mercato questa modalità di autenticazione non è accettata.

[3] Codificato MIME Base 64

Il sistema GME marca come “letto” il messaggio restituito; una successiva invocazione del metodo DownloadMessage non restituisce più quel messaggio. Per scaricare un messaggio già letto si può ricorrere a ForceDownloadMessage
Il messaggio di notifica e’ nel formato accettato dal sistema del Mercato Elettrico (vedi “XML Implementation Guide for Market Participant” (cfr. [2]).

- **GetNextMessage**
  Permette di ottenere la lista dei messaggi di notifica ancora da scaricare (non letti). Restituisce una lista in formato XML che contiene il nome e l’identificativo di ciascun messaggio di notifica scaricabile. Il numero massimo dei messaggi inclusi nella lista può essere indicato all’invocazione.

- **ForceDownloadMessage**
  Consente di scaricare uno specifico messaggio di notifica individuato tramite il suo identificativo, anche se questo era già stato scaricato.
  Il sistema GME comunque marca come “letto” il messaggio restituito.

4. Uso del GMEWS

Di seguito sono fornite indicazioni sull'impiego del GMEWS, secondo quanto descritto nel WSDL.

4.1 Login

È il metodo del GMEWS che deve essere chiamato per primo.
Stabilisce una sessione di lavoro con il sistema GME.

Il metodo ha due parametri di input, UserName e Password

Login(UserName, Password)

Di seguito è riportato un esempio messaggio SOAP che contiene l'invocazione del metodo login

```
<?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>
```

Si evidenzia quanto segue:

- il SOAP Header deve contenere un elemento SessionId vuoto (nel WSDL il SOAP header infatti e' definito con attributo required='true')
- I parametri Username e Password possono essere valorizzati con valore qualsiasi dal momento che, essendo richiesta l'autenticazione SSL client, il GMEWS identifica l'utente in base al SubjectName del certificato digitale usato per l'autenticazione SSL.

In Appendice D e Appendice E sono fornite alcune indicazioni su come invocare il metodo login e stabilire una sessione con il GMEWS.

4.1.1 Login response

A seguito della invocazione del metodo login il GME WS restituisce una Login response, a conferma che si è stabilita una sessione di lavoro.
Di seguito è riportato un esempio messaggio SOAP che contiene una 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>
```

Si evidenzia quanto segue:

- Se la risposta e' positiva (Result = TRUE) il SessionId nel SOAP header è ora valorizzato con l'identificativo che il sistema GME ha assegnato a quella sessione utente. Nelle successive invocazioni di metodi del GMEWS (Upload, Download, Logout) l'applicazione client deve indicare questo valore del SessionId.

- Se la risposta e' negativa viene inviato un messaggio di tipo SOAP Fault, avente ad esempio la seguente struttura:

```
<?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>
```

Nota

Sul sistema GME una sessione scade superato un tempo di inattività.
Il GMEWS respinge invocazioni che fanno riferimento ad una sessione scaduta con messaggi di tipo SOAP Fault (vv. seguito).

4.2 DownloadMessage

Il metodo DownloadMessage viene invocato passando un parametro in input (**MPNumber)** e due parametri vuoti di uscita (**MessageName, MessageContent)**:

DownloadMessage(MPNumber, MessageName, MessageContent)

Di seguito è riportato un esempio messaggio SOAP che contiene l'invocazione del metodo DownloadMEssage

```
<?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>
```

Si evidenzia quanto segue:

- Il SessionId nel SOAP header deve indicare il valore restituito dal metodo Login
- **MPNumber** riporta l'identificatore alfanumerico (nel sistema GME) del Market Partecipant al quale sono indirizzate le notifiche che si vogliono scaricare.

4.2.1 DownloadMessage response

A seguito della invocazione del metodo DownloadMessage il GME WS restituisce una DownloadMessage response.
Di seguito è riportato un esempio messaggio SOAP che contiene una 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>
```

Si evidenzia quanto segue:

- Se la risposta contiene un messaggio con un tag Result valorizzato a 'true', il tag **MessageName** riporta il nome del messaggio di notifica e in **MessageContent** il contenuto XML (HTML encoded) del messaggio. L'esempio si riferisce ad un messaggio di Functional Acknowledgement. In generale può essere anche un messaggio di bid notification o unit schedale tra quelli indicati "XML Implementation Guide for Market Participant" (cfr. [2])
- Se la risposta contiene un messaggio con un tag Result valorizzato a 'false' allora non ci sono messaggi da scaricare e la risposta non contiene alcun messaggio

Se l'utente non e' autorizzato a scaricare i messaggi per il Market Participant indicato, oppure la sessione e' scaduta o per qualche altro motivo, al posto del messaggio precedente viene inviato un messaggio di tipo SOAP Fault, avente ad esempio la seguente struttura (il messaggio e' relativo ad una sessione scaduta):

```
<?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

Il metodo UploadMessage viene invocato passando tre parametri in input (**MPNumber, MessageName, MessageContent)**:

UploadMessage(MPNumber, MessageName, MessageContent)

Di seguito è riportato un esempio messaggio SOAP che contiene l'invocazione del metodo UploadMessage

```
<?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
```

Si evidenzia quanto segue:

- Il SessionId nel SOAP header deve indicare il valore restituito dal metodo Login
- **MPNumber** riporta riporta l'identificatore alfanumerico (nel sistema GME) del Market Partecipant al quale si riferiscono i dati contenuti nel documento XML che viene trasmesso
- **Message Content** è il documento XML firmato, in formato ***PKCS#7***[4] (indicato sopra come "**…..PKCS#7 Signed Document …..**") che viene trasmesso (sottomissione offerte, sottomissione Margini);
  Il messaggio PKCS #7 deve essere codificato PKCS 7 ASN e deve contenere al suo interno:
  - il documento XML da trasmettere a GME, che deve essere conforme ad un messaggio di sottomissione offerte/sottomissione margini, come indicato in "XML Implementation Guide for Market Participant" (cfr. [2])
  - la firma del documento
  - il certificato di firma digitale del firmatario, in codifica X.509 ASN

  Non vanno aggiunti altri certificati, liste di revoche o attributi qualificati.
  La firma deve essere prodotta con gli algoritmi:
  - sha-1 (hashing)[5] + rsa (encryption)

  L' Appendice F fornisce alcune indicazioni su come ottenere un messaggio firmato in formato PKCS #7 tramite l'uso di Microsoft CryptoAPI
- **Message Name** e' il nome che l'utente assegna al messaggio che viene trasmesso

Il metodo effettua l'upload del documento XML sul sistema GME, a nome del market participant indicato nell'**MPNumber**.

[4] Codificato MIME Base 64

[5] Attualmente il sistema GME supporta anche firma con hashing md5, ma a breve questo algoritmo di digest non sara' più accettato e documenti firmati con esso verranno rifiutati. Il suo uso è quindi sconsigliato.

4.3.1 UploadMessage response

A seguito della invocazione del metodo UploadMessage il GME WS restituisce una UploadMessage response.
Di seguito è riportato un esempio messaggio SOAP che contiene una UploadMessage response nel caso l'operazione di upload sia stata completata con successo

```
<?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>
```

Sostanzialmente in caso di successo nel tag Result viene restituitoun XML (con encoding HTML) con il seguente formato:

```
 <?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>
```

Si evidenzia quanto segue:

- Il campo **<REQUEST_STATUS>** valorizzato COMPLETED indica che l'operazione di upload si è completata con successo
- **<MESSAGE_NAME>** riporta il nome assegnato dall'utente al messaggio
- **<MESSAGE_ID>** iporta l'identificativo assegnato dal sistema GME al messaggio (il quale consente il reperimento del messaggio utilizzando il portale GME, menu Transazioni XML/Messaggi)
- **<DATE>** e **<TIME>** indicano data e ora (timestamp) di ricezione del messaggio (in base ai quali il sistema GME assegna le priorita')

Se l'utente non e' autorizzato a scaricare i messaggi per il Market Participant indicato, oppure la sessione e' scaduta o per qualche altro motivo, al posto del messaggio precedente viene inviato un messaggio di tipo SOAP Fault, avente ad esempio la seguente struttura (il messaggio e' relativo ad una sessione scaduta):

```
<?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

Il metodo GetNextMessage viene invocato passando due parametri in input (MPNumber e MaxNumberOfMessages) e due parametri vuoti di uscita (NumberOfMessages, MessageList):

GetNextMessage(MPNumber, MaxNumberOfMessages, NumberOfMessages, MessageList)

Di seguito è riportato un esempio messaggio SOAP che contiene l'invocazione del metodo GetNextMessage

```
<?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>
```

Si evidenzia quanto segue:

- Il SessionId nel SOAP header deve indicare il valore restituito dal metodo Login
- **MPNumber** riporta l'identificatore alfanumerico (nel sistema GME) del Market Partecipant per il quale si vuole sapere quali notifiche sono presenti
- **MaxNumberOfMessages** è il numero massimo di messaggi che si vuole ottenere nella risposta

4.4.1 GetNextMessage response

A seguito della invocazione del metodo GetNextMessage il GME WS restituisce una GetNextMessage response.
Di seguito è riportato un esempio di messaggio SOAP che contiene una 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>
```

Si evidenzia quanto segue:

- Il tag **<MessageList>** contiene un elenco di messaggi <Message> di ciascuno dei quali riporta l'identificativo della notifica (tag MessageId) e il nome assegnato alla notifica (tag MessageName) assegnati dal sistema GME;
- Se non ci sono messaggi da scaricare (lista vuota) viene ritornato un messaggio nel seguente formato:

```
<?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>
```

Se l'utente non e' autorizzato a scaricare i messaggi per il Market Participant indicato, oppure la sessione e' scaduta o per altro motivo, al posto del messaggio precedente viene inviato un messaggio di tipo SOAP Fault, avente ad esempio la seguente struttura (il messaggio e' relativo ad una sessione scaduta):

-

```
<?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

Il metodo ForceDownloadMessage viene invocato passando due parametri in input (MPNumber e MessageId) e due parametri vuoti (MessageName, MessageContent):

ForceDownloadMessage(MPNumber, MessageId, MessageName, MessageContent)

Il metodo consente di scaricare uno specifico messaggio di notifica.

Di seguito è riportato un esempio messaggio SOAP che contiene l'invocazione del metodo ForceDownloadMessage:

```
<?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>
```

Si evidenzia quanto segue:

- Il SessionId nel SOAP header deve indicare il valore restituito dal metodo Login
- **MPNumber** riporta l'identificatore alfanumerico (nel sistema GME) del Market Partecipant di cui si vuole scaricare la notifica
- **MessageId** è l'identificativo assegnato dal sistema GME al messaggio di notifica che si vuole scaricare

4.5.1 ForceDownloadMessage response

A seguito della invocazione del metodo ForceDownloadMessage il GMEWS restituisce una ForceDownloadMessage response.
Di seguito è riportato un esempio di messaggio SOAP che contiene una 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> …..vedi analogo contenuto in DownloadMessage…..
              </MessageContent>
              </ForceDownloadMessageResponse>
         </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Si evidenzia quanto segue:

- Se il messaggio richiesto esiste (**Result** = TRUE), il tag **MessageName** riporta il nome del messaggio di notifica e in **MessageContent** il contenuto del messaggio

In questa versione del GMEWS (1.0.4.3) se il messaggio non esiste al posto del messaggio precedente viene inviato un messaggio di tipo SOAP Fault, avente ad esempio la seguente struttura:

```
<?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>
```

Se l'utente non e' autorizzato a scaricare i messaggi per il Market Participant indicato, oppure la sessione e' scaduta o per qualche altro motivo, al posto del messaggio precedente viene inviato un messaggio di tipo SOAP Fault, avente ad esempio la seguente struttura (il messaggio e' relativo ad una sessione scaduta):

```
<?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

Il metodo Logout viene invocato senza passare alcun parametro:

Logout()

Il metodo termina la sessione di lavoro.

Di seguito è riportato un esempio messaggio SOAP che contiene l'invocazione del metodo Logout

```
<?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>
```

Si evidenzia quanto segue:

- Il SessionId nel SOAP header deve indicare il valore restituito dal metodo Login, che individua la sessione che verrà terminata

4.6.1 Logout response

A seguito della invocazione del metodo Logout il GME WS restituisce una Logout response che conferma la chiusura della sessione.
Di seguito è riportato un esempio di messaggio SOAP che contiene un 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>
```

Se la sessione e' scaduta o per qualche altro motivo, al posto del messaggio precedente viene inviato un messaggio di tipo SOAP Fault, avente ad esempio la seguente struttura (il messaggio e' relativo ad una sessione scaduta):

```
<?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>
```

Appendice A : WSDL di GMEWS

Di seguito viene descritto il WSDL del Web Service, ottenibile dagli URL:

http://www.ipex.it/gmews/gmews.wsdl
https://www.ipex.it/gmews/gmews.wsdl (richiede autenticazione con certificato SSL client)

################################ 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:wdlns='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>
```

Appendice B : WSDL di GMEWS compatibile .NET Framework 1.1

Di seguito è riportato il WSDL del Web Service compatibile .NET Framework 1.1, ottenibile dagli URL:

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

Date le peculiarità del framework .NET si è preferito produrre uno specifico WSDL compatibile con esso.

Questo WSDL indirizza aspetti di compatibilità non supportati dal WSDL indicato in Appendice A, compatibile con SOAP Toolkit. In particolare, questo WSDL utilizza tipi complessi nel SOAP header e si è sperimentata con esso la possibilità di generazione automatica di codice .NET per un applicativo client del Web Service.

########## 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>
```

Appendice C : Suggerimenti sull'uso dei certificati digitali in ambiente Windows

1) L'applicazione client avrà necessità di due certificati, uno di autenticazione e uno di firma. In ambiente Windows con installato Internet Explorer 6.0.28 e superiori non e' possibile autenticarsi utilizzando un certificato di firma.
   Assicurarsi di selezionare un certificato di autenticazione per stabilire la connessione https

2) In ambiente Windows i certificati devono essere importati nei key store di Windows. L'importazione di un certificato presente sul dispositivo hardware (SmartCard o token) Windows si esegue usando i tool software forniti insieme al dispositivo. Come default solitamente questi tool importano i certificati nello store di CURRENT_USER. Se l'applicazione client è multiutente e si vuole un accesso ai certificati indipendente dal tipo di utente collegato e' necessario spostare fisicamente i certificati sullo store di LOCAL_MACHINE

3) Affinché un'applicazione client in ambiente Windows possa usare con successo i certificati utente di autenticazione e di firma, questi devono essere trusted sulla macchina client.
   Un certificato non trusted sul client produce errori (croce a sfondo rosso) oppure warnings (punti di domanda gialli) perche' Windows cerca sempre di recuperare la catena di trust completa, e questo può causare malfunzionamenti nella applicazione client.
   Oltre ai ceritificati utente, è necessario quindi importare negli store di CURRENT_USER (o di LOCAL_MACHINE) anche il certificato della CA che ha emesso quei certificati. Se questa CA è una CA subordinata, occorre caricare tutta la catena dei certificati di CA che conduce da quella fino ad una CA root.
   I certificati di CA root devono essere importati nello store: Trusted Root Certification Authorities
   I certificati di CA subordinate devono essere importati nello store: Intermediate Certification Authorities
   I certificati utente vanno invece importati nel Personal store

4) Sono stati riscontrati problemi nello scaricamento in HTTPS del WSDL utilizzando il SOAP Toolkit. È possibile scaricare il WSDL anche in http all'indirizzo indicato in Appendice A

5) Nel .NET Framework 1.1 non esistono classi native per l'accesso diretto a certificati caricati nel key store di Windows. Possibili alternative aggirare questa limitazione sono:

   A) accedere ai key store di Windows attraverso platform invoke (Pinvoke) delle corrispondenti primitive Microsoft CryptoAPI (che risiedono sulla DLL crypt32.dll)

   B) utilizzare il componente CAPICOM di Microsoft che supporta l'accesso agli store dei certificati (la release 2.0.0.3 attualmente scaricabile dal sito Microsoft fornisce anche esempi in linguaggio C#)

   C) installare il pacchetto aggiuntivo Web Service Extension (WSE); tuttavia la versione 2.0 (compatibile con .NET Framework 1.1) non è ancora ufficiale e comunque non e' integrata nel Framework

   La prossima versione del Framework .NET denominata "Longhorn" (attualmente ancora in versione alpha, si veda link:http://longhorn.msdn.microsoft.com/lhsdk/ndp/ovrwhatsnewinwhidbeyalpha.aspx), dovrebbe integrare nel Framework .NET anche alcune funzionalita' di CAPICOM per l'accesso agli store dei certificati

Appendice D : esempio di invocazione del metodo Login con SOAP Toolkit 3.0

Si fornisce uno stralcio di codice in linguaggio Visual Basic 6.0 per esemplificare come si può invocare il metodo Login usando SOAP Toolkit 3.0:

Classe ClientHeaderHandler (utilizzata per il management dell'header SOAP nel 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 ###################################

Implementazione metodo Login:

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) del Certificato utilizzato

Il metodo:

SetProxy oSoapClient

puo' essere usato per settare eventualmente le caratteristiche del proxy nel caso si usi un proxy server per la connessione a Internet (Property dell'oggetto HttpConnector30 con nomi "ProxyServer", "ProxyUser", "ProxyPassword").

########## 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

'
' contestualmente all'esecuzione di questo metodo, come step intermedio dell'SSL Handshaking, se tutto e' stato installato/configurato
'  correttamente viene  mostrata una dialog dove  l'utente puo' inserire il PIN necessario per accedere alla chiave privata del  certificato
'  residente sul token USB
'
bRes = oSoapClient.Login(LoginId, Password)

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

########## Fine metodo Login ##########

Appendice E : esempio di invocazione dei metodi Login/Download con .NET Framework 1.1

Viene fornito un esempio di codice in VB.NET per l'invocazione dei metodi Login/Download utilizzando il .NET Framework 1.1.
L'esempio presuppone che la parte pubblica del certificato sia esportata su file usando la funzionalità di export del certificato su file a partire dal certificato visualizzato sullo snap-in apposito di Microsoft Management Console.

Per indicazioni su come accedere dal .NET Framework ai certificati digitali caricati sul Windows key store si rimanda a Appendice C – punto 5.

La classe proxy **WebReference** viene generata automaticamente in Visual Studio .NET 2003 con la funzionalita' "Add Web Reference" specificando il WSDL **gmews_NET.wsdl** definito nell'Appendice 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

      '
      ' Setting del proxy utilizzato per la connessione
      '
      SetProxy  "proxyuser", "proxypassword", "proxydomain"

      '
      '  Caricamento del certificato X509: la parte pubblica del certificato di autenticazione e' stata esportata su file, usando
      '  l'export del certificato su file a partire dal certificato visualizzato sullo snap-in apposito di Microsoft Management Console
      '
      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 = ""

' contestualmente all'esecuzione di questo metodo, come step intermedio dell'SSL Handshaking, se tutto e' stato installato/configurato
'  correttamente viene  mostrata una dialog dove  l'utente puo' inserire il PIN necessario per accedere alla chiave privata del  certificato
'  residente sul token USB

         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
        '
        ' Setting del proxy utilizzato per la connessione
        '
        SetProxy  “proxyuser”, “proxypassword”, “proxydomain”
        ‘
        ‘  Caricamento del certificato X509: la parte pubblica del certificato di autenticazione e’ stata esportata su file, usando
        ‘  l’export del certificato su file a partire dal certificato visualizzato sullo snap-in apposito di Microsoft Management Console
        ‘
        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
```

############################## Fine ClientWS.vb ##

Appendice F : esempi di produzione di un messaggio firmato in formato PKCS#7

Di seguito viene fornito uno stralcio di codice C++ che descrive come si può produrre un PKCS #7 signed message utilizzando la libreria Microsoft CryptoAPI:

```
//
// Occorre inizialmente acquisire il contesto del certificato di firma e assegnarlo a pCertContext
//
pCertContext : context del certificato di firma selezionato
//
// Definizione Algoritmo di hashing:  SHA-1
//
#define HASH_ALGORITHM              szOID_OIWSEC_sha1
// #define HASH_ALGORITHM             szOID_RSA_MD5
//
// definizione encoding PKCS#7 per il messaggio firmato e X.509 per il certificato
//
#define MY_ENCODING_TYPE  (PKCS_7_ASN_ENCODING | X509_ASN_ENCODING)
```

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

```
//   occorre costruire la struttura CRYPT_SIGN_MESSAGE_PARA che contiene i parametri utilizzati
//   per la firma dei messaggi incluso il context del certificato di firma
   memset(&SignMessagePara, 0, sizeof(CRYPT_SIGN_MESSAGE_PARA));
   SignMessagePara.cbSize = sizeof(CRYPT_SIGN_MESSAGE_PARA);
   SignMessagePara.HashAlgorithm.pszObjId = HASH_ALGORITHM;  //imposta algoritmo di hashing (digest algorithm)
   SignMessagePara.pSigningCert = pCertContext;              //imposta il certificato da usare per produrre la firma
   SignMessagePara.dwMsgEncodingType = MY_ENCODING_TYPE;     //imposta la codifica del PKCS#7 message e del certificato
   SignMessagePara.cMsgCert = 1;                             //inserisce solo un certificato all'interno del messaggio PKCS #7
   SignMessagePara.rgpMsgCert = &pCertContext;               //seleziona il certificato da inserire nel messaggio PKCS #7
                                                              //NOTA: l'algoritmo di hash encryption (RSA) per produrre la
                                                              firma è implicitamente dedotto dall'algoritmo indicato nel
                                                              certificato per la chiave pubblica
   rgcbToBeSigned[0] = InDataLen;
   rgpbToBeSigned[0] = (BYTE *)InData;
   // Ottiene la dimensione massima del buffer contenente il documento finale con la firma
   //
   rc=CryptSignMessage(&SignMessagePara,
                  FALSE,          // undetached signature
                  1,
                  rgpbToBeSigned,
                  rgcbToBeSigned,
                  NULL,
                  &OutDataLenSign);
   //
   // effettua la firma del documento
   //
   *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)

A seguire dovra' essere effettuato l'encoding MIME BASE64 del buffer (*OutData).

Indicazioni su come produrre un PKCS#7 signed message nel Framework .NET 1.1

Il Framework .NET versione 1.1 non supporta in modo diretto la produzione di messaggi firmati in formato PKCS#7: supporta l'hashing SHA1 ma non l'encryption dell'hashing, necessario per ottenere la firma del contenuto del messaggio PKCS#7.
La prossima versione del Framework .NET, denominata "Longhorn", (ancora in fase alpha si veda il link: http://longhorn.msdn.microsoft.com/lhsdk/ndp/ovrwhatsnewinwhidbeyalpha.aspx) dovrebbe supportare la firma di documenti in formato PKCS#7.

Possibili alternative per aggirare questa limitazione e produrre un messaggio firmato PKCS#7 usando il framework .NET 1.1. sono:

A) ricorrere a platform invoke (Pinvoke) delle corrispondenti funzioni delle CryptoAPI Microsoft (che risiedono sulla DLL crypt32.dll)

B) utilizzare (nell'ambito dell'ambiente .NET Framework 1.1, ma anche da VB6) il componente CAPICOM di Microsoft per effettuare la firma digitale (questo componente per default firma in formato PKCS#7 con hashing SHA1).

Nota

Il componente CAPICOM converte internamente il testo da firmare secondo la codifica Unicode (2 byte per carattere). Il sistema GME si attende invece che il documento XML ricevuto (per il quale e' previsto un encoding "iso-8859-1, ovvero iso Latin-1 – cfr. [2]) sia trasmesso con codifica ANSI (1 byte per carattere).
Di seguito vengono suggeriti due workaround (per VB6 e per C# del Framework .NET 1.1) per aggirare questo problema

B1) In VB6 si può forzare una conversione da Unicode a ByteString sul contenuto da firmare, invocando una routine come quella sotto descritta :

```
'******************************** Esempio VB6  **********************************

Content: contenuto del documento da firmare
SignedData: oggetto CAPICOM.SignedData
Signer: oggetto CAPICOM.Signer

'*****************************************************************************
' 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)
….

'******************************** Fine Esempio VB6  **********************************
```

B2) In .NET 1.1. si può sostituire il tipo "string" con "native int" e ridefinire il tipo della property Content dal tipo string a native int scavalcando così la conversione automatica delle stringhe in Unicode. Di seguito si descrive come:

Nel Framework .NET il tipo dato stringa (string) e' a tutti gli effetti una stringa con caratteri Unicode.
Si puo' importare il CAPICOM nel Framework .NET 1.1 con i seguenti passi:

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

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

3) rimpiazzare "string " con "native int" nelle seguenti righe del file capicom.il:

```
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) Rimpiazzare "string marshal(bstr)" con "native int" nelle seguenti righe del file capicom.il:

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

Avendo così ridefinito il tipo della property Content dal tipo string dell'ambiente .NET a native int e' possibile scavalcare la conversione automatica delle stringhe in Unicode.
Per fare la firma digitale, si deve quindi codificare in C# una porzione di codice "unsafe", come esemplificato qui sotto:

```
'******************************** Esempio C#  ****************************************************

byteArrayContent: byte array del documento da firmare
signedData: oggetto CAPICOM.SignedDataClass importato nel framework .NET (tlbimp.exe)
signer: oggetto CAPICOM.SignerClass importato nel 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);

'******************************** Fine Esempio C#  ****************************************************
```