Urzędowa Poczta Elektroniczna (UPE)

Dokument zawiera opis sposobu implementacji wysyłania wniosków wraz z załączonymi aktami prawnymi do publikacji w Dzienniku Urzędowym poprzez dedykowaną Urzędową Pocztę Elektroniczną

Data dokumentu: 27 września 2012
Wersja: 1.1
Autor: Adam Rakowski
Konsultacja techniczna: Krzysztof Radzimski
Konsultacja merytoryczna: Piotr Jegorow

Zawartość

Informacje podstawowe

Lokalizacja usługi sieciowej (WebService):
http://upe.com.pl/ws/upe.asmx
WSDL:
http://upe.com.pl/ws/upe.asmx?wsdl

W celu przekazania aktów normatywnych lub innych aktów prawnych do ogłoszenia w Wojewódzkim Dzienniku Urzędowym za pomocą Urzędowej Poczty Elektronicznej (zwanej dalej **UPE**) należy wykonać następujące kroki:

Krok 1
Przygotowanie wniosku

1. Stworzyć plik XML o nazwie **Wniosek.xml**, który będzie zawierał podstawowe informacje niezbędne do utworzenia właściwego wniosku o ogłoszenie w Wojewódzkim Dzienniku Urzędowym. Plik XML musi zawierać następujące dane:
   1) Symbol sprawy
   2) Data sprawy
   3) Imię składającego wniosek
   4) Nazwisko składającego wniosek
   5) Email składającego wniosek
   6) Stanowisko składającego wniosek
   7) Lista aktów prawnych dołączonych do wniosku

   Przykładowy plik:

```
<?xml version="1.0" encoding="utf-8"?>
<wniosek wersja="1.0">
   <data>2012-09-03</data>
   <symbol>XCV/2012/11/23</symbol>
   <imie>XCV/2012/11/23</imie>
   <nazwisko>Kowalski</nazwisko>
   <email>jan@kowalski.pl</email>
   <stanowisko>Informatyk</stanowisko>
   <zalaczniki>
         <zalacznik nazwa="Uchwala.XVI.80.2012.zip" />
         <zalacznik nazwa="Uchwala.XII.221.2012.zip" />
   </zalaczniki>
</wniosek>
```

   Uwagi
   1) Nazwa załączonego do wniosku pliku aktu prawnego **nie może** zawierać polskich liter, spacji, ani znaków specjalnych
   2) Załączone pliki aktów muszą być zgodne z rozporządzeniem Prezesa Rady Ministrów z dnia 27 grudnia 2011 r. w sprawie wymagań dla dokumentów

elektronicznych zawierających akty normatywne i inne akty prawne, dzienników urzędowych wydawanych w postaci elektronicznej oraz środków komunikacji elektronicznej i informatycznych nośników danych (Dz. U. Nr 289, poz. 1699)

2. Spakować plik wniosek.xml wraz z załączonymi plikami zawierającymi akty prawne w archiwum ZIP.

Krok 2
Logowanie do UPE.

W celu zalogowania do UPE należy wywołać metodę, w której podajemy nazwę użytkownika oraz skrót hasła (algorytm SHA256).

```
private String ComputeHashPassword(String password){
        return BitConverter.ToString(
            new SHA256CryptoServiceProvider().ComputeHash(
                  Encoding.UTF8.GetBytes(password))).Replace("-", "");
}
…

SPSession session = upe.StartNewSession(userLogin,
ComputeHashPassword(userPassword));
```

Przykład:
Dla „qwe” ‘hash’ będzie wyglądał następująco:
`489cd5dbc708c7e541de4d7cd91ce6d0f1613573b7fc5b40d3942ccb9555cf35`

Identyczny rezultat można osiągnąć korzystając z języka JAVA oraz bibliotek Apache Commons Codec (http://commons.apache.org/codec/).

```
import org.apache.commons.codec.digest.DigestUtils;
…
DigestUtils.sha256Hex(userPassword);
```

Jeżeli użytkownik zostanie poprawnie uwierzytelniony, metoda zwróci obiekt *SPSession*, który jest wymagany w kolejnych krokach. W przypadku nieprawidłowego uwierzytelnienia zostanie zwrócony wyjątek, informujący o błędzie.

Krok 3
Wysyłanie wniosku

1. Przed rozpoczęciem wysyłania archiwum ZIP, należy pobrać identyfikator wniosku oraz zainicjalizować transfer:

```
int issueID = upe.CreateEmptyIssue(session);
PackageInfo packageInfo = upe.SetupPackageTransfer(session,
```

```
                    issueID, packageFile.Length);
```

PackageInfo zawiera informację, o tym, na ile paczek należy podzielić archiwum ZIP oraz ile zajmuje każda paczka.

2. Po zainicjalizowaniu należy wysyłać archiwum ZIP podzielone na pliki wielkości 512KB (524288):

```
// inicjalizacja transferu: wysyłamy częściami po 512KB (524288) każda
PackageInfo packageInfo = upe.SetupPackageTransfer(session,
                                    issueID, packageFile.Length);

// wyslanie paczek
foreach (PackagePart part in packageInfo.Parts){
          upe.SendPackagePart(session, packageInfo, part.PartId, filePart);
}
```

3. Po wysłaniu ostatniej część wysyłamy informację do serwera o zakończeniu wysyłania:

```
// inicjalizacja transferu: wysyłamy częściami po 512KB (524288) każda
PackageInfo packageInfo = upe.SetupPackageTransfer(session, issueID,
                                    packageFile.Length);

// wyslanie paczek
foreach (PackagePart part in packageInfo.Parts){
          upe.SendPackagePart(session, packageInfo, part.PartId, filePart);
}

// potwierdzanie wysłania paczki
upe.ConfirmTransfer(session, packageInfo);
```

Krok 4
Pobieranie przygotowanego wniosku do podpisu

Usługa sieciowa UPE, przekonwertuje załączone akty prawne w formacie ZIP na format wymagany przez UPE (ZIPX) oraz wygeneruje właściwy wniosek z pełnymi danymi, tj. danymi właściwego Urzędu Wojewódzkiego, do którego wysyłany jest wniosek oraz danymi instytucji wysyłającej. Tak przygotowaną paczkę zawierającą powyższe dokumenty należy ponownie pobrać, podpisać bezpiecznym podpisem elektronicznym i wysłać do UPE. Powyższa procedura składa się z następujących kroków:

1. Inicjalizacja pobierania archiwum ZIP:

```
PackageInfo downloadPackage = upe.SetupDownloadPackageTransfer(session,
issueID, packageInfo.PackageId);
```

2. *PackageInfo* zawiera wszystkie niezbędne informacje do poprawnego pobrania archiwum ZIP postaci pojedynczych paczek z serwera:

```
PackageInfo downloadPackage = upe.SetupDownloadPackageTransfer(session,
issueID, packageInfo.PackageId);

// utworzenie bufora
byte[] buffer = new byte[downloadPackage.TotalSize];
```

```
foreach (PackagePart part in downloadPackage.Parts){
        // pobranie części pliku
        byte[] data = upe.DownloadPackagePart(session,
              downloadPackage, part.PartId);
        // przekopiowanie danych do bufora
        Array.Copy(data, 0, buffer, (part.PartId - 1) *
              downloadPackage.PartSize, data.Length);
}

// zapisanie pliku
File.WriteAllBytes(packagePath, buffer);
```

3. Po zakończeniu pobierania archiwum ZIP, nie ma potrzeby przesyłania informacji do usługi o zakończeniu pobierania.

Krok 5
Podpisanie wniosku

1. Rozpakowujemy pobrane archiwum ZIP.
2. Podpisujemy plik **wniosek.xml** wraz za łączonymi aktami prawnymi w formacie ZIPX, dołączając do sygnatury podpisu stosowne referencje. Format podpisu musi być zgodny z specyfikacją techniczną formatu podpisu określoną w załączniku nr 2 do rozporządzenia Prezesa Rady Ministrów z dnia 27 grudnia 2011 r. *w sprawie wymagań dla dokumentów elektronicznych zawierających akty normatywne i inne akty prawne, dzienników urzędowych wydawanych w postaci elektronicznej oraz środków komunikacji elektronicznej i informatycznych nośników danych* (Dz. U. Nr 289, poz. 1699)
3. Po złożeniu bezpiecznego podpisu elektronicznego pliku wniosku o ogłoszenie oraz załączone akty prawne w formacie ZIPX pakujemy do archiwum ZIP.

Krok 6
Wysłanie podpisanego wniosku

Powtarzamy operacje opisane w kroku 3, wskazując plik ZIP zawierający właściwy plik wniosku ogłoszenie podpisany bezpiecznym podpisem elektronicznym wraz z aktami prawnymi w formacie ZIPX.

1. Przed rozpoczęciem wysyłania archiwum ZIP, należy zainicjalizować transfer:

```
PackageInfo packageInfo = upe.SetupPackageTransfer(session,
                                    issueID, packageFile.Length);
```

PackageInfo zawiera informację, o tym, na ile paczek należy podzielić archiwum ZIP oraz ile zajmuje każda paczka.

2. Po zainicjalizowaniu należy wysyłać archiwum ZIP podzielone na pliki wielkości 512KB (524288):

```
// inicjalizacja transferu: wysyłamy częściami po 512KB (524288) każda
PackageInfo packageInfo = upe.SetupPackageTransfer(session,
                                    issueID, packageFile.Length);
```

```
// wyslanie paczek
foreach (PackagePart part in packageInfo.Parts){
         upe.SendPackagePart(session, packageInfo, part.PartId, filePart);
}
```

3. Po wysłaniu ostatniej część wysyłamy informację do serwera o zakończeniu wysyłania:

```
// inicjalizacja transferu: wysyłamy częściami po 512KB (524288) każda
PackageInfo packageInfo = upe.SetupPackageTransfer(session, issueID,
                                packageFile.Length);

// wyslanie paczek
foreach (PackagePart part in packageInfo.Parts){
         upe.SendPackagePart(session, packageInfo, part.PartId, filePart);
}

// potwierdzanie wysłania paczki
upe.ConfirmTransfer(session, packageInfo);
```

Podsumowanie

Po wykonaniu kroku 6, UPE utworzy właściwy plik paczki archiwalnej zawierający przekazany wniosek o ogłoszenie wraz z aktami przeznaczonymi do ogłoszenia oraz metadane właściwe dla dokumentów elektronicznych przekazywanych do archiwum państwowego. Tak wytworzona paczka zostanie przekazana do właściwej Redakcji Dziennika Urzędowego. Na wskazany w ustawieniach UPE adres e-mail użytkownika zostanie przekazane Urzędowe Poświadczenie Przedłożenia. Ponadto wszystkie dokumenty będą dostępne z poziomu portalu internetowego UPE pod adresem https://upe.com.pl.

Pobieranie Urzędowego Poświadczenia Przedłożenia

Usługa sieciowa umożliwia sprawdzenie i pobranie UPP, które informuje o tym, że Redakcja Dziennika Urzędowego, otrzymała wniosek o ogłoszenie.

W celu pobrania UPP należy wywołać metodę:

```
PackageInfo packageInfo = upe.SetupDownloadUpp(session, issueID);
```

Jeżeli Redakcja Dziennika Urzędowego nie odebrała wniosku wraz z załącznikami to wartość `packageInfo` będzie pusta (`null`). W przeciwnym wypadku instancja klasy `PackageInfo` będzie zawierać wszystkie niezbędne informacje potrzebne do pobrania paczki z Urzędowym Poświadczeniem Przedłożenia.

Pobrać paczkę z UPP można w następujący sposób:

```
byte[] buffer = new byte[packageInfo.TotalSize];

foreach (PackagePart part in packageInfo.Parts) {
      byte[] data = upe.DownloadUppPackagePart(
            session, packageInfo, part.PartId);
      Array.Copy(data, 0, buffer, (part.PartId - 1) * packageInfo.PartSize,
             data.Length);
}
```

Pobrany plik to archiwum ZIP zawierające:

- podpisany plik XML UPP
- podpisany plik PDF UPP

WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://upe.com.pl/ws/upe"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" targetNamespace="http://upe.com.pl/ws/upe"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://upe.com.pl/ws/upe">
      <s:element name="StartNewSession">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="login" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="password" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="StartNewSessionResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="StartNewSessionResult" type="tns:SPSession"
/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="SPSession">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="DatabaseId" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="IsActive" type="s:boolean" />
          <s:element minOccurs="1" maxOccurs="1" name="UserId" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="InstitutionId" type="s:int" />
          <s:element minOccurs="0" maxOccurs="1" name="SessionId" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" name="SpItemId" type="s:int" />
        </s:sequence>
      </s:complexType>
      <s:element name="EndSession">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="session" type="tns:SPSession" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="EndSessionResponse">
        <s:complexType />
      </s:element>
      <s:element name="SetupPackageTransfer">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sessionHandle" type="tns:SPSession" />
            <s:element minOccurs="1" maxOccurs="1" name="issueId" type="s:int" />
            <s:element minOccurs="1" maxOccurs="1" name="packageSize" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="SetupPackageTransferResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SetupPackageTransferResult"
type="tns:PackageInfo" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="PackageInfo">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="InstitutionId" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="UserId" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="IssueId" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="TransferMode" type="tns:PackageTransferMode"
/>
          <s:element minOccurs="0" maxOccurs="1" name="PackageId" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" name="PartSize" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="TotalSize" type="s:int" />
```

```
          <s:element minOccurs="0" maxOccurs="1" name="Parts" type="tns:ArrayOfPackagePart" />
        </s:sequence>
      </s:complexType>
      <s:simpleType name="PackageTransferMode">
        <s:restriction base="s:string">
          <s:enumeration value="Download" />
          <s:enumeration value="Upload" />
        </s:restriction>
      </s:simpleType>
      <s:complexType name="ArrayOfPackagePart">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="unbounded" name="PackagePart" nillable="true"
type="tns:PackagePart" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="PackagePart">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="PartId" type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="Size" type="s:int" />
        </s:sequence>
      </s:complexType>
      <s:element name="SetupDownloadPackageTransfer">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sessionHandle" type="tns:SPSession" />
            <s:element minOccurs="1" maxOccurs="1" name="issueID" type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="packageID" type="s:string" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="SetupDownloadPackageTransferResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="SetupDownloadPackageTransferResult"
type="tns:PackageInfo" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="DownloadPackagePart">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sessionHandle" type="tns:SPSession" />
            <s:element minOccurs="0" maxOccurs="1" name="info" type="tns:PackageInfo" />
            <s:element minOccurs="1" maxOccurs="1" name="partId" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="DownloadPackagePartResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="DownloadPackagePartResult"
type="s:base64Binary" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="SendPackagePart">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sessionHandle" type="tns:SPSession" />
            <s:element minOccurs="0" maxOccurs="1" name="info" type="tns:PackageInfo" />
            <s:element minOccurs="1" maxOccurs="1" name="partId" type="s:int" />
            <s:element minOccurs="0" maxOccurs="1" name="content" type="s:base64Binary" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="SendPackagePartResponse">
        <s:complexType />
      </s:element>
      <s:element name="ConfirmTransfer">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sessionHandle" type="tns:SPSession" />
            <s:element minOccurs="0" maxOccurs="1" name="package" type="tns:PackageInfo" />
          </s:sequence>
        </s:complexType>
      </s:element>
```

```
      <s:element name="ConfirmTransferResponse">
        <s:complexType />
      </s:element>
      <s:element name="CreateEmptyIssue">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="sessionHandle" type="tns:SPSession" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="CreateEmptyIssueResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="CreateEmptyIssueResult" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="StartNewSessionSoapIn">
    <wsdl:part name="parameters" element="tns:StartNewSession" />
  </wsdl:message>
  <wsdl:message name="StartNewSessionSoapOut">
    <wsdl:part name="parameters" element="tns:StartNewSessionResponse" />
  </wsdl:message>
  <wsdl:message name="EndSessionSoapIn">
    <wsdl:part name="parameters" element="tns:EndSession" />
  </wsdl:message>
  <wsdl:message name="EndSessionSoapOut">
    <wsdl:part name="parameters" element="tns:EndSessionResponse" />
  </wsdl:message>
  <wsdl:message name="SetupPackageTransferSoapIn">
    <wsdl:part name="parameters" element="tns:SetupPackageTransfer" />
  </wsdl:message>
  <wsdl:message name="SetupPackageTransferSoapOut">
    <wsdl:part name="parameters" element="tns:SetupPackageTransferResponse" />
  </wsdl:message>
  <wsdl:message name="SetupDownloadPackageTransferSoapIn">
    <wsdl:part name="parameters" element="tns:SetupDownloadPackageTransfer" />
  </wsdl:message>
  <wsdl:message name="SetupDownloadPackageTransferSoapOut">
    <wsdl:part name="parameters" element="tns:SetupDownloadPackageTransferResponse" />
  </wsdl:message>
  <wsdl:message name="DownloadPackagePartSoapIn">
    <wsdl:part name="parameters" element="tns:DownloadPackagePart" />
  </wsdl:message>
  <wsdl:message name="DownloadPackagePartSoapOut">
    <wsdl:part name="parameters" element="tns:DownloadPackagePartResponse" />
  </wsdl:message>
  <wsdl:message name="SendPackagePartSoapIn">
    <wsdl:part name="parameters" element="tns:SendPackagePart" />
  </wsdl:message>
  <wsdl:message name="SendPackagePartSoapOut">
    <wsdl:part name="parameters" element="tns:SendPackagePartResponse" />
  </wsdl:message>
  <wsdl:message name="ConfirmTransferSoapIn">
    <wsdl:part name="parameters" element="tns:ConfirmTransfer" />
  </wsdl:message>
  <wsdl:message name="ConfirmTransferSoapOut">
    <wsdl:part name="parameters" element="tns:ConfirmTransferResponse" />
  </wsdl:message>
  <wsdl:message name="CreateEmptyIssueSoapIn">
    <wsdl:part name="parameters" element="tns:CreateEmptyIssue" />
  </wsdl:message>
  <wsdl:message name="CreateEmptyIssueSoapOut">
    <wsdl:part name="parameters" element="tns:CreateEmptyIssueResponse" />
  </wsdl:message>
  <wsdl:portType name="UPESoap">
    <wsdl:operation name="StartNewSession">
      <wsdl:input message="tns:StartNewSessionSoapIn" />
      <wsdl:output message="tns:StartNewSessionSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="EndSession">
      <wsdl:input message="tns:EndSessionSoapIn" />
      <wsdl:output message="tns:EndSessionSoapOut" />
    </wsdl:operation>
```

```
    <wsdl:operation name="SetupPackageTransfer">
      <wsdl:input message="tns:SetupPackageTransferSoapIn" />
      <wsdl:output message="tns:SetupPackageTransferSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="SetupDownloadPackageTransfer">
      <wsdl:input message="tns:SetupDownloadPackageTransferSoapIn" />
      <wsdl:output message="tns:SetupDownloadPackageTransferSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="DownloadPackagePart">
      <wsdl:input message="tns:DownloadPackagePartSoapIn" />
      <wsdl:output message="tns:DownloadPackagePartSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="SendPackagePart">
      <wsdl:input message="tns:SendPackagePartSoapIn" />
      <wsdl:output message="tns:SendPackagePartSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="ConfirmTransfer">
      <wsdl:input message="tns:ConfirmTransferSoapIn" />
      <wsdl:output message="tns:ConfirmTransferSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="CreateEmptyIssue">
      <wsdl:input message="tns:CreateEmptyIssueSoapIn" />
      <wsdl:output message="tns:CreateEmptyIssueSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="UPESoap" type="tns:UPESoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="StartNewSession">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/StartNewSession" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EndSession">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/EndSession" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetupPackageTransfer">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/SetupPackageTransfer" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetupDownloadPackageTransfer">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/SetupDownloadPackageTransfer"
style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DownloadPackagePart">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/DownloadPackagePart" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SendPackagePart">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/SendPackagePart" style="document" />
      <wsdl:input>
```

```
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ConfirmTransfer">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/ConfirmTransfer" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="CreateEmptyIssue">
      <soap:operation soapAction="http://upe.com.pl/ws/upe/CreateEmptyIssue" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="UPESoap12" type="tns:UPESoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="StartNewSession">
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/StartNewSession" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="EndSession">
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/EndSession" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetupPackageTransfer">
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/SetupPackageTransfer" style="document"
/>
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetupDownloadPackageTransfer">
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/SetupDownloadPackageTransfer"
style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DownloadPackagePart">
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/DownloadPackagePart" style="document"
/>
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SendPackagePart">
```

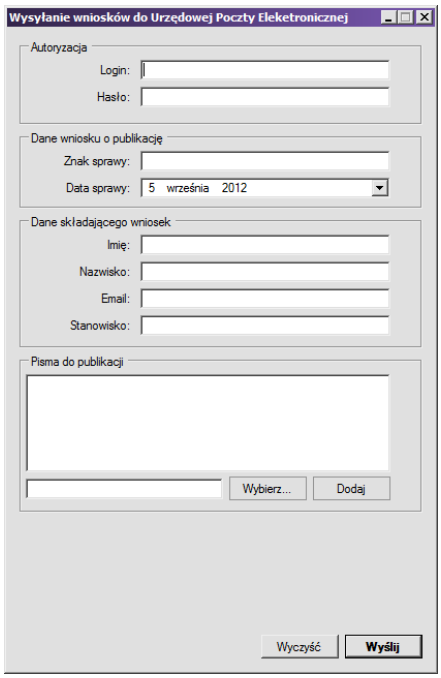
```
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/SendPackagePart" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ConfirmTransfer">
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/ConfirmTransfer" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="CreateEmptyIssue">
      <soap12:operation soapAction="http://upe.com.pl/ws/upe/CreateEmptyIssue" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="UPE">
    <wsdl:port name="UPESoap" binding="tns:UPESoap">
      <soap:address location="http://upe.com.pl/ws/upe.asmx" />
    </wsdl:port>
    <wsdl:port name="UPESoap12" binding="tns:UPESoap12">
      <soap12:address location="http://upe.com.pl/ws/upe.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Program przykładowy

Przykładowy program został przygotowany za pomocą środowiska Visual Studio 2012 oraz pakietu .NET Framework 4.5. Wykorzystano jedynie biblioteki dostępne w tym pakiecie. W przypadku korzystania ze starszej wersji .NET Framework, należy wykorzystać bibliotekę firmy trzeciej służącej do operacji na archiwach ZIP (Ionic.ZIP, ICSharpZipLib lub podobne). W przypadku korzystania z innych środowisk i języków programowania, należy wykorzystać właściwe narzędzia do komunikacji z usługa sieciową oraz wykonania wszystkich kroków.

Przykładowy program nie zawiera bibliotek odpowiedzialnych za składanie bezpiecznego podpisu elektronicznego.

Do utworzenia proxy wykorzystano kreator tworzenia referencji do usług WWW. Opisane powyżej kroki realizuje metoda SendRequest() w klasie MainWindow.




```
using System;
using System.Collections.Generic;
using System.IO;
using System.IO.Compression;
using System.Security.Cryptography;
using System.Text;
using System.Windows.Forms;
using System.Xml.Linq;
using Abc.UPEv4.Sample.RequestSender.UPE;

namespace Abc.UPEv4.Sample.RequestSender
{

    /// <summary>
    /// Main window
    /// </summary>
    public partial class MainWindow : Form
    {

        #region Fields
        private List<RequestFile> files;
        #endregion
```

```
#region Constructors

/// <summary>
/// Inicjalizuje klasę <see cref="MainWindow"/>
/// </summary>
public MainWindow()
{
    InitializeComponent();

    files = new List<RequestFile>();
}

#endregion

#region Methods

/// <summary>
/// Waliduje formularz
/// </summary>
/// <returns>Flaga, która informuje czy formularz jest poprawny</returns>
private bool ValidateForm()
{

    bool errors = false;
    errorProvider.Clear();


    if (string.IsNullOrEmpty(txtLogin.Text))
    {
        errorProvider.SetError(txtLogin, "Wpisz login");
        errors = true;
    }

    if (string.IsNullOrEmpty(txtPassword.Text))
    {
        errorProvider.SetError(txtPassword, "Wpisz hasło");
        errors = true;
    }

    if (string.IsNullOrEmpty(txtSymbol.Text))
    {
        errorProvider.SetError(txtSymbol, "Wpisz znak sprawy");
        errors = true;
    }

    if (string.IsNullOrEmpty(txtEmail.Text))
    {
        errorProvider.SetError(txtEmail, "Wpisz email");
        errors = true;
    }

    if (string.IsNullOrEmpty(txtSymbol.Text))
    {
        errorProvider.SetError(txtSymbol, "Wpisz imię");
        errors = true;
    }

    if (string.IsNullOrEmpty(txtLastName.Text))
    {
        errorProvider.SetError(txtLastName, "Wpisz nazwisko");
        errors = true;
    }

    if (string.IsNullOrEmpty(txtEmail.Text))
    {
        errorProvider.SetError(txtEmail, "Wpisz stanowisko");
        errors = true;
    }

    if (listFiles.Items.Count == 0)
    {
        errorProvider.SetError(listFiles, "Dołącz conajmniej jeden plik");
        errors = true;
    }
```

```
        return !errors;
    }

    /// <summary>
    /// Tworzy wniosek gotowy do wysłania
    /// </summary>
    /// <returns>Ścieżka do paczki z wnioskiem</returns>
    private String CreateRequestPackage()
    {

        String requestFilePath = Path.Combine(Path.GetTempPath(), Guid.NewGuid().ToString());
        String xmlRequestFilePath = Path.GetTempFileName();

        // towrzymy plik XML z wnioskiem
        // bardzo ważne! Paczka bez tego pliku nie zostanie przyjęta.
        XDocument doc = new XDocument();
        XElement xRoot = new XElement("wniosek",
            new XAttribute("wersja", "1.0"),
            new XElement("data", dateRequest.Value.ToString("yyyy-MM-dd")),
            new XElement("symbol", txtSymbol.Text),
            new XElement("imie", txtFirstName.Text),
            new XElement("nazwisko", txtLastName.Text),
            new XElement("email", txtEmail.Text),
            new XElement("stanowisko", txtPosition.Text)
        );

        doc.Add(xRoot);

        // załaczniki
        XElement xAttachments = new XElement("zalaczniki");
        xRoot.Add(xAttachments);

        foreach (RequestFile file in files)
        {
            xAttachments.Add(new XElement("zalacznik", new XAttribute("nazwa", file)));
        }

        doc.Save(xmlRequestFilePath);

        // wszystko pakujemy do zpia
        using (ZipArchive zip = ZipFile.Open(requestFilePath, ZipArchiveMode.Create))
        {

            zip.CreateEntryFromFile(xmlRequestFilePath, "Wniosek.xml");

            // zalaczniki

            foreach (RequestFile file in files)
            {
                zip.CreateEntryFromFile(file.FileName, file.ToString());
            }
        }

        return requestFilePath;
    }

    /// <summary>
    /// Tworzy hash hasła
    /// </summary>
    private String ComputeHashPassword(String password)
    {
        return BitConverter.ToString(new SHA256CryptoServiceProvider().ComputeHash(
           Encoding.UTF8.GetBytes(password))).Replace("-", "");
    }

    /// <summary>
    /// Wysyła wniosek
    /// </summary>
    private void SendRequest()
    {

        UpdateStatus("Autoryzacja");

        String packagePath = null;
```

```
try
{
    using (UPE.UPE upe = new UPE.UPE())
    {

        // haslo hashujemy algorytmem SHA256
        SPSession session = upe.StartNewSession(txtLogin.Text,
          ComputeHashPassword(txtPassword.Text));

        UpdateStatus("Utworzenie paczki z wnioskiem");
        packagePath = CreateRequestPackage();

        UpdateStatus("Wysyłanie wniosku");

        int issueID = upe.CreateEmptyIssue(session);

        #region Wysyłanie paczki do serwera

        byte[] packageFile = null;
        using (FileStream fs = new FileStream(packagePath, FileMode.Open))
        {
            packageFile = new byte[fs.Length];
            fs.Read(packageFile, 0, packageFile.Length);
        }

        // inicjalizacja transferu: wysyłamy częściami po 512KB (524288) każda
        PackageInfo packageInfo = upe.SetupPackageTransfer(session, issueID,
                  packageFile.Length);

        // wyslanie paczek
        foreach (PackagePart part in packageInfo.Parts)
        {

            UpdateStatus(string.Format("Wysyłanie paczki {0} z {1}",
                  part.PartId, packageInfo.Parts.Length));

            byte[] filePart = new byte[part.Size];
            Array.Copy(packageFile, (part.PartId - 1) * packageInfo.PartSize,
                  filePart, 0, part.Size);

            upe.SendPackagePart(session, packageInfo, part.PartId, filePart);
        }

        UpdateStatus("Potwierdzanie wysłania paczki");
        upe.ConfirmTransfer(session, packageInfo);

        #endregion

        // usuniecie starej paczki - nie bedzie potrzebna
        File.Delete(packagePath);

        // po wyslaniu paczki z zalacznikami zostanie zwrocona paczka
        // z wygenerowanym wnioskiem oraz przekonwertowanymi zalacznikami,
        // które trzeba podpisać podpisem elektronicznym

        #region Pobranie paczki

        UpdateStatus(
          "Pobieranie wygnerowanej paczki z wnioskiem do podpisania przez serwer");
        PackageInfo downloadPackage = upe.SetupDownloadPackageTransfer(
                  session, issueID, packageInfo.PackageId);

        byte[] buffer = new byte[downloadPackage.TotalSize];

        foreach (PackagePart part in downloadPackage.Parts)
        {
            byte[] data = upe.DownloadPackagePart(session,
                  downloadPackage, part.PartId);
            Array.Copy(data, 0, buffer, (part.PartId - 1) *
                  downloadPackage.PartSize, data.Length);
        }

        // zapisanie paczki
        File.WriteAllBytes(packagePath, buffer);

        #endregion
```

```
                // TODO rozpakowanie podpisanie wniosku (Wniosek.xml)

                // TODO spakowanie paczki

                // Odeslanie podpisanego wniosku z powrotem do UPE
                // jezeli wszystko bedzie ok, to zostanie pobrany przez redakcję

                #region Wysłanie podpisanej paczki

                packageFile = File.ReadAllBytes(packagePath);
                packageInfo = upe.SetupPackageTransfer(session, issueID, packageFile.Length);

                foreach (PackagePart part in packageInfo.Parts)
                {

                    UpdateStatus(string.Format("Wysyłanie paczki {0} z {1}",
                          part.PartId, packageInfo.Parts.Length));

                    byte[] filePart = new byte[part.Size];
                    Array.Copy(packageFile, (part.PartId - 1) * packageInfo.PartSize,
                          filePart, 0, part.Size);

                    upe.SendPackagePart(session, packageInfo, part.PartId, filePart);
                }

                UpdateStatus("Potwierdzanie wysłania paczki");
                upe.ConfirmTransfer(session, packageInfo);

                #endregion

                UpdateStatus("Zamykanie połączenia");
                upe.EndSession(session);
            }

            UpdateStatus("Paczka wysłania poprawnie");

        }
        catch (Exception ex)
        {
            UpdateStatus(ex.Message);
        }
        finally
        {

            if (!String.IsNullOrEmpty(packagePath) && File.Exists(packagePath))
            {
                File.Delete(packagePath);
            }

        }
    }

    /// <summary>
    /// Czyści formularz
    /// </summary>
    private void ClearForm()
    {

        txtFilePath.Text = null;
        txtSymbol.Text = null;
        txtLastName.Text = null;
        txtLogin.Text = null;
        txtPassword.Text = null;
        txtEmail.Text = null;

        files.Clear();
        BindRequestFiles();
    }

    /// <summary>
    /// Aktualizuje status
    /// </summary>
    /// <param name="message">Treść wiadomośći</param>
    private void UpdateStatus(String message)
    {
```

```
        lblStatus.Visible = !string.IsNullOrEmpty(message);
        lblStatus.Text = message;
    }

    /// <summary>
    /// Binduje liste z plikami
    /// </summary>
    private void BindRequestFiles()
    {
        listFiles.DataSource = null;
        listFiles.DataSource = files;
    }

    #endregion

    #region Event handlers

    private void btnSend_Click(object sender, EventArgs e)
    {

        if (!ValidateForm())
        {
            UpdateStatus("Wystąpiły błędy podczas walidacji formularza");
            return;
        }

        SendRequest();
    }

    private void btnClear_Click(object sender, EventArgs e)
    {

        ClearForm();
    }

    private void btnBrowse_Click(object sender, EventArgs e)
    {

        using (OpenFileDialog openFileDialog = new OpenFileDialog())
        {

            openFileDialog.Multiselect = false;
            openFileDialog.Filter = "EAP XML Legislator |*.zipx|EAP|*.lapx|Edytor MSWiA|*.zip";

            if (openFileDialog.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
            {
                txtFilePath.Text = openFileDialog.FileName;
            }
        }
    }

    private void btnAdd_Click(object sender, EventArgs e)
    {

        if (!String.IsNullOrEmpty(txtFilePath.Text) && File.Exists(txtFilePath.Text))
        {

            // sprawdzenie czy nie ma juz dodanego zalacznika o takiej samej nazwie
            string filename = Path.GetFileNameWithoutExtension(txtFilePath.Text).ToLower();
            if (files.Exists(x =>
                  Path.GetFileNameWithoutExtension(x.FileName).ToLower() == filename))
            {
                return;
            }

            files.Add(new RequestFile() { FileName = txtFilePath.Text });
            BindRequestFiles();

            txtFilePath.Text = null;
        }
    }

    #endregion

    #region Classes
```

```
        /// <summary>
        /// Plik dołaczany do wniosku
        /// </summary>
        protected class RequestFile
        {

            #region Properties
            public String FileName { get; set; }
            #endregion

            #region Methods
            public override string ToString()
            {
                return Path.GetFileName(FileName);
            }
            #endregion
        }

        #endregion
    }
}
```