

# Podstawy Web Serwisów w SAS

Zespół Wsparcia Technicznego, SAS Institute Polska

## Wstęp

Technologia Usług Internetowych (web serwisów) jest najprostszym sposobem integracji systemów informatycznych i budowania systemów rozproszonych. W ostatnich latach zdobyła duże znaczenie i popularność. Dokument ten opisuje podstawy używania i tworzenia web serwisów w produktach SAS Institute. Zawiera proste przykłady umożliwiające szybkie rozpoczęcie korzystania z usług internetowych i stanowi uzupełnienie oficjalnej dokumentacji SAS Institute. Opisuje dostęp do web serwisów za pomocą języka SAS 4GL oraz tworzenie web serwisów w aplikacji SAS BI Web Services. Dokument dotyczy usług internetowych opartych na protokole Simple Object Access Protocol (SOAP) i opisywanych językiem Web Services Description Language (WSDL). Przykłady były testowane na wersji SAS 9.3.

W wersji 9.4 jedyną różnicą jest numer portu aplikacji web'owej przy domyślnej konfiguracji. W wersji 9.3 był to port 8080. W wersji 9.4 domyślnym portem jest 7980 na systemie Linux i 80 na systemie Windows.

## Spis treści

Wstęp .....	1
Cechy Web Serwisów .....	1
Testowanie Web Serwisów .....	1
Przykładowy Web Serwis.....	1
Dostęp do usługi internetowej z programu SoapUI .....	3
Dostęp do usługi internetowej z poziomu SAS 4GL.....	5
Tworzenie Usług internetowych w SAS BI Web Services. ....	7
Prosty Web Serwis dodający dwie liczby.....	7
Web serwis transponujący zbiory SAS.....	19
Dodanie schematu dla danych wejściowych lub wyjściowych.....	28

## Cechy Web Serwisów

- Opublikowane w sieci Internet
- Przesyłają komunikaty w formacie XML
- Samoopisujące się
- Niezależne od platformy i języka programowania
- Cechy: skalowalność, prostota, rozproszenie

## Testowanie Web Serwisów

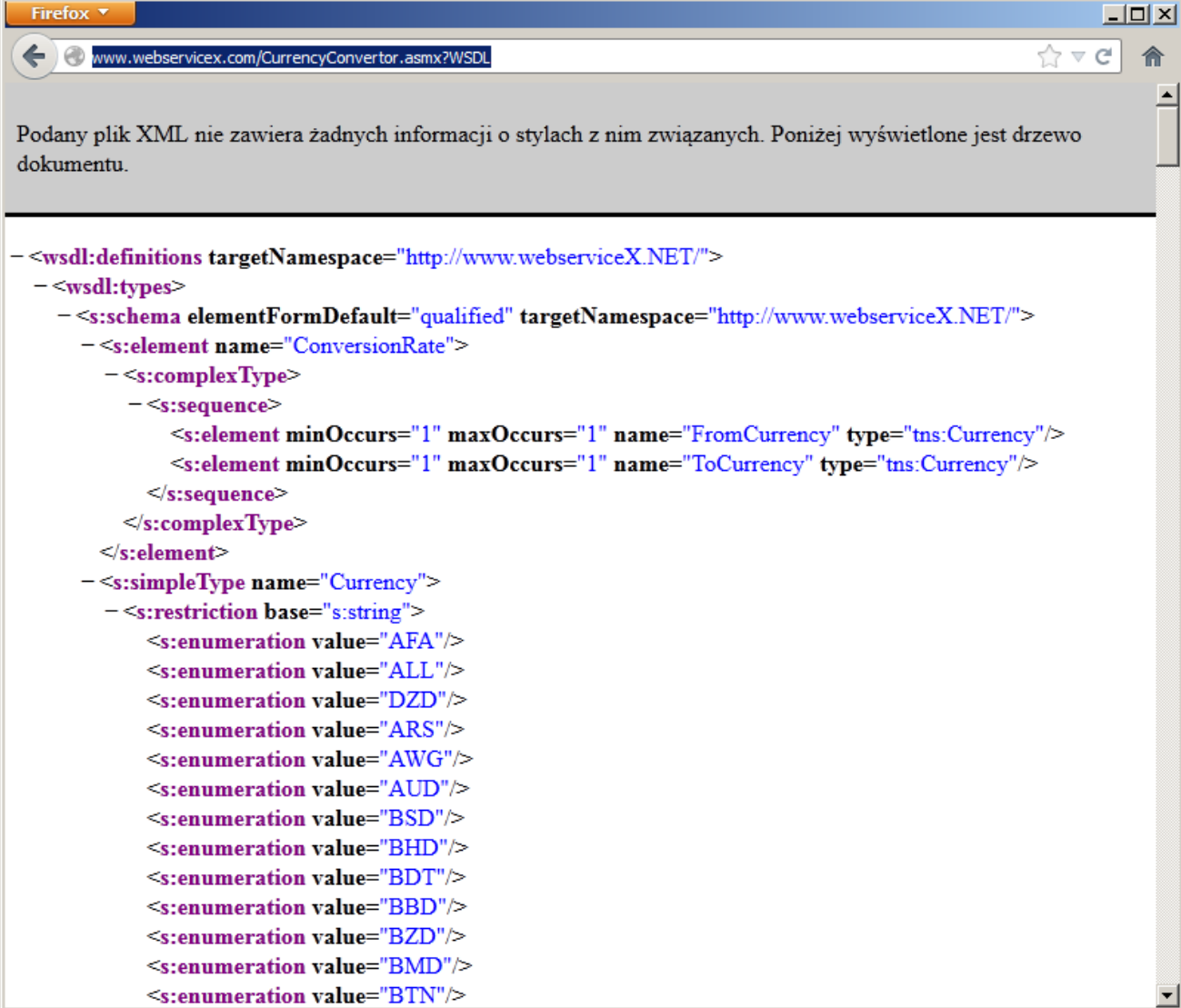
Do testowania web serwisów użyć można aplikacji SoapUI dostępnej na licencji GNU Lesser General Public License (LGPL). Aplikację można pobrać ze strony <http://sourceforge.net/projects/soapui/>.

## Przykładowy Web Serwis

Do testowania użyjemy przykładowego Web Serwisu udostępniającego kursy walut dostępnego pod adresem <http://www.websvicex.com/CurrencyConvertor.asmx>

Usługi internetowe mogą udostępniać własny opis w opartym na XML'u języku WSDL. Można go obejrzeć dodając do adresu usługi końcówkę „?wsdl”. WSDL dla przykładowego web serwisu dostępny jest pod adresem:

<http://www.webserviceX.com/CurrencyConvertor.asmx?wsdl>



```
-<wSDL:definitions targetNamespace="http://www.webserviceX.NET/">
  -<wSDL:types>
    -<s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET/">
      -<s:element name="ConversionRate">
        -<s:complexType>
          -<s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="FromCurrency" type="tns:Currency"/>
            <s:element minOccurs="1" maxOccurs="1" name="ToCurrency" type="tns:Currency"/>
          </s:sequence>
        </s:complexType>
      </s:element>
    -<s:simpleType name="Currency">
      -<s:restriction base="s:string">
        <s:enumeration value="AFA"/>
        <s:enumeration value="ALL"/>
        <s:enumeration value="DZD"/>
        <s:enumeration value="ARS"/>
        <s:enumeration value="AWG"/>
        <s:enumeration value="AUD"/>
        <s:enumeration value="BSD"/>
        <s:enumeration value="BHD"/>
        <s:enumeration value="BDT"/>
        <s:enumeration value="BBD"/>
        <s:enumeration value="BZD"/>
        <s:enumeration value="BMD"/>
        <s:enumeration value="BTN"/>
      </s:restriction>
    </s:simpleType>
  </s:schema>
</wSDL:types>
</wSDL:definitions>
```

Opis usługi internetowej umożliwia skonstruowanie zapytania w formacie SOAP. Narzędzie takie jak SoapUI potrafi automatycznie skonstruować szablon zapytania na podstawie WSDL'a.

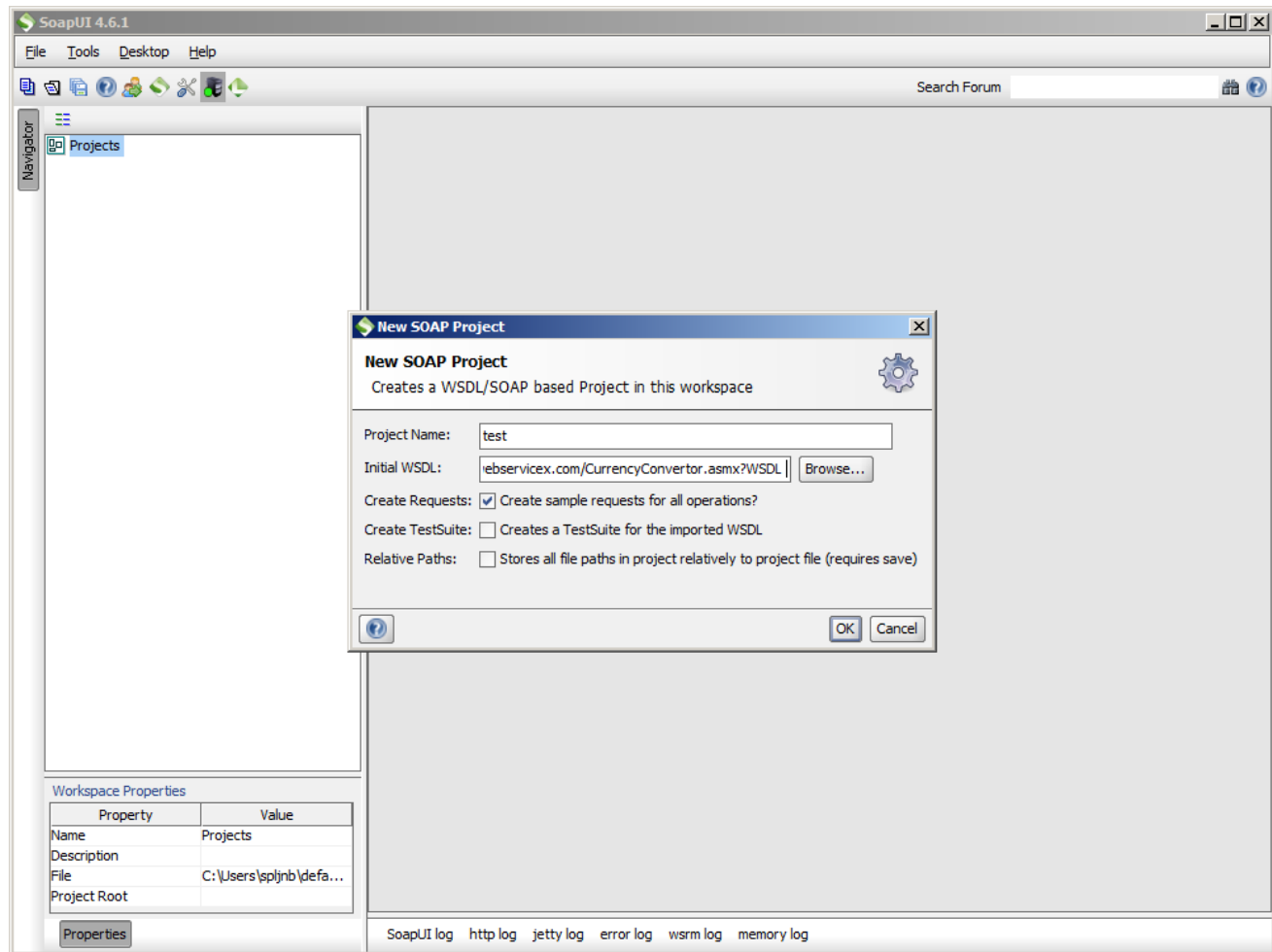
## Dostęp do usługi internetowej z programu SoapUI

Wybieramy Projects -> New SOAP Project

Wpisujemy nazwę projektu i podajemy adres pod którym dostępny jest WSDL:

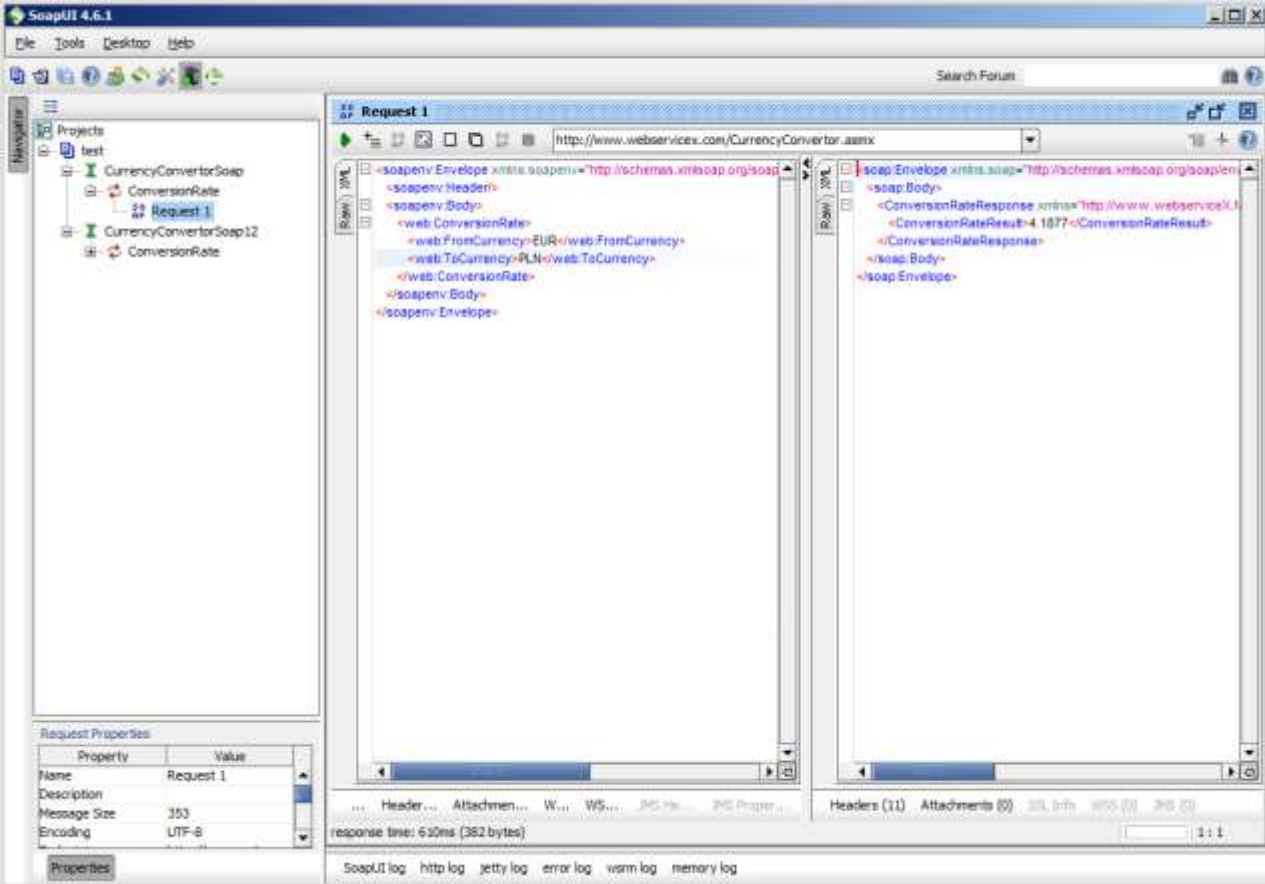
<http://www.websvcx.com/CurrencyConvertor.asmx?wsdl>

*Przy problemach z połączeniem należy się upewnić czy nie jesteśmy za serwerem proxy i wpisać odpowiednie ustawienia w Preferences -> Proxy Settings. Program nie wykrywa automatycznie ustawień proxy dla sieci jak popularne przeglądarki internetowe.*



Program automatycznie utworzy szablon XML'a z zapytaniem odpowiednim dla tej usługi internetowej. Uzupełniamy go i naciskamy zieloną strzałkę aby wysłać zapytanie do usługi internetowej i dostajemy

odpowieź (kurs euro w złotych) także w formacie XML.



The screenshot displays the SoapUI 4.6.1 interface. The main window shows a SOAP request and response for the service `http://www.webservices.com/CurrencyConverter.asmx`. The request is a SOAP message with the following XML structure:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header/>
  <soap:Body>
    <web:ConversionRate>
      <web:FromCurrency>EUR</web:FromCurrency>
      <web:ToCurrency>PLN</web:ToCurrency>
    </web:ConversionRate>
  </soap:Body>
</soap:Envelope>
```

The response is a SOAP message with the following XML structure:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ConversionRateResponse xmlns="http://www.webserviceX.I" >
      <ConversionRateResult>4.1877</ConversionRateResult>
    </ConversionRateResponse>
  </soap:Body>
</soap:Envelope>
```

The response time is 630ms (382 bytes). The interface also shows a project tree on the left with 'Request 1' selected, and a 'Request Properties' table at the bottom left.

Property	Value
Name	Request 1
Description	
Message Size	353
Encoding	UTF-8

## Dostęp do usługi internetowej z poziomu SAS 4GL

Takie samo zapytanie możemy wysłać z poziomu języka SAS 4GL za pomocą procedury HTTP lub procedury SOAP. Obie te procedury są dostępne od wersji SAS 9.2. Są one dostępne w ramach licencji na produkt SAS Base.

```
filename request temp ;
filename response "c:\webservice\response.xml";

*plik z zapytaniem xml;
data _null_;
  file request;
  input;
  put _infile_;
  datalines4;
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://www.webserviceX.NET/">
  <soapenv:Header/>
  <soapenv:Body>
    <web:ConversionRate>
      <web:FromCurrency>EUR</web:FromCurrency>
      <web:ToCurrency>PLN</web:ToCurrency>
    </web:ConversionRate>
  </soapenv:Body>
</soapenv:Envelope>
;;;

proc http
  in=request
  out=response
  url="http://www.webserviceX.com/CurrencyConvertor.asmx"
  method="post"
  ct="text/xml; charset=utf-8"
/* Trzeba podać jeśli jesteśmy za serwerem proxy */
  proxyhost="adres.serwera.proxy.com"
  proxyport=80
  ;
run;
```

W wyniku działania procedury odpowiedź została zapisana w pliku c:\webservice\response.xml. Za pomocą silnika xml można taki plik zapisać jako bibliotekę SAS i wypisać zawartość zbioru.

```
filename map "c:\webservice\response.map";
libname resp xmlv2 xmlfileref=response xmlmap=map;
proc print data=resp.ConversionRateResponse;
run;
```

Do przypisania pliku xml jako biblioteki SAS potrzebna jest w tym przypadku mapa. Mapę tworzy się za pomocą narzędzia "SAS XML Mapper", które może być zainstalowane z depotu lub pobrane ze strony:

<http://support.sas.com/demosdownloads/setupcat.jsp?cat=Base+SAS+Software>. Film : "How to Automatically Generate XMLMap Files"

można znaleźć na stronie <http://support.sas.com/rnd/base/xmlengine/>.

Zawartość pliku response.map utworzonego automatycznie za pomocą programu SAS XML Mapper:

```
<?xml version="1.0" encoding="windows-1250"?>
<!-- ##### -->
<!-- 2013-06-13T12:15:59 -->
<!-- SAS XML Libname Engine Map -->
<!-- Generated by XML Mapper, 903000.0.0.20110518190000_v930 -->
<!-- ##### -->
<!-- ### validation report ### -->
<!-- ##### -->
<!-- XMLMap validation completed successfully. -->
<!-- ##### -->
<SXLEMAP name="AUTO_GEN" version="2.1">

  <NAMESPACES count="4">
    <NS id="1" prefix="soap">http://schemas.xmlsoap.org/soap/envelope/</NS>
    <NS id="2" prefix="xsi">http://www.w3.org/2001/XMLSchema-instance</NS>
    <NS id="3" prefix="xsd">http://www.w3.org/2001/XMLSchema</NS>
    <NS id="4" prefix="">http://www.webserviceX.NET/</NS>
  </NAMESPACES>

  <!-- ##### -->
  <TABLE description="ConversionRateResponse" name="ConversionRateResponse">
    <TABLE-PATH syntax="XPathENR">/{1}Envelope/{1}Body/{4}ConversionRateResponse</TABLE-PATH>

    <COLUMN name="ConversionRateResult">
      <PATH syntax="XPathENR">/{1}Envelope/{1}Body/{4}ConversionRateResponse/{4}ConversionRateResult</PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>double</DATATYPE>
    </COLUMN>

  </TABLE>

</SXLEMAP>
```

Web serwis można wywołać także za pomocą procedury SOAP:

```
proc soap
  in=request
  out=response
  url="http://www.webserviceX.com/CurrencyConvertor.asmx"
  soapaction="http://www.webserviceX.NET/ConversionRate"
/* Trzeba podać jeśli jesteśmy za serwerem proxy */
  proxyhost="adres.serwera.proxy.com"
  proxyport=80
;
run;
```

Dokumentacja:

[HTTP Procedure](#), [SOAP Procedure](#), [SAS® 9.3 XML LIBNAME Engine](#).

## Tworzenie Usług internetowych w SAS BI Web Services.

Aplikacja SAS BI Web Services jest częścią SAS Integration Technologies wchodzącej w skład pakietów takich jak SAS BI Server i wielu rozwiązań SAS. Jest to aplikacja webowa działająca na serwerze aplikacji takim jak JBOSS (SAS 9.2, SAS 9.3) lub SAS Web Application Server (SAS 9.4). Umożliwia ona udostępnianie procesów gotowych jako web serwisów. Od wersji 9.3 wszystkie procesy gotowe automatycznie udostępniane są jako web serwisy. W wersji 9.2 konieczny jest dodatkowy krok „Deploy As Web Service” opisany na końcu tego przykładu.

### Prosty Web Serwis dodający dwie liczby.

- Logujemy się do programu SAS Management Console
- Na wybranym folderze klikamy prawym przyciskiem myszy i wybieramy New -> Stored Process
- Wpisujemy nazwę tworzonego procesu gotowego
- Proces gotowy będzie wykonywał następujący kod SAS (w tym przykładzie jest to zawartość pliku c:\webservice\dodaj.sas):

```
%global liczba1 liczba2;  
%let suma = %eval(&liczba1 + &liczba2);
```

- Obrazy na kolejnych stronach przedstawiają właściwości procesu gotowego :

**dodaj Properties**

General | Execution | Parameters | Data | Authorization

Name:

Type: Stored process

Description:

Location: /przyklady

Created: 6/25/13 12:08 PM

Modified: 6/25/13 12:08 PM

Keywords:

Responsibilities:

Name	Role

Hide from user



**dodaj Properties** [X]

General Execution Parameters Data Authorization

Application server:  
SASApp

Server type:

- Default server  
Select this option to allow the client application to specify the server.
- Stored process server only  
Select this option if the stored process uses sessions or if it uses replay (for example, to produce graphics in streaming output).
- Workspace server only  
Select this option if the stored process must be run under the client identity.

Source code location and execution:

- Allow execution on other application servers (store source code in metadata)
- Allow execution on selected application server only
  - Store source code in metadata
  - Store source code on application server

Source code repository: c:\webservice [v] [Manage...]

Source file: dodaj.sas

[Edit Source Code...]

Result capabilities:  Stream  Package

[OK] [Cancel] [Help]

**dodaj Properties** [X]

General | Execution | **Parameters** | Data | Authorization

Prompts (input parameters):

Displayed Text	Name	Type
Parameters		Standard group
liczba 1	liczba 1	Numeric (integer)
liczba 2	liczba 2	Numeric (integer)

Buttons: New Prompt..., New Group..., Edit..., Delete, Move Up, Move Down, Add Shared..., Save as Shared..., Unshare, Test Prompts...

Output parameters:

Label	Name	Type	Description
	suma	Integer	

Buttons: New..., Edit..., Delete

OK Cancel Help

**Edit Prompt**

General | Prompt Type and Values

Name:  
liczba1

Displayed text:  
liczba1

Description:

Parent group:  
Parameters

Options

Hide at run time     Requires a non-blank value

Read-only values

OK    Cancel    Help

The image shows a screenshot of the 'Edit Prompt' dialog box in SAS, specifically the 'Prompt Type and Values' tab. The dialog box has a title bar with the text 'Edit Prompt' and a close button (X). The main area is divided into several sections:

- Prompt type:** A dropdown menu is set to 'Numeric'.
- Method for populating prompt:** A dropdown menu is set to 'User enters values'.
- Number of values:** A dropdown menu is set to 'Single value'.
- Allow only integer values:** A checked checkbox.
- Minimum value allowed:** An empty text input field.
- Maximum value allowed:** An empty text input field.
- Include Special Values:** A section with two unchecked checkboxes: 'All possible values' and 'Missing values'.
- Default value:** A text input field containing the number '2'.

At the bottom right of the dialog box, there are three buttons: 'OK', 'Cancel', and 'Help'.

**Edit Prompt**

General | Prompt Type and Values

Name:  
liczba2

Displayed text:  
liczba2

Description:

Parent group:  
Parameters

Options

Hide at run time     Requires a non-blank value

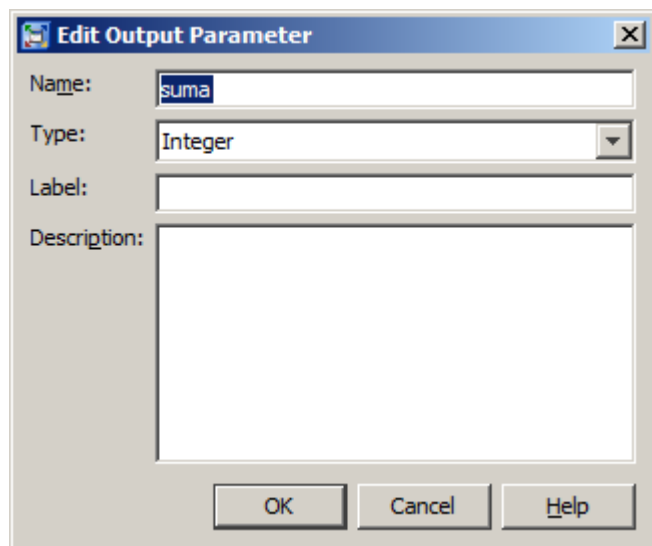
Read-only values

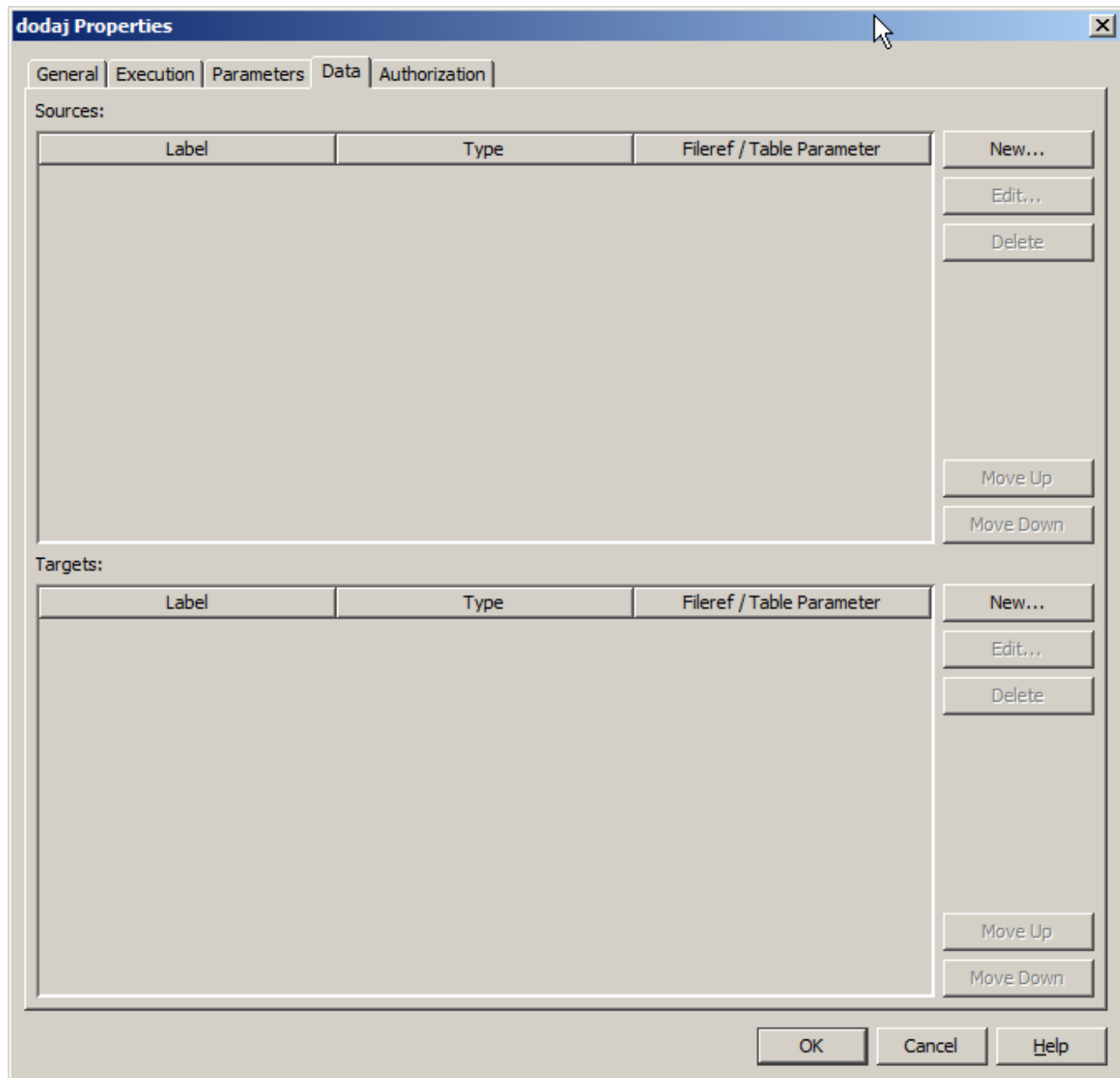
OK    Cancel    Help

The image shows a screenshot of the 'Edit Prompt' dialog box in SAS, specifically the 'Prompt Type and Values' tab. The dialog box has a title bar with the text 'Edit Prompt' and a close button (X). The main area is divided into several sections:

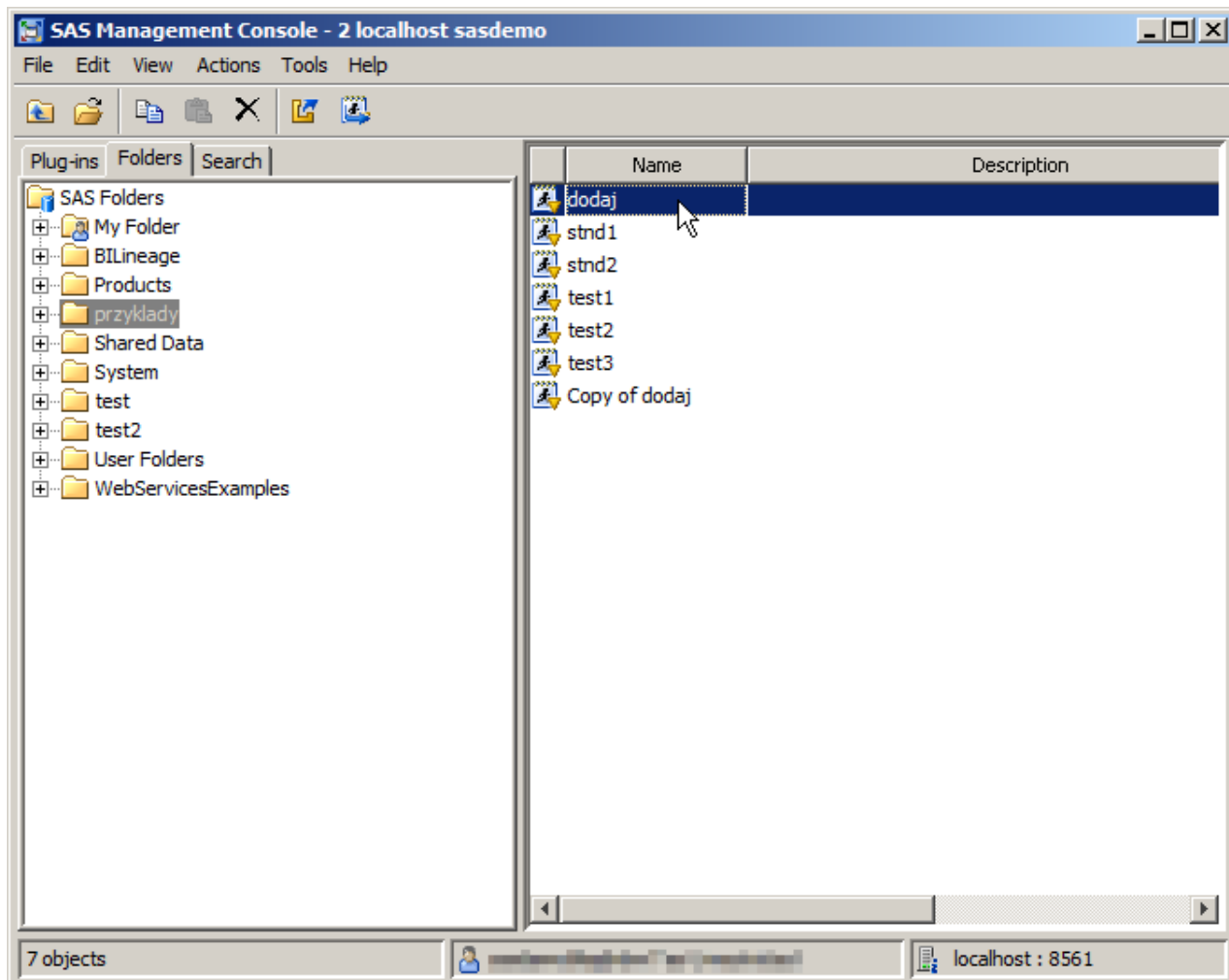
- Prompt type:** A dropdown menu is set to 'Numeric'.
- Method for populating prompt:** A dropdown menu is set to 'User enters values'.
- Number of values:** A dropdown menu is set to 'Single value'.
- Allow only integer values:** A checkbox is checked.
- Minimum value allowed:** An empty text input field.
- Maximum value allowed:** An empty text input field.
- Include Special Values:** Two checkboxes are present: 'All possible values' (unchecked) and 'Missing values' (unchecked).
- Default value:** A text input field containing the number '2'.

At the bottom right of the dialog box, there are three buttons: 'OK', 'Cancel', and 'Help'.









Utworzony proces gotowy jest dostępny jako web serwis bez żadnych dodatkowych zabiegów (wersje od SAS 9.3) pod adresem [http\(s\)://nazwa\\_serwera:port/SASBIWS/services/\[lokalizacja\\_procesu\\_w\\_metadanych\]](http(s)://nazwa_serwera:port/SASBIWS/services/[lokalizacja_procesu_w_metadanych]). Chodzi tu o adres serwera aplikacji webowych i jego port. Dla tego konkretnego procesu WSDL znajdzie się pod następującym adresem: <http://nazwa-maszyny:8080/SASBIWS/services/przyklady/dodaj?WSDL>. Jeśli ścieżka w metadanych zawiera znaki specjalne to muszą być one zakodowane. Na przykład kod znaku spacji to %20. URL ze spacją <http://nazwa-maszyny:8080/SASBIWS/services/przyklady/Copy%20of%20dodaj?wsdl>

Web serwis może być także widoczny pod adresem niezależnym od lokalizacji w metadanych. Należy kliknąć na proces gotowy prawym przyciskiem myszy i wybrać „Deploy As Web Service...”. Wybieramy potem

domyślny Web Service Marker URL i wpisujemy nazwę pod jaką chcemy aby serwis był dostępny:

**Deploy As Web Service**

**Web Service Information**

Specify the URL for the SAS BI Web Services WebServiceMaker and a name for the new Web service to be deployed.

Web Service Maker URL:

New Web Service Name:

Use my current credentials to deploy

Use these credentials to deploy:

User ID:

Password:

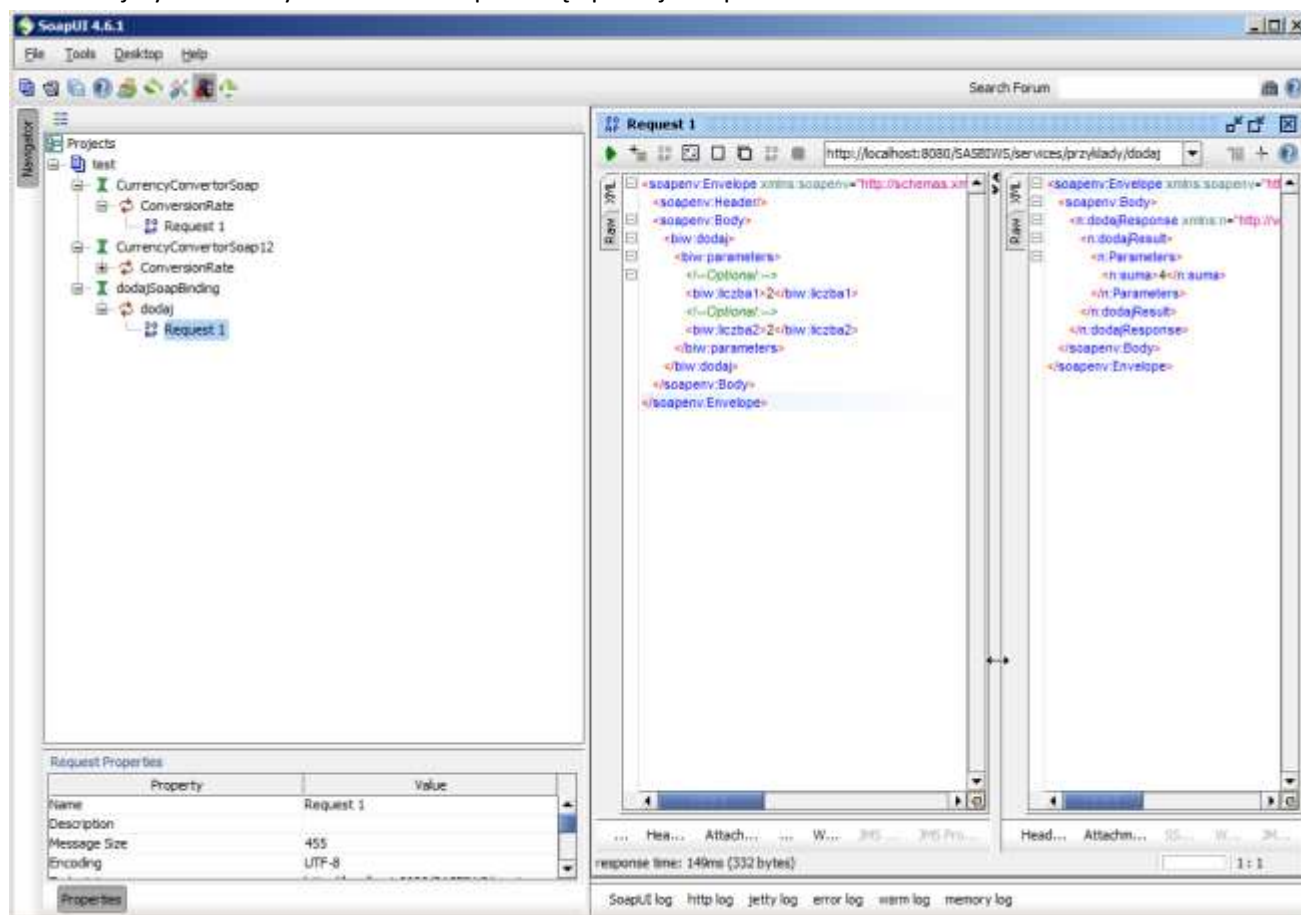
< Back   Next >   Finish   Cancel   Help

WSDL Web serwisu jest widoczny pod adresem

[http://spljnbw7.emea.sas.com:8080/SASBIWS/services/dodaj\\_liczby?wsdl](http://spljnbw7.emea.sas.com:8080/SASBIWS/services/dodaj_liczby?wsdl)

Ten krok jest konieczny w wersji SAS 9.2 aby udostępnić proces gotowy jako web serwis.

Przetestujmy stworzony web serwis za pomocą aplikacji SoapUI:



*Przy testowaniu usług internetowych na serwerze w lokalnej sieci należy wyłączyć proxy w aplikacji SoapUI (Preferences -> Proxy Settings)*

## Web serwis transponujący zbiory SAS

W poprzednim przykładzie web serwis przyjmował na wejście pojedyncze wartości przekazywane do procesu gotowego SAS jako makrozmiennicze i zwracał także pojedyncze wartości. W tym przykładzie wejściem będzie plik xml zawierający tabele w formacie używanym przez silnik bibliotek xml w SAS. Wyjściem także będzie plik xml.

- Logujemy się do programu SAS Management Console
- Na wybranym folderze klikamy prawym przyciskiem myszy i wybieramy New -> Stored Process
- Wpisujemy nazwę tworzonego procesu gotowego
- Proces gotowy będzie wykonywał następujący kod SAS:

```
libname we XML;
libname wy XML;

* stworzenie zbioru zawierającego nazwy zbiorów na wejściu ;
proc contents noprint data=we._all_ out=zbiory; quit;
run;

*stworzenie makrozmiennych zawierających nazwy zbiorów na wejściu;
data _null_;
set zbiory end=end;
by memname;
retain nr_zbioru 0;

if first.memname then do;
    nr_zbioru+1;
    call symput(compress('zbior_'||nr_zbioru),trim(memname));
end;

if end then      call symput('liczba_zbiorow',trim(nr_zbioru));
run;

%macro transpose;
%do i=1 %to &liczba_zbiorow;
proc transpose data=we.&&zbior_&i out=&&zbior_&i;
var _all_;
run;
%end;
%mend;
%transpose;

*skopiowanie przetransponowanych zbiorów do biblioteki wyjściowej;
%macro wy;
proc copy in=work out=wy;
select
%do i=1 %to &liczba_zbiorow;
&&zbior_&i
%end;
;
run;
%mend;
%wy;

libname we;
libname wy;
```

- Obrazy na kolejnych stronach przedstawiają właściwości procesu gotowego :

**transpose Properties**

General | Execution | Parameters | Data | Authorization

Name: transpose

Type: Stored process

Description:

Location: /przyklady

Created: 11/15/13 4:24 PM

Modified: 11/15/13 4:43 PM

Keywords:

Add...  
Edit...  
Delete

Responsibilities:

Name	Role
------	------

Add...  
Delete

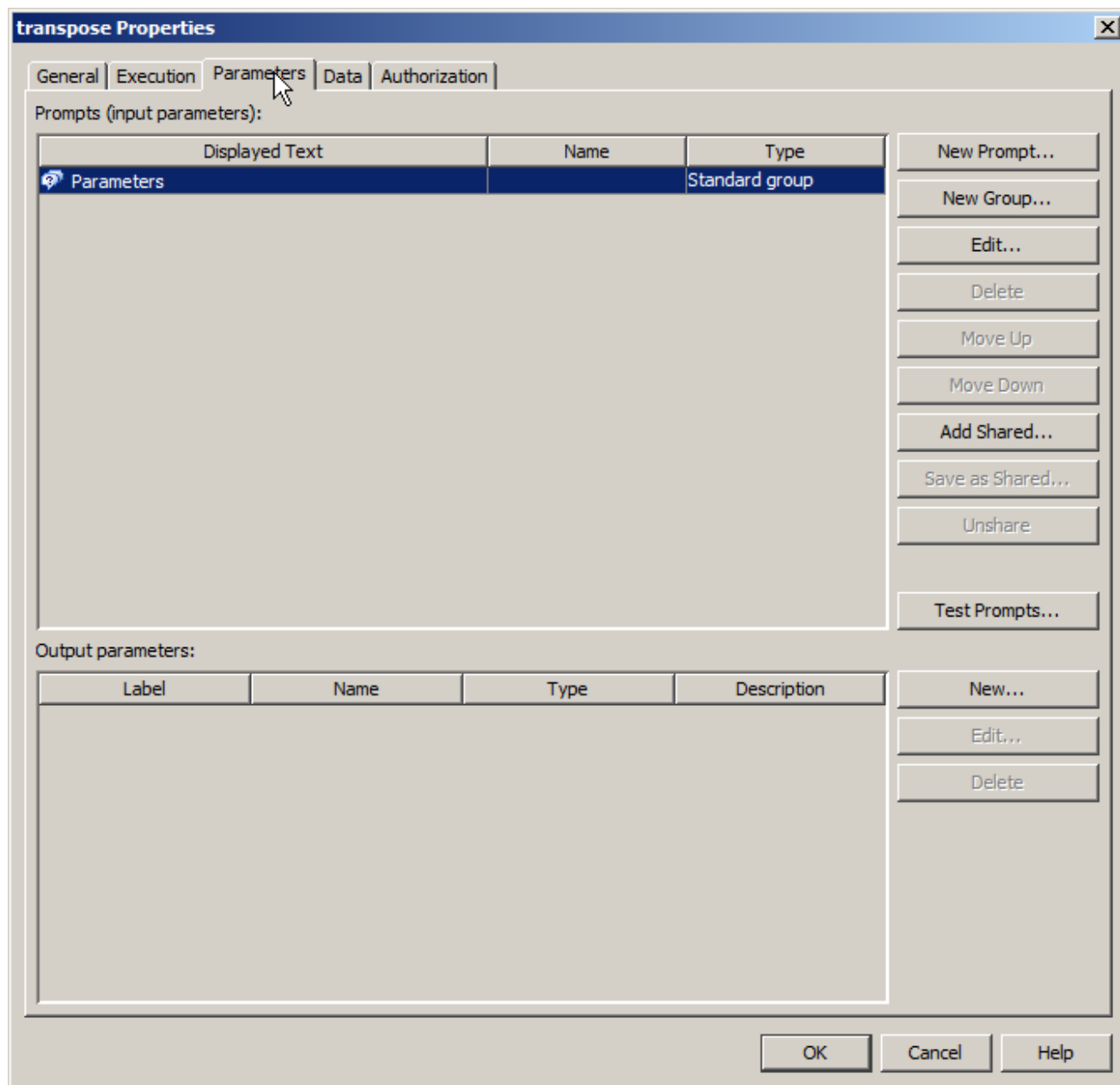
Hide from user

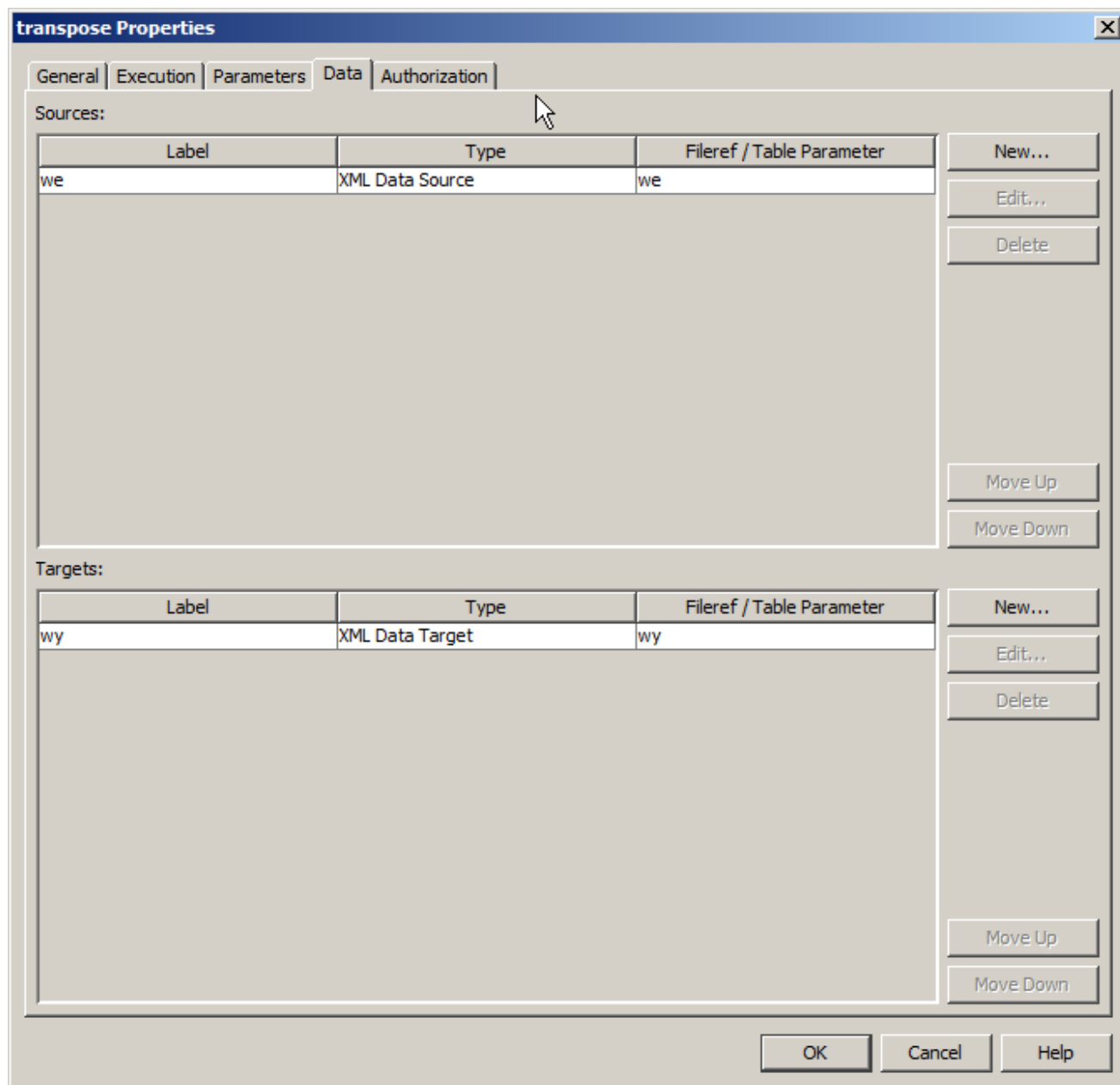
OK Cancel Help

The image shows a screenshot of the 'transpose Properties' dialog box, specifically the 'Execution' tab. The dialog has a title bar with the text 'transpose Properties' and a close button. Below the title bar are five tabs: 'General', 'Execution', 'Parameters', 'Data', and 'Authorization'. The 'Execution' tab is currently selected. The main content area is divided into several sections:

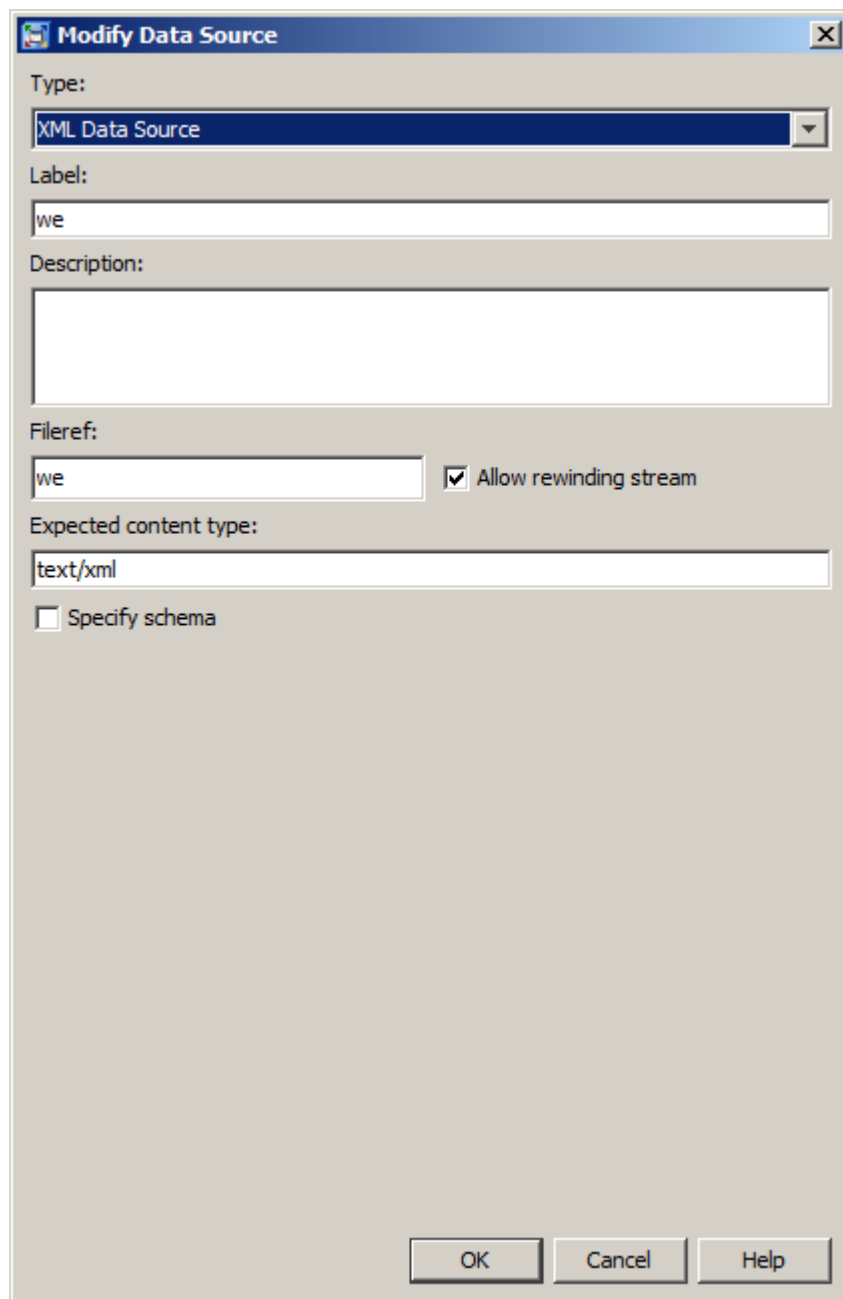
- Application server:** A dropdown menu showing 'SASApp'.
- Server type:** Three radio button options:
  - Default server  
Select this option to allow the client application to specify the server.
  - Stored process server only  
Select this option if the stored process uses sessions or if it uses replay (for example, to produce graphics in streaming output).
  - Workspace server only  
Select this option if the stored process must be run under the client identity.
- Source code location and execution:** Two radio button options:
  - Allow execution on other application servers (store source code in metadata)
  - Allow execution on selected application server only
    - Store source code in metadata
    - Store source code on application server
- Source code repository:** A text box containing 'c:\webservice' and a 'Manage...' button.
- Source file:** A text box containing 'transpose.sas'.
- Edit Source Code...** button.
- Result capabilities:** Two checkboxes: 'Stream' (unchecked) and 'Package' (unchecked).

At the bottom right of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.









**Modify Data Source**

Type:  
XML Data Source

Label:  
we

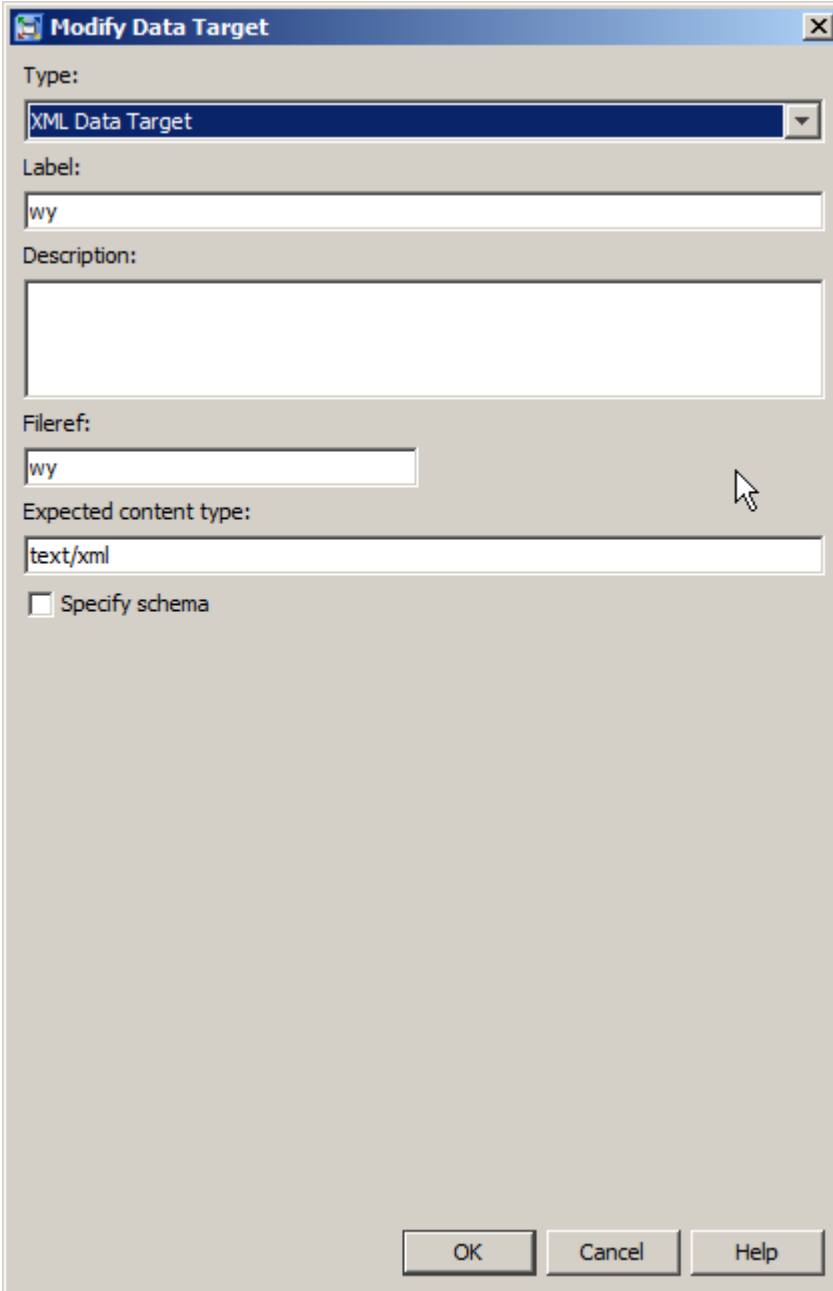
Description:

Fileref:  
we  Allow rewinding stream

Expected content type:  
text/xml

Specify schema

OK Cancel Help



**Modify Data Target**

Type:  
XML Data Target

Label:  
wy

Description:

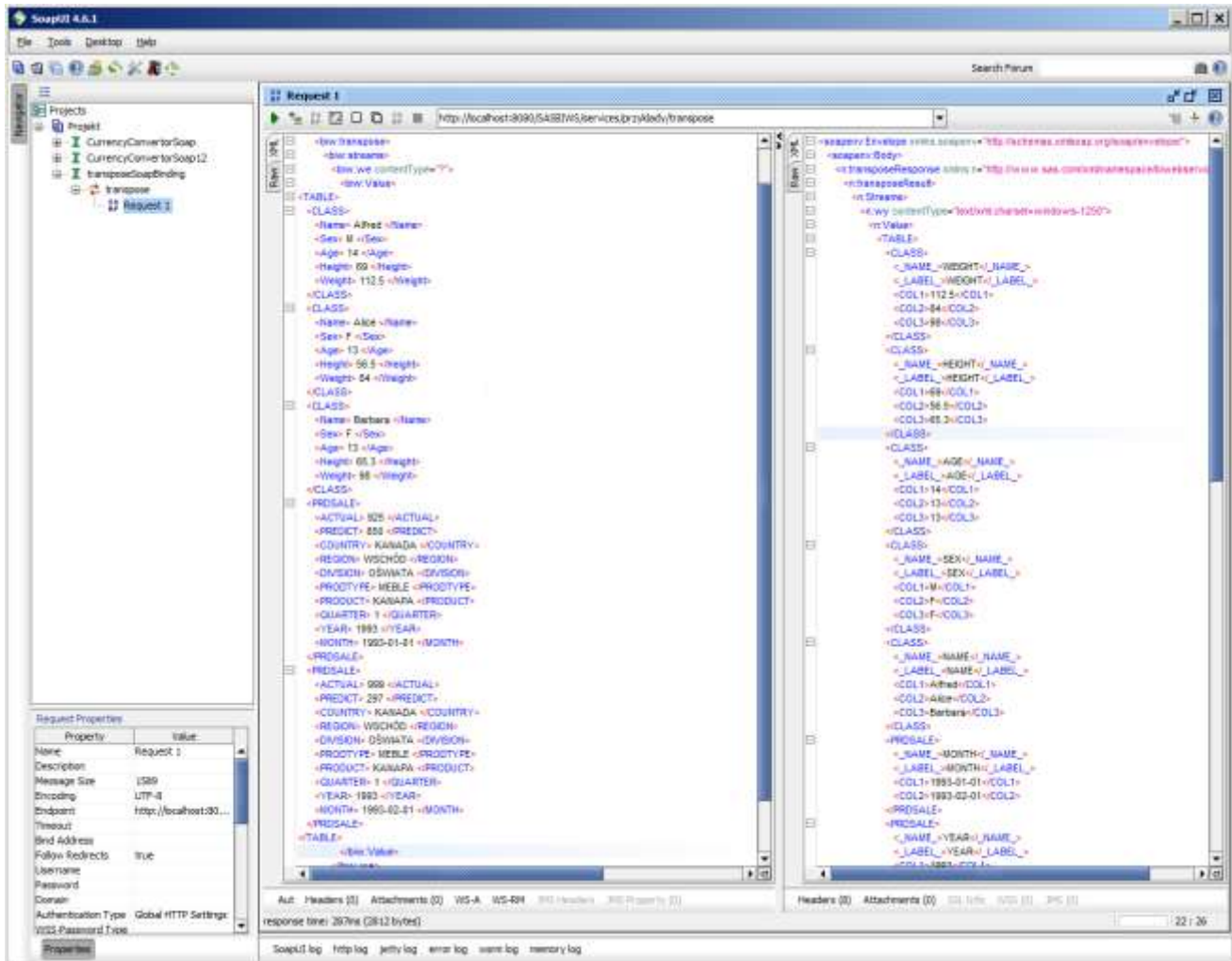
Fileref:  
wy

Expected content type:  
text/xml

Specify schema

OK Cancel Help

Przetestujmy stworzony web serwis za pomocą aplikacji SoapUI:



Dane użyte do testu:

```
<TABLE>
  <CLASS>
    <Name> Alfred </Name>
    <Sex> M </Sex>
    <Age> 14 </Age>
    <Height> 69 </Height>
    <Weight> 112.5 </Weight>
  </CLASS>
  <CLASS>
    <Name> Alice </Name>
    <Sex> F </Sex>
    <Age> 13 </Age>
    <Height> 56.5 </Height>
    <Weight> 84 </Weight>
  </CLASS>
  <CLASS>
    <Name> Barbara </Name>
    <Sex> F </Sex>
    <Age> 13 </Age>
    <Height> 65.3 </Height>
    <Weight> 98 </Weight>
  </CLASS>
</PRDSALE>
```

```

<ACTUAL> 925 </ACTUAL>
<PREDICT> 850 </PREDICT>
<COUNTRY> KANADA </COUNTRY>
<REGION> WSCHÓD </REGION>
<DIVISION> OŚWIATA </DIVISION>
<PRODTYPE> MEBLE </PRODTYPE>
<PRODUCT> KANAPA </PRODUCT>
<QUARTER> 1 </QUARTER>
<YEAR> 1993 </YEAR>
<MONTH> 1993-01-01 </MONTH>
</PRDSALE>
<PRDSALE>
  <ACTUAL> 999 </ACTUAL>
  <PREDICT> 297 </PREDICT>
  <COUNTRY> KANADA </COUNTRY>
  <REGION> WSCHÓD </REGION>
  <DIVISION> OŚWIATA </DIVISION>
  <PRODTYPE> MEBLE </PRODTYPE>
  <PRODUCT> KANAPA </PRODUCT>
  <QUARTER> 1 </QUARTER>
  <YEAR> 1993 </YEAR>
  <MONTH> 1993-02-01 </MONTH>
</PRDSALE>
</TABLE>

```

### Dodanie schematu dla danych wejściowych lub wyjściowych

Dla danych wejściowych lub wyjściowych możemy także określić schemat XML. Wtedy WSDL będzie zawierał informację o postaci danych wejściowych lub wyjściowych. W tym przykładzie zmodyfikujemy web serwis transponujący dane dodając schemat XML dla danych wejściowych.

Schemat może być utworzony automatycznie za pomocą następującego kodu SAS:

```

filename schemat 'C:\webservice\class.xsd';
libname xmllib xml 'C:\webservice\class.xml' xmlmeta=schemadata
xmlschema=schemat;

```

```

data xmllib.class;
set sashelp.class;
run;

```

Wygenerowany schemat trzeba jeszcze zmodyfikować przez dodanie do elementu xs:schema atrybutów:

```
targetNamespace="http://www.tempuri.org/xml/namespace/class"
```

```
xmlns="http://www.tempuri.org/xml/namespace/class" elementFormDefault="qualified"
```

Schemat gotowy do użycia przez web serwis wygląda tak:

```

<?xml version="1.0" encoding="windows-1250" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.tempuri.org/xml/namespace/class"
xmlns="http://www.tempuri.org/xml/namespace/class"
elementFormDefault="qualified">
  <xs:element name="TABLE">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="CLASS" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="CLASS">
    <xs:complexType>

```

```
<xs:sequence>
  <xs:element name="Name" minOccurs="0">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="8" />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="Sex" minOccurs="0">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:maxLength value="1" />
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="Age" minOccurs="0" type="xs:double" />
  <xs:element name="Height" minOccurs="0" type="xs:double" />
  <xs:element name="Weight" minOccurs="0" type="xs:double" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Następnie wchodzimy we właściwości źródła danych (właściwości procesu, zakładka Data – zaznaczamy XML Data Source i klikamy „Edit...”)

Modify Data Source

Type:  
XML Data Source

Label:  
we

Description:

Fileref:  
we  Allow rewinding stream

Expected content type:  
text/xml

Specify schema

Schema URI:  
file:///C:\webservice\class.xsd

Reference namespace:  
http://www.tempuri.org/xml/namespace/class

Reference name:  
TABLE

Reference type:  
 Schema element  Schema type

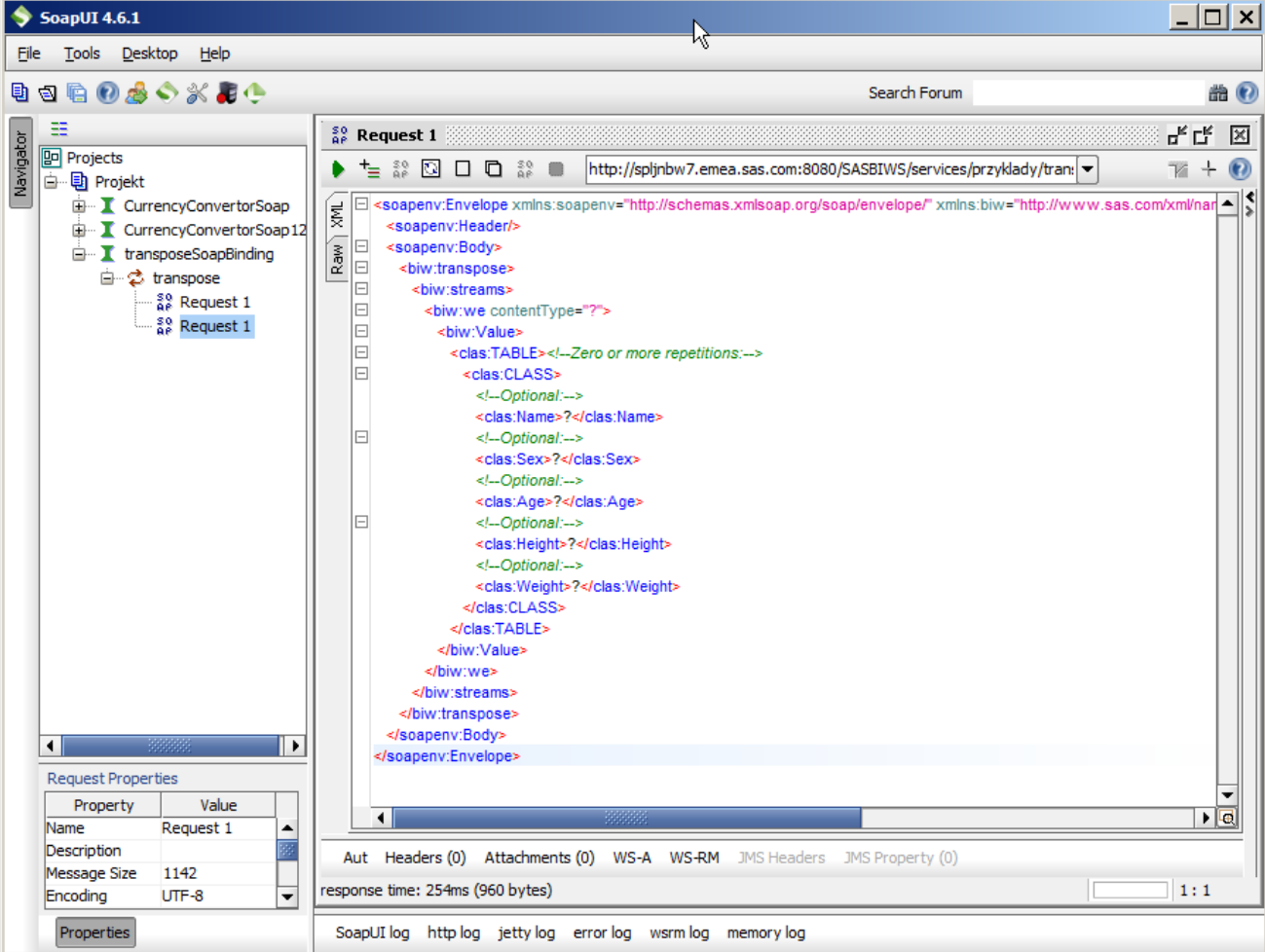
WSDL generation options:  
 Embedded  Referenced

OK Cancel Help

- Zaznaczamy opcję „Specify schema”
- W polu „Schema URI” podajemy lokalizację pliku xsd w następujący sposób:  
<file:///C:\webservice\class.xsd>
- W polu „Reference namespace” podajemy wartość taką samą, jaką wprowadziliśmy wcześniej do wygenerowanego schematu  
<http://www.tempuri.org/xml/namespace/class>
- W polu Reference name wpisujemy „TABLE”
- Jako Reference type wybieramy „Schema element”
- Jako WSDL generation options wybieramy „Embedded”

Teraz możemy otworzyć w przeglądarce link

<http://spljnbw7.emea.sas.com:8080/SASBIWS/services/przyklady/transpose?wsdl&reload=yes> i widzimy, że zawiera on informację o strukturze danych wejściowych. Informacja ta może być wykorzystana przez aplikację, na przykład SoapUI.



The screenshot displays the SoapUI 4.6.1 application window. The main area shows a SOAP request for a service at `http://spljnbw7.emea.sas.com:8080/SASBIWS/services/przyklady/transpose`. The request is in XML format, showing a `soapenv:Envelope` containing a `soapenv:Header` and a `soapenv:Body`. The body contains a `biw:transpose` element with a `biw:streams` sub-element. The `biw:we` element has a `contentType` attribute and a `Value` element. The `Value` element contains a `TABLE` with a `CLASS` attribute and several optional elements: `Name`, `Sex`, `Age`, `Height`, and `Weight`. The `CLASS` attribute is set to `TABLE`. The `Value` element is enclosed in a `biw:we` element, which is enclosed in a `biw:streams` element, which is enclosed in a `biw:transpose` element, which is enclosed in a `soapenv:Body` element, which is enclosed in a `soapenv:Envelope` element.

The interface also shows a Navigator on the left with a tree view of projects and requests. The 'Request Properties' table is visible at the bottom left:

Property	Value
Name	Request 1
Description	
Message Size	1142
Encoding	UTF-8

At the bottom of the window, there are logs for SoapUI, http, jetty, error, wsrm, and memory. The status bar shows 'response time: 254ms (960 bytes)' and '1 : 1'.