Exchange of data over internet using web services (e.g., SOAP and REST) in SAS environment

Kevin Lee, Accenture Accelerated Research & Development Services, Berwyn, PA

ABSTRACT

We are living in the world of abundant information, and the ability to seamlessly exchange information between customers, partners and internal business units is vital for success for any organization. Today, much of the information can be accessed and exchanged between different systems over the internet using web services. Web services allow different systems to exchange data over internet. The paper will show how SAS® can exchange the data with the different software system over internet using web services.

The paper will introduce the basic concepts of web service and its method of communication: SOAP(Simple Object Access protocol) and REST(Representational state transfer). First, the paper will briefly describe SOAP and its structure – HTTP header and SOAP envelop. The paper will show the examples of how SAS programmers send a request to the web service and receive the response from the web service using SOAP in SAS environment. The paper will also show how SAS programmers can create a SOAP request using SOAPUI, the open-source software which allows the users to create SOAP and test the connectivity with the web service. The paper will explain how FILENAME and SOAPWEB function send SOAP request file and receive response file. The paper will also explain the structure of SOAP response file in XML.

The paper will show the structure of REST, and it will instruct how SAS programmers write SAS codes to get the data from other system using REST. The paper will introduce SAS FILEMANE, its url and debug options.

INTRODUCTION OF WEB SERVICE

A web service is a method of communication that allows two software systems to exchange the data over the internet. Two primary architectures for web services are SOAP and REST.

INTRODUCTION OF SOAP

Simple Object Access protocol (SOAP) is a protocol specification for data exchange in web services. It is platform, system and language independent and communicates through the internet. It uses XML format and Hypertext Transfer Protocol (HTTP). It sends request files in XML and receives response file in XML. Since HTTP is supported by all internet browsers and servers, SOAP provides a way to communicate between applications running on different systems, technologies and languages.

It has three elements.

1. An envelope element that identifies the XML document as a SOAP message.
2. A header element that contains header information.
3. A body element that contains call and response information.

INTRODUCTION OF REST

Representational state transfer (REST) is a simpler data exchange format than SOAP data exchange. It is also platform, system and language independent and communicates through the internet. It also uses HTTP, but unlike SOAP, response files come ready to be used, not wrapped in SOAP envelope. So, REST does not need to use XML format to send and receive data through web services. In the examples of REST response files, programmers will see how request and response files from REST can be different from those of SOAP.

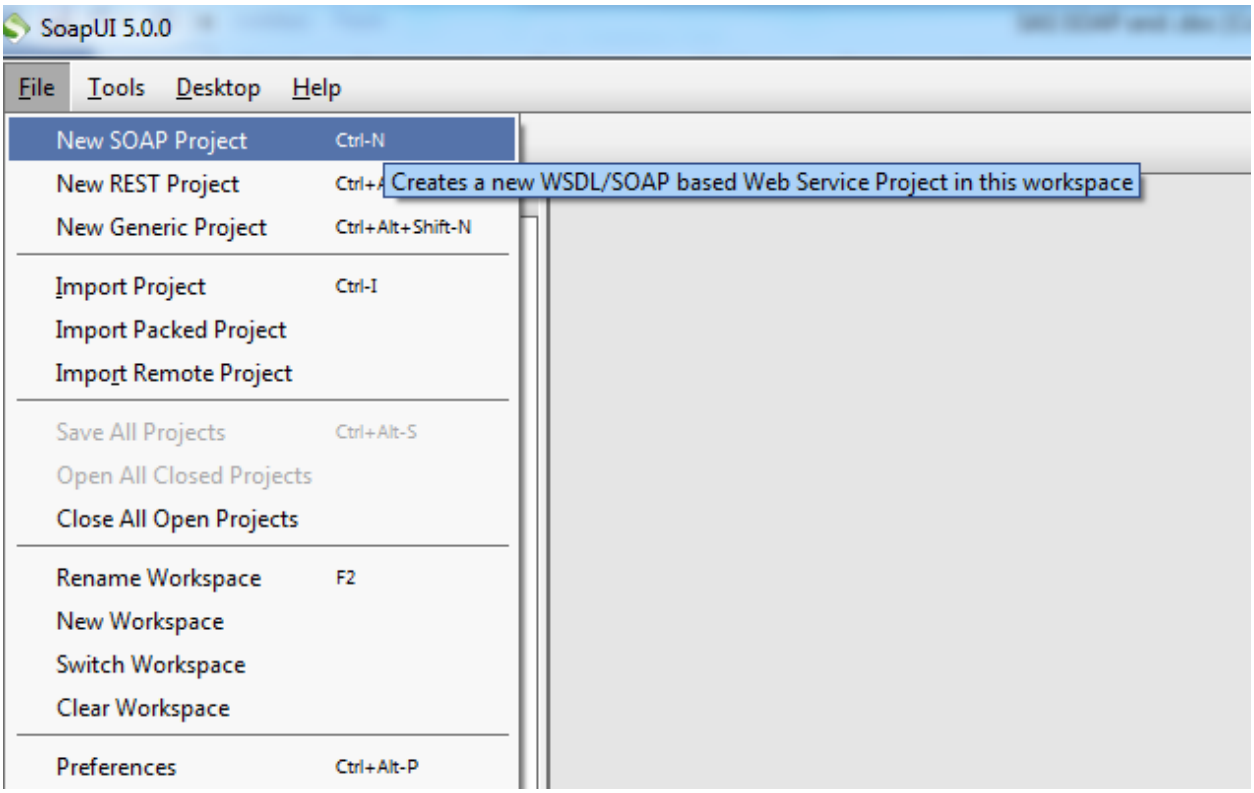
HOW TO OBTAIN DATA USING SOAP

The paper will introduce two case studies and provide step by step instructions of how the programmers can receive the data using SOAP and convert it to SAS data sets for further analysis and reporting.

CASE STUDY 1 – GETTING THE LIST OF CITIES

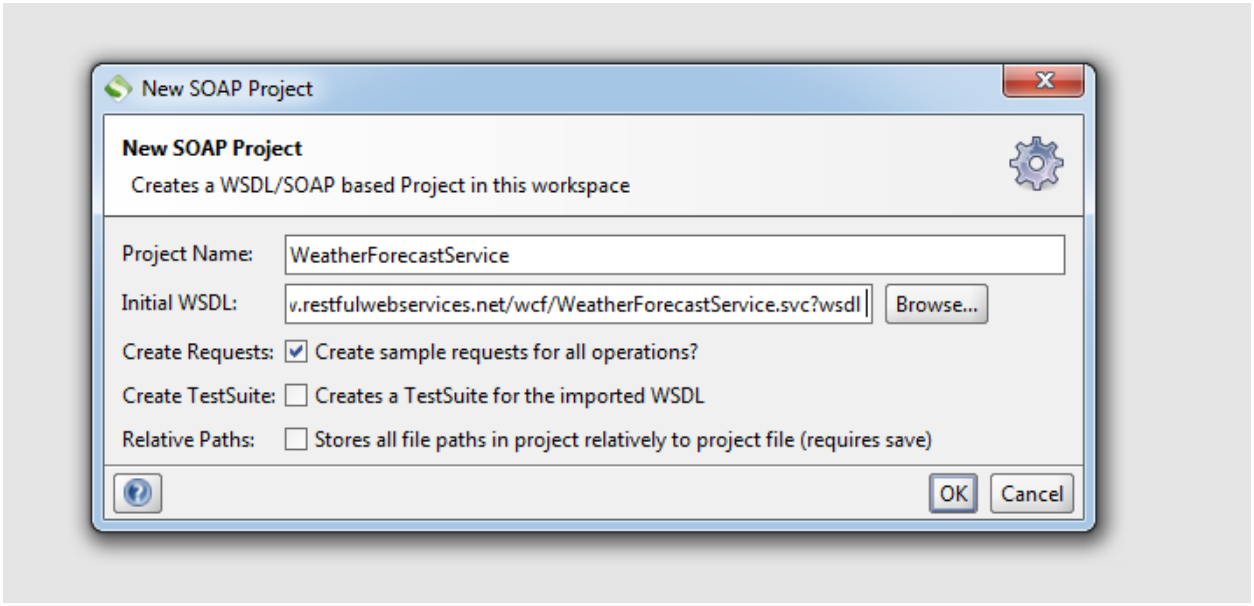
HOW TO CREATE SOAP USING SOAPUI

SOAPUI is the free software package that helps the programmers to develop SOAP request. The programmers can use SOAPUI to test SOAP API, create SOAP request file and receive SOAP response file.
In order to create SOAP request file, programmers open SOAPUI software and go to “File” and create “New SOAP Project” as shown in Display 1.



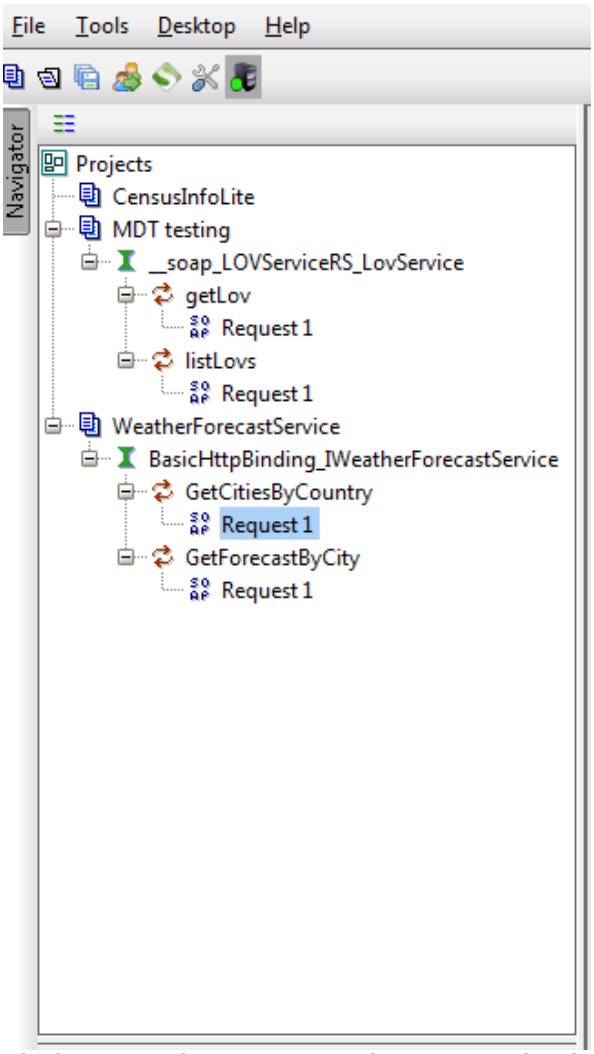
Display 1: Open new SOAP project in SOAPUI.

Programmers can add WDSL to get the necessary information on web services as shown in Display 2. Web Service Description Language (WSDL) is an XML-based interface definition language that is used for describing the functionality offered by a web service of system.



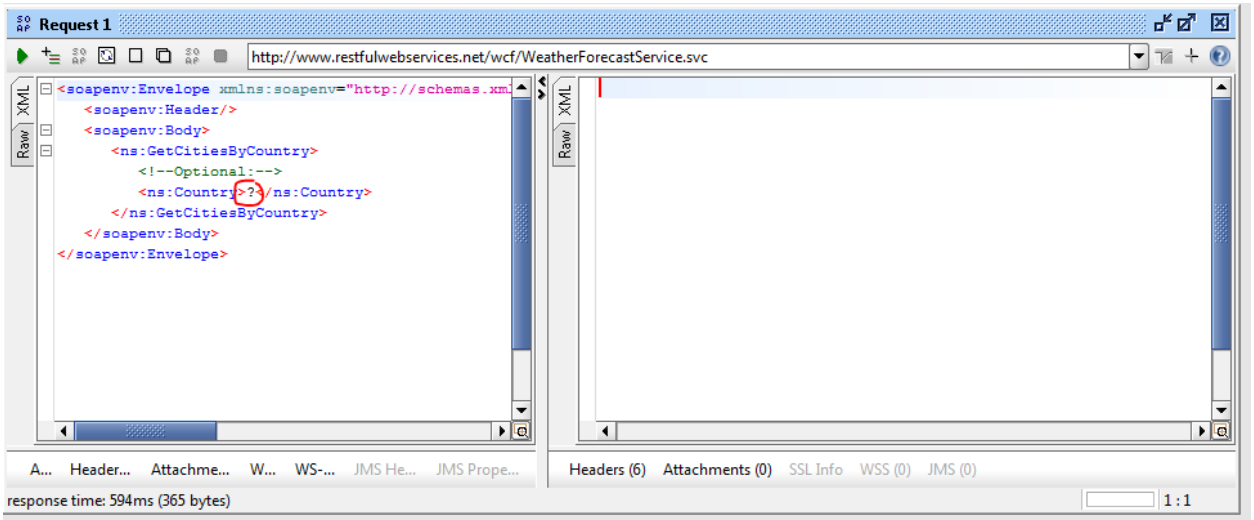
Display 2: provide WSDL to new SOAP project in SOAPUI.

As shwon in Display 3, programmers see that new project of “WeatherForecastService” is created along with GetCitiesByCountry and GetForecastByCity objects.



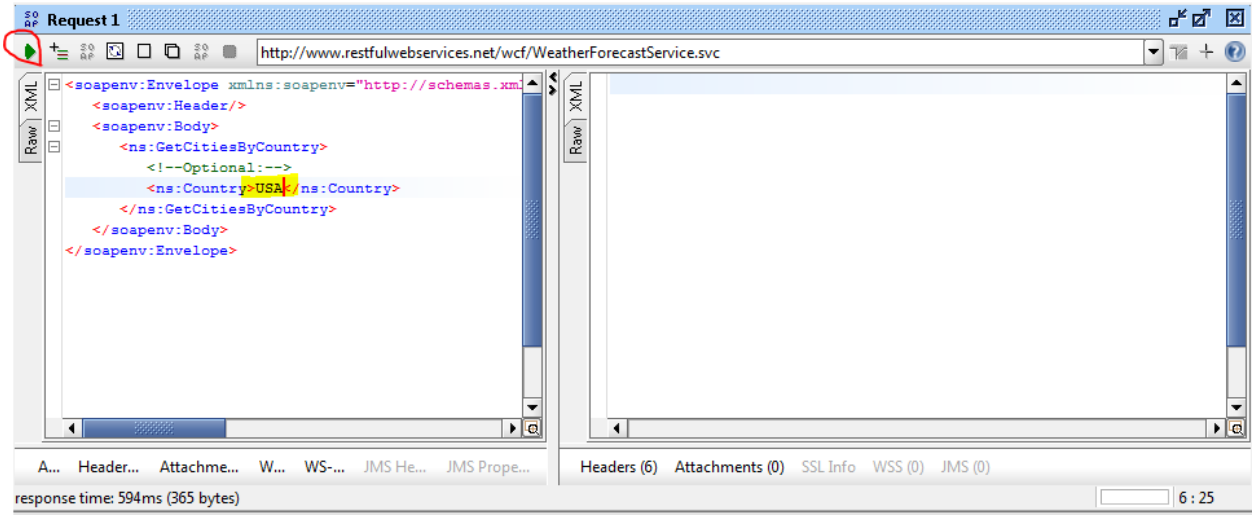
Display 3: WeatherForecastService SOAP project in SOAPUI.

Once “Request 1” object is clicked as highlighted in Display 3, programmers will have a window of “Request 1”. This is SOAP request for GetCitiesByCountry command. “?” red-highlighted in Display 4 is a parameter input on this SOAP request file.



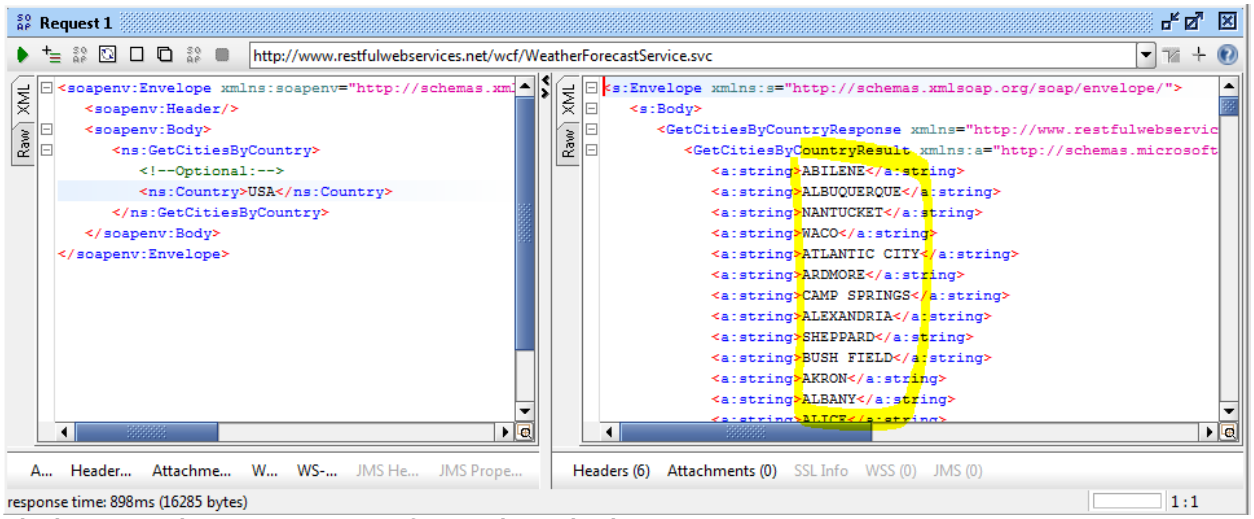
Display 4: Request window of WeatherForecastService SOAP project in SOAPUI.

After adding “USA” as shown in Display 5 and clicking on the green process button as shown in Display 5, the programmers will receive SOAP response files shown in Display 6.



Display 5: Add the parameter to SOAP request of WeatherForecastService SOAP project in SOAPUI.

The programmers will receive SOAP response file as yellow-highlighted in Display 6.



Display 6: Receive SOAP response from web service in SOAPUI.

SAS programmers will be also able to send SOAP request file and receive SOAP response file.

HOW TO USE SOAP IN SAS

SAS programmers can send SOAP request file that is created in SOAPUI. The first part of Code 1 is obtained from RAW tab in SOAPUI and is HTTP. The second part is obtained from XML tab in SOAPUI and is a SOAP request.

```
POST http://www.restfulwebservices.net/wcf/WeatherForecastService.svc HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "GetCitiesByCountry"
Content-Length: 357
Host: www.restfulwebservices.net
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)

<soapenv:Envelope  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://www.restfulwebservices.net/ServiceContracts/2008/01">
```

```
  <soapenv:Header/>
  <soapenv:Body>
    <ns:GetCitiesByCountry>
      <!--Optional:-->
      <ns:Country>USA</ns:Country>
    </ns:GetCitiesByCountry>
  </soapenv:Body>
</soapenv:Envelope>
```

Code 1: Full SOAP request of WeatherForecastService SOAP project in SOAPUI.

Upon Full SOAP request shown in Code 1, SAS programmers can create SAS codes shown in Code 2 and save REQUEST XML files shown in Code 3.

```
*********************************************************************************
*  Send saved SOAP request file and receive SOAP response file.
*********************************************************************************;
FILENAME request 'T:\KL\Request.xml' ;
FILENAME response 'T:\KL\Response.xml';

DATA _NULL_;
      ** URL is from SOAP Header in POST;
      url="http://www.restfulwebservices.net/wcf/WeatherForecastService.svc";
      ** SOAPACTION is from SOAP Header in POST;
      SOAPACTION='GetCitiesByCountry';
      rc=SOAPWEB("request",url,"response",soapaction,,,,,,);
RUN;
```

Code 2: SAS codes for SOAP request.

SAS programmers save SOAP request xml file shown in Code 3 in local drive ('T:\KL\Request.xml') and send SOAP request file to web service in URL of "http://www.restfulwebservices.net/wcf/WeatherForecastService.svc" as a call of 'GetCitiesByCountry'. In Code 2, SOAPWEB function will contain all the necessary information:

- Request: SOAP request xml file
- URL: address of web service
- Response: SOAP response xml file
- Soapaction: call method

SOAPWEB function sends SOAP request xml files to URL address of web service of different system through the internet, calls the method of GetCitiesByCountry to the system and gets SOAP response file. SAS programmers also put the parameter values in request.xml; for example, SAS programmers can write the codes to put parameter values in <Country> tag in Code 3.

```
<soapenv:Envelope  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://www.restfulwebservices.net/ServiceContracts/2008/01">
  <soapenv:Header/>
  <soapenv:Body>
    <ns:GetCitiesByCountry>
      <!--Optional:-->
      <ns:Country>USA</ns:Country>
    </ns:GetCitiesByCountry>
  </soapenv:Body>
</soapenv:Envelope>
```

Code 3: REQUEST.xml

The successful run of SOAPWEB can receive a response file in Code 4. In Code 4, SAS programmers receive the list of cities. SAS programmers now can convert xml files to SAS datasets for further analysis or transformation.

```
<?xml version="1.0"?>
-<GetCitiesByCountryResponse xmlns="http://www.restfulwebservices.net/ServiceContracts/2008/01">
        -<GetCitiesByCountryResult xmlns:a=http://schemas.microsoft.com/2003/10/Serialization/Arrays
        xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
                <a:string>ABILENE</a:string>
                <a:string>ALBUQUERQUE</a:string>
                <a:string>NANTUCKET</a:string>
                <a:string>WACO</a:string>
                ……
                <a:string>SOUTH HAWAII MET</a:string>
```

```
                <a:string>HICKAM</a:string>
                <a:string>HONOLULU ARTCC</a:string>
                <a:string>OLIKTOK POINT</a:string>
                <a:string>POINT LAY</a:string>
            </GetCitiesByCountryResult>
</GetCitiesByCountryResponse>
```

Code 4: RESPONSE.xml

The successful run of SOAPWEB can receive a response file in Code 4. In Code 4, the programmers receive the list of cities.

The SAS programmer can use SAS XML mapper to map xml file to SAS datasets. In SAS 9.3, SAS programmers create XMLMap file using XMLV2 LIBNAME engine. XMLMap file can create SAS datasets from a hierarchical XML file. The Code 5 will show how SAS programmers can create XMLMap file.

```
*********************************************************************************
*  Create XMLMap file and create SAS datasets from XML file
*********************************************************************************;
** SOAP response xml files;
filename  myresp "T:\KL\Response.xml";

**** Create response xml map file;
filename  respmap " T:\KL\response.map";
libname myresp xmlv2 xmlmap=respmap automap=replace;

**** Convert SOAP response xml files to SAS temporary dataset in work area;
proc copy in=myresp out=work;
run;
```

Code 5: SAS codes that can create XMLMap file and also convert XML file to SAS datasets using XMLMap file.

In SAS working library, SAS dataset of "String" is created, and it has variables and contents as indicated in Table 1.

GetCitiesByCountryResult_ORDINAL	String_ORDINAL	string
1	1	ABILENE
1	2	ALBUQUERQUE
1	3	NANTUCKET
1	4	WACO
….		

Table 1: SAS dataset of String

HIGH LEVEL DESIGN

Figure 1 shows the high level design between SAS and external system over internet using SOAP. It shows how SAS programmers can build the integration of SAS and external system using SOAP. A SOAP request was sent to web service of external system by SAS. SAS will receive SOAP response file in XML and convert XML to SAS dataset for analysis and reporting.

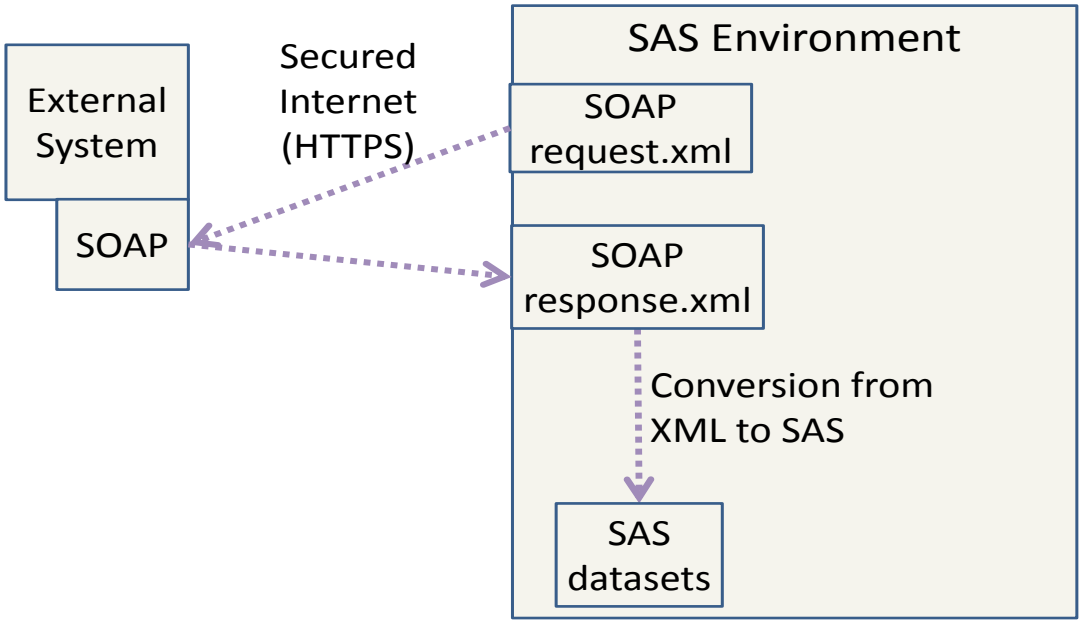
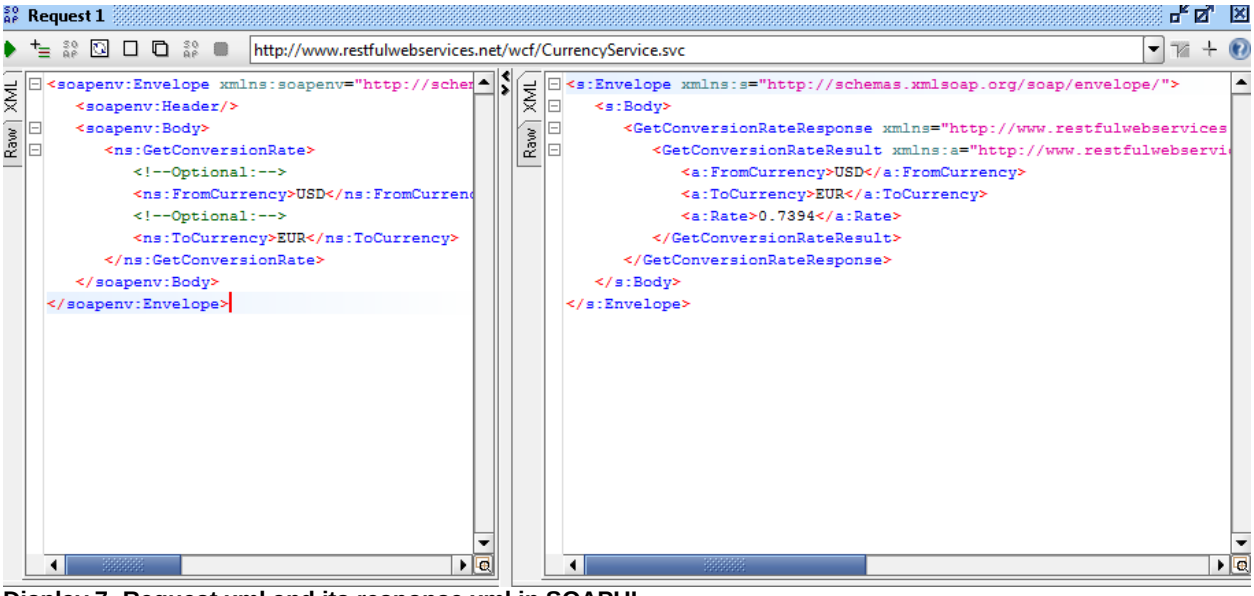


Figure 1. High Level Design of integration of SAS and external system over internet using SOAP

CASE STUDY 2 – GETTING THE CURRENTY CONVERSION RATE

If adding WSDL of http://www.restfulwebservices.net/wcf/CurrencyService.svc?wsdl in SOAPUI, the programmers can create SOAP request file. In Display 7, the programmer added USD in <FromCurrency> tag and EUR in <ToCurrency> tag in request.xml. Once the request file was sent to web service through SOAPUI, the programmer received 0.7334 as Rate.



Display 7: Request.xml and its response.xml in SOAPUI

The SAS programmers can create SAS codes shown in Code 6 that send “currency request.xml” in Code 7 to obtain the currency rate of “currency response.xml” in Code 8.

```
*** Request2 - this works;
FILENAME request 'T:\KL\webservice\currency request.xml' ;
FILENAME response 'T:\KL\webservice\currency response.xml';
```

```
DATA _NULL_;
     url="http://www.restfulwebservices.net/wcf/CurrencyService.svc";
     SOAPACTION='GetConversionRate';
     rc=SOAPWEB("request",url,"response",soapaction,,,,,,,);
RUN;
```
Code 6: SAS codes for currency exchange.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://www.restfulwebservices.net/ServiceContracts/2008/01">
  <soapenv:Header/>
  <soapenv:Body>
    <ns:GetConversionRate>
      <!--Optional:-->
      <ns:FromCurrency>USD</ns:FromCurrency>
      <!--Optional:-->
      <ns:ToCurrency>EUR</ns:ToCurrency>
    </ns:GetConversionRate>
  </soapenv:Body>
</soapenv:Envelope>
```
Code 7: currency request.xml

```
<GetConversionRateResponse xmlns="http://www.restfulwebservices.net/ServiceContracts/2008/01">
<GetConversionRateResult xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:a="http://www.restfulwebservices.net/DataContracts/2008/01">
 <a:FromCurrency>USD</a:FromCurrency>
 <a:ToCurrency>EUR</a:ToCurrency>
 <a:Rate>0.7394</a:Rate>
</GetConversionRateResult>
</GetConversionRateResponse>
```
Code 8: currency response.xml

The SAS programmers can convert currency response xml file to SAS datasets as shown in Code 5.

HOW TO OBTAIN DATA USING REST

Unlike SOAP-based web service, REST does not require request xml file to receive the data from web service. REST simply sends request through HTTP with parameters in it.

In Code 9, SAS programmers will call yahoo finance REST web service to receive the data of apple stock price from 11/29/2004 to 01/01/2005. SAS statement FILENAME and its option URL can set up REST communication. The URL up to ?(`http://ichart.finance.yahoo.com/table.csv)` is the path to web service and URL after ? (`s=AAPL&a=11&b=29&c=2004&d=01&e=1&f=2005&g=d&ignore=.csv`) is parameters and their inputs to web service. The URL will call daily quotes of apple stocks from 11-29-2004 to 01-01-2005 and SAS dataset of price can contain its information.

```
*** FILENAME and URL will call REST;
filename price url
'http://ichart.finance.yahoo.com/table.csv?s=AAPL&a=11&b=29&c=2004&d=01&e=1&f=2005&g=d
&ignore=.csv' debug;
*** SAS dataset of price will contain daily stock quotes of Apple stock;
data price;
      infile price length=len;
      input record $varying200. len;
run;
```
Code 9: SAS codes for REST

SAS programmers can also use PROC HTTP to receive Apple daily quotes in CSV file format. SAS codes in Code 10 will show how to call REST using PROC HTTP and save CSV response files to local drive.

```
*** Create csv file in local drive ;
filename out "&local_url.\resp-stock-appl.csv";  ** will get response output;
*** URL option in PROC HTTP send a GET request to Yahoo Finance REST;
proc http
    out=out
```

```
    url="http://ichart.finance.yahoo.com/table.csv?s=AAPL&a=11&b=29&c=2004&d=01&e=1&f=
2005&g=d&ignore=.csv"
    method="GET";
run;
```

Code 10: SAS codes for REST

Code 11 shows how SAS programmers can also receive CDASH Control Terminology from NCI website.

```
*** Codes will receive CDASH CT from NCI URL to local drive;
filename cdash "G:\KL\web connection\URL connection\CDASH Terminology.xls";
**** This gives the exact same results;
proc http out=cdash
url="http://evs.nci.nih.gov/ftp1/CDISC/SDTM/CDASH%20Terminology.xls"

method="get"
;
run;
```

Code 11: SAS codes to receive CDASH Control Terminology

CONCLUSION

The ability to exchange data and information using web services over the internet will be extremely useful and powerful. SAS introduces web services related SAS functions, statements and options such as SOAPWEB, FILENAME URL, PROC HTTP, PROC SOAP and LIBNAME XMLV2. Using SAS web service codes, SAS programmers are able to receive real-time data over internet into SAS environment for reporting and analysis. SAS programmers can also build data and system integration over the internet.

REFERENCES

- The SAS programmer's guide to XML and Web Services, Chirstopher W. Schacherer.
- The Ins and Outs of Web-Based Data with SAS, Bill McNeill
- A simple way of importing from A REST Web Service into SAS in Three Lines of Code, Philip Busby
- Extreme Web Access: What to do when FILENAME URL is not enough, Garth Helf.
- Using Base SAS to talk to the outside world: consuming SOAP and REST Web Services using SAS 9.1 and the new features of SAS 9.2, Curtis E. Mack.
- SAS XML LIBNAME Engine: User's guide
- SAS PROC HTTP: User's guide
- SAS SOAPWEB FUNCTION: User's guide
- SOAPUI User's guide
- World Wide Web Consortium. SOAP Version 1.2

CONTACT INFORMATION

Your comments and questions are valued and welcomed. Please contact the author at

Kevin Lee
Accenture Life Sciences
Kevin.s.lee@accenture.com
610-407-1767

TRADEMARKS