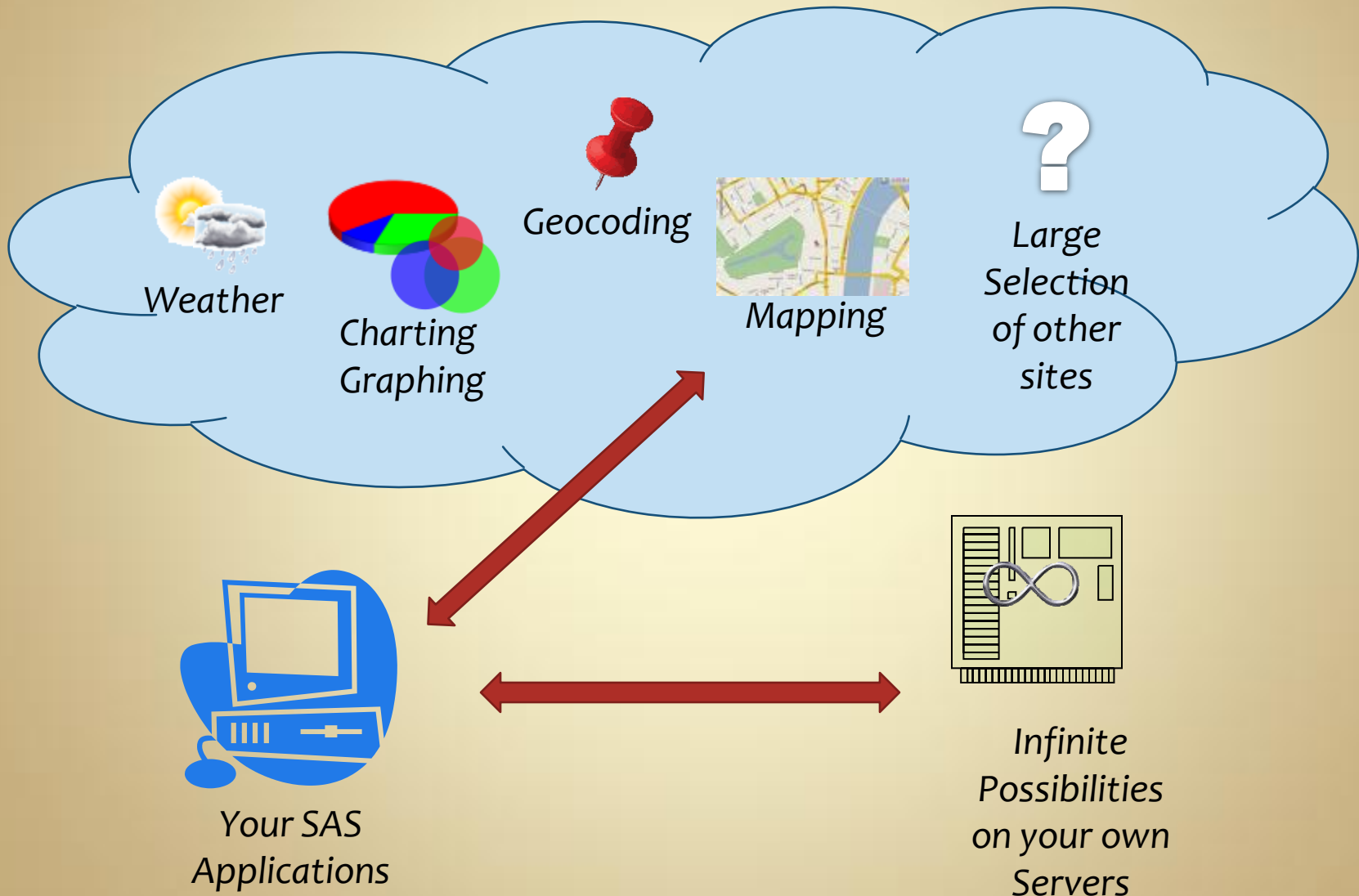


# Consuming SOAP and REST Web Services from Base SAS.

Curtis Mack  
Curtis.Mack@lgan.com  
Looking Glass Analytics  
[www.LGAN.com](http://www.LGAN.com)

# Web Services



# Two Main Types of Web Services

- SOAP

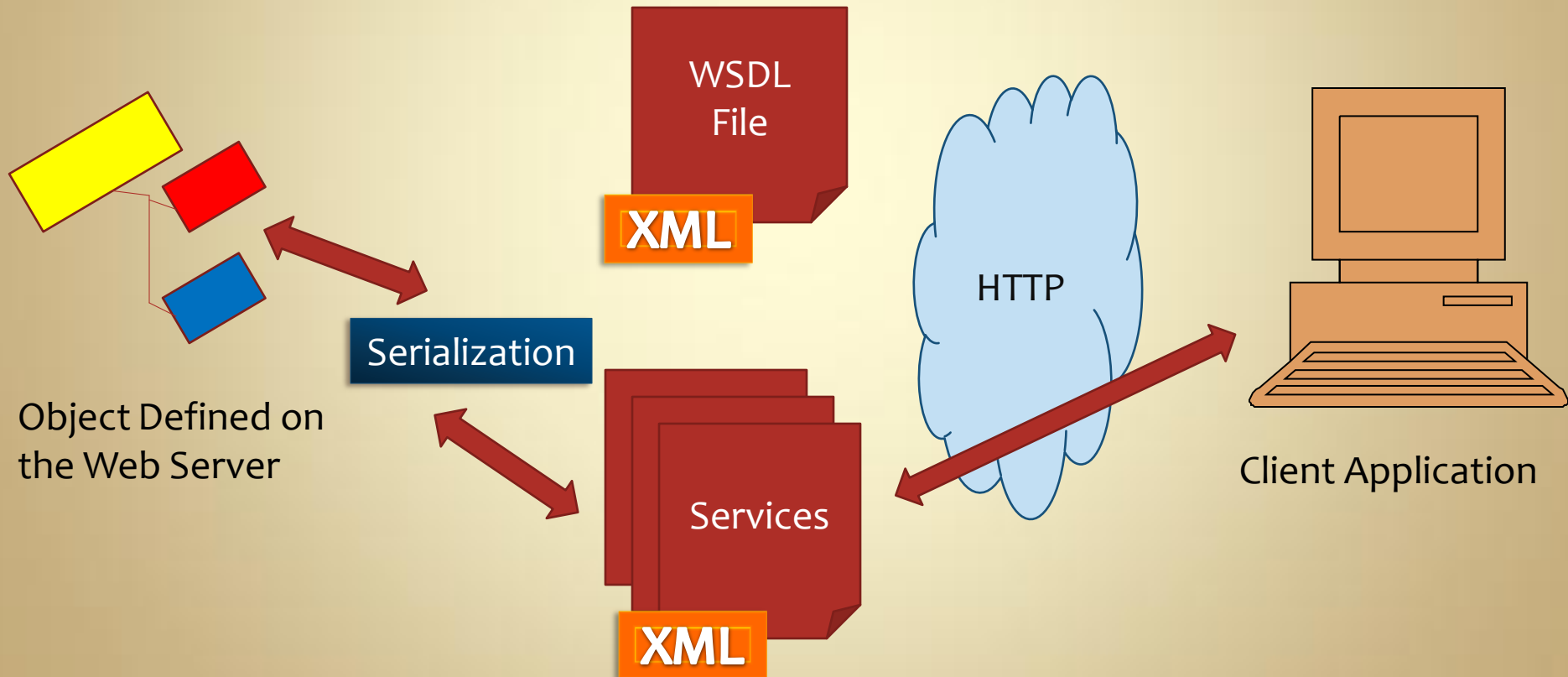
- Object based
- More structured
- Well integrated into code development tools.
- More powerful
- Harder to code

- REST

- Parameter Based
- Less Structured
- More Human Readable
- Easier to code
- Has become more popular

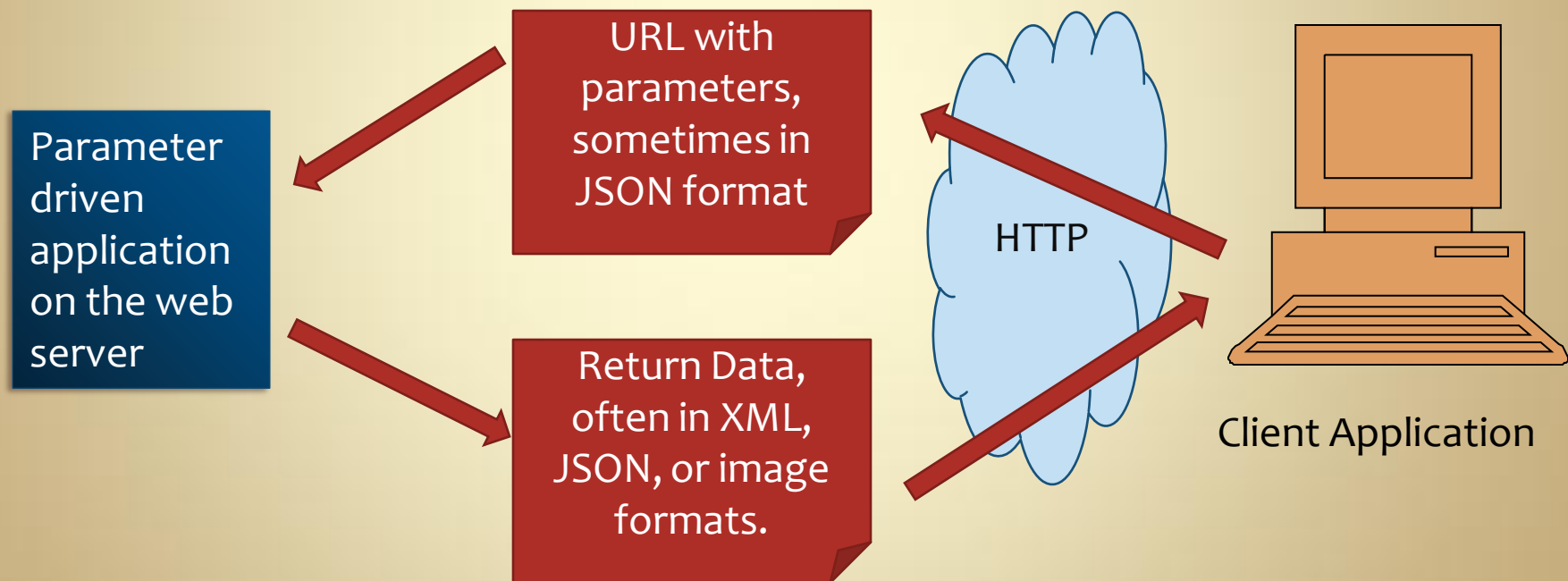
# The SOAP Protocol

- Simple Object Access Protocol (Quick Overview)



# The REST Protocol

- Representational State Transfer (Quick Overview)



# Working With SOAP Services

- Get the WSDL file from the service
  - <http://www.service.com/service.ext?WSDL>
- Select the desired action
- Find the definition for the object the action request expects
- Find the definition for the object the service returns.

# Typical WSDL File



These SOAP examples are using services from the company Strike Iron.

Their free CensusLite service is used in this demonstration.

[www.StrikeIron.com](http://www.StrikeIron.com)

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<-wsdl:definitions xmlns:soap=
```

```
-xmlns:tm="http://mi
```

```
-xmlns:soapenc="http
```

```
-xmlns:mime="http://
```

```
-xmlns:tns="http://ww
```

```
-xmlns:s="http://www.
```

```
-xmlns:soap12="http://
```

```
-xmlns:http="http://s
```

```
-targetNamespace="h
```

```
-xmlns:wsdl="http://s
```

```
<wsdl:documentation xmlns
```

```
Bureau's Census 2000. Provid
```

```
</wsdl:documentation>
```

```
<wsdl:types>
```

```
<s:schema elementFo
```

```
<s:element na
```

```
<s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

```
</s:cor
```

ation from the U.S. Census  
Code.

ring" />

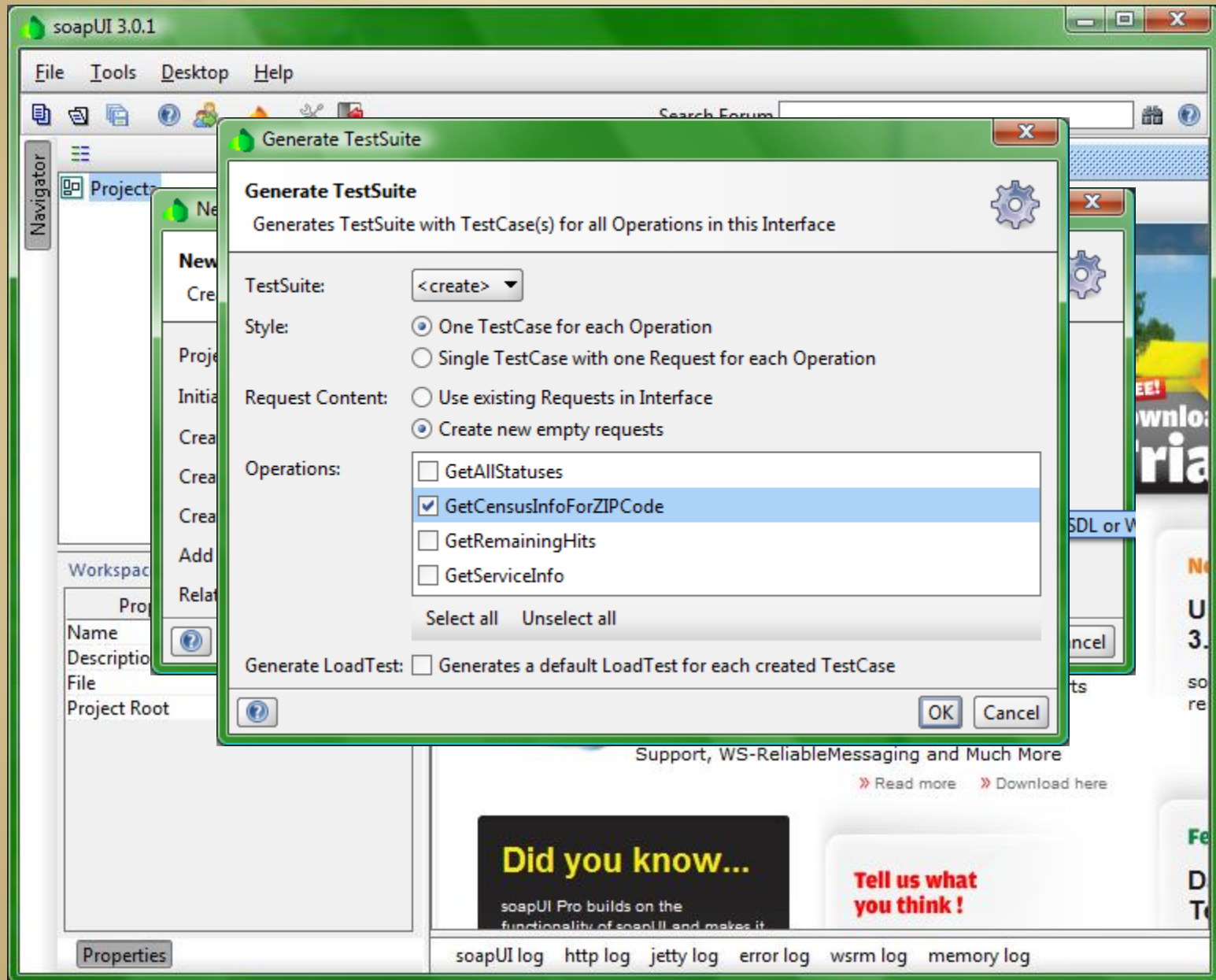
# SoapUI



Free open source tool for reading WSDL files and generating sample SOAP calls

<http://www.soapui.org/>

# Using SoapUI to Build a Soap Call



# The Resulting Soap Call

The screenshot displays the soapUI 3.0.1 interface. On the left, the Navigator pane shows a project tree with 'CensusInfoLiteSoap' as the main item, containing several test cases and a test suite. The 'TestSteps (1)' for 'GetCensusInfoForZIPCode' is selected. Below the navigator, the 'TestRequest Properties' tab is active, showing a table of properties for the selected test step.

| Property         | Value                  |
|------------------|------------------------|
| Name             | GetCensusInfoFor...    |
| Description      |                        |
| Message Size     | 344                    |
| Encoding         | UTF-8                  |
| Endpoint         | http://wslite.strik... |
| Bind Address     |                        |
| Follow Redirects | false                  |
| Interface        | CensusInfoLiteSoap     |
| Operation        | GetCensusInfoFor...    |
| Username         |                        |
| Password         |                        |
| Domain           |                        |

The main pane shows the raw XML response for the 'GetCensusInfoForZIPCode' operation. The URL bar indicates the endpoint: `http://wslite.strikeiron.com/censusinfolite01/CensusInfo...`. The raw XML content is as follows:

```
POST http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/xml; charset=UTF-8
SOAPAction: "http://www.wslite.strikeiron.com/GetCensusInfoForZIPCode"
User-Agent: Jakarta Commons-HttpClient/3.1
Host: wslite.strikeiron.com
Content-Length: 344

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <wsl:GetCensusInfoForZIPCode>
      <!-- Optional: -->
      <wsl:ZIPCode>?</wsl:ZIPCode>
    </wsl:GetCensusInfoForZIPCode>
  </soapenv:Body>
</soapenv:Envelope>
```

At the bottom of the main pane, the 'Request Log (1)' tab is active, showing the response time: 579ms (633 bytes). The bottom status bar includes links to various logs: soapUI log, http log, jetty log, error log, wsrn log, and memory log.

# Populating the Body

The screenshot displays the soapUI 3.0.1 application window. The left sidebar contains a 'Navigator' pane with a tree view of test artifacts, including 'CensusInfoLiteSoap', 'GetCensusInfoForZIPCode', and 'Test Steps (1)'. The 'Test Properties' pane at the bottom left shows details for the selected test step, such as 'Name', 'Description', 'Message Size', 'Encoding', 'Endpoint', and 'Operation'. The main workspace shows the 'GetCensusInfoForZIPCode' test step selected, with its URL 'http://wslite.strikeiron.com/censusinfolite01/CensusInfo...' and a 'Raw' XML view. The XML body is a SOAP envelope containing a 'ws1:ZIPCode' element with the value '98501'. The bottom status bar shows 'response time: 1169ms (5528 bytes)' and a list of log files.

soapUI 3.0.1

File Tools Desktop Help

Search Forum

Navigator

- CensusInfoLiteSoap
  - GetAllStatuses
  - GetCensusInfoForZIPCode
    - Request 1
  - GetRemainingHits
  - GetServiceInfo
- CensusInfoLiteSoap12 TestSuite
- CensusInfoLiteSoap TestSuite
  - GetCensusInfoForZIPCode TestCase
    - Test Steps (1)
      - GetCensusInfoForZIPCode
    - Load Tests (0)

TestRequest Properties Test Properties

| Property         | Value                  |
|------------------|------------------------|
| Name             | GetCensusInfoFor...    |
| Description      |                        |
| Message Size     | 348                    |
| Encoding         | UTF-8                  |
| Endpoint         | http://wslite.strik... |
| Bind Address     |                        |
| Follow Redirects | false                  |
| Interface        | CensusInfoLiteSoap     |
| Operation        | GetCensusInfoFor...    |
| Username         |                        |
| Password         |                        |
| Domain           |                        |

Properties

GetCensusInfoForZIPCode

http://wslite.strikeiron.com/censusinfolite01/CensusInfo...

XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <ws1:GetCensusInfoForZIPCode>
      <!--Optional:-->
      <ws1:ZIPCode>98501</ws1:ZIPCode>
    </ws1:GetCensusInfoForZIPCode>
  </soapenv:Body>
</soapenv:Envelope>
```

Aut Headers (0) Attachments (0) WS-A WS-RM

Assertions (0) Request Log (2)

response time: 1169ms (5528 bytes)

soapUI log http log jetty log error log wsrn log memory log

# An Example of the Returned XML

The screenshot displays the soapUI 3.0.1 interface. The left sidebar shows a project tree with 'CensusInfoLiteSoap' and a 'Test Steps' section containing 'GetCensusInfoForZIPCode'. The main window shows the 'Raw' XML response for this test. The XML is a SOAP envelope with a body containing a 'GetCensusInfoForZIPCodeResponse' message. The response includes a 'ServiceStatus' (211), a 'StatusDescription' ('Census Information Found'), 'RemainingHits' (99), and a 'ServiceResult' with a 'Count' of 1 and a 'CensusInfo' object. The 'CensusInfo' object contains various demographic data for ZIP code 98501.

**Test Properties**

| Property     | Value       |
|--------------|-------------|
| Name         | GetCens...  |
| Descripti... |             |
| Message...   | 348         |
| Encoding     | UTF-8       |
| Endpoint     | http://w... |
| Bind Ad...   |             |
| Follow R...  | false       |
| Interface    | CensusI...  |
| Operation    | GetCens...  |
| Username     |             |

**TestRequest Properties**

response time: 1169ms (5528 bytes) 1:1

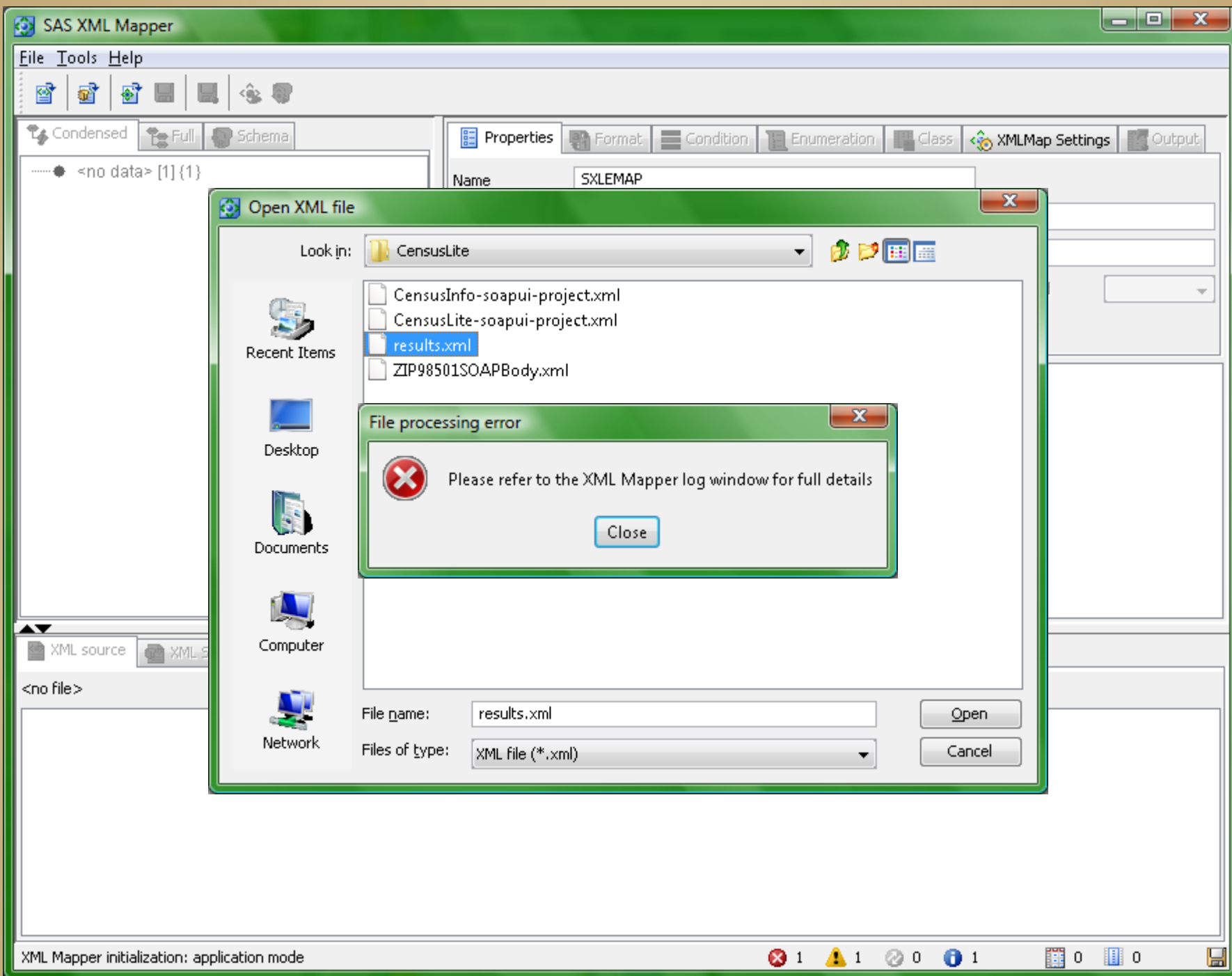
soapUI log http log jetty log error log wsrm log memory log

# Working with the Returned XML

- Could write code to parse the results
- SAS XML Mapper, is a much better

## Approach

- GUI interface that reads an existing XML file
- The user selects the desired data elements
- Creates a .MAP file mapping XML structures into a rectangular format that can be stored in a SAS dataset



# Simplify the XML for Mapper

- XML Mapper occasionally fails when reading complicated XML structures
  - Edit the XML file in a text editor
  - Remove unneeded branches
  - Maintain the Hierarchy of the needed information
  - Open the edited file in XML Mapper to create the XML Map.
  - May need to edit more than once to get difference data elements out of the same XML file

SAS XML Mapper

File Tools Help

Condensed Full Schema

soap:Envelope [1]{1}

- soap:Body [1]{1}
- GetCensusInfoForZIPCodeResponse [1]{1}
- GetCensusInfoForZIPCodeResult [1]{1}
- ServiceStatus [1]{1}
- RemainingHits [1]{1}
- ServiceResult [1]{1}
- Count [1]{1}
- CensusInfo [1]{1}
- CensusInformation [1]{1}
- ZIPCode [1]{1}
- Total\_pop [1]{1}
- Male\_pop [1]{1}
- Female\_pop [1]{1}
- Age\_Under\_5\_pop [1]{1}
- Age\_5\_9\_pop [1]{1}
- Age\_10\_14\_pop [1]{1}
- Age\_15\_19\_pop [1]{1}
- Age\_20\_24\_pop [1]{1}

Properties

Name SXLEMAP

Description

Path

End Path Begin/End

☐ Retain ☐ Replace

SXLEMAP

XML source XML Schema source XMLMap SAS Code Example Table view Contents Validate Log

XML: G:\PNWSUG\Papers\Soap\CensusLite\results.xml

nsusInformation></CensusInfo></ServiceResult></GetCensusInfoForZIPCodeResult></GetCensusInfoForZIPCodeResponse></soap:Body></soap:Envelope>

XML file loaded: results.xml

1 1 0 1 0 0

SAS XML Mapper

File Tools Help

Condensed Full Schema

ServiceStatus [1]{1}

RemainingHits [1]{1}

ServiceResult [1]{1}

Count [1]{1}

CensusInfo [1]{1}

CensusInformation

ZIPCode [1]

Total\_pop [1]

Male\_pop [1]

Female\_pop [1]

Age\_Under [1]

Age\_5\_9\_pop [1]{1}

Age\_10\_14\_pop [1]{1}

Age\_15\_19\_pop [1]{1}

Age\_20\_24\_pop [1]{1}

Age\_25\_34\_pop [1]{1}

Age\_35\_44\_pop [1]{1}

Age\_45\_54\_pop [1]{1}

Age\_55\_59\_pop [1]{1}

Get

Expand

Collapse

AutoMap this Branch

Properties

Name: AUTO\_GEN

Description:

Path:

End Path: Begin/End

Retain: Replace

AUTO\_GEN

CensusInformation

XML source XML Schema source XMLMap SAS Code Example Table view Contents Validate Log

XML: G:\PNWSUG\Papers\Soap\CensusLite\results.xml

nsusInformation></CensusInfo></ServiceResult></GetCensusInfoForZIPCodeResult></GetCensusInfoForZIPCodeResponse></soap:Body></soap:Envelope>

Auto generated map

0 0 0 1 1 95

# Making a SOAP Call in SAS 9.1

- Must create code to communicate with server via TCP/IP port 80.
- SOAP\_CALL Macro
  - A relatively simple macro that can handle many SOAP calls
  - Parameters:
    - Host = The URL of the server domain.
    - ServiceLocation = URL of the service
    - SoapAction = The name of the Action, including the namespace.
    - SchemaReference = Any actions or object definitions used in the SOAP body must be prefixed with a namespace alias. This parameter contains the definition of that alias.
    - ContentXML = Name of file containing the XML being sent to the action.
    - OutputXML = Name of the file to which the results will be written.

# The SOAP\_CALL

```
options mprint;  
%let WorkDir = G:\PNWSUG\Papers\Soap\CensusLite;  
  
%SOAP_Call(wslite.strikeiron.com,  
    http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx,  
    http://www.wslite.strikeiron.com/GetCensusInfoForZIPCode,  
    %nrstr(wsl="http://www.wslite.strikeiron.com"),  
    &WorkDir\ZIP98501SOAPBody.xml,  
    &WorkDir\results.xml);  
  
filename CensMap "&WorkDir\CensusInformation.map";  
libname Census xml "&WorkDir\results.xml" xmlmap=CensMap  
    access=READONLY;  
  
Data ZIP98501Census;  
    set Census.Censusinformation;  
run;
```

```

filename SoapSrv socket "wslite.strikeiron.com:80" lrecl=32767 termstr=CRLF;
filename SoapRtrn "G:\PNWSUG\Papers\Soap\CensusLite\results.xml" RECFM=NB;
filename ContXML "G:\PNWSUG\Papers\Soap\CensusLite\ZIP98501SOAPBody.xml";

data _ContXML(drop = TotalXMLLength);
  retain TotalXMLLength 0;
  length content $32767;
  infile ContXML end=xmleof;
  input;
  content = strip(_infile_);
  TotalXMLLength = TotalXMLLength + length(content) + 1;
  if xmleof then call symput('TotalXMLLength',TotalXMLLength);
run;

data _null_;
  length content1 content2 $32767;
  retain mode 1 TotalReturn ReturnLength 0;
  infile SoapSrv truncover;
  file SoapRtrn;
  if _n_ = 1 then do;
    content1 = '<soapenv:Envelope ||
               'xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" || " " ||
               'xmlns:' || "wsl="http://www.wslite.strikeiron.com" || ">" ||
               '<soapenv:Body>';
    content2 = "</soapenv:Body>" || "</soapenv:Envelope>";
    ContentLength = length(content1) + length(content2) + 101 + 4 + int(101 / 32767);
    call symput ('ContentLength',ContentLength);
    put "POST http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx HTTP/1.1" /
        'Content-Type: text/xml; charset=utf-8' /
        'SOAPAction: ""http://www.wslite.strikeiron.com/GetCensusInfoForZIPCode""' /
        'Host: wslite.strikeiron.com' /
        'Content-Length: ' ContentLength /;
    put content1 @;
    currentLineLength = length(content1);
    fc = open("_ContXML");
    do until(fetch(fc));
      thisLine = strip(getvarc(fc,1));
      if sum(length(thisLine),currentLineLength) + 1 >= 32767 then do;
        put;
        currentLineLength = 0;
      end;
      put thisLine@@;
      currentLineLength = sum(length(thisLine),currentLineLength) + 1;
    end;
    rc = close(fc);
    put;
    put content2 ;
    put 'OPTIONS / HTTP/1.1' /
        'Host: http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx' /
        'Connection: Close' /;
  end;

```

```

if mode = 1 then do;
  input thisRec $ 1 - 32767;
  if thisRec =: "Content-Length:" then do;
    ReturnLength = input(scan(thisRec,2,' '),10.);
  end;
  call symput('ReturnLength',ReturnLength);
  if thisRec = " " then mode = 2;
end;
if mode = 2 then do;
  nextReturn = ReturnLength - TotalReturn;
  input thisRec $varying32767. nextReturn;
  file SoapRtrn;
  put thisRec;
  TotalReturn = TotalReturn + length(thisRec);
  if TotalReturn >= ReturnLength then mode = 3;
end;
if mode = 3 then do;
  input;
  end;
run;

```

Code generated by a call  
to the SOAP\_CALL macro.

Some code review.

```
filename SoapSrv socket "wslite.strikeiron.com:80" lrecl=32767 termstr=CRLF;
filename SoapRtrn "G:\PNWSUG\Papers\Soap\CensusLite\results.xml" RECFM=N;
filename ContXML "G:\PNWSUG\Papers\Soap\CensusLite\ZIP98501SOAPBody.xml";
```

```
data _ContXML(drop = TotalXMLLength);
retain TotalXMLLength 0;
length content $32767;
```

Length of the Envelope Start +  
Length of the Envelope End +  
Size of the Body Text +  
End-of-Lines between them +  
End-of-Lines between logical  
records.

Start of port writing Block.  
Define variables containing  
the envelope start and end  
text.  
Calculate the total Bytes.

```
content2 = '</soapenv:Body>';
ContentLength = length(content2);
call symput('ContentLength', ContentLength);
put "POST http://ws.

HTTP/1.1 /
```

```
if _data _ContXML(drop = TotalXMLLength);
retain TotalXMLLength 0;
length content $32767;
infile ContXML end=xmleof;
input;
content = strip(_infile_);
TotalXMLLength = TotalXMLLength + length(content) + 1;
if xmleof then call symput('TotalXMLLength', TotalXMLLength);
run;
```

" || " " ||

```

fc = open("_ContXML");
do until(fetch(fc));
  thisLine = strip(getvarc(fc,1));
  if sum(length(thisLine),currentLineLength) >= 32767 then do;
    put;
    put 'OPTIONS / HTTP/1.1' / "Host:
    http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx" /
    'Connection: Close' /;
  end;
  put thisLine@@;
  currentLineLength = sum(length(thisLine),currentLineLength) + 1;
end;
rc = close(fc);

```

```

call symput('ContentLength',length(content));
put "POST http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx HTTP/1.1" /
'Content-Type: text/xml' /
"SOAPAction: ""http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite"" /
"Host: wslite.strikeiron.com" /
'Content-Length: ' ContentLength /
put content1 @;
currentLineLength = length(content1);
fc = open("_ContXML");
do until(fetch(fc));
  thisLine = strip(getvarc(fc,1));
  if sum(length(thisLine),currentLineLength) >= 32767 then do;
    put;
    currentLineLength = 0;
  end;
  put thisLine@@;
  currentLineLength = sum(length(thisLine),currentLineLength) + 1;
end;
rc = close(fc);
put;
put content2 ;
put 'OPTIONS / HTTP/1.1' /
"Host: http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx" /
'Connection: Close' /;
end;

```

## Close the Soap Envelope

Post an extra  
“OPTIONS” service  
request to force an extra  
end of line character.

```
filename SoapSrv socket "wslite.strikeiron.com:80" lrecl=32767 termstr=CRLF;
filename SoapRtrn "G:\PNWSUG\Papers\Soap\CensusLite\results.xml" RECFM=N;
filename ContXML "G:\PNWSUG\Papers\Soap\CensusLite\ZIP98501SOAPBody.xml";
```

Ignore the results from  
the OPTIONS request.  
Write it out to the  
destination file.

```
file SoapSrv;
if _n_ = 1 then do;
  content1 = '<soapenv:Envelope '||
    'xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" '|| " " ||
    'xmlns:'|| "wsl="||"http://www.wslite.strikeiron.com"" '|| ">" ||
    '<soapenv:Body>';
  content2 = "</soapenv:Body>" || "</soapenv:Envelope>";
  ContentLength = length(content1) + length(content2) + 101 + 4 + int(101 / 32767);
  call symput ('ContentLength',ContentLength);
  put "POST http://wslite.strikeiron.com/censusinfolite01/CensusInfolite.asmx HTTP/1.1" /
```

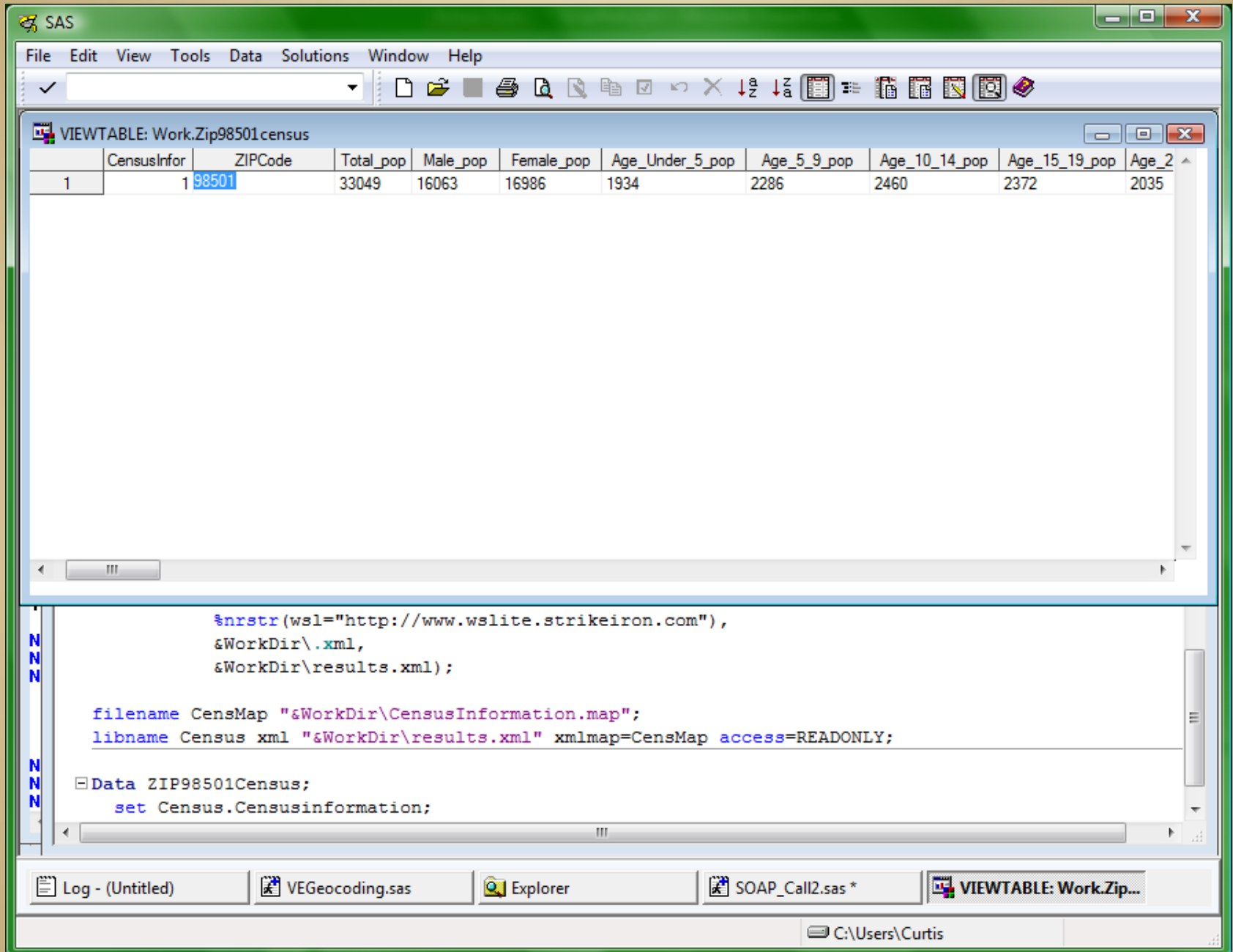
```
if mode = 1 then do;
  input thisRec $ 1 - 32767;
  if thisRec =: "Content-Length:" then do;
    ReturnLength = input(scan(thisRec,2,' '),10.);
  end;
  call symput('ReturnLength',ReturnLength);
  if thisRec = " " then mode = 2;
end;
if mode = 2 then do;
  nextReturn = ReturnLength - TotalReturn;
  input thisRec $varying32767.$nextReturn;
  file SoapRtrn;
  put thisRec;
  TotalReturn = TotalReturn + length(thisRec);
  if TotalReturn >= ReturnLength then mode = 3;
end;
if mode = 3 then do;
  input;
  end;
run;
```

```
if mode = 2 then do;
  nextReturn = ReturnLength - TotalReturn;
  input thisRec $varying32767.$nextReturn;
  file SoapRtrn;
  put thisRec;
  TotalReturn = TotalReturn + length(thisRec);
  if TotalReturn >= ReturnLength then mode = 3;
end;
if mode = 3 then do;
  input;
  end;
run;
```

# Working with The Resulting XML

```
filename XMLMap "c:\CensusLite.map";
```

```
libname results XML "c:\results.xml" xmlmap=  
XMLMap access=READONLY;
```



# A Better Way

## PROC SOAP

- Only Available in 9.2 phase 1 and later
- Handles Authentication
- Handles Proxy Servers

**proc soap**

in=request

out=response

url=" http://www.soapserver.com/SoapService.ext "

soapaction="http://www.soapnamespace.com/MethodName";

**run;**

# Same Call in PROC SOAP

```
FILENAME request "temprq.xml";  
FILENAME response "tempre.xml";
```

```
proc soap in=request  
    out=response  
    url="http://wslite.strikeiron.com/censusinfolite01/CensusInfoLite.asmx"  
    soapaction="http://www.wslite.strikeiron.com/GetCensusInfoForZIPCode"  
    ;  
run;
```

- The result file is processed the same way, using an XMLMap

# Working With REST Services

- Get the service definition and parameters from the service documentation
- Find the definition for the data the service returns.
- Construct a URL containing:
  - The Service URL
  - Each Parameter specified
    - &parametername=value
    - Values must be encoded if they contain anything but letters and numbers
- The code is usually much simpler than SOAP

# Reading the REST Results

- No standard return format. Common formats include:
  - JSON Strings
  - XML
  - Images
- Read the documentation for each service
- SoapUI has tools for working with REST services as well

# Typical REST Call

```
%let AddressVal = %qsysfunc(URLENCODE(1600 Pennsylvania Avenue NW));  
%let CityVal = %qsysfunc(URLENCODE(Washington));  
%let StateVal = DC
```

```
%let url = %nrstr(  
%nrstr(?appid=X)  
%nrstr(&street=)  
%nrstr(&city=)%s  
%nrstr(&state=)%s  
eocode)
```

```
%put &url;
```

This Example uses the  
Yahoo! Geocoding Service

[www.Yahoo.com](http://www.yahoo.com)

```
http://local.yahoo.comXXXXXXXXXX-  
&street=1600%20Pennsylvania%20Avenue%20NW&city=Washington&state=DC
```

```
%let OutXML noprint notitles recm=n;  
put record $varying4000. len;  
run;
```

# The Results

```
<?xml version="1.0" ?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:yahoo:maps" xsi:schemaLocation="urn:yahoo:maps
http://api.local.yahoo.com/MapsService/V1/GeocodeResponse.xsd">
  <Result precision="address">
    <Latitude>38.898590</Latitude>
    <Longitude>-77.035971</Longitude>
    <Address>1600 Pennsylvania Ave NW</Address>
    <City>Washington</City>
    <State>DC</State>
    <Zip>20006</Zip>
    <Country>US</Country>
  </Result>
</ResultSet>
- <!-- ws01.ydn.gq1.yahoo.com uncompressed Fri Sep 25 10:06:25 PDT 2009 -->
```

**Can be easily read using an XMLMap and a XML Libname Reference**

# A Little Better Way

## PROC HTTP

- Only Available in 9.2 phase 1 and later
- Handles many protocol issues
  - Authentication
  - Proxy Servers
  - Specifying the Content Type
  - Custom HTTP Headers
  - GET and POST calls

**proc HTTP**

**in=request**

**out=response**

**url=" http://www.soapserver.com/SoapService.ext ";**

**run;**

# Same Yahoo Call in PROC HTTP

```
filename OutXML "OutXML.xml";  
filename address "address.txt";
```

```
data _null_;  
    file address;  
    put 'appid=XXXXXXXXXXXXX-' @;  
    put '&street=1600 Pennsylvania Avenue NW' @;  
    put '&city=Washington' @;  
    put '&state=DC';  
run;
```

```
PROC HTTP  
    in=address  
    out=OutXML  
    url="http://local.yahooapis.com/MapsService/V1/geocode";  
RUN;
```

# New to SAS 9.2 Phase 2

Libname myxml XML XMLType = WSDL

```
filename NWS url
```

```
"http://www.weather.gov/forecasts/xml/SOAP_server/ndfdXMLserver.php?wsdl";  
libname NWS XML92 xmltype=WSDL;
```

```
proc datasets library=NWS details;  
run;
```

- Brand new in 9.2 Phase 2
- Allows direct calls to SOAP services via a LIBNAME reference.
- Very little documentation yet
- I can't get it to work ☹

# Other Examples

# Fun REST PROC HTTP Example

Google Charts



This Example uses the  
Google Chart API

<http://code.google.com/apis/chart/>

```
%let WorkDir
```

```
filename in '  
filename out
```

```
data _null_;  
  file in;  
  put 'cht=v&  
run;
```

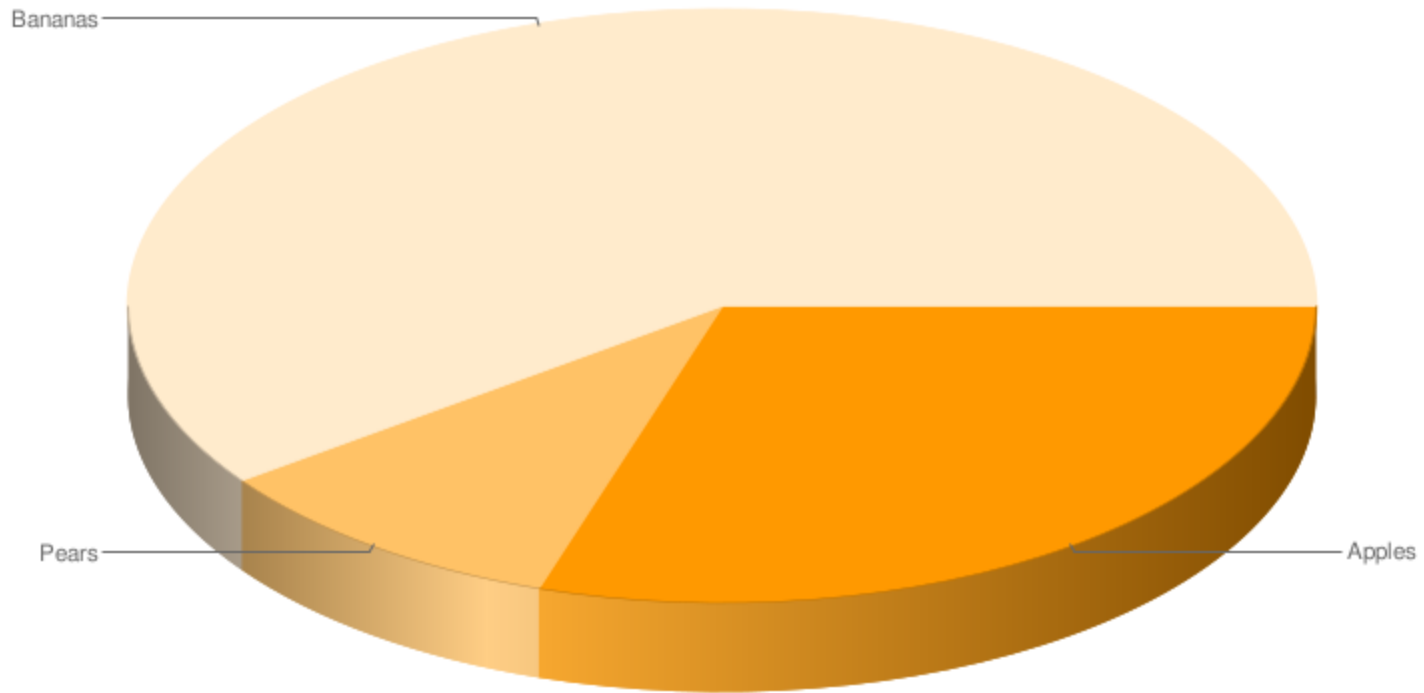
```
proc http in;
```

hl='

The first three values specify the relative sizes of three circles: A, B, and C.  
The fourth value specifies the area of A intersecting B.  
The fifth value specifies the area of A intersecting C.  
The sixth value specifies the area of B intersecting C.  
The seventh value specifies the area of A intersecting B intersecting C.

# More Fun With Google Charts

- clear
- M



es|P

# Warning!

All of these public web services have Terms of Use agreements. Please read them before implementing an application that uses these services!

# Conclusion

- There are many different ways to consume web services from within base SAS.
- Depending SAS has made great improvements in its tools for doing this.
- They are continuing development in this area.

# The End

**Code examples will be available in the  
proceeding after the conference**

Curtis Mack  
Curtis.Mack@lgan.com  
Looking Glass Analytics  
[www.LGAN.com](http://www.LGAN.com)