Paper 2029-2014

Five things to Do when using SAS® BI Web services

Neetha Sindhu, Vimal Raj
Kavi Associates LLC, Barrington IL

ABSTRACT

Traditionally, web applications interact with back-end databases by means of JDBC/ODBC connections to retrieve and update data. With the growing need for real-time charting and complex analysis types of data representation on these web applications, SAS computing power can be put to use by adding a SAS web service layer between the application and the database. With the experience that we have with integrating these applications to SAS® BI Web Services, this is our attempt to point out five things to do when using SAS BI Web Services. 1) Input Data Sources: always enable "Allow rewinding stream" while creating the stored process. 2) Use LIBNAME statements to define XML filerefs for the Input and Output Streams (Data Sources). 3) Define input prompts and output parameters as global macro variables in the stored process if the stored process calls macros that use these parameters. 4) Make sure that all of the output parameters' values are set correctly as defined (data type) before the end of the stored process. 5) The Input Streams (if any) should have a consistent data type; essentially, every instance of the stream should have the same structure. This paper consists of examples and illustrations of errors and warnings associated with the previously mentioned cases.

INTRODUCTION

Web services are used to allow communication of distributed systems. It facilitates applications on different platforms to use web based protocols to communicate. SAS offers BI Webservices through which SAS stored processes can be exposed to external applications. The number of web based applications and mobile applications that would benefit to use the power of SAS have grown drastically. A lot of complex data manipulation, calculations, statistical analysis and modeling can be achieved on the SAS side. This can be made available to the applications developed using technologies like JAVA which can consume these web services using the HTTP.

SAS BI Web Services automatically exposes a WSDL file for each and every stored process in your system. These WSDL files use XML to include detailed information about the inputs and outputs of each stored process using XML schema descriptions. Also, the WSDL file includes the URLs of endpoints to use to invoke these stored processes by using the SOAP protocol over HTTP. Typically, you use these WSDL files to automatically generate code in your client framework that can be used to invoke the Web services. SAS BI Webservices provides inputs to the underlying SAS stored processes using filerefs and macro variables. [1]

In this paper we are using an example to show how input/output parameters should be handled while creating stored processes to be deployed as Webservices. It is a comprehensive learning obtained by continuous use of SAS BI Webservices for our client applications. These Webservices allow for streaming in and streaming out data using xml. This streamed in data can be read into SAS datasets in the stored process. Similarly any SAS dataset can be streamed out using xml message format.

CASE

We have created a very simple demonstration project to illustrate the usage of inputs and outputs in SAS BI Webservices. It consists of a SAS BI Webservice that queries the dataset CARS provided in the SASHELP library. This service accepts user specified 'Make of the Car' and pulls all the details of this make and outputs it to the client application by streaming. It also accepts a stream input through which horsepower and cylinders are sent to the SAS stored process. Another dataset is streamed back to the client and it consists of all the cars that have the number of cylinders and horsepower as entered.

SAS Dataset: SASHELP.CARS

Stored Process Created: cars_report

Input Prompts: Make

Input Stream: Config

Output Parameters: StatusCode, StatusMessage

Output Streams: MakeRep, HPRep

We are using SOAP UI as a client to consume this Webservice. SOAP UI sends and receives xml requests and responses.

PURPOSE

The following are some simple tips to keep in mind while using input/output prompts and data sources [streams]. If these are taken care of, development of SAS BI Webservices becomes easier. Many a times developers do not pay much attention to these steps and end up spending a great deal of time debugging the errors. A conscious effort has been made to describe the need for following these tips and errors associated with them if they are not followed.

INPUT DATA SOURCES: ALWAYS ENABLE "ALLOW REWINDING STREAM" WHILE CREATING THE STORED PROCESS.

While using XML or SOAP method of invoking a web service and there are input streams to be passed as input to the stored process, we need to create input data sources while defining the metadata of the stored process. In the new stored process creation wizard always check the "Allow rewinding stream" check box. This option must be checked when using the LIBNAME engine to dynamically interpret XML data. The LIBNAME engine must make multiple passes over the data, and these multiple passes can occur only if the stream allows its contents to be "rewound" (restarted from the beginning). [2]

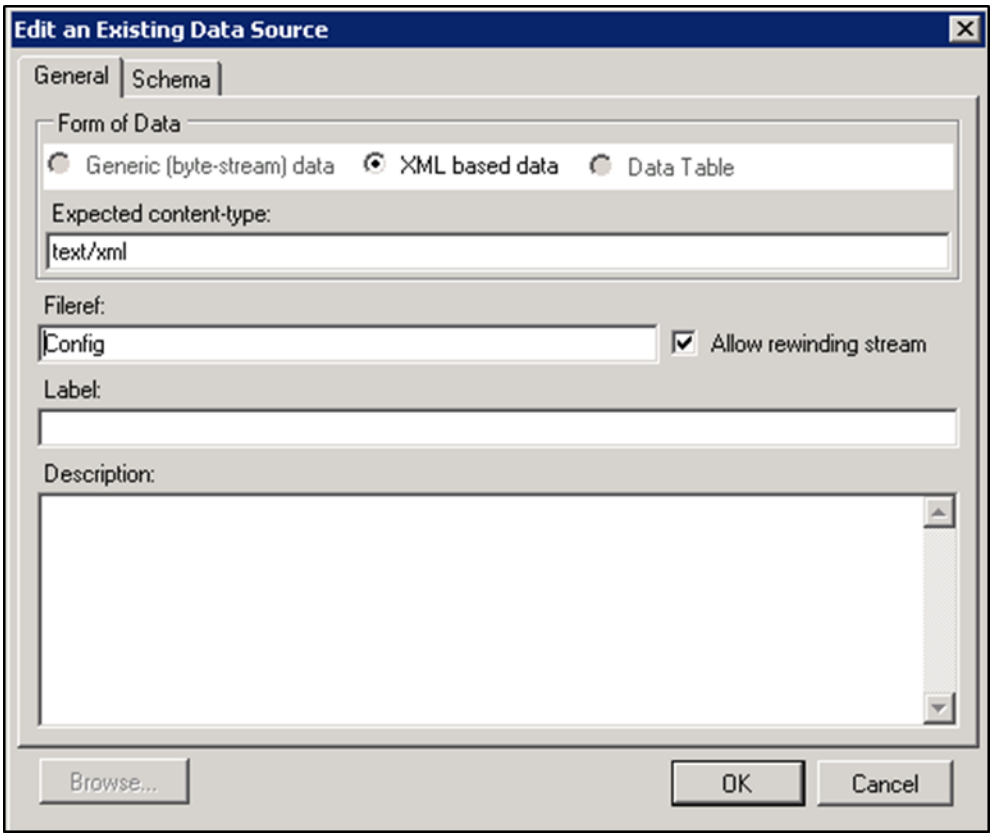

Figure 1. Create Input Stream

This option plays a major role when using a data step to read the incoming stream as a sas dataset. If this option is

not checked and the stored process code tries to read the incoming data source stream, it encounter an I/O failure.

```
44         +data temp1;
45         +set Config.Config;
46         +run;

ERROR: Undetermined I/O failure.
ERROR: Encountered during XMLInput parsing at or near line 1, column 1.
ERROR: The definition for the "CONFIG" table is not well-formed or is corrupt.
NOTE: The DATA step has been abnormally terminated.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: SAS set option OBS=0 and will continue to check statements. This might cause NOTE: No observations in data set.
WARNING: The data set WORK.TEMP1 may be incomplete.  When this step was stopped there were 0 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Figure 2. Error Log

USE LIBNAME STATEMENTS TO DEFINE XML FILEREFS FOR THE INPUT AND OUTPUT STREAMS (DATA SOURCES).

For SAS a stream is just an ordered flow of data. The following code using the xml libname engine can be used to convert the incoming streams into data sets. If the incoming message type is XML you can define the incoming data sources/streams as xml librefs. This converts the incoming streams into structured SAS Library.

```
/*create XML libref*/
libname Config XML;
libname MakeRep XML;
libname HPRep XML;
```

Figure 3. Code Snippet to create XML Fileref

These can then be used as a regular SAS Library. In a case that the xml libname is not defined and you try to access the input data source as a SAS library, the following error message is thrown by the SAS stored process.

```
44         +data temp1;
45         +set Config.Config;
ERROR: Libname CONFIG is not assigned.
46         +run;
```

Figure 4. Error Log

DEFINE INPUT PROMPTS AND OUTPUT PARAMETERS AS GLOBAL MACRO VARIABLES IN THE STORED PROCESS IF THE STORED PROCESS CALLS MACROS THAT USE THESE PARAMETERS.

The input prompts or output parameters can be used to pass information to and from the SAS stored processes. These are created when the stored process metadata is created. These prompts need to be declared as global variables in the SAS stored process code. Below is a code snippet of how the global variables are defined in the stored process code.

```
/*Define input and output parameters as Global macro variables*/
%global StatusCode StatusMessage Make;
```

Figure 5. Code snippet to declare global parameters

If a code file invoked from the stored process tries to access one of these prompts that have not been declared as global, the following error message is thrown by the SAS stored process.

```
The exception follows: Invalid value type for output parameter 'StatusCode' was returned from stored process execution.
```

Figure 6. Error Log

MAKE SURE THAT ALL OF THE OUTPUT PARAMETERS' VALUES ARE SET CORRECTLY AS DEFINED (DATA TYPE) BEFORE THE END OF THE STORED PROCESS.

The SAS stored process code needs to set appropriate values to all the defined output parameters. These output parameters can be of any data type. If the value assigned to the macro variable denoted by the output parameter is not the same data type as defined in the metadata, the following error will be thrown by the SAS stored process.

```
The exception follows: Invalid value type for output parameter 'StatusMessage' was returned from stored process execution
```

Figure 7. Error Log

The output parameters can be set like any other macro variable is assigned a value. Below is Code snippet of how the output parameters are set.

```
/*Set Output Parameters*/
%let StatusCode=0;
%let StatusMessage=Success;
```

Figure 8. Code snippet to set output parameters

THE INPUT STREAMS (IF ANY) SHOULD HAVE A CONSISTENT DATA TYPE; ESSENTIALLY, EVERY INSTANCE OF THE STREAM SHOULD HAVE THE SAME STRUCTURE.

While streaming in XML input data, make sure that the values of the one XML element are all of the same data type. When the stream is read into a SAS dataset the first record determines the data type of the column. If there is a discrepancy in the data type of values coming in the following records an error will be thrown.

Below is a part of a SOAP/XML request sent to the webservice. The Stream CONFIG has two elements HORSEPOWER and CYLINDERS. The first main CONFIG element corresponds to the first record in the SAS dataset that this stream is read into and HORSEPOWER and CYLINDERS are the columns of the numeric datatype. When SAS tries to read the second record from the stream and a character value is being passed, an error is thrown.

```
<cars:Config contentType="?">
   <cars:Value>
    <TABLE>
    <CONFIG>
            <HORSEPOWER>200</HORSEPOWER>
            <CYLINDERS>4</CYLINDERS>
    </CONFIG>
    <CONFIG>
            <HORSEPOWER sd/HORSEPOWER>
            <CYLINDERS>6</CYLINDERS>
    </CONFIG>
    </TABLE>
   </cars:Value>
</cars:Config>
```

Figure 9. Input Stream Structure

CONCLUSION

SAS BI Webservices combined with the facility of having input/ouput parameters and streams makes it a very powerful and useful concept. This has enabled many of our applications and I assume a lot of other web based applications to incorporate complex analysis types of solutions.

This paper is a cheat sheet that you can use while developing the stored process to be exposed as Webservices.

REFERENCES

[1] http://support.sas.com/documentation/cdl/en/wbsvcdg/62759/PDF/default/wbsvcdg.pdf

[2] Tim Beese and Greg Granger, SAS Institute Inc., Cary, NC 'Excelling with Excel' SAS Global Forum 2012

http://support.sas.com/resources/papers/proceedings12/036-2012.pdf

ACKNOWLEDGMENTS

We would like to thank Kavi Associates LLC to provide us with all the necessary infrastructure and resources to be able to strengthen our skillset in SAS. We are also grateful to SAS for giving us this opportunity to present and publish our work and learnings.

RECOMMENDED READING

http://support.sas.com/documentation/cdl/en/engxml/62845/PDF/default/engxml.pdf

http://support.sas.com/documentation/cdl/en/wbsvcdg/62759/PDF/default/wbsvcdg.pdf

CONTACT INFORMATION <HEADING 1>

Please reach out to us for any comments, feedback or questions.

Name: Neetha Sindhu
Enterprise: Kavi Associates LLC
1250 S Grove Ave Suite 300
Barrington, IL - 60010
Work Phone: (847)387-6760 [extn: 201]
E-mail: neetha.sindhu@kaviglobal.com

Name: Vimal Raj
Enterprise: Kavi Associates LLC
1250 S Grove Ave Suite 300
Barrington, IL - 60010
Work Phone: (847)387-6760 [extn: 311]
E-mail: vimal.raj@kaviglobal.com