Paper 2027-2014

SAS® and Java Application Integration for Dummies

Neetha Sindhu, Hari Hara Sudhan, Mingming Wang
Kavi Associates, Barrington, IL 60010

ABSTRACT

Traditionally, Java 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 types of web applications, SAS® computing power can be put to use by adding a SAS web service layer between the application and the database. This paper shows how a SAS web service layer can be used to render data to a JAVA application in a summarized form using SAS® Stored Processes. This paper also demonstrates how inputs can be passed to a SAS Stored Process based on which computations/summarizations are made before output parameter and/or output data streams are returned to the Java application.

SAS Stored Processes are deployed as SAS® BI Web Services using SAS® Management Console, which are available to the JAVA application as a URL. The SOAP method is used to generate the XML's which is the primary method of interaction between the web services and JAVA. XML data representation is used by various applications as a method of communication. We then illustrate how RESTful web services can be used with JSON objects being the communication medium between the JAVA application and SAS® 9.3. Once this pipeline communication between the application, SAS engine, and database is set up, any complex manipulation or analysis as supported by SAS can be incorporated into the SAS Stored Process. We then illustrate how graphs and charts can be passed as outputs to the application.

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 demonstrate how SAS can be integrated with a JAVA application using the SAS BI Webservices.

ARCHITECTURE

The basic architecture of how a JAVA application and communicates with the SAS web services is illustrated below. Figure 1 shows the different layers on how the communication between JAVA and web service layer happens.

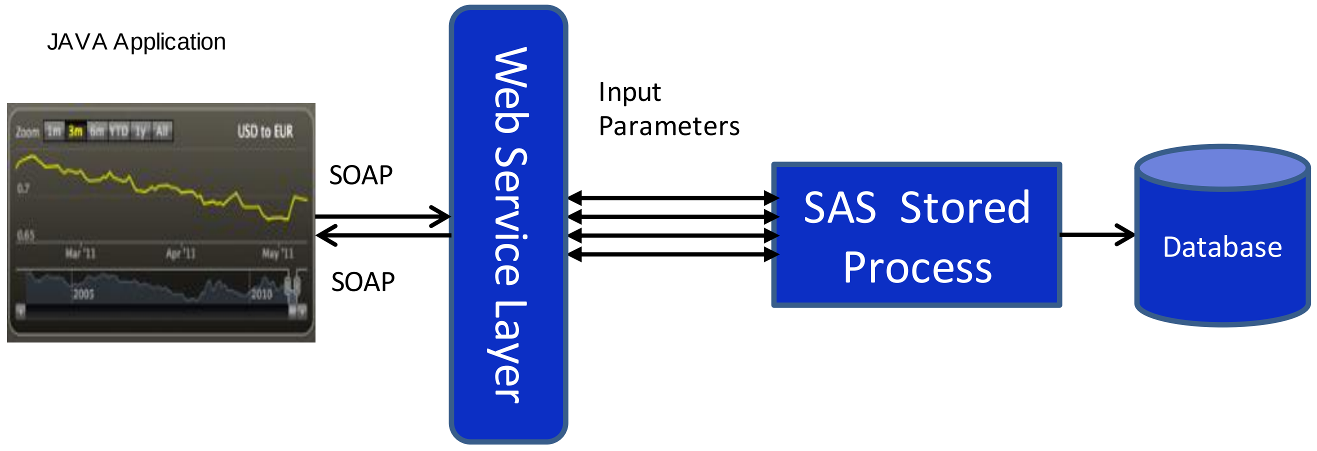
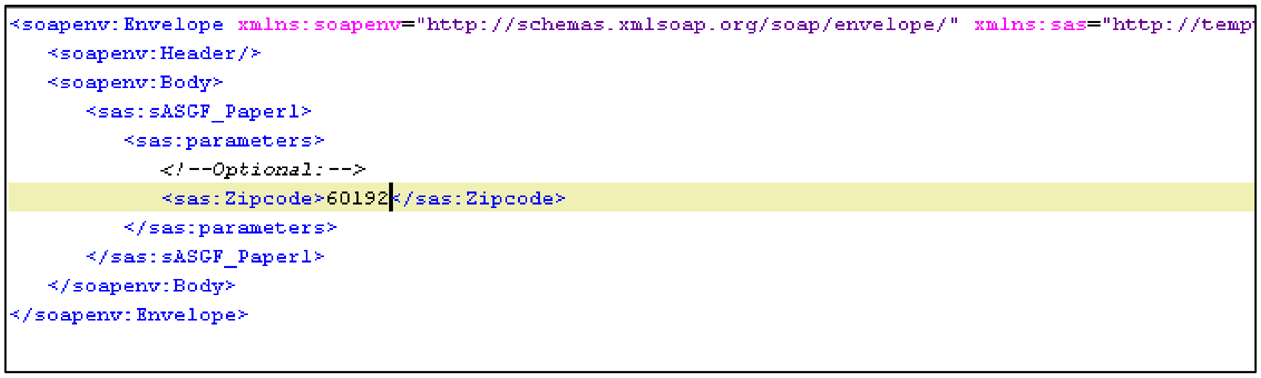


Figure 1 Architecture

The JAVA application gets the required inputs from the user and passes to the web service. The inputs parameters/streams are then sent as a SOAP-XML request to the web service. Based on the given inputs the stored process is executed and the results are sent back to the JAVA Application as a SOAP-XML response. A sample SOAP-XML response is shown below.



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sas="http://temp
   <soapenv:Header/>
   <soapenv:Body>
      <sas:sASGF_Paper1>
         <sas:parameters>
            <!--Optional:-->
            <sas:Zipcode>60192</sas:Zipcode>
         </sas:parameters>
      </sas:sASGF_Paper1>
   </soapenv:Body>
</soapenv:Envelope>
```

Figure 2

CASE

We have created a very simple demonstration project to illustrate the integration of a JAVA application to SAS via the SAS BI Webservices. It consists of a SAS BI Web service that queries the dataset ZIPCODE provided in the SASHELP library. This service accepts user specified 'Zipcode' and pulls all the details of this location such as latitude, longitude, city, state and outputs it to the client application by streaming. This information is then displayed to the user by the JAVA Application.

The sample data being considered here is the geographic details of the United States. The description of the data is available in Table1.

Variable description	Data Type	SAS Variable Format
Zipcode	INTEGER	15.
Latitude	CHARACTER	$*w*.
County	INTEGER	15.

Table 1. Variable Description

JAVA APPLICATION

A simple JAVA application is built to get a zip code as an input from the user. In this example the user enters the zip code and the location information specific to that zip code such as city, state, county, latitude and longitude is displayed.

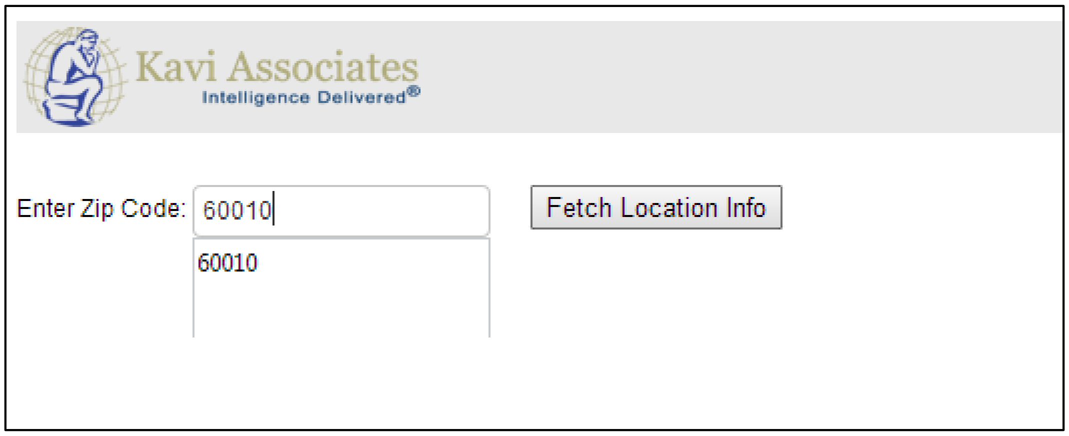
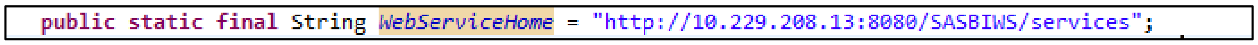


Figure 3: Java UI

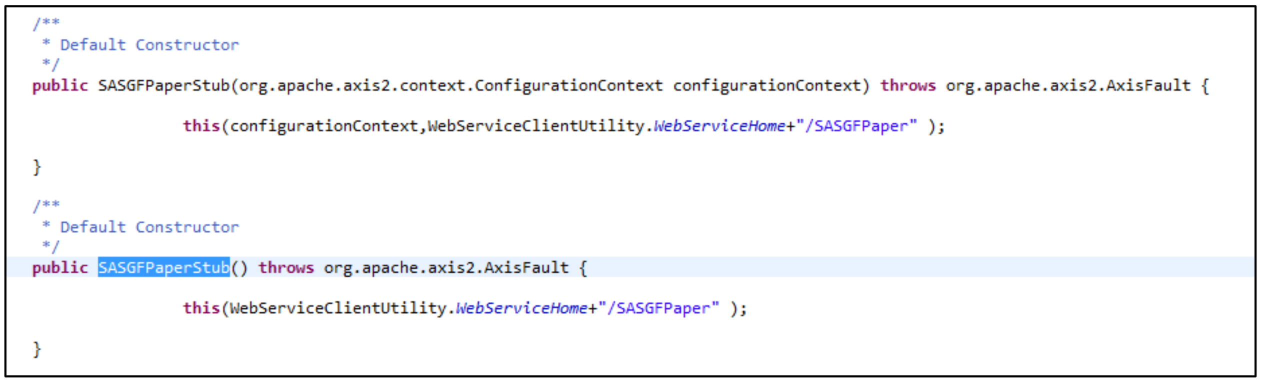
The input parameter is w rapped in a SOAP request and sent to the SAS BI Webservice and the SOAP response received is parsed. The follow ing is the WSDL endpoint provided by the SAS BI Webservice.



```
public static final String WebServiceHome = "http://10.229.208.13:8080/SASBIWS/services";
```

Figure 4: JAVA web service definition

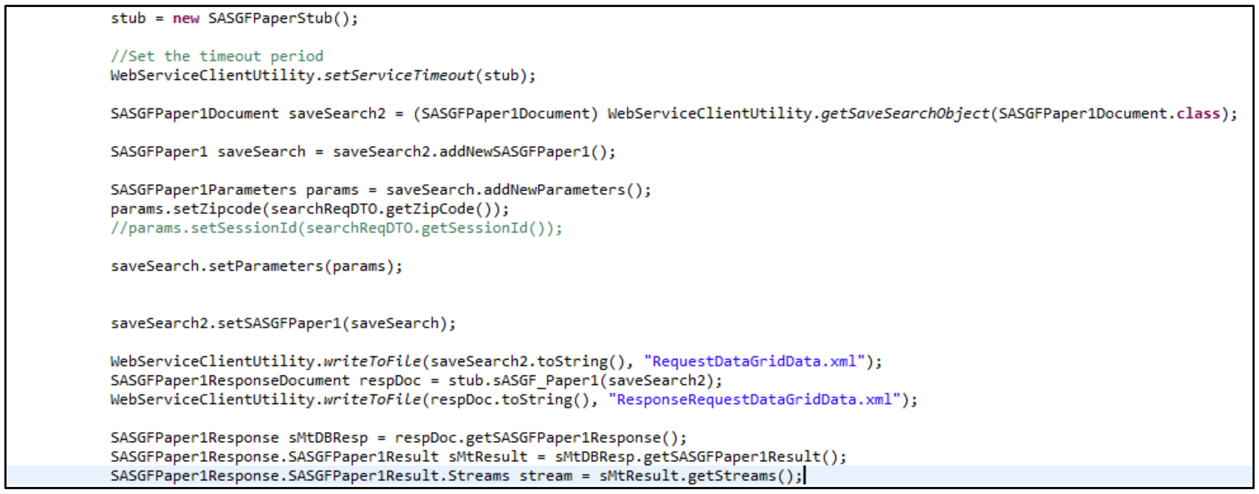
In order to invoke the w eb service from JAVA w e are using Apache Axis2 API. The follow ing is the stub code generated using Apache Axis2 tool automatically.



```
/**
 * Default Constructor
 */
public SASGFPaperStub(org.apache.axis2.context.ConfigurationContext configurationContext) throws org.apache.axis2.AxisFault {

                this(configurationContext,WebServiceClientUtility.WebServiceHome+"/SASGFPaper" );

}

/**
 * Default Constructor
 */
public SASGFPaperStub() throws org.apache.axis2.AxisFault {

                this(WebServiceClientUtility.WebServiceHome+"/SASGFPaper" );

}
```

Figure 5: JAVA Objects definition

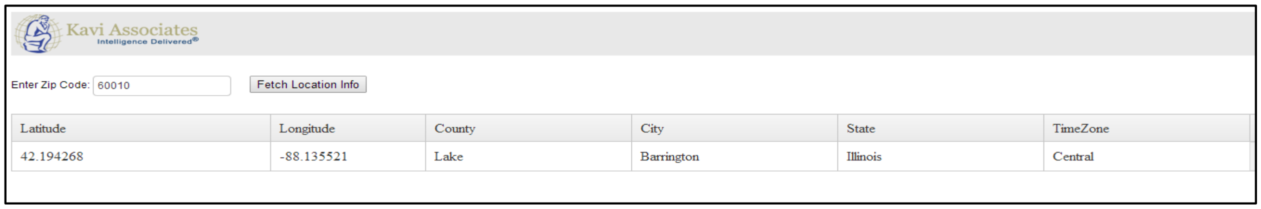
The follow ing is the code snippet to invoke the w eb service by populating the input as per the WSDL.



```
stub = new SASGFPaperStub();

//Set the timeout period
WebServiceClientUtility.setServiceTimeout(stub);

SASGFPaper1Document saveSearch2 = (SASGFPaper1Document) WebServiceClientUtility.getSaveSearchObject(SASGFPaper1Document.class);

SASGFPaper1 saveSearch = saveSearch2.addNewSASGFPaper1();

SASGFPaper1Parameters params = saveSearch.addNewParameters();
params.setZipcode(searchReqDTO.getZipCode());
//params.setSessionId(searchReqDTO.getSessionId());

saveSearch.setParameters(params);


saveSearch2.setSASGFPaper1(saveSearch);

WebServiceClientUtility.writeToFile(saveSearch2.toString(), "RequestDataGridData.xml");
SASGFPaper1ResponseDocument respDoc = stub.sASGF_Paper1(saveSearch2);
WebServiceClientUtility.writeToFile(respDoc.toString(), "ResponseRequestDataGridData.xml");

SASGFPaper1Response sMtDBResp = respDoc.getSASGFPaper1Response();
SASGFPaper1Response.SASGFPaper1Result sMtResult = sMtDBResp.getSASGFPaper1Result();
SASGFPaper1Response.SASGFPaper1Result.Streams stream = sMtResult.getStreams();
```

Figure 6: JAVA web service

Using XML DOM parser the response stream is converted into JAVA object, w hich is then displayed back to the user as show n in the Figure below .



Latitude	Longitude	County	City	State	TimeZone
42.194268	-88.135521	Lake	Barrington	Illinois	Central

Figure 7: UI Output

SAS STORED PROCESS

A SAS stored process is a SAS program that is hosted on a server and described by metadata [2].The steps to create a SAS Stored process using Enterprise guide is described in the follow ing sections. The stored process illustrated in this paper uses one input parameter and one output parameter. There is also a table output that is streamed as an XML to the JAVA application.

Creating a SAS Stored process using SAS Enterprise Guide®

A SAS Stored process can be created using a SAS Management console or SAS Enterprise guide. The follow ing stored process is being created in SAS 9.3 using SAS Enterprise guide 5.1.

Step 1: Specify name and location

The first step w hile creating a stored process is to specify the name of the stored process and the location w here it has to be stored in the metadata server. A short description of the stored process can also be specified.

Stored process name: SASGF_Paper

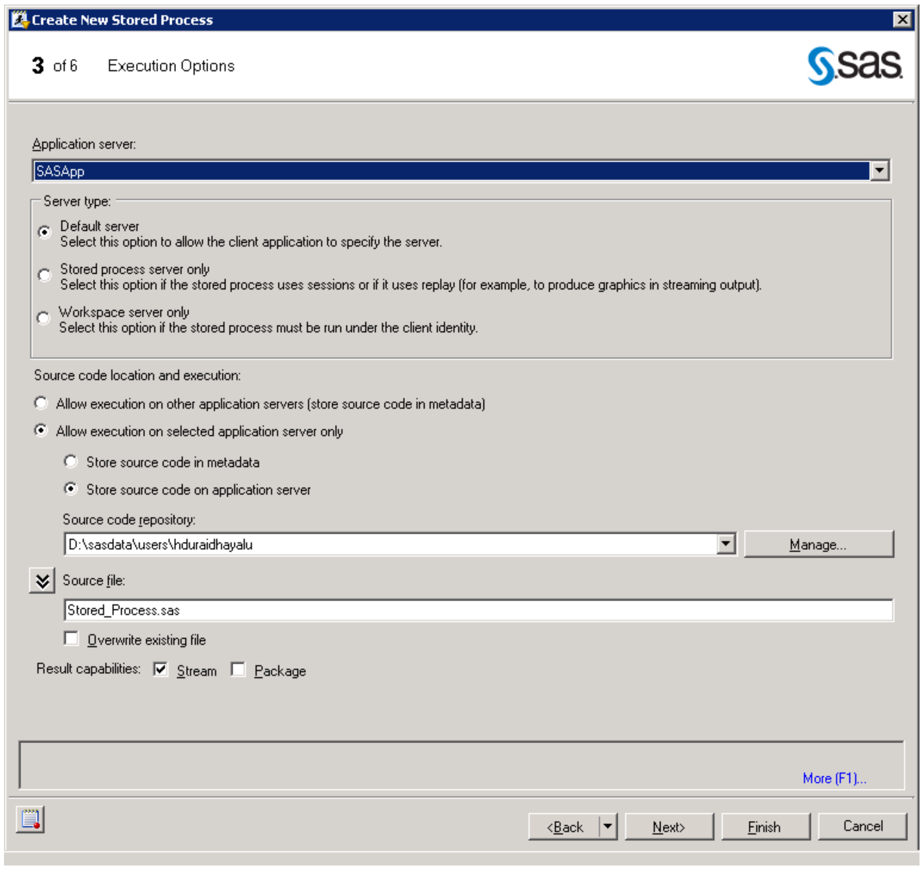
Location: /Projects/

Step 2: SAS Code

The actual code to be executed when the web service is called is available here.

Step 3: Specify the execution server and source code location

The next step w ould be to specify the server w here the stored process has to be executed. The stored process can be executed in stored process server or Workspace server. Stored processes running on a SAS Workspace Server w ill execute using the client application logon credentials for the end user. The stored processes executed on a SAS Stored process server w ill execute using the credentials of the sassrv account [3]. If Default server option is selected the SAS Stored process server is used.



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: D:\sasdata\users\hduraidhayalu

Source file: Stored_Process.sas

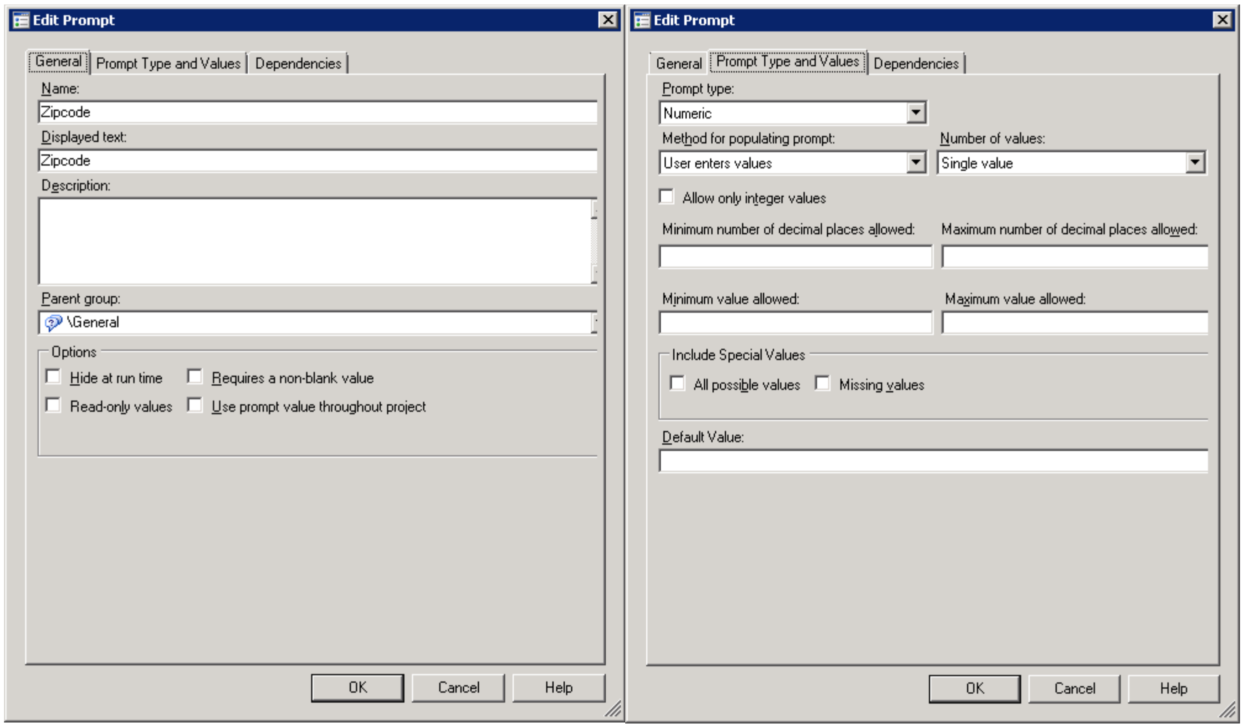
☐ Overwrite existing file

Result capabilities: ☑ Stream ☐ Package

Figure 8: Specify execution server

Step 4: Create prompts

The next step is to create prompts. For this paper one input prompt and one output parameter is created. In Figure 2 the Name specifies the name of the macro variable that stores the value of the input prompt. In this case the zipcode entered by the user is stored in the macro variable *Zipcode.* In the ***Prompt Type and Values*** tab the type of input (numeric or character) and the method of how the user inputs the data are specified.


Edit Prompt

General | Prompt Type and Values | Dependencies

Name: Zipcode

Displayed text: Zipcode

Description:

Parent group: \General

Options

☐ Hide at run time ☐ Requires a non-blank value

☐ Read-only values ☐ Use prompt value throughout project

OK Cancel Help

Edit Prompt

General | Prompt Type and Values | Dependencies

Prompt type: Numeric

Method for populating prompt: User enters values

Number of values: Single value

☐ Allow only integer values

Minimum number of decimal places allowed:

Maximum number of decimal places allowed:

Minimum value allowed:

Maximum value allowed:

Include Special Values

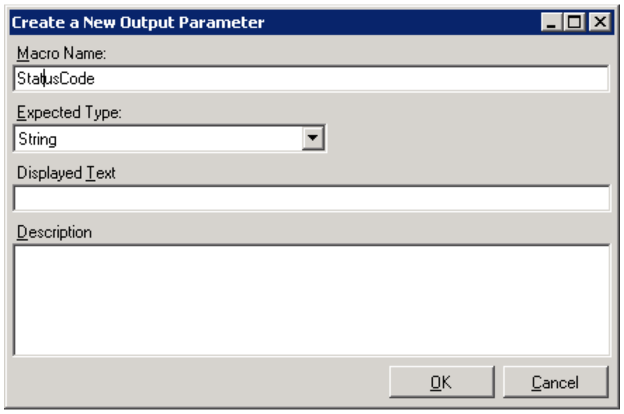
☐ All possible values ☐ Missing values

Default Value:

OK Cancel Help

Figure 9: Input parameter

The output parameter created is StatusCode. This specifies if the stored process w as executed successfully. The macro variable created here is StatusCode and this can be assigned any value in the SAS code mentioned in step 2.


Create a New Output Parameter

Macro Name: StatusCode

Expected Type: String

Displayed Text

Description

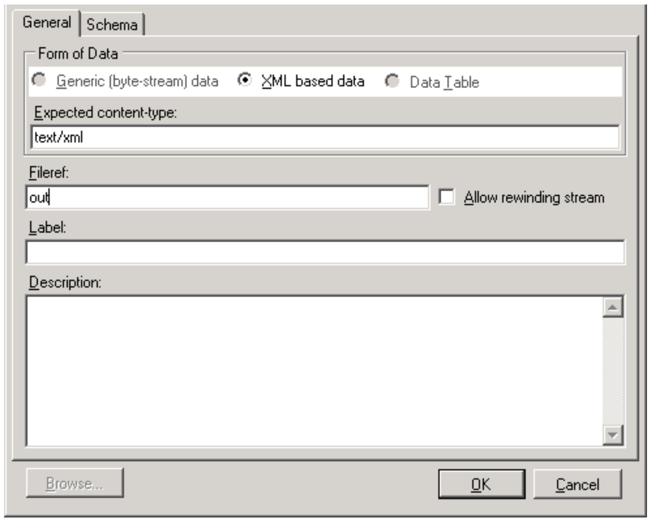
OK Cancel

Figure 10: Output prompts

Step 5: Specify the output stream

In this case study since w e do not have any input streams the steps to create output streams is show n below . When creating a new output stream the type of stream - XML has to be specified and the Fileref. By giving the Fileref the resulting output is streamed as an XML to the JAVA application. The code to stream a dataset as an XML using the Fileref is as below ;

```
Libname  out XML;
Data  out.<dataset_name>;
Set <result_data_set>;
Run;
```



Form of Data: [✓ XML based data]

Expected content-type: text/xml

Fileref: out

Allow rewinding stream

Label:

Description:

Figure 11: Data Output streams

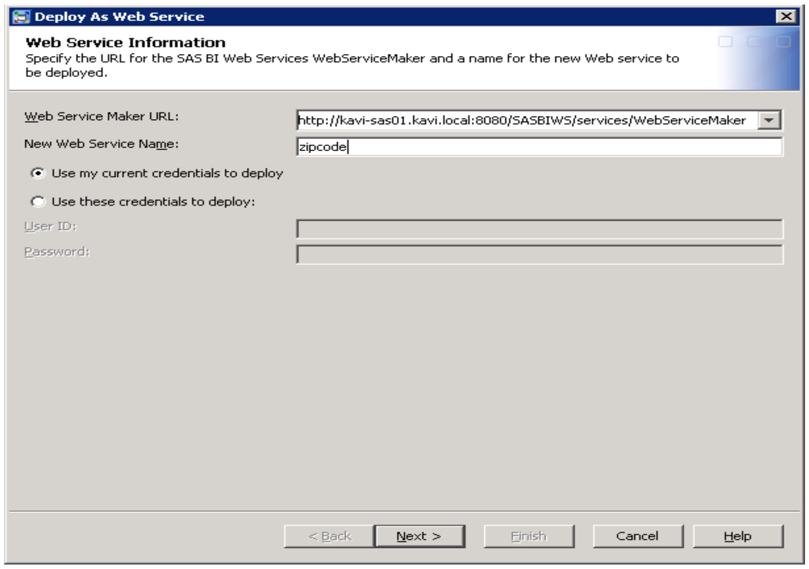
Step 6: Summary

This step summarizes all the information that was provided in the earlier steps. This step can be used to verify all the properties that were specified.

Deploy stored process as a web service

SAS 9.3 exposes all the stored processes as web services without having to deploy them as web services using Management Console.

For earlier versions of SAS, stored process is deployed as a web service using SAS Management Console. Browse to the location of the stored process in the management console as specified in Step 1 and right click and select the option Deploy as web service. Specify a name for the web service and the stored process would be deployed as a web service and the URL to access the same is supplied.



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: http://kavi-sas01.kavi.local:8080/SASBIWS/services/WebServiceMaker

New Web Service Name: zipcode

[✓ Use my current credentials to deploy]

Use these credentials to deploy:

User ID:

Password:

Figure 12: Deploy web service

CONCLUSION

Introduction of the SAS layer in between complex applications built on JAVA and the database is a good approach to allow for the application to achieve complex statistical analysis types of function. In similar lines to the illustration shown in this paper more can be achieved by integrating multiple SAS BI Web services to applications with multiple functionalities.

REFERENCES

[1] 5 things to do when using SAS® BI Web services; Neetha Sindhu, Vimal Raj, Kavi Associates LLC, SAS Global Forum 2014

[2] Creating and Using SAS® Stored Processes with SAS® Enterprise Guide® ;Eric Rossland, Kari Richardson ;SAS Institute. Available at http://www2.sas.com/proceedings/sugi30/135-30.pdf.

[3] http://support.sas.com/kb/31/155.html; SAS Support services.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Neetha Sindhu
Organization: Kavi Associates LLC
Work Phone: 847-387-6760 extn:201
Email: neetha.sindhu@kaviglobal.com

Name: Hari Hara Sudhan
Organization: Kavi Associates LLC
Work Phone: 847-387-6760 extn:203
Email: hari.duraidhayalu@kaviglobal.com

Name: Ming Ming Wang
Organization: Kavi Associates LLC
Work Phone: 847-387-6760 extn:210
Email: mingming.wang@kaviglobal.com