Paper 017-2009

Developing Rich Internet Applications (RIA) Using the SAS® Intelligence Platform

Robert Lill, BNL Consulting LLC, Rockville, MD
Steven Tuttle, SAS Institute Inc., Houston, TX

ABSTRACT

This paper discusses how to develop rich Internet applications (RIA) using tools such as Flex with SAS® as a service provider. Exposing the powerful SAS platform as a service endpoint through its Java API, we are able to inject real intelligence into business applications. We provide an architectural overview and technical discussion of each component used including SAS, Eclipse, RIAFusion plug-in, Flex, Jboss, and Java. We highlight the SAS Integration, Service registration, and Flex development steps required to realize this SOA application development construct.

INTRODUCTION

The purpose of this paper is to describe and showcase how SAS can play an important role as service provider in the emerging world of Rich Internet Application development. We will focus our discussion on the current production release of SAS, 9.1.3 SP4. With SAS 9.2, there are several new features added to the SAS Business Intelligence suite that provide for authoring and deployment of SAS Stored Processes as web services.[1] This will only improve the ease of integration when developing RIAs.

RICH INTERNET APPLICATIONS

As both Java and SAS developers for many years, we have both seen the movement from rich client interfaces to zero clients and now to a synthesis of the two called rich Internet applications (RIA). A RIA is a Web application designed to deliver the same features and functions normally associated with desktop applications[2]. With all the benefits web browsers provide, there is still gap in the quality of the user experience when compared with true client applications. RIAs fill this gap by providing a quality user experience though attractive and functional interfaces. For the most part RIAs are back end agnostic and are about building better interfaces. RIAs of course resonate with the consumer, but if you take a look at the benefits RIAs provide however, you'll see that there are some very useful synergies with Software as a Service and a general Service Oriented Architecture strategy. As we know, service architectures can provide significant value to the enterprise for many reasons and are becoming more and more prevalent. It just takes more work to build applications against services as opposed to traditional data connections such as JDBC/ODBC. With RIAs though, it turns out to be a perfect case of the consumer side and the enterprise side coming together for mutual benefit. For these reasons, they are on the rise. According to Gartner, "*By 2010, at least 60 percent of new application development projects will include RIA technology, and at least 25 percent will rely primarily on RIA*".

Even with all the out of the box functionality vendors like SAS and others provide, there is still a need for very high end custom developed portlets and applications. Many of the leading BI vendors are adding an RIA development toolkit as extensions to their platforms .These are purpose built and can draw from the arsenal of features and functions that come with the platform. SAS is moving in that direction but in the meantime, we have created a solution using the current release of SAS 9.1 sp4.

[1] For a more detailed description of these capabilities, see SUGI paper Paper 310-2008 titled Using SAS® BI Web Services and PROC SOAP in a Service-Oriented Architecture by Dan Jahn of SAS, Cary NC.

[2] An RIA normally runs inside a Web browser and usually does not require software installation on the client side to work. However, some RIAs may only work properly with one or more specific browsers. For security purposes, most RIAs run their client portions within a special isolated area of the client desktop called a sandbox. The sandbox limits visibility and access to the file and operating system on the client to the application server on the other side of the connection.

This paper will detail a set of components we created and integrated in order to more easily develop RIAs using SAS as the data and compute server provider. We call this toolkit "**RIAFusion**". The following architecture diagram shows how developers and consumers would interact with our implementation of a RIA/SAS Services architecture. Our discussion below does not discuss the Service Bus[3], but since it is commonly part of an SOA approach, we include it.

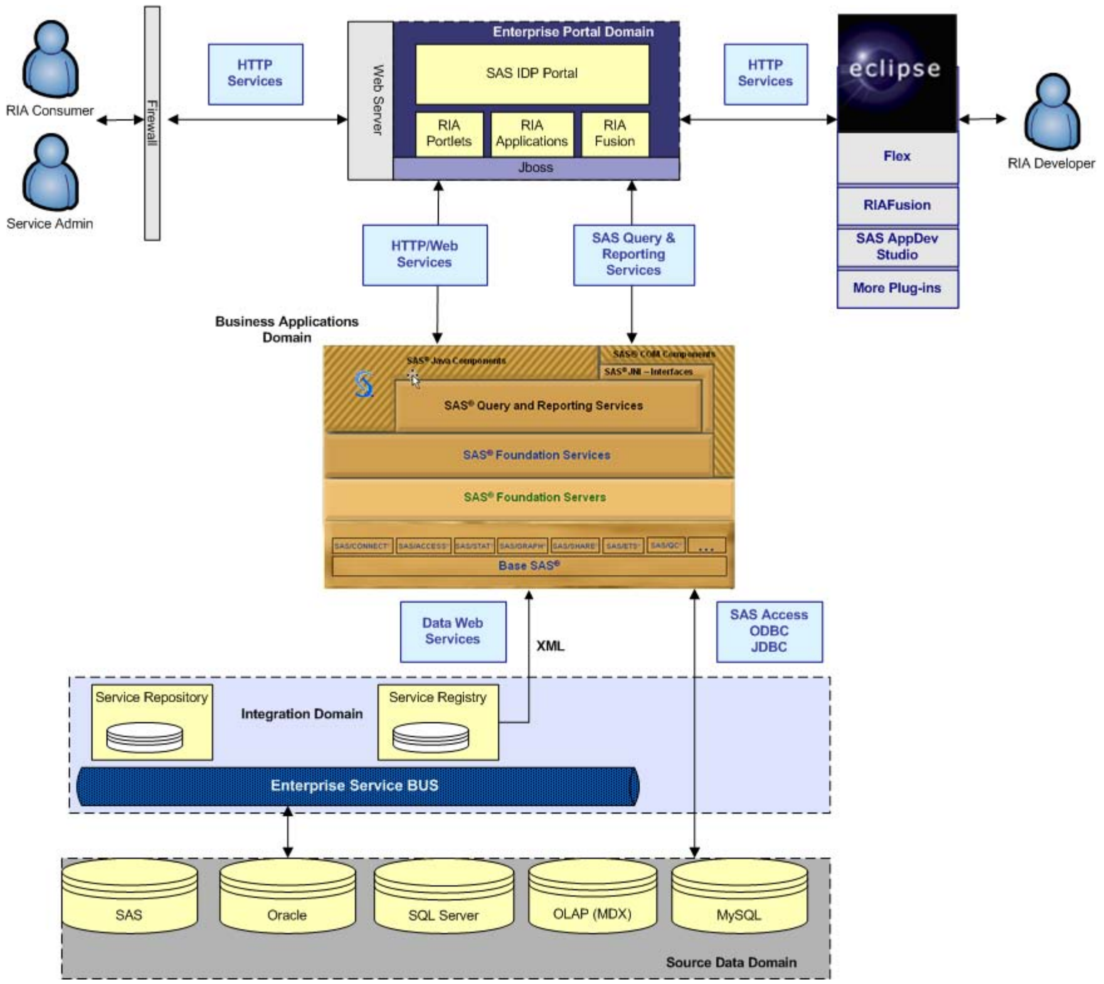


Figure 1 RIAFusion Architecture

This diagram shows how RIA developers using Eclipse, have access to all of their Eclipse plug-ins such as Flex, SAS AppDev studio and others to develop from a single and consistent UI. From there, they can connect through to the RIAFusion Registry server application running under a Java servlet container such as Jboss, Tomcat, Weblogic and Websphere. As you can see, developers connect just like end-users through the Enterprise Portal layer. The development tools also use HTTP services to drive metadata access. Therefore developers can be spread out over the enterprise without having to have direct point to point access. The SAS services registered to the application provide the metadata on service endpoint and column information. SAS of course can access almost any database through its access engines. It can also consume another data web service that could be surfaced within a Stored Process. With the SAS platform at our fingertips, we can develop our RIAs. The following section describes this platform and components used by RIAFusion.

[3] An ESB is the piece of software that lies between the business applications and enables communication among them. An ESB generally provides an abstraction layer on top of an implementation of an enterprise messaging system, which allows integration architects to exploit the value of messaging without writing code.

SAS INTELLIGENCE PLATFORM

Since our back-end service provider is SAS, we felt it was important to provide some background of the platform. The SAS Intelligence Platform is a complete infrastructure for developing, controlling, protecting, and delivering intelligence about the enterprise. The platform encompasses the complete information life cycle from data integration, optimized data storage, authentication and authorization, advanced analytics, and business intelligence. There are many software components that make up the SAS Intelligence platform, but this paper will focus on SAS Metadata Server, SAS Management Console, Stored Processes, and the SAS BI Java API.

The *SAS Metadata Server* is the cornerstone of the Intelligence Platform. The Metadata Server is the central repository for storing information about servers, data sources, users, authorization/security rights, stored processes and reports among others.

The *SAS Management Console* is the control center for Metadata Server management. The SAS Management Console allows a user to import table data from an RDBMS, create stored processes, configure and maintain user security along with other administrative functions.

The *SAS Stored Process* is a fundamental part of the SAS Intelligence Platform. A Stored Process is a SAS program that has been registered in the SAS Metadata Server and can then be consumed by various SAS Business Intelligence clients. A Stored Process can have input parameters to increase flexibility of the program. It might run some advanced analytics on a complex data structure and then generate a custom report.

The *SAS BI Java API* was used to build custom Java web applications to interact with the SAS Metadata Server. Java was the language of choice since SAS provides a very robust Java application programming interface.

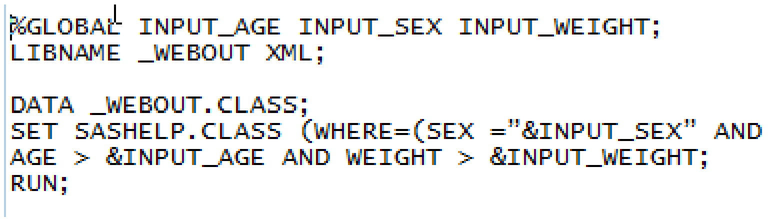
SERVICE REGISTRATION

We created a Java web application using Struts to handle service registration and execution. We created this as a web application since more individuals spread across the enterprise may want to create and register services to be used by the more narrow set of RIA developers. The RIA Fusion Registry server provides a standards based interface using HTTP and web services to access any available data source. Using this application we can register both JDBC services and SAS STP services[4]. The focus of building rich internet applications around the SAS Intelligence Platform starts with the SAS Metadata Server and the registration of a SAS program code as a Stored Process. We detail each step of this process below.

Step 1. Write SAS Program

The first step is to write a SAS program. The program may have multiple parameter inputs and there might be a large amount of complicated logic, but the main goal of the program is to render a table as XML output.

The example for this application will be a simple program that accepts three input parameters, subsets a simple dataset CLASS from the SASHELP demonstration data, and then renders the results of the query as an XML response back to the user.



```
%GLOBAL INPUT_AGE INPUT_SEX INPUT_WEIGHT;
LIBNAME _WEBOUT XML;

DATA _WEBOUT.CLASS;
SET SASHELP.CLASS (WHERE=(SEX ="&INPUT_SEX" AND
AGE > &INPUT_AGE AND WEIGHT > &INPUT_WEIGHT;
RUN;
```

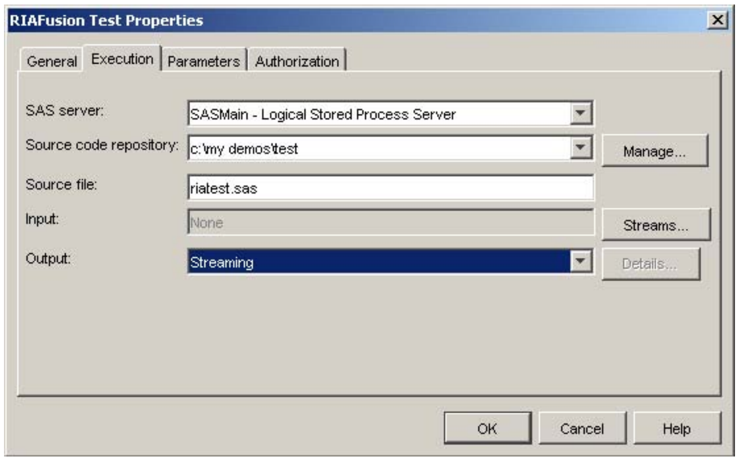
Figure 2 SAS Stored Process Code -riatest.sas

Create a file called "riatest.sas" and enter the SAS code as shown in Figure 2. The program defines three global macro variables whose values are expected to be passed-in as parameters: INPUT_AGE, INPUT_SEX, and INPUT_WEIGHT. The second key feature of the program is the use of the SAS XML libname engine (SXLE). The SXLE has the ability to read in XML files or write out XML files via this libname engine. This example will be writing a subset of data from SASHELP.CLASS as XML output via the _WEBOUT output stream via the SAS Stored Process Server.

[4] For this paper we will only discuss the SAS STP registration process.

Step 2. Create a Stored Process

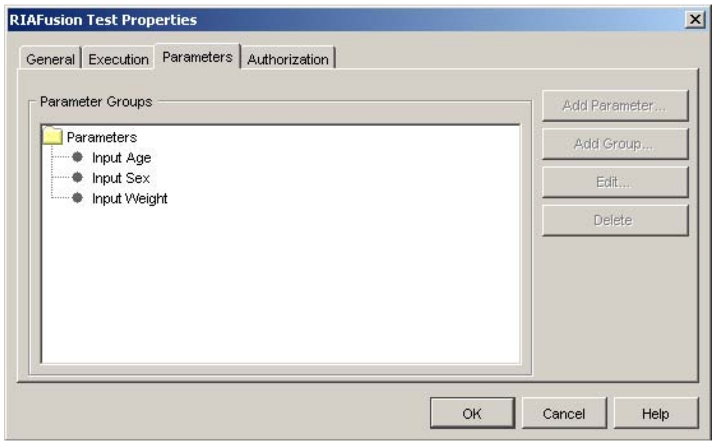
The second step is to register the “riatest.sas” file as a stored process in the SAS Management Console. Start up the SAS Management Console to begin the creation of a Stored Process Entry. Use the BI Manger Plug-in to create the SAS Stored Process, called “RIAFusion Test”. Add a keyword to the Stored Process called “RIAFusion” on the General tab. RIAFusion keyword is used by the service registration web application to list only those stored processes that are specifically built for use within the Rich Internet Application framework. Make sure that a Stored Process Server is the selected SAS Server, and the output is Streaming. Figure 3 shows the configuration of the Execution Tab.



SAS server: SASMain - Logical Stored Process Server
Source code repository: c:\my demos\test
Source file: riatest.sas
Input: None
Output: Streaming

Figure 3 RIAFusion Test – Execution values

The “RIAFusion Test” stored process is configured to use parameters. As with the code shown in Figure 2, three parameters were created for this stored process. “Input Age” parameter has a SAS variable name of input_age, with a variable type of integer, and has a default value of 12. “Input Weight” parameter has a SAS variable name of input_weight, with a variable type of integer, and has a default value of 80. “Input Sex” parameter has a SAS variable name of input_sex, with a variable type of string, and has a default value of M.

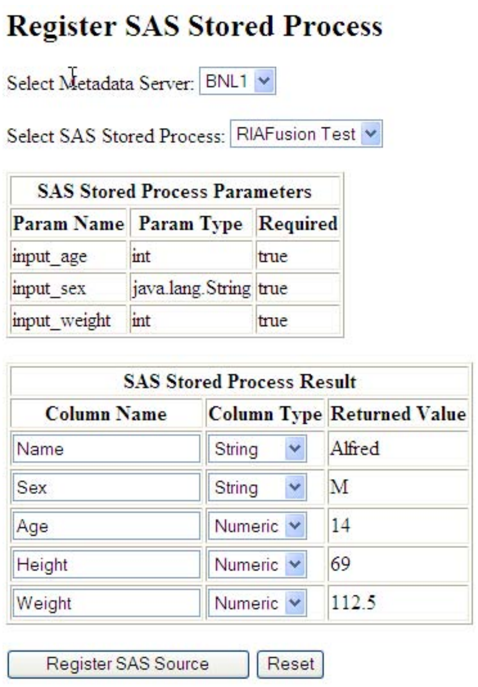


Parameters
- Input Age
- Input Sex
- Input Weight

Figure 4 RIAFusion test - parameters

Step 3. STP Service Registration

The third step to using a SAS Stored Process in the Rich Interface Application framework is to register the service to be an available data source. The registration of the "RIAFusion Test" is done via the registration wizard. First, the user selects an available SAS Metadata Server from the drop down list. Next, the registration wizard communicates with the SAS Metadata server to retrieve all of the stored processes that have the keyword "RIAFusion" . Next, select a stored process to use. The registration wizard will again communicate with the SAS Metadata server to find out more detailed information about the selected stored process and as shown in Figure 5, all of the parameters that were configured during the creation of the stored process. If the selected stored process has no parameters required, the user will see a message that no parameters were found. Next, the user will be presented with a list of columns, their column type, and a sample row of data that was returned by executing the stored process. The user can change the column type if necessary. Finally the user clicks the Register SAS Source button to add the stored process to the Service Registration Database. These SAS services are now available to be discovered and then executed.


Register SAS Stored Process

Select Metadata Server: BNL1

Select SAS Stored Process: RIAFusion Test

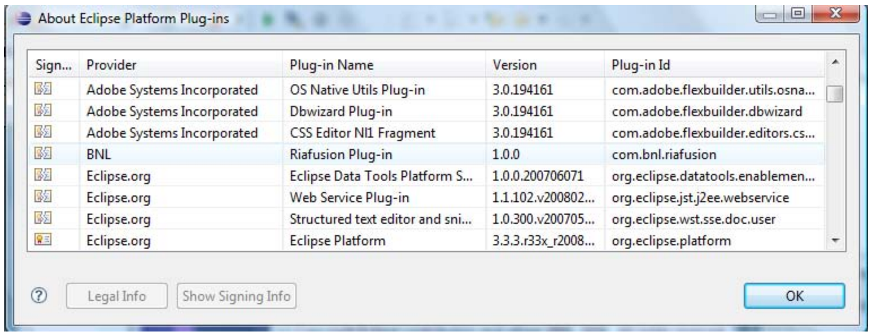
SAS Stored Process Parameters		
Param Name	Param Type	Required
input_age	int	true
input_sex	java.lang.String	true
input_weight	int	true

SAS Stored Process Result		
Column Name	Column Type	Returned Value
Name	String	Alfred
Sex	String	M
Age	Numeric	14
Height	Numeric	69
Weight	Numeric	112.5

Figure 5 Registration wizard

ECLIPSE INTEGRATION

Eclipse is a software platform comprising extensible application frameworks, tools and a runtime library for software development and management. It is written primarily in Java to provide software developers and administrators an integrated development environment (IDE). The Eclipse Rich Client Platform (RCP) is a special version of Eclipse that allows for the development of plug-ins to Eclipse. Many vendors including SAS are developing their client applications as Eclipse plug-ins. Another example is Adobe Flex. BNL Consulting has followed suit to create a plug-in for Eclipse called RIAFusion.


About Eclipse Platform Plug-ins

Sign...	Provider	Plug-in Name	Version	Plug-in Id
	Adobe Systems Incorporated	OS Native Utils Plug-in	3.0.194161	com.adobe.flexbuilder.utils.osna...
	Adobe Systems Incorporated	Dbwizard Plug-in	3.0.194161	com.adobe.flexbuilder.dbwizard
	Adobe Systems Incorporated	CSS Editor Nl1 Fragment	3.0.194161	com.adobe.flexbuilder.editors.cs...
	BNL	Riafusion Plug-in	1.0.0	com.bnl.riafusion
	Eclipse.org	Eclipse Data Tools Platform S...	1.0.0.200706071	org.eclipse.datatools.enablemen...
	Eclipse.org	Web Service Plug-in	1.1.102.v200802...	org.eclipse.jst.j2ee.webservice
	Eclipse.org	Structured text editor and sni...	1.0.300.v200705...	org.eclipse.wst.sse.doc.user
	Eclipse.org	Eclipse Platform	3.3.3.r33x_r2008...	org.eclipse.platform

Figure 6 RIAFusion Eclipse plug-in

One of the main issues a developer faces when building RIAs with tools like Flex is connecting to and locating the data sources that drive the user interface. The data sources can be wide ranging from relation databases, web services, http services and local XML. Flex has the capability to make HTTP or Web services calls to dynamically interact with data and feed the Flash objects such as tables, graphs etc. The problem with what comes out of the box is that you have to know specific URLs and the data items that are available from these service endpoints. In the heterogeneous world of data, this can be problematic, especially as organizations move towards SOA architectures. To fill this void, we created the RIA Fusion Registry server web application (described and illustrated above) and the RIA Fusion Eclipse plug-in. In order to increase the productivity of the report developer we decided to enhance the Adobe Flex development experience by creating the RIA Fusion Eclipse plug-in. This plug-in gives the developer access to all of the data sources that have been defined in the RIA Fusion Registry from the RIA Services View inside of Eclipse. Figure 7 below shows the RIA Services View which is available from the Show View menu or by right clicking on the Flex design page.

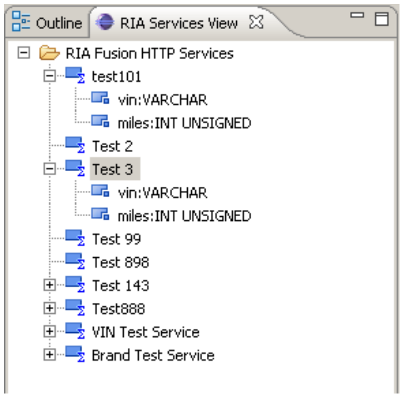
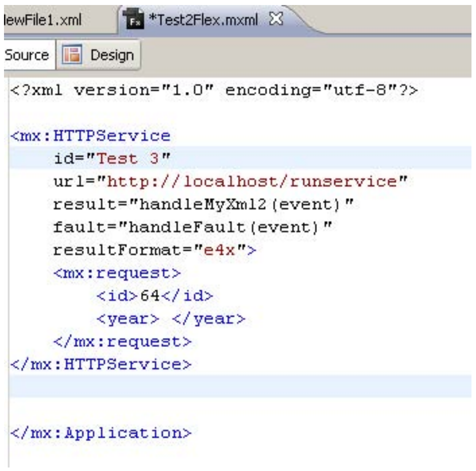


Figure 7 RIA Services View

From the RIA Services View a developer can drag and drop a service onto a Flex .mxml file and the associated .mxml datasource is automatically created from the information defined in the Service Registry. Figure 8 below shows the resulting code. The developer also has drag and drop access to any of the result columns for a datasource. When the report is run a call is made to the RIA Fusion Registry to execute the defined service with the given parameters. The result is an XML document that conforms to the Flex standards. The developer can easily test the application from Eclipse by clicking on the Run menu. Once they package up the RIA application, they can deploy it directly to any web server or servlet container. In the next section we illustrate the results of this process.



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

<mx:HTTPService
    id="Test 3"
    url="http://localhost/runservice"
    result="handleMyXml2(event)"
    fault="handleFault(event)"
    resultFormat="e4x">
    <mx:request>
        <id>64</id>
        <year> </year>
    </mx:request>
</mx:HTTPService>


</mx:Application>
```

Figure 8 RIA Services MXML tag

EXAMPLE RIA

With this simple addition to the powerful Flex development library, we can create an unlimited set of dashboards or applications, ranging from simple to the complex. To showcase the RIAFusion concept, we quickly developed a simple Flex application. The view below shows the Flex design perspective including a table object with a submit button.

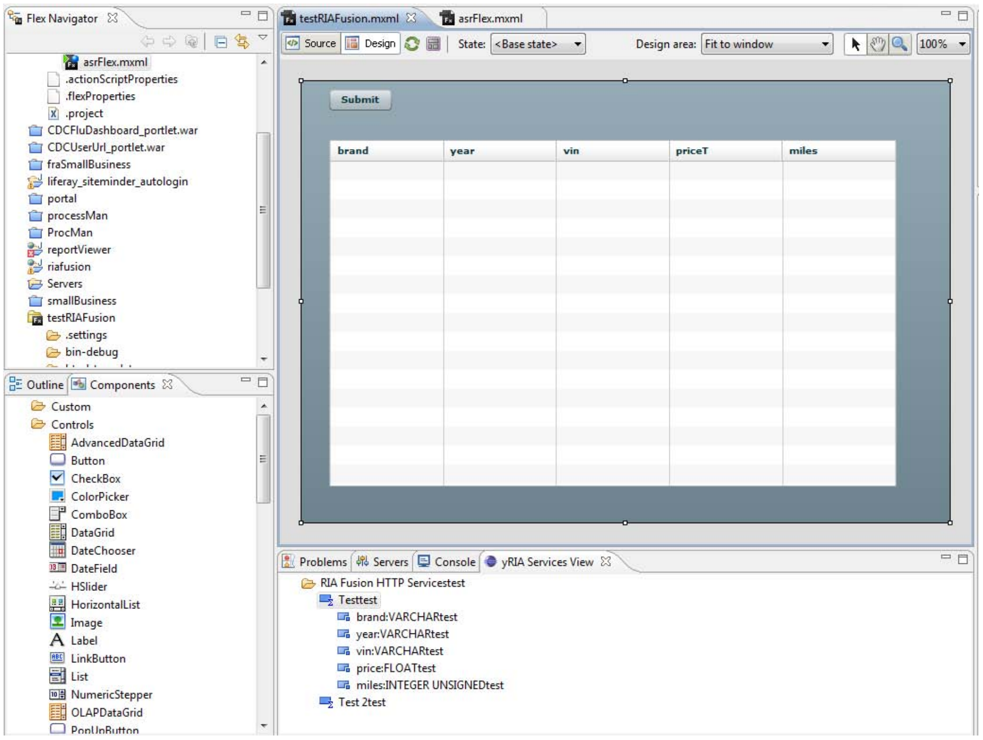
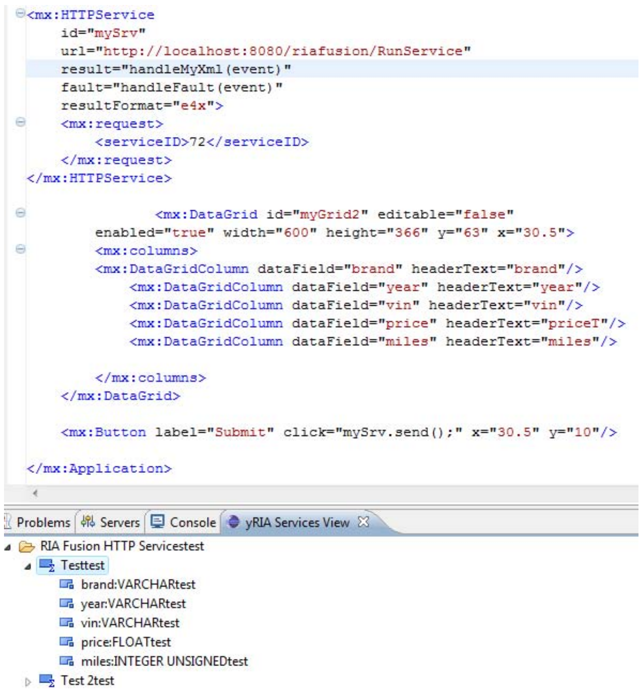


Figure 9 Flex design view

To view and edit the source code, we switch to the source view. The source view shows the results of dragging and dropping the first RIAFusion service listed in our Services View.

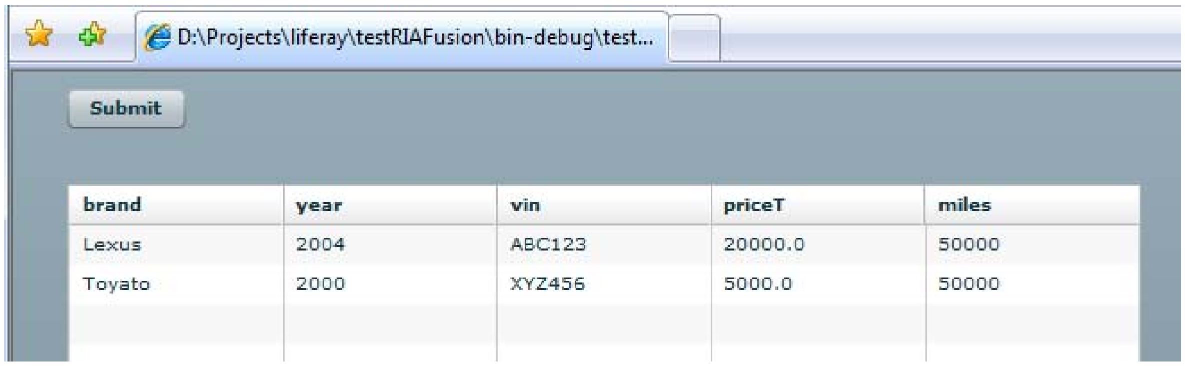


```
<mx:HTTPService
    id="mySrv"
    url="http://localhost:8080/riafusion/RunService"
    result="handleMyXml(event)"
    fault="handleFault(event)"
    resultFormat="e4x">
    <mx:request>
        <serviceID>72</serviceID>
    </mx:request>
</mx:HTTPService>

                <mx:DataGrid id="myGrid2" editable="false"
        enabled="true" width="600" height="366" y="63" x="30.5">
        <mx:columns>
        <mx:DataGridColumn dataField="brand" headerText="brand"/>
            <mx:DataGridColumn dataField="year" headerText="year"/>
            <mx:DataGridColumn dataField="vin" headerText="vin"/>
            <mx:DataGridColumn dataField="price" headerText="priceT"/>
            <mx:DataGridColumn dataField="miles" headerText="miles"/>

        </mx:columns>
    </mx:DataGrid>

    <mx:Button label="Submit" click="mySrv.send();" x="30.5" y="10"/>

</mx:Application>
```

```
RIA Fusion HTTP Servicestest
  Testtest
    brand:VARCHARtest
    year:VARCHARtest
    vin:VARCHARtest
    price:FLOATtest
    miles:INTEGER UNSIGNEDtest
  Test 2test
```

Figure 10 Flex source view

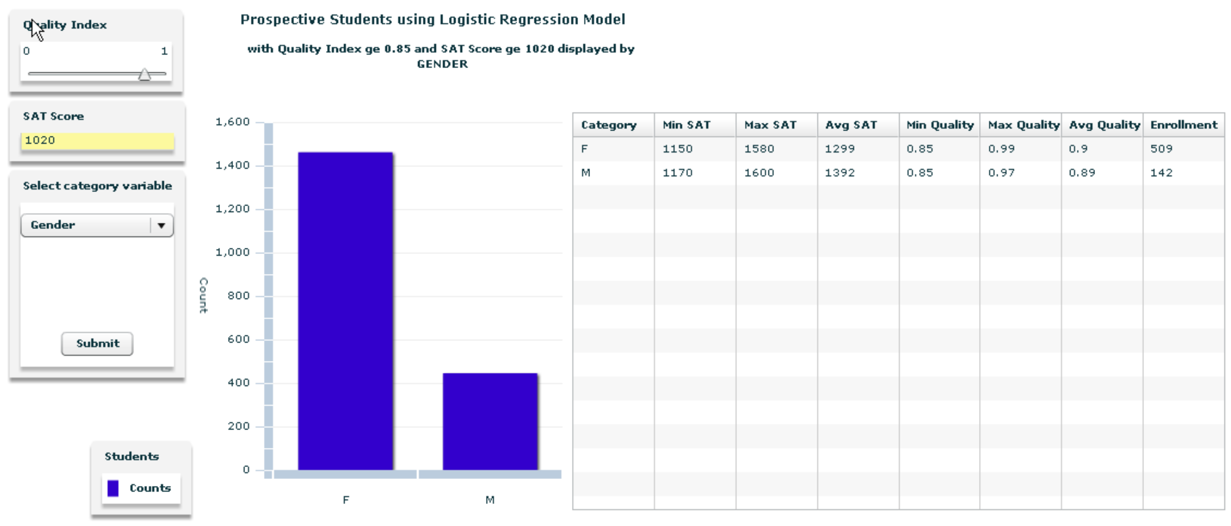
To test our code, we simply click on the run menu and run as Flex application. The results show the execution of the Flex application within a browser. This requires the Flash plug-in to be installed.



brand	year	vin	priceT	miles
Lexus	2004	ABC123	20000.0	50000
Toyato	2000	XYZ456	5000.0	50000

Figure 11 Flex results using RIAFusion services

With the SAS integration, we can now also run very sophisticated data routines through the SAS Stored Process mechanism and surface the results to the Flex objects. This opens up all the analytic power of SAS, data manipulation through the 4GL and access to the enormous library of SAS PROCs to provide a host of business and analytic intelligence functions. The results of the routines are then exported as XML and consumed by the RIA. The example application below takes in a set of parameters defining a cohort of prospective students. A regression routine is run by SAS as a service against the database of students. The resulting coefficients are applied against this database of prospective students filtered by the requested Quality Index and minimum SAT score. The counts and other metrics are then displayed in the bar chart and table. This is all done in real-time.



SAT Score: 1020

Select category variable: Gender

Prospective Students using Logistic Regression Model

with Quality Index ge 0.85 and SAT Score ge 1020 displayed by GENDER

Category	Min SAT	Max SAT	Avg SAT	Min Quality	Max Quality	Avg Quality	Enrollment
F	1150	1580	1299	0.85	0.99	0.9	509
M	1170	1600	1392	0.85	0.97	0.89	142

Figure 12 Complex RIA application with SAS analytics

CONCLUSION

With the development of the RIAFusion application and Eclipse plug-in, we have opened the gate to easily develop a RIA using SAS as a service provider. This simple but powerful integration will further serve to exemplify the power of the SAS platform and how it can go beyond traditional query and reporting tools to process and render data. While we used Flex for our RIA development, there are other RIA development tools available or soon to be released such as Microsoft's Silverlight. RIAs are here to stay and providing vendor and platform agnostic access to data will only improve upon their value and use across the enterprise. SAS with its rich history of data processing and powerful analytics, along with its large customer base will be a key service provider in this movement.

CONTACT INFORMATION

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

Robert Lill
President
BNL Consulting, LLC
301-760-0709
Rob.lill@bnl-consulting.com

Steven Tuttle
SAS Institute Inc.
713-962-5090
Steven.tuttle@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.