#### Paper 004-2011

# **Building Enterprise Applications Using SAS® Real-Time Services**

Falko Schulz, SAS Institute Inc., Brisbane, Australia

# ABSTRACT

SAS<sup>®</sup> Web services can help you meet the challenges of integrating the processes in your service-oriented architecture (SOA). Real-time services allow you to leverage the power of SAS across the enterprise and beyond. This paper demonstrates how to create SAS Web services and how to call these Web services from a third-party application. Increase your return on investment by integrating more processes in your SOA with SAS Web services.

**Note**: A zip file with code examples from this paper is available at <a href="http://www.sascommunity.org/wiki/Building\_Enterprise\_Applications\_using\_SAS\_real-time\_services">http://www.sascommunity.org/wiki/Building\_Enterprise\_Applications\_using\_SAS\_real-time\_services</a>.

### INTRODUCTION

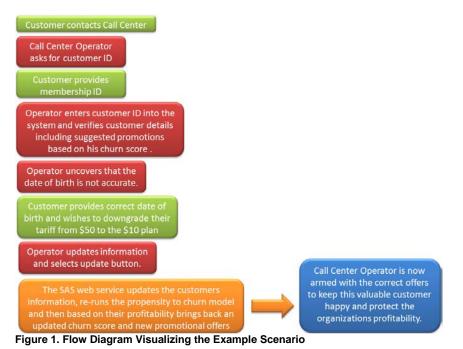
SAS 9.2 has excellent capabilities to create Web services that surface SAS advanced analytics. Using Web services standards ensures that these services are available to all applications in your SOA.

SAS BI Web services expose SAS stored processes as Web services, making it easy for other applications to call SAS code. A Web service is described by a Web Service Description Language (WSDL). WSDL is an XML file that describes the set of operations that a service contains as well as the inputs and outputs of each operation.

# **EXAMPLE SCENARIO**

For the purpose of this document the following scenario is used:

There is the requirement to build a new call center interface to help call center operatives make the best possible tariff recommendations to customers. The interface should be able to retrieve the customer's details by providing the customer's phone number or ID. The operator should also be able to make changes to the customer's details. Finally, the interface needs to retrieve the customers churn score and make promotion offers in real time.



The following figure shows the proposed interface. Technology used to develop the user interface is Adobe Flex.

di m	<b>orld Corp Call Ce</b> line Client Query Applicati t View Help				and the
ID: *	12 Segment	Quick Adopters		hurn Score	Call Rate by Month
Title:	Mr First: Bob	MI: Last: Muttom	100	\$700 9	A
Address1: Address2:	1 Eagle St		80	- \$600 - \$500 6	$ \sim                                   $
Postcode: State:		Dane Phone: 0298418888 Muttom@cor Gender: U - Undefined	60	5 - \$400 4	$\wedge \vee \vee$
DOB: Commen	21/01/1968 Tariff: P	repay \$50	40	- \$300 <sup>3</sup> 2 - \$200 1	
		📓 Update 💋 Sea	20		Jan Feb Mar Apr Ma
Current	History	Open:		\$0	Calls Claims
Call Number	0298418888	O Solved: 08/17/2010	Promotion		
ACTIVE	PAUSE RESUME	Agent ID:	Detai	Is: 200 free minutes call tim	e to any network each month
Call in Queu	Je: 0	Notes:	O Promotion :	2: HST004	
Time:	17/08/2010 11:23:26		Detai	Is: Free handset upgrade an	d 500MB free data per month
		🛃 Upd	ate		Apply Promotion

Figure 2. Call Center Application

# **CREATING THE SAS WEB SERVICE**

SAS BI Web services are a part of SAS<sup>®</sup> Integration Technologies, which runs on the middle (Web) tier. Given the call center application design, there are two Web services that need to be built.

ID: 🐐	12	S	egment:	Quick Ad	lopters		
Title:	Mr Fir	st: Bob		MI:	Last	Muttom	
Address1:	1 Eagle St	i -					
Address2:							
Postcode:	4001	City:	Brisba	ne	Phone:	0298418888	
State:	QLD	Email:	Bob.M	uttom@cor	Gender:	U - Undefined	
DOB:	21/01/19	68 📰 Tariff	: Pre	pay \$50			
Commen						~	
				-	-		
					Updat	e 🔍 Sea	rch



- Customer Detail Record Service a Web service returning the customer details by providing the customer ID or phone number
- Customer Update Service a Web service updating the customer detail record and returning the recalculated churn score

Each Web service will be executed by clicking either the **Update** or the **Search** button.

# CUSTOMER DETAIL RECORD SERVICE

This Web service returns the customer details by a given customer ID or phone number. A service is a SAS program executed by the SAS<sup>®</sup> Stored Process Server. Parameters passed to this service are available as SAS macro variables and results of the SAS program are streamed back as XML as part of the Simple Object Access Protocol (SOAP).

This paper uses the following sample table as data source (you can download the full table at <a href="http://www.sascommunity.org/wiki/Building\_Enterprise\_Applications\_using\_SAS\_real-time\_services">http://www.sascommunity.org/wiki/Building\_Enterprise\_Applications\_using\_SAS\_real-time\_services</a>).

	custID custTitle	cust Segment	cust First Name	custLastName	custAddress1	custCity	cust State	custPostcode	custPhone	custEmail	custGender	custDoB	cust Tariff	custProb	custLastMonthVal
1	1	E	Kenneth	Foreman	P.O. Box 562, 7657 Orci St.	Nevada City	NJ	54400	(745) 100-2227	scelerisque.mollis	F	01/22/1962	18	0	57.
2	2 Dr.	В	Leslie	Crawford	496-5397 A, Av.	Waycross	Montana	82822	(426) 530-2889	a.facilisis@aliqua	F	08/24/1966	17	0	63
3	3 Dr.	С	Igor	Lott	493-6218 Facilisis Rd.	Salem	OK	8779	(155) 192-7676	non.ante@Nuncm	F	06/06/1979	8	1	69
4	4	С	Raphael	Gillespie	P.O. Box 256, 2192 Ante Rd.	New Britain	Colorado	55537	(808) 888-9647	per.inceptos@quis	М	06/18/1984	4	1	66
5	5 Dr.	E	Amy	Philips	524-7754 Tellus, Avenue	Palo Alto	WA	43395	(205) 104-4473	sed.hendrent.a@n	М	02/11/1961	12	0	68
6	6 Mr.	С	Jamalia	Chase	Ap #903-3093 Ullamcorper Avenue	Kenosha	MO	43877	(668) 487-4405	convallis.ante@au	М	12/06/1970	4	0	69
7	7	E	Ryan	Hatfield	665 Gravida Street	Spokane Valley	Montana	44030	(340) 414-3312	et@Donectempus	M	06/20/1969	3	0	69
8	8 Mr.	A	Melodie	Simpson	P.O. Box 447, 3081 Nec, St.	Fremont	Alaska	50649	(512) 941-7119	in.consequat.enim	F	08/19/1974	17	1	51
9	9	С	Beau	Bentley	P.O. Box 536, 7781 Nec. Av.	Clairton	Louisiana	89143	(636) 217-2147	vel.venenatis.vel	F	10/18/1973	1	1	67
10	10 Ms.	D	Raphael	Holcomb	Ap #839-7056 Feuglat Avenue	Mesquite	AK	40719	(198) 876-3462	euismod.uma@ne	м	04/25/1974	16	1	67
11	11 Dr.	A	Kessie	Ellis	Ap #976-1288 Risus. Av.	Brown Deer	New Jersey	56599	(496) 365-7412	lacus@purusNulla	F	08/28/1960	8	0	68.
12	12 Dr.	E	Deirdre	Cook	Ap #884-2367 Dignissim Avenue	Ventura	Alabama	9580	(964) 973-4389	Duis.a.mi@neccur	М	05/24/1970	16	0	63
13	13	В	August	Kemp	6963 Ultrices, Rd.	Miami Gardens	MO	57851	(584) 105-7449	dolor.Fusce@phar	F	01/25/1963	5	0	593
14	14 Mr.	A	Sierra	Woodward	679-5118 lpsum Rd.	Watertown	Nevada	61938	(463) 497-8909	amet@InforemDon	M	03/31/1987	16	0	67
15	15 Mr.	E	Kendall	Valentine	5541 Ac Road	Hoboken	MA	11943	(526) 360-1669	Aliquam@Nunc.c	M	01/12/1978	3	.1	68
16	16 Dr.	C	Nathaniel	Shepard	Ap #710-8711 Lorem Av.	Norfolk	AK	77986	(836) 106-5944	tristique@augueid.	F	05/04/1976	17	0	54
17	17 Mrs.	В	Aimee	Floyd	588-9628 Tincidunt Rd.	Moline	New Hampshire	9397	(663) 995-0078	vel.lectus.Cum@f	F	08/19/1977	4	0	55
18	18 Dr.	C	Dieter	Gonzalez	450-1948 Ridiculus Street	Evanston	Indiana	69939	(262) 965-7515	massa@egetdictu	F	01/05/1978	4	1	52
	19 Mr.		Daphne	Nunez	P.O. Box 459, 4758 Sed Avenue	Huntington Beach	LA	55528		Nain ad nulla@Fu	Ę	04/25/1987	5		59

Figure 4. Sample Customer Records

For demonstration purposes, the following SAS program is used. You will need to change the program accordingly to retrieve data from other data sources.

The program below requires the following parameter to work correctly:

• \_custID = Customer ID

```
Returns the data for a given table
*/
%macro getCustomerDetails;
       /* filter the customer data set by given customer ID */
       data data;
               set temp.customers;
               where custID = &_custID;
               custComments = "";
               custPromotion = "";
               custPromolID = "";
               custPromo1Desc = "";
               custPromo2ID = "";
               custPromo2Desc = "";
       run;
%mend;
/* basic parameter validation */
%macro validateParameters;
       %if not %symexist( custID) %then %do;
               %global custID;
               %let _custID=;
       %end;
       %if not %symexist(_custPhone) %then %do;
               %global custPhone;
               %let _custPhone=;
       %end;
       %else %if not %symexist( custID) or not %symexist( custPhone) %then %do;
               data data;
                 error = "1";
                 message = "Parameter ' custID' or ' custPhone' is missing.";
               run;
       %end; %else %do;
               %checkds(temp.customers);
               %getCustomerDetails;
       %end;
%mend;
%validateParameters;
```

/* write out XML stream */ libname _WEBOUT xml;
<pre>data _WEBOUT.CALL_CENTER_WEB_SERVICE_OUTPUT;    set data;</pre>
run;

Figure 5. Customer Detail Record Service

The SAS program needs to be registered as a stored process so it can be deployed as a Web service. This is done via SAS<sup>®</sup> Management Console. The documentation for SAS Management Console shows the process for registering the SAS program as a stored process. There are two important steps to deploy a SAS Stored Process as a Web service:

- specifying the XMLA Web Service keyword (Figure 6. General Properties)
- filling in the dialog boxes to define the parameters (Figure 7. Parameters)

Customer Detail	Service Properties			Customer Detai	l Service Properties			X
General Exec	ution Parameters Data Authorization			General Exec	ution Parameters Da	ta Authorization		
Na <u>m</u> e:	Customer Detail Service			Prompts (input	parameters):			
Type:	Stored process				Displayed Text	Name	Туре	New Prompt
Description:				🧭 Parameter			Standard group	New Group
				Custor	mer ID mer Phone Number	_custID custPhone	Text Text	
						-	The second secon	Edit
								Delete
								Move Up
								Move Down
Location:	/Projects/SGF2011							
Created:	16/08/10 13:38							Add Shared
Modified:	17/08/10 13:00							Save as Shared
Keywords:	XMLA Web Service		<u>A</u> dd					Unshare
			Delete					
				Output parame	aters:			Test Prompts
				Label	Name	Туре	Description	New
				Labor .	- Tanto	1)00	b obtription i	Edit
		Ť.						
Responsibilities	Name	Role	A <u>d</u> d					Delete
			Delete					
	1							
		OK Car	ncel <u>H</u> elp				ОК	Cancel <u>H</u> elp
				U				

Figure 6. General Properties



Once the SAS program has been registered as a stored process it can be deployed as a Web service. This is done using the SAS Web Service wizard from SAS Management Console. Access this wizard by right-clicking the stored process and selecting **Deploy As Web Service** from the context menu:



Figure 8. Access 'Deploy As Web Service' Wizard

Deploy As Web Service		Deploy As Web Service	
Web Service Information Specify the URL for the SAS BI Web Services WebServiceMaker and a name for the new Web service be deployed.	to	Web Service Keywords and Namespace Enter the namespace and keywords associated with this web service.	5
Web Service Maker URL: http://ssafas.apac.sas.com:8080/SASBIWS/services/Web	ServiceMaker 👻	Namespace: http://tempuri.org/customerDetailService	1
New Web Service Name: customerDetailService		Keywords: Add keyword	il
Use my current credentials to deploy		Delete keyword	
O Use these credentials to deploy:			e.
User ID:			
Password:			
<back next=""> Finish Cancel</back>	Help	< Back Next > Finish Cancel Help	]
Figure 9. Web Service Information		Figure 10. Web Service Keywords	
Deploy As Web Service		Deploy As Web Service	
Confirm Web Service Deployment Confirm the following parameters which will be passed to the WebServiceMaker to deploy a new Web service. This operation may take several minutes and cannot be cancelled once started.		Confirm Web Service Deployment Confirm the following parameters which will be passed to the WebServiceMaker to deploy a new Web service. This operation may take several minutes and cannot be cancelled once started.	
Web Service Maker URL: http://ssafas.apac.sas.com:8080/SASBIWS/services/WebServiceMaker		Web Service Maker URL: http://ssafas.apac.sas.com:8080/SASBIW5/services/WebServiceMaker	1
New Web Service Name: customerDetailService		New Web Service Name: customerDetailService	il
Namespace: http://tempuri.org/customerDetailService		Namespace: Web Service Successfully Deployed	]
Stored Processes:		Stored Process IProjects/SGF Web service successfully deployed to the following location:	
/Projects/SGF2011/Customer Detail Service		//Projects/SGF       Image: Sast Come: 8080/SASBIWS/services/customerDetailService         Keywords:       Image: Sast Come: 8080/SASBIWS/services/customerDetailService	
< Back Next > Finish Cancel	Help	<back next=""> Finish Cancel Help</back>	]

Figure 11. Web Service Deployment

Figure 12. Web Service Deployed

After the Web service is deployed to the application server a confirmation dialog box will be displayed with the endpoint URL for the newly created service (Figure 12. Web Service Deployed).

The Web Service Definition Language (WSDL) file can be accessed by adding "?wsdl" to the endpoint URL (Figure 13. WSDL File).

Hint: To copy this URL from the dialog box, use Ctrl-C from the keyboard.

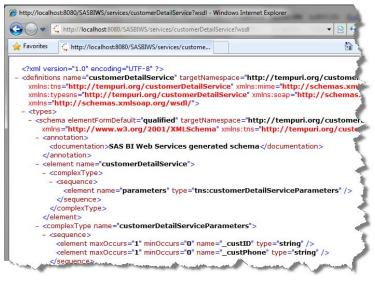


Figure 13. WSDL File

#### **CUSTOMER UPDATE SERVICE**

This service will be used to update the customer detail record. It should be capable of retrieving new customer details, which then get updated in the underlying database. This service will also rescore the likelihood of churn based on the new details received. Finally, this service will provide recommended promotions, which can be offered to a customer to prevent churn.

The program below requires the following parameters to work correctly:

- custID = Customer ID
- \_custPhone, \_custTitle, \_custSegment, \_custFirstName, \_custLastName, \_custAddress1, \_custCity, \_custState, \_custPostcode, \_custPhone, \_custEmail, \_custGender, \_custDoB, \_custTariff

```
Returns the data for a given table
*/
%macro updateCustomerDetails;
       /* update customer data set with new values */
       proc sal;
               update temp.customers
               set custTitle = "&_custTitle",
                       custSegment = "& custSegment",
                      custFirstName = "& custFirstName",
                       custLastName = "&_custLastName",
                       custAddress1 = "& custAddress1",
                       custCity = "& custCity",
                       custState = "& custState",
                       custPostcode = & custPostcode,
                       custPhone = "&_custPhone",
                       custEmail = "& custEmail",
                       custGender = trim("& custGender"),
                      custDoB = input("& custDoB", ddmmyy10.),
                      custTariff = & custTariff
               where custID = \& custID;
       quit;
       /* re-calculate the churn score */
       data data;
               set temp.customers;
               where custID = & custID;
```

```
/* Replace churn score calculation with real SAS Enterprise Miner model: */
               custProb = ranuni(-1) * 100;
               if (custProb > 80) then do;
                   custPromolID="TAR001"; custPromolDesc="Any tariff Upgrade free for 1 month";
                   custPromo2ID="FAM003"; custPromo2Desc="Family and Friends trial package free
for 3 months";
               end:
               else if (custProb > 60) then do;
                   custPromolID="FAM002"; custPromolDesc="Family and Friends trial package free
for 2 months";
                   custPromo2ID="MIN001"; custPromo2Desc="100 free minutes call time to any
network each month";
               end;
               else do;
                   custPromolID="MIN002"; custPromolDesc="200 free minutes call time to any
network each month";
                   custPromo2ID="HST004"; custPromo2Desc="Free handset upgrade and 500MB free
data per month";
               end;
       run;
%mend:
/* some basic parameter validation */
%macro validateParameters;
       %if not %symexist(_custID) %then %do;
               %global custID;
               %let _custID=;
       %end;
       %if not %symexist(_custPhone) %then %do;
               %global custPhone;
               %let custPhone=;
       %end;
       %else %if not %symexist( custID) or not %symexist( custPhone) %then %do;
               data data;
                 error = "1":
                 message = "Parameter ' custID' or ' custPhone' is missing.";
               run;
       %end; %else %do;
               %updateCustomerDetails;
       %end;
%mend;
%validateParameters;
/* write out XML stream */
libname WEBOUT xml;
data _WEBOUT.CALL CENTER WEB SERVICE OUTPUT;
 set data;
run;
```

Register this SAS program following the same steps described for the "Customer Detail Service". The only difference is the service name (Figure 14. Customer Update Service) and parameters used (Figure 15. Parameters).

General Execution Parameters       Data Authorization         Name:       Outcome Nuclease Service         Type:       Stored process         Description:       Custome Fort Name         Customer Signers       Customer Signers         October:       Projects/SGF2011         Created:       1709(10 15:36         Modified:       1709(10 13:00         Responsibilities:       Name         Name       Responsibilities:         Name       Responsibilities:	Customer Update	e Service Properties		<b>X</b>	Customer Update Service Properti	es			×
Type:       Displayed Text       Name       Type       New Prompt         Description:       P Customer Tible       custTible       Text       New Group         Q Customer First Name       custRiddensame       Text       Edit         Q Customer First Name       custRiddensame       Text       Edit         Q Customer First Name       custRiddensame       Text       Edit         Q Customer First Name       custRiddensame       Text       Delete         Q Customer First Name       custRiddensame       Text       Delete         Q Customer Finder Name       custRiddensame       Text       Delete         Q Customer Address 1       custRiddensame       Text       Delete         Q Customer Address 2       custRiddensame       Text       Add Shared         Q Customer Address 1       custRiddensame       Text       Save as Shared         Q Customer Finder       custRiddensame       Text       Save as Shared         Q Customer Taiff       custRiddensame       Text       Text       Text         Q Customer Finder       custRiddensame       Text       Text       Text       Text         Q Customer Finder       custRiddensame       Text       Text       Text	General Execu	ition Parameters Data Authorization			General Execution Parameters	Data Authorization			
Description: <sup>2</sup> Customer ID <sup>2</sup> Custom	Name:	Customer Update Service			Prompts (input parameters):				
Description: <ul> <li>Q Customer Title</li> <li>Customer Ti</li></ul>	Type:	A Stored process			Displayed Text	Name	Туре		New Prompt
Location:     /Projects/SGF2011					Customer ID	_custID	Text		
Location:       //Projects/SGF2011	Description				Customer Title	_custTitle	Text		New Group
Location:       /Projects/SGF2011       Customer Kink Name       custAddress1       Text         -2       Customer Address 1       custAddress1       Text         -2       Customer Address 2       custAddress2       Text         -2       Customer Price       custFinal       Customer Final       Customer Final         -2       Customer Price       custFinal       Customer Final       Customer Final       Customer Final         -2       Customer Final       custFinal       custFinal       Customer Final       Text         -2       Customer Comments									Edit
Location:       Projects/SGF2011       Customer Last Name       CustAddress:       Text       Move Up         Created:       16/08/10 15:36       Customer Address:       Customer City       CustOmer City       CustOmer City       Exet       Add Shared         Wodified:       17/08/10 13:00       Customer Address:       Customer Phone       CustOmer Phone       CustOmer Phone       CustOmer Address:       Save as Shared         Qustomer State       CustOmer Gender       CustOmer Gender       CustOmer Mone       Save as Shared       Unchare         Qustomer State       CustOmer State       CustOmer Mone       CustOmer Mone       Save as Shared       Unchare         Qustomer Gender       CustOmer Mone       CustOmer Mone       CustOmer Mone       Text       Text         Qustomer Gender       CustOmer Mone       CustOmer Mone       CustOmer Mone       Text       Text<							Text		Lakin
Location:       /Projects/SGF2011       Customer Address 1       custAddress 1       text         Created:       16/08/10 15:36       Customer State       custAddress 2       custAddress 1       text         Modfied:       17/08/10 13:00       Customer State       custCare Text       Add Shared         Vegtomer Comments:       Customer Comments       custCare Text       Save as Shared         Output parameters:       Customer Comments       custCare Text       Text       Text         Output parameters:       Customer Comments       custComerts       Customer Comments       custComents       Text         Output parameters:       Customer Comments       custComerts       Customer Comments       Customer Comments       Text         Output parameters:       Customer Comments       custComerts       Customer Comments       Customer Comments       Customer Comments       Edit         Delete       Name       Type       Description       Edit									Delete
Location:       /Projects/5672011       Customer City       _custCity       Text         Created:       16/06/10 15:36									
Location:       [Projects/SGF2011       Customer City       custCity       Text       Add Shared         Created:       16/08/10 15:36       Customer Phone       custPhone       Text       Add Shared         Keywords:       XMLA. Web Service       Add       Delete       Customer Phone       custComerts       Text       Save as Shared         Q: Customer Phone       custComert DoB       custComert Text       Unshare       Text       Text       Text       Text       Text       Text       Text       Unshare       Unshare       Customer Phone       CustOmert Text       Save as Shared       Unshare       Unshare       Text       Text </td <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>Move Up</td>									Move Up
Location:       /Projects/SGF2011         Created:       16/06/10 15:36         Modified:       17/06/10 13:00         Keywords:       XPLA Web Service         Add       Q Customer Postcode         Q Customer Postcode       _custPostcode         Q Customer Comments       _custPostcode <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>Maus Daws</td>									Maus Daws
Created:       16/08/10 15:36       Customer Postcode       custPostcode       Text       Add Shared         Modified:       17/08/10 13:00       Customer Phone       custPostcode       Text       Save as Shared         Keywords:       XMLA Web Service       Add       Delete       Customer Content Final       custContent Final       Text       Unshare         Output parameters:       Customer Comments       custContent Final       custContent Final       Text       Test Prompts         Output parameters:       Customer Comments       custContent Final       Customer Comments       custContent Final       Test Prompts         Customer Social       Name       Name       Type       Description       Edt         Delete       Add       Delete       Delete       Delete       Delete       Delete	l annihina i	Investor the ICCEPOD 1						E	Move Down
Modified:       17/09/10 13:00       Customer Phone       custPhone       Text       Save as Shared         Keywords:       MLA Web Service       Add       Outcomer Email       custPhone       Text       Unshare         Q Customer Comments       custOmer Phone       custOmer Text       Outcomer Text       Unshare       Text       Text       Text       Text       Text       Unshare         Q Customer Comments       custComments       custComments       custComments       custComments       Text       Tex									Add Shared
Keywords:     XPILA Web Service     Add     Q     Customer Gender									
Keywords:       XMLA Web Service       Add         Delete       Add         Qustomer Gender      oustGender       Text         Qustomer Coll      oustGender       Text         Qustomer Coll      oustGender       Text         Qustomer Commerts      oustGender       Text         Delete       Name       Type       Description         Edit       Delete      oustGender      oustGender	Modified:	17/08/10 13:00							Save as Shared
Responsibilities     Name     Role     Add	Keywords:	XMLA Web Service	10	Add					( United and
Responsibilities:     Name     Role     Add								-111	Unshare
Responsibilities:         Name         Role         Add         Add         Image: Control of the				Delete				-	
Responsibilities:     Name     Role     Add     Image: Contraction of the second of th									
Responsibilities:     Name     Role     Add     Image: Construction of the construction of					Customer Comments	_custcomments	Text	-	Test Prompts
Responsibilities: Name Role Add Add					Output parameters:				
Responsibilities: Name Role Add Add					Label Name	Turne	Description		New
Responsibilities: Name Role Add Delete					Laber Ivalle	Type	Description		
I Valle NUE PULL									Edit
I Valle NUE PULL	Responsibilities								Delete
Delete	responsibilities.	Name	Role	Add					
				Delete					
					1				
OK Cancel Help OK Cancel Help			OK Cancel	] [ Help ]			OK		Cancel Help
							-		

Figure 14. Customer Update Service

Figure 15. Parameters

# **BUILDING THE CALL CENTER INTERFACE**

The technology used to build the user interface is Adobe Flex. *Rich Internet Applications* (RIA) are Web applications that have the usability of desktop software and give users a more interactive and enriching experience. They are deployable on all major browsers and operating systems, which removes the hassle of performing a client installation. Most systems that run an RIA have Adobe Flash Player installed, which is a prerequisite to running a Flex application.

**Note**: A zip file with code examples and the full application source code from this paper will also be available at <a href="http://www.sascommunity.org/wiki/Building\_Enterprise\_Applications\_using\_SAS\_real-time\_services">http://www.sascommunity.org/wiki/Building\_Enterprise\_Applications\_using\_SAS\_real-time\_services</a>

The following diagram visualizes the Web service process.



Figure 16. Overview of the Web Service Process

The Adobe Flex client application communicates directly with the middle-tier Java code, making a standardized call (SOAP) to the Web service. The middle tier then makes an Integrated Object Model (IOM) call to the SAS Application Server and executes the stored process. The results are streamed back to the middle tier and are finally returned to the client application.

### CALLING A WEB SERVICE IN FLEX

Flex consists of two languages: MXML and ActionScript. MXML is an XML-based language, which incorporates many built-in functions and is generally used to lay out the user interface. When compiled, each MXML tag is generated into ActionScript. ActionScript is a powerful object-oriented programming language. This section of the paper provides a brief overview of the important parts of the code that call the previously created Web service. A basic knowledge of Flex programming is required for understanding the content of this section.

To call your Web service, use the <mx:WebService> tag. This tag requires the WSDL address (http://localhost:8080/SASBIWS/services/customerDetailService?wsdl), the operation name

(customerDetailService), and any parameters ( custID, custPhone) that you want to pass into the Web service.

Here is an example of how your Web service call should be set up in Flex:

The only fields that are unfamiliar at this point should be in the <mx:operation name="">tag, specifically resultFormat, result, and fault. The resultFormat function lets the compiler know that the return type of the Web service will be an object. The result function executes when the operation is completed, and the fault function executes if any problems are encountered. Instead of specifying fix values for your parameters (for example, <\_custID>12</\_custID>12</\_custID>), you can use ActionScript to assign dynamic values using the following syntax: <\_custID> {custID.toString()}</custID>.

To call a Web service, the application uses a sendWSRequest function. The Web service gets executed by calling the send() function on the operation object as shown in the following code:

```
private function sendWSRequest(service:WebService, reqObj:Object):void {
    var operation:Operation = service.operations[0];
    // add the request object to the operation. The request object contains call parameters
    operation.request = reqObj;
    operation.send();
```

The onCustomerRecordRecieved function retrieves the results that are generated by the Web service and places them into an ArrayCollection:

```
private function onCustomerRecordRecieved(evt:ResultEvent):void {
       //retrieve the SAS table name from the XML property file
       var drWS:XMLList = xmlProperties..service.(@name == "DETAILRECORD");
       var tableName:String = String(drWS[0].@outputTable);
       // do we have a result stream
       if (evt.result.Streams) {
               // create a new array collection using the ArrayUtil class
               var orArray:ArrayCollection = new
ArrayCollection(ArrayUtil.toArray(evt.result.Streams. WEBOUT.Value.TABLE[tableName].source));
               if (orArray.length == 0)
                      orArray = new
ArrayCollection(ArrayUtil.toArray(evt.result.Streams._WEBOUT.Value.TABLE[tableName]));
               // if the array contains values use the field mappings to assign values
               if (orArray.length > 0) {
                       var firstRow:Object = orArray[0];
                      var fieldMappings:XMLList = drWS..field;
                      loadFields(firstRow, fieldMappings);
               }else{
                      Alert.show("Customer ID " + this.custID.text + " not found.");
               }
       }else{
                      Alert.show("Customer ID " + this.custID.text + " not found.");
       }
```

The preceding code snippet reads the SAS output table name from a property file, which configures the application. This way you do not need to re-compile your application if the Web service output table changes. The application also calls the loadFields function, which reads each customer record detail and assigns it to an input field.

Once all fields are loaded, the user interface shows the customer details:

ID: 🕸	12	Se	gment:	Quick Adopters				
Title:	Mr First	Bob		MI:	Last	Muttom		
Address1:	1 Eagle St							
Address2:								
Postcode:	4001	City:	Brisba	ne	Phone:	0298418888		
State:	QLD	Email:	Bob.M	uttom@cor	Gender:	U - Undefined		
DOB:	21/01/1968	Tariff:	Pre	pay \$50			•	
Commen								
				(	Updat	e 🖉 🔎 Sea	rch	

#### The complete Flex code can be downloaded at

http://www.sascommunity.org/wiki/Building Enterprise Applications using SAS real-time services. This code also produces an interactive bar chart, line plot, and other call center interface-related outputs. Figure 17 shows the final call center application interface.

A delaw	ine Client Query Applica			
ID: *	12 Segme	nt: Quick Adopters	Chu	rn Score Call Rate by Month
Address1: Address2: Postcode: State: DOB: Commen		isbane Phone: 0298418888 bb.Muttom@cor Gender: U - Undefined Prepay \$50	<ul> <li>80 -</li> <li>60 -</li> <li>▼</li> <li>40 -</li> <li>20 -</li> <li>b</li> </ul>	- \$600 8 7 - \$500 6 5 - \$400 4 - \$300 3 - \$200 1 - \$100 0 Jan Feb Mar Apr May
Current Call Number ACTIVE Call in Queu Time:	PAUSE RESUME	Open:     Solved: 08/17/2010  Agent ID:  Notes:  Upda	Promotion 1: Details: Promotion 2: Details:	HST004

(c) 2009 World Corp Inc. Privacy Policy and Discla

#### Figure 17. Final Call Center Application Interface

The application also provides the user with the capability to update the customer record. The updated fields can be written back to the server by pressing the **update** button. This will initiate a request to execute the 'customerUpdateService' Web service, which updates the SAS data set.

Field values are passed to the SAS Web service using a request object (reqObj):

```
private function updateCustByID(event:Event):void {
       var reqObj:Object = new Object();
       reqObj.parameters = {
              _custID: this.custID.text.toString(),
              _custTitle: this.custTitle.text.toString(),
              custSegment: String(this.custSegment.selectedItem.@name),
              _custFirstName: this.custFirstName.text.toString(),
              _custMiddleName: this.custMiddleName.text.toString(),
               custLastName: this.custLastName.text.toString(),
              custAddress1: this.custAddress1.text.toString(),
              custAddress2: this.custAddress2.text.toString(),
              _custCity: this.custCity.text.toString(),
               custState: this.custState.text.toString(),
              _custPostcode: this.custPostcode.text.toString(),
              _custPhone: this.custPhone.text.toString(),
               custEmail: this.custEmail.text.toString(),
              _custDoB: dfShort.format(this.custDOB.selectedDate),
              _custTariff: String(this.custTariff.selectedItem.@name),
              custComments: this.custComments.text.toString(),
              custPromotion: this.custPromotion.text.toString()
       };
       // use our utility class to send the web service request
       sendWSRequest(urService, reqObj);
```

This service will re-calculate the recommended promotion based on the new information. For the purpose of this paper, the customer churn probability is randomly generated (that is, replace this with a SAS<sup>®</sup> Enterprise Miner<sup>™</sup> scoring model). Based on the churn probability, the new promotion offers are calculated. This prototype application uses a simplified SAS code that needs to be updated in a production environment. See the "Customer Update Service" above for more details.

### GLOSSARY

- WSDL The Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services. (<u>http://en.wikipedia.org/wiki/Web Services Description Language</u>)
- SOA Service-oriented architecture (SOA) is a flexible set of design principles used during the phases of
  systems development and integration in computing. A system based on an SOA architecture will package
  functionality as a suite of interoperable services that can be used within multiple separate systems from
  several business domains. (<u>http://en.wikipedia.org/wiki/Service-oriented\_architecture</u>)
- SOAP SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web services in computer networks. It relies on Extensible Markup Language (XML) for its message format, and usually relies on other Application Layer protocols, most notably Remote Procedure Call (RPC) and Hypertext Transfer Protocol (HTTP), for message negotiation and transmission. (http://en.wikipedia.org/wiki/SOAP)
- IOM The Integrated Object Model (IOM) in SAS Integration Technologies provides distributed object interfaces to SAS software features. To call these interfaces, clients can use industry-standard languages, programming tools, and communication protocols. The interfaces are built into SAS and are available to clients whenever SAS is executed as an object server. (http://support.sas.com/documentation/onlinedoc/inttech)

# CONCLUSION

Using SAS BI Web service is an easy and convenient way to stream results from SAS analytics processes into enterprise applications. Real-time services allow you to leverage the power of SAS across the enterprise and beyond, while integrating the processes in your SOA.

### REFERENCES

Flynn, Joe. 2010. "Flex Your SAS<sup>®</sup> Muscle." *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC: SAS Institute Inc.

Vincent, Stephen A. 2010. "SAS<sup>®</sup> Application Messaging: How to Integrate Disparate Processes in Your Service-Oriented Architecture." *Proceedings of the SAS Global Forum 2010 Conference.* Cary, NC: SAS Institute Inc.

# ACKNOWLEDGMENTS

I would like to thank Jeremy Rankcom for his help in reviewing this paper.

### **CONTACT INFORMATION**

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

lko Schulz
S Institute Inc., Australia
Eagle St
sbane, QLD, 4001
-3233 320
lko.Schulz@sas.com
p://www.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.