Paper AD09

Integrating SAS® Analytics with Microsoft®-based Desktop Applications

Dr. Sam Yamamura, Nortel Networks, Research Triangle Park, North Carolina
James Tarantino, Nortel Networks, Research Triangle Park, North Carolina
John Payne, Nortel Networks, Research Triangle Park, North Carolina

ABSTRACT

Many corporations today use Microsoft-based applications (e.g. Visual Basic executables, Access Data Projects, OLAP for Excel, .NET services) to exploit the value of warehoused enterprise information. Several enabling technologies exist to leverage the SAS system as a "compute server", adding advanced analytical components to core functionality in the Microsoft Office productivity suite. This paper compares the pros and cons of two SAS technologies, Integrated Technologies and SAS/INTRNET, as means for integrating the SAS system with Microsoft Excel and Access applications. Through case study, examples, and discussion, the authors will present tips and considerations for using SAS/INTRNET with Excel web queries, Access Data Projects, and VB executables.

INTRODUCTION

In the current trends of enterprise resource planning and data warehousing, organizations have become increasingly savvy in collecting, organizing and analyzing data. Ongoing use of relational databases and internetworking technologies has given rise to new generations of business intelligence (BI) applications known under different headings, including EIS, DSS, and OLAP. Early versions of BI applications were client-intensive and often requiring custom application development from an external vendor or an organization's own IT department. Now, most BI application developers can readily utilize web technologies to deploy analysis and reporting applications in ways that require very little footprint on the your computers.

Though companies are continually pushing their applications to the web, some analysis, reporting and data management still are done directly within your productivity suite (i.e., various applications that comprise MS Office) itself. The "cottage IT" groups that spring up in organizations often rely on their proficient use of tools like MS Excel and MS Access to manage their data and produce reports. It is not just technical staff using productivity suite functionality; everybody seems to be in the reporting business. It is rare to find a BI applications developer who has not heard the question, "Can I download this to Excel?". Like it or not, presentations and spreadsheets remain a ubiquitous form of organizational communication, data sharing, and reporting.

The functionality and availability of productivity suites provide some interesting opportunities for application developers as they develop analysis and reporting environments. To start, BI applications designed with productivity suite components have the advantage of keeping you in an environment that is already familiar. For instance, macros or Visual Basic add-ins are often just an extra button or menu within the generally understood MS Excel workbook. An organization's IT department might even expect to see faster application adoption since there is less learning curve for you to experience. Moreover, an organization may also be able to develop applications more rapidly by leveraging the inherent data connectivity and analytics within the productivity suite. This idea can be extended to include ease of application support of since many objects and controls used for BI functions are deployed within the operating system software and productivity suite. Finally, there are still some functions required in applications that are just not practically available on the web. Some functionality still requires some type of thick client footprint on your computers. For instance, ad hoc updating of cell level data, like you can do in an MS Excel worksheet, is still challenging to do in the web environment.

MS Office is one productivity suite with a continuously expanding set of functionality to access, analyze and report enterprise data. Within Excel alone is the ability to conduct web queries, ODBC database queries, and OLAP. These capabilities also exist in MS Access. The inclusion of Visual Basic for Applications (VBA), Visual Basic (VB), C, C++, and/or C# add-ins can further enhance connectivity, data manipulation, and analysis. BI developers wary of having to spend time coding VB or C++ to get data access and query functionality may be surprised at the embedded functionality in the productivity suite that can be leveraged to avoid complex coding and custom development.

As a practical alternative to intensive custom code development, Microsoft created project format in MS Access 2000 that provides a front-end access engine to MS SQL Server managed through an OLE-DB connection. This technology, called Access Data Project (ADP), allows you to develop SQL tables, views and stored procedures from inside MS Access 2000. Advanced BI developers can also take advantage of VBA to add additional functionality to database applications. With VBA comes the ability to access MS Windows APIs and controls. When integrated, developers can quickly create client-server applications that use a server-based SQL Server database and a client-side set of forms and MS Access functionality. As an aside, these applications could be completed in VB exclusively, but ADPs have forms, environment variables and ActiveX automation components within them, eliminating the need for a full-blown VB software development kit (SDK) when creating simple BI applications.

Notwithstanding the ease-of-use and ease-of-development of ADPs, their most dramatic shortfall is their limited offering of statistics and functions. ADPs have generally the analytic capability of MS Excel. To extend the analytic capabilities while preserving the benefits inherent in ADPs, BI developers can utilize third-party plug-ins and/or custom code. SAS Institute is a software solutions provider with existing capability that can be coupled effectively with the MS Office productivity suite.

This paper provides you an actual example of adding advanced analytic capabilities to an ADP client-server application by integrating the SAS system with Microsoft's WebBrowser control object. You will be presented an MS ADP having requirements for analytics beyond the scope of what MS Access 2000 could easily provide. The BI development team mentioned here used SAS/INTRNET to embed data imputation and regression algorithms into a thick-client MS Access 2000 ADP while limiting the total amount of client-side software installation.

CASE STUDY: ADDING ANALYTICS TO ACCESS DATA PROJECTS

THE DILEMMA

The compensation organization ("customer") within Nortel Networks needed support in mass-producing market-based salary guidelines to be used for planning employees' compensation. Accuracy and timeliness of these numeric guidelines are critical since they enable senior leaders to establish payroll budgets as well as guide people managers in distributing employee compensation. Base-salary guidelines are derived through analysis of geographically based salary and wage data for hundreds of external jobs that correspond to internal positions. The data are complex; they often have different suppliers, formats, schemas, and delivery schedules. Historically, compensation analysts have managed elaborate systems of interrelated MS Excel spreadsheets to organize and analyze the

salary data. Also, due to the regional nature of the work, it was common for analysts, in their respective parts of the world, to develop their own idiosyncratic approach to salary guideline determination. Consequently, this led to process inefficiencies and inconsistencies at the enterprise level. Compounding the issue, the data modeling and regression analyses required for imputing missing data and deriving the final salary bands was beyond the scope of conventional desktop software. Ultimately, the client group approached the Information Services department for a solution that would integrate the data, analyze it with standard procedures, and publish the final salary guidelines to core reporting and salary planning systems. To address the customer's needs, a BI development team within Information Services was tasked with developing an IT-based solution called the Market Data Analysis Tool (MDAT).

In order to meet the multiple constraints of time-to-delivery, user acceptance, and programmer availability, the BI development team selected the core application architecture to be an MS ADP connected to a MS SQL Server database. The critical gap in functionality was data imputation and regression algorithms that could be supplied by The SAS System. The challenge was how to integrate this niche analytic capability into the ADP without disrupting the 'user experience.' That is, how to make the SAS System transparent to the you so you can enjoy a holistic experience using MDAT without the feeling the are toggling between multiple applications.

A TALE OF TWO INTEGRATION STRATEGIES

As most BI developers know, there are multiple ways to tackle almost any IT problem. The chosen path is often dictated by the constraints of the situation. SAS Institute provides two core technologies to build client-server applications - SAS\Integration Technologies and SAS/INTRNET. The choice of technology option ultimately boils down to the nature of the application (e.g., thick vs. thin), what SAS modules the organization has licensed, and the proficiency of the developers. Both options enable information delivery from SAS application servers to client-side computers. However, there are certain advantages and restrictions associated with each approach. First, SAS\Integration Technologies offers full integration of SAS system capabilities with other productivity tools such as MS Excel, Access, and VB executables. Using object-oriented (OO) programming techniques, developers can instantiate and invoke SAS objects, connect a client application to SAS remote server, and interact with data using SAS capabilities within programming environments, including VB, C and C++. Using Lightweight Data Access Protocol (LDAP), security and permission around Integration Technologies can be centrally managed. These features enable thick clients to tap the data processing capabilities of SAS without requiring you to have SAS thick client or formal knowledge of SAS computing. However, an ability to seamlessly integrate SAS with other applications comes with potential maintenance overhead. Since client-side applications are built with SAS objects, developers may need to constantly maintain client-side software run under wide range of operating system and driver configurations.

SAS/INTRNET helps deliver enterprise information through a CGI interface to standard web browsers. Web-oriented information delivery environment reduces the need for software maintenance on client computers. Furthermore, it cuts development cost and learning cycle of developing SAS/INTRNET applications by taking advantage of standard, and existing, SAS application development methods. Instead of object oriented programming within Visual Basic or C++, developers can leverage simply their existing SAS programming skills. The result of analysis executed by analysis server can be viewed in wide range of formats, from non-interactive summary results taking advantage of SAS ODS HTML, to highly interactive Java interface and ActiveX Graph objects. SAS/INTRNET applications can also take advantage of existing web security framework. On the other hand, web interface limits interactivity between client and server computers. For example, it would be more challenging for the application server to analyze the data on your computer using SAS/INTRNET than SAS Integration Technologies.

BUSINESS DECISION & RATIONALE

The BI development team evaluated issues discussed above to help choose an optimal SAS solution for MDAT. The customer requested the following features to be included in the tool: First, missing data points in the external salary survey dataset needs to be imputed using the descriptive statistics of the same data set. When the number of observations in the data set is insufficient, the customer-supplied rule is used to impute the missing value. The data before and after the imputation needs to be displayed in the tables. The imputed data is then used to update the table in the SQL database. Second, MDAT required the ability to perform regression analysis on the salary survey data. You can choose from linear, polynomial, logarithmic and exponential regression methods. Results of the analysis such as R-squared, slope coefficient, and statistical significance needed to be displayed. The predicted regression line and the observed values are then plotted over the entire company salary range.

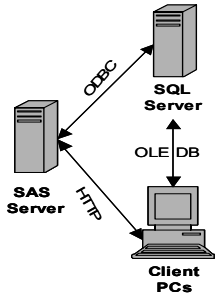
Prior to the development of MDAT, the customer used MS Excel to perform the imputation and regression analyses. While Excel offered a flexible environment to manipulate the dataset, it resulted in the lack of standardized analysis method among the client group members. It became obvious during requirements capture that the customer often used different underlying regression models without such knowledge. On the other hand, Excel offered the convenience of generating predicted lines, since the customer was not equipped with programming and in-depth statistical knowledge required to operate SAS/STAT.

The BI development team's decision was to incorporate MS ADP environment of MDAT with SAS/INTRNET technology. The decision is based on several factors. First, using SAS/INTRNET allows rapid deployment of required components through conventional SAS programming. Second, MDAT was able to take advantage of SAS ODS features via SAS/INTRNET. When SAS code is called within a WebBrowser control of MDAT, the results delivered via the ODS appeared seamlessly. Third, the ActiveX ODS component provided greater flexibility in plotting regression lines. Unlike JPEG files, graphs using ActiveX drivers let you customize graph attributes such as type of graph, color, graph style and labels. The use of flexible information delivery medium enables you to customize the graphical output, while saving developer's time in unnecessary cosmetic enhancements. Lastly, invocation of SAS macros via the ADP's menu bars helped maintain the user-friendly aspect of data analysis that you are accustomed to in a windows environment.

IMPLEMENTING THE SOLUTION

The implementation of client/server SAS computation environment consists of three parts: (a) your computer running the MDAT ADP; (b) the SQL Server database containing the application data; and (c) the SAS "compute" server for imputing missing data values and regressing data (see Figure 1).

Figure 1. MDAT client/server environment



Computers running Microsoft Windows 95 or greater (2000 or above recommended) are able to run the client-side of the MDAT application. The ADP portion of MDAT requires that Microsoft Data Access Components (MDAC) 2.6 be installed on your computer. Most required ADP components are pre-installed if your machine is running Windows 2000 or higher. If you are on a Windows release prior to Windows 2000, you can obtain MDAC 2.6 as a free download from Microsoft's corporate website. In order for you to take advantage of the SAS/ODS output, you should also install SAS Client Side Components.

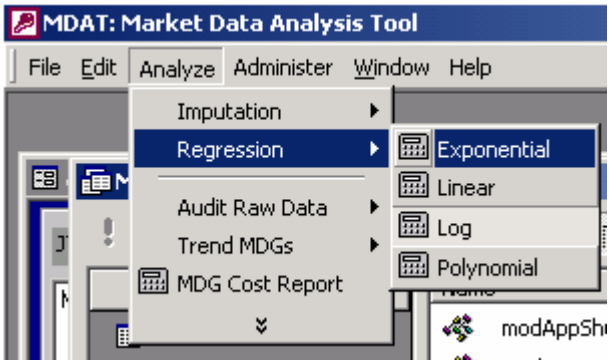
The first task when opening MDAT is to initialize your unique analysis environment on the data server. The initialization of your work environment on the server improves application efficiency by reducing the data transfer needs between the servers and you on the client-side. The following VB codes creates your temporary user tables at startup:

```
' define ADODB connection
Dim cnn As New ADODB.Connection
Dim cmd As New ADODB.Command
Dim arrSQL(1 To 4) As String
Dim varAny As Variant

'arrSQL(1) through (4) contains SQL
'statements to create unique TABLES and VIEWS
'using strUserName as an identifier.
arrSQL(1) = "CREATE TABLE " & _
strUserName & _
strTableScript
arrSQL(2) = ……
arrSQL(3) = ……
arrSQL(4) = ……

'Define Connection to SQL server and open
With cnn
    .ConnectionString = _
      "Provider=SQLOLEDB.1; " & _
      "Persist Security Info=False;" & _
      "User ID=USER_ID;" & _
      "Password=*******;" & _
      "Initial Catalog=tempdb;" & _
      "Data Source=SERVERNAME"
    .Open
End With

cmd.ActiveConnection = cnn

For Each varAny In arrSQL
    cmd.CommandText = varAny
    cmd.Execute
Next varAny

'Close connections and commands to conserve
'system resources
cnn.Close
Set cnn = Nothing
Set cmd = Nothing
```

MDAT retrieves strUserName from your network ID, and creates four temporary user tables based on arrSQL(1 to 4). Connection string includes User ID and Password that you are required to enter when launching MDAT.

Once your data is successfully staged in your user tables. You can request MDAT send analysis requests to the SAS compute server via the 'Analyze' menu bar (see Figure 2).

Figure 2. MDAT Analyze Menu



In MS Access (and ADPs), menu bars can be customized to launch tables, forms and/or VB macros. In the menu displayed in Figure 2, the regression options run the following VB macro:

fLoadWebForm(fRegressData(3,-1),TRUE,"Polynomial Regression")

The macro fLoadWebForm takes three arguments, and opens a form with an embedded WebBrowser control object. The first argument takes web URL, given by the following fRegressData function:

```
Function fRegressData _
    (strModel As String, _
    intCommit As Integer) As String

Dim strModelType As String
Dim strCommit As String

'Select Regression Model
Select Case strModel
    Case 1
        strModelType = "LINEAR"
    Case 2
        strModelType = "EXP"
    Case 3
        strModelType = "POLY"
    Case 4
        strModelType = "LOG"
End Select

'Select If the predicted values need to be
stored.
Select Case intCommit
    Case -1
        strCommit = "YES"
    Case 0
        strCommit = "NO"
End Select

'Construct URL to SAS broker.
fRegressData = "http://URL/cgi-bin/" & _
     "broker.exe?_service=SSD" & _
     "&_program=CODES.MDAT_GLM.SAS" & _
     "&user=" & strUserName & _
     "&model=" & strModelType & _
     "&commit=" & strCommit

End Function
```

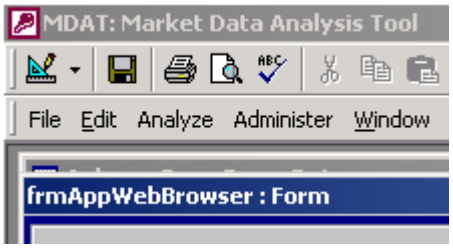
The function constructs a URL that invokes the SAS broker (broker.exe) and the code (MDAT_GLM.SAS) using three parameters, strUserName (used upon initialization to create your temporary tables), GLM model type, and data save option. The macro variable '&user' helps MDAT_GLM.SAS to load GLM source data from and save predicted results to appropriate

temporary tables.

The second argument in fLoadWebForm sets the visibility property of the form containing the WebBrowser control. When the value is set to FALSE, it is possible to use SAS/INTRNET services to emulate a background operation; that is, running SAS without returning procedure output to you. Specifically, the ADP sends a request to the broker, but simply does not display the form containing the output. This is somewhat inefficient, but it does serve as a way to have a background operation without distracting you with unrequested output. The third argument is a caption that appears on the top left corner of the form containing the WebBrowser control.

The form object in MDAT is displayed after choosing the regression option (see Figure 3).

Figure 3. Form window launch



The result of the regression is displayed as illustrated in Figure 4.

Figure 4. Regression output in WebBrowser control

ANALYSIS 2 : Location_Country = CA , Currency_Code = CAD
Metro_Code = MONTREAL , Job_family = RDD , Job_Sub_family = SWR

The GLM Procedure

J_Base_50th

R-Square	Coeff Var	Root MSE	ADJ_Base_50th Mean
0.309227	167.1028	37400.38	22381.67

Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	-0.00000	80214.88531	-0.00	1.0000
JCI	32173.64583	60221.68222	0.53	0.6302
JCI*JCI	-6994.27083	9731.88824	-0.72	0.5242

The data server supporting MDAT is MS SQL Server 2000 running on a two-way MS Windows 2000 server. Actually, any data sources that SAS and MS Access can access via ODBC can be used.

Two databases are used to support MDAT data retrieval and analysis. The first database is the main data repository where the records are stored permanently. The second database is temporary database (tempdb), which is the default location in the SQL server to store any of your temporary information. TempDB facilitates the maintenance of your temporary user tables that MDAT client application creates, because MS SQL Server deletes the tables after a period of inactivity. The MDAT client application accesses the databases via ADODB connection strings, whereas SAS/INTRNET service connects to the data using ODBC.

SAS/INTRNET was housed in a separate server within MDAT environment. The rationale is to distribute the server workload between MS SQL server and a SAS compute server.

SAS/INTRNET is set up as a web service within Microsoft Internet Information Services (IIS). SAS/INTRNET was configured to "listen" for the analysis request through a port in IIS service.

SAS broker is configured at the startup with the following command in broker.cfg_v8:

```
SocketService SSD " "
ServiceDescription "SSD Service"
ServiceAdmin "Osamu Yamamura"
ServiceAdminMail "mail@mail.com"
Server 47.142.16.155
IdleTimeout 30
Port 5500 5501 5502 5503 5504
#  Remove the following line for any servers
before V8.1
FullDuplex True
```

Port 5500 through 5504 was allocated for SAS/INTRNET broker in this example.

When MDAT invokes the form with the WebBrowser control, the code in MDAT_GLM is processed through the SAS/INTRNET broker. There are several steps inside the code. The first step is to initialize the ODS, so that the output is directed through the broker to the WebBrowser control.

```
*  Capture output to HTML;
ods path tmplib.template(read)
sashelp.tmplmst(read);
options fullstimer;
ods listing close;
ods html body=_webout
         path=&_tmpcat (url=&_replay)
         rs=none
         style = nortel;
```

'tmplib' is a SAS library defined at startup. It is used to store the temporary data that SAS/INTRNET server retrieved from the SQL server. ODS HTML BODY and PATH are set to _webout and &tmpcat (url=&_replay), the system (macro) variables that point to SAS broker's URL.

The second step is to retrieve the data from SQL Server to templib to transform the data for the analysis. PROC SQL's ODBC connection and pass through features are used to accomplish this:

```
/* Use the Path through feature of PROC SQL
to obtain the data from the SQL server */
%macro Bring_Data_From_SQL(indata, outdata);
proc sql;
    /* TEMPDB is defined in ODBC connection
       on the server */
    connect to ODBC (dsn='TEMPDB');
    create table
    %sysfunc(compress(&IntrNetlib. .
             &outdata.))
    as select * from
    connection to ODBC
    (
    SELECT *
    from &indata. where BASE_WEIGHT > 0
    );
    disconnect from odbc;
quit;
```

ODBC pass through ('connection to ODBC') is an efficient method of having SAS and SQL server interact, since the SQL code is executed on SQL server where the data processing is faster. The macro variables IntrNetlib and outdata specify the SAS data set created from the SQL table (indata) accessed via ODBC 'TEMPLIB.'

After a series of data cleansing, the third step of MDAT_GLM.sas is to run the regression analysis using PROC GLM, and to save the predicted value in the temporary SAS dataset. A portion of data manipulation and GLM macros are shown here:

```
/* If the regression option is Exponential or
Logarithmic, convert the value of X */
data %sysfunc(compress(&IntrNetlib. .
       &outdata.));
  set %sysfunc(compress(&IntrNetlib. .
       &outdata.));
  %IF "EXP" = "&MODEL." %THEN
    EXP_JCI = EXP(JCI);
  %IF "LOG" = "&MODEL." %THEN
    LOG_JCI = LOG(JCI);
  ;
run;

By default PROC GLM does not analyze
exponential and logarithmic models.  Value of
the regressor is therefore transformed prior
to the analysis to allow the use of GLM
framework.  However, the use of PROC NLIN is
ideal for most non-linear models.


/* run separate PROC GLM for each Y */
%do i = 1 %to %eval(&Y_count.);

    /* SELECT desired outputs */
    ODS SELECT FitStatistics
               ParameterEstimates;

    proc glm data =
        %sysfunc(compress(&IntrNetlib. .
                 &indata. _G&g.));
                ;

    /* model statement */
    /* polynomial */
    %IF "POLYNOMIAL" = "&MODEL." %THEN
        model &&Y&i. = &x. &x.*&x.;
    /* linear */
    %ELSE %IF "LINEAR" = "&MODEL." %THEN
        model &&Y&i. = &x.;
    /* exponential */
    %ELSE %IF "EXP" = "&MODEL." %THEN
        model &&Y&i. = exp_&x.;
    /* logarithmic */
    %ELSE %IF "LOG" = "&MODEL." %THEN
        model &&Y&i. = log_&x.;
    /* ELSE default is polynomial*/
    %ELSE
        model &&Y&i. = &x. &x.*&x.;
    ;

    output out=
       %sysfunc(compress(&IntrNetlib. .
                &indata. _G&g.))
             p=
       %sysfunc(compress(&&Y&i.. _hat));
    run;
    quit;

%END; /* i */
```

The macro variables &&Yi.., where &i. Is 1 through &Y_count. Is regressed on the macro variable &x. in separate PROC GLM statements. First, the ODS SELECT statement is submitted to display only the necessary statistics. Second, model statement is selected based on the value in the macro variable &MODEL, a parameter passed via the URL string from MDAT client. Lastly, the predicted values of the analysis is stored in the SAS table specified by "output out=" statement.

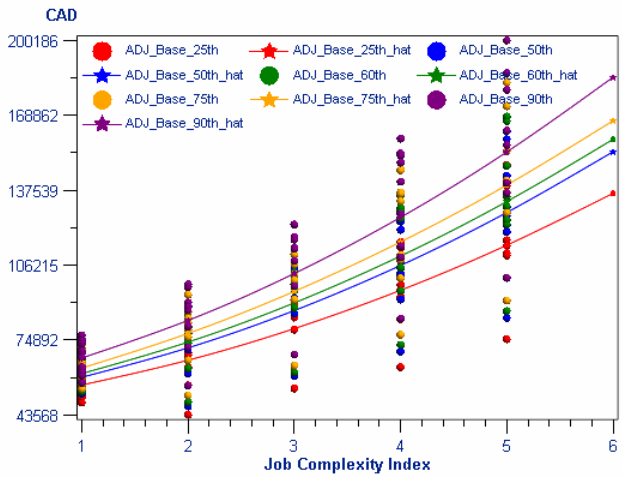
The predicted and the observed valued are then over-layed in the 2D plot using PROC GPLOT with ODS Active X driver. Although ACTIVE X driver does not currently support some GOPTIONS, it offers some advantages over JPG image files. First, the interactive nature of ACTIVE X graphs allows the separation of content and format. Using Active X drivers, developers can focus on code development while the users of the graph can own the ownership of the format, such as color, font and style. Second, Active X graphs are better suited for presentations within MS Office environment. The graph can be easily copied and pasted into a presentation or a document as a Windows Metafile or as an Active X object. The downside of this approach is that unlike JPEG files, client computers are required to install SAS Graph components.

The following is a section of PROC GPLOT macro:

```
   /* PROC GPLOT */
   /* axis statement */
   axis1 label=("&&currency&g..");

   /* GPLOT */
   /* Nested within a loop, the combination of
    &g. (group defined by geography)
    and &t. (Total number of Y values regressed
    by X) defines the number of plots. */
   PROC GPLOT
       data=%sysfunc(compress(&IntrNetlib. .
            &gsource. _G&g.));

       plot
       %do t=1 %to %eval(&Y_count.);
           &&Y&t.*&x.=%eval(&t.*2 -1)
           %sysfunc(compress(&&Y&t..
               _hat))*&x.=%eval(&t.*2)
       %end;

       /* plot options */
       / overlay legend=legend1 vaxis=axis1
                 haxis=axis2;
       run;
   quit;
```

SAS/INTRNET redirects Active X graphs broker to the WebBrowser control embedded in the form object like other output for MDAT_GLM.SAS (see Figure 5).

Figure 5. SAS/GRAPH output sent to WebBrowser Control



The final step of MDAT_GLM.SAS is to save the predicted values in the temporary table in SQL server. Again, PROC SQL with ODBC pass through is used to save the results back to the server.

```
/* Based on Regression outputs, UPDATE SQL
   statements are built for each predicted
   value , and assigned to the macro
   variables &update1. to &&update&NumSQL..
   PROC SQL with ODBC passthrough is used
   to upload the data                        */


%IF 0 < %eval(&NumSQL.) %THEN
    %DO;
    proc sql;
    connect to odbc (dsn='tempdb');
        %IF 1 = %eval(&g.) %THEN
            %DO;
                EXECUTE
                (DELETE FROM &outset.     )
by odbc;
            %END;
                EXECUTE
                (
                %DO i=1 %TO %eval(&NumSQL.);
                &&update&i.
                %END;
                ) by odbc;
    disconnect from odbc;
    quit;
%END;
```

In this example, UPDATE SQL statements stored in the macro &&update&i. are executed in the SQL server. This method is useful when the data needs to be manipulated prior to being stored in a SQL table. Alternatively, a LIBNAME statement with ODBC connection to a SQL database can be used. In this case, the DATA step offers another method to copy a table from one library (SAS datasets) to another (SQL server), based on a set of conditions/formulas within the DATA statement.

CONCLUSION

The MDAT application capitalized on short development cycle, one of the advantages that SAS/INTRNET offers. The BI developers leveraged existing SAS programming capabilities; macros for imputing and regressing the selected survey data were developed quickly. Lack of advanced object-oriented programming also drove server-side SAS developers and client-side VB developers to concentrate in their respective areas. This lead to a more expedient development path. The application also took advantage of SAS ODS and styles which were standardized and reused from other SAS applications. Using VB's WebBrowser Control object and corporate standard color schemes, the analysis results were displayed within the client-side application seamlessly. The results can be easily imported into popular productivity tools such as Excel, using either web-query or simple cut-and-paste.

The use of SAS/INTRNET over SAS/Integration Technologies is in no way an endorsement of one solution over the other. The intent of this is paper is simply to show how you can integrate an application in the MS Office suite to SAS via SAS/INTRNET. This solution may be an easy alternative for those organizations that have not licensed Integration Technologies.

CONTACT INFORMATION

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

John Payne
Nortel Networks
P.O. Box 13010
RTP, NC 27709
Work Phone: 1 (919) 997-3719
Email: johpayne@nortelnetworks.com
Web: http://www.nortelnetworks.com

James W. Tarantino
Nortel Networks
P.O. Box 13010
RTP, NC 27709
Work Phone: 1 (919) 997-4849
Email: jtaranti@nortelnetworks.com
Web: http://www.nortelnetworks.com

Dr. Osamu "Sam" Yamamura
Nortel Networks
P.O. Box 13010
RTP, NC 27709
Work Phone: 1 (919) 997-5209
Email: samyama@nortelnetworks.com
Web: http://www.nortelnetworks.com