SAS/IntrNet™ Software: A Roadmap

Donald J. Henderson, SAS Institute Inc.

Abstract

Web Publishing, CGI, Java™, ActiveX®, JavaScript™, Dynamic HTML, Web Publishing, Compute Services, Data Services, Report Distribution, Application Distribution, Application Dispatcher™, htmSQL™, JConnect™ . . . .are all terms used to describe the types of processing and the tools used to define and quantify Web applications.

Each of these terms will be described in the context of what problems they address and how they address those problems. Next, criteria will be discussed that attendees can use to determine when a particular technology is most appropriate.

SAS/IntrNet software provides capabilities in all of these areas. A road map that explains how each of the components of SAS/IntrNet software will be presented so that users can determine which of the technologies is most appropriate for their application.

NOTE: The World Wide Web has had a tremendous impact on the time involved in delivering new software technology. At the time this paper was written (early January 1998) it accurately reflected the components of SAS/IntrNet software. However, the author expects that some of the material will be out-of-date by the time the paper is presented at the SUGI 23 Conference. The reader should check with his/her Software Sales representative or the following URL for the most up to date information on SAS/IntrNet software:

http://www.sas.com/web

Introduction

SAS/IntrNet software provides a powerful suite of tools that allow you to create and deploy Web-enabled reports and applications. They can provide remote access to your organization's data and bring the computing power of SAS software to any desktop, regardless of whether SAS software is installed on the desktop or a central server platform.

Because organizations often use a diverse collection of technologies to store and manage information, many factors must be considered when developing a Web-based information delivery system. The factors that impact your choice of Web technologies can include:

- user interface requirements
- data access and analysis requirements
- application performance
- the costs of development, deployment, and maintenance

This paper addresses how the various components of SAS/IntrNet software can be used in an information delivery infrastructure to help you turn data into strategically useful information and knowledge by utilizing a technology is available to virtually everyone and that most users already have experience with: the World Wide Web.

The following areas will be specifically addressed:

- the terms and technologies that form the basis of Web technology
- a brief description of each component of SAS/IntrNet Software
- software and tools that can be integrated with SAS/IntrNet in order to build web applications
- the evolution of web applications to a component based architecture composed of pieces utilizing various technologies
- product dependencies for the SAS/IntrNet components
- criteria to help determine what components to use based on the application need, and available skill set

Definitions

Web technology has had a dramatic impact on information delivery and the applications development process. It has simplified the client server model in that it is no longer necessary to distribute and install applications on each desktop. The Web provides the infrastructure to ease the deployment of applications where a client can request services such as data access or processing from another machine.

Web enabled Information Delivery Systems can be described both in terms of the function they serve, how they work, and the types of technology they utilize. We will begin by defining some of the more common terms used in describing the capabilities of SAS/IntrNet software.

Web Publishing is the creation of static reports which are made available by storing them in a location accessible by a Web Server. Any user with just a web browser (such as Internet Explorer or Netscape) can access the information. The generated reports can be produced in a variety of ways, including, for example, scheduled batch jobs. The only constraint is that the files containing the reports must be in *Internet Content* form. HTML (text

based) and GIF or JPEG (graphical) are some examples of valid *Internet Content*.

CGI (Common Gateway Interface) is a programming interface that allows a Web server to communicate with an external program. CGI programs are typically small, written in a script or high-level language, and reside on the Web server to act as the interface between a Web Browser and a Content Server (providing Data and/or Compute Services). When a Web browser accesses a URL (Uniform Resource Locator) for a CGI program, the Web server executes the CGI program. Typically, the CGI program invokes a session with a database or application server, which in turn processes the request from the client and returns the result to the browser by way of the Web server. Purely CGI based applications require interaction with server(s) in order to provide interactivity with the user.

Java is an object oriented programming language developed and promoted by Sun Microsystems that developers can use to create Web-enabled programs, known as *Java applets*. A Java applet consists of two parts:

- an applet HTML page
- a set of Java classes that contain the program logic

The applet HTML page specifies what Java applet is to be invoked (in fact, a single HTML page can include multiple applets). When a Web server returns an applet HTML page to a Web browser, the browser reads the applet information and requests the applet classes from the Web server. After these classes are downloaded, the applet is executed based on any user input (note that upon leaving the HTML page containing the applet, the applet is deleted). For example, the applet can establish a connection to a database or application server, submit requests for processing, and receive and display the results. Since the Java applet is executing locally, it can provide a much richer interactive environment than a CGI program can.

ActiveX controls are a Windows only implementation that take advantage of the Microsoft Windows implementation of COM™ (Component Object Models) technologies to provide interactivity and interoperability with other types of COM components and services. ActiveX controls are a next generation of OLE controls (OCX), providing a number of enhancements specifically designed to facilitate distribution of components over the Web. They can be embedded in HTML pages much like Java applets.

Java applets and **ActiveX** controls differ in that ActiveX controls only run on Windows platforms; once downloaded they remain locally installed; and can access the local PC file system; and can communicate with any machine on the user's local network.
JavaScript is a lightweight interpreted programming language with simple object-oriented capabilities. It operates at the client side (i.e., the browser) and allows for executable content to be included in web pages.

Dynamic HTML is the next generation of HTML currently supported by Netscape Communicator 4 and Internet Explorer 4. It adds significant capability both in terms of interactivity and visual effects.

Compute Services is running a SAS program on demand and delivering results back to the users browser. Typically the Compute Server (e.g., the SAS program) is actively involved in determining how the results are the be displayed by the users browser (e.g., by generating the actual HTML returned to the users browser or by returning a delimited data file to a Java applet).

Data Services is a logical extension of SQL services and describes the ability to use SQL to access, query and display data. The Data Server is only involved in querying the data and returning the result set. It is the responsibility of the requesting program (e.g., a CGI program, a Java applet or the Web Server itself) to format and/or process the results appropriately.

Report distribution is the provision of information to a number of people in an organization using the Web. This can be static information that is generated periodically and provided for passive viewing, or it can be information that is generated on demand. The report can be customized to address specific user needs, and this customization can be performed by the solution provider or the end-user.

Application distribution is the provision of reporting and analysis applications to a number of people in an organization through HTML pages or a Java applet. Using SAS/IntrNet software, those applications do not require that the SAS System be locally installed on the user's desktop; instead SAS Software services are accessed from a server and delivered to the user's desktop.

SAS/IntrNet Components

SAS/IntrNet software contains a variety of components that can either be used individually or in combination to address specific needs of a Web-enabled information delivery solution. It is important to realize that you can use one component or you can create a solution that involves multiple components, where each component provides services that best meet individual requirements. The power of SAS/IntrNet software lies in the remarkable *range* of each component.

The **Web Publishing Tools** are a family of tools that enable you to generate static Web pages using your SAS data and output. Use these tools with existing SAS products to quickly and easily create Web content. The Web Publishing Tools are available at no extra charge. The **HTML Formatting Tools** are a collection of macros that enable you to format your SAS data sets and procedure output into HTML-formatted pages that you can share with Web users. The current Formatting Tools are the Output Formatter, the Data Set Formatter, and the Tabulate Formatter. (In Version 7 of the SAS System the Output Delivery System, ODS will allow for direct creation of HTML files.) **The GIF**, **JPEG** and **GIF Animation** drivers enable you to produce *Internet Content* images with SAS/GRAPH® procedures. The GIF Animation driver enables you to combine GIF images created with SAS/GRAPH procedures so that you can add animations to your Web pages. The **GGRAF procedure** enables SAS

programmers to create three-dimensional, clickable charts in VRML (Virtual Reality Modeling Language). The Java **VRML Browser** enables you to view the contents of VRML 2.0 files, such as those created by the GGRAF procedure. Our VRML Browser takes advantage of Java's cross-platform portability and the VRML standard file format for Web accessible graphics.

The **Application Dispatcher** is a CGI gateway between a Web browser and SAS software. It enables users to build dynamic applications that give users access to the power of SAS software from their Web browsers. It provides the capability to run any SAS program that can be executed in batch mode. The user need only concern themselves with the details of the SAS program. All of the details relating to CGI, the communication of the parameter values from the HTML page, and returning the results to the users browser are handled by the Application Dispatcher framework. The Application Dispatcher allows the deployment of these programs to any number of users (whether they have SAS software installed or not).

htmSQL offers a gateway to your SAS data from a Web browser that allows you to build dynamic queries. It consists of a CGI program that resides on your Web server and can be used to access and update a SAS data set (including read access through views to external DBMS). The user provides an input file (an .hsql file) containing SQL statements embedded in HTML, and htmSQL submits the statements to a SAS Data Server (either SAS/SHARE software or the Scalable Performance Data Server). It then retrieves and formats the results according to the HTML embedded in the .hsql file. htmSQL can be used to create sophisticated, dynamic applications that let users manipulate report data to address their particular information requirements.

JConnect is a set of Java classes you can use to create Java applets and applications that communicate with SAS software on a server, allowing you to take advantage of remote SAS computing resources. It provides similar functionality to what a SAS software client can do with SAS/CONNECT® software, except that the functionality is available to any Java program and does not require SAS software to be locally installed. The programs that you create using JConnect can start a SAS session, connect to that session, create data sets, access existing SAS data, run SAS programs to analyze SAS data, and retrieve the results.

The **SAS/SHARE*NET Driver for JDBC**™ is a set of Java classes you can use to create Java applets and applications that communicate with a SAS Data Server (either SAS/SHARE® software or the Scalable Performance Data Server™). The Java programs that you create can enable a user to view and update data by submitting SQL queries and statements through a direct connection to the SAS server.

The **SAS SQL Library for C**™ is an API (Applications Programming Interface) that enables you to create applications that communicate with a SAS/SHARE server. The **SAS ODBC** driver provides ODBC-compliant Windows applications with read and write access to local and remote SAS data sets. Both of these are provided as part of SAS/IntrNet software to provide open access to SAS data. Many Windows Web Servers include functionality to access ODBC compliant data sources.

JTunnel™ employs HTTP Tunneling to allow applets to communicate with remote systems via a CGI program running on the Web server. As a security measure, Java specifications dictate that an applet cannot make network connections to a machine other than the machine from which it was downloaded. When deploying Java applets this security feature may force the use of a server configuration that is less than ideal because that server would have to be installed on the same machine as your Web server. In addition, many firewalls prohibit applets from communicating beyond the firewall, a restriction that can further reduce server configuration options. JTunnel addresses both of these configuration problems. You can use JTunnel with Java applets written using the SAS/IntrNet Java components to eliminate the restriction on where your SAS server runs in relation to your Web server and firewall.

Lightweight Java applets are specialized visual components. The reason these applets are lightweight is because they do not establish a persistent connection to a server but receive all information they need from applet parameters. Lightweight applets are best used in conjunction with either the Application Dispatcher or htmSQL, which can generate the applet HTML calls along with the custom data values which are used as input to the visual component.. Graphical applets such as that shown later in Figure 4 are an example of such applets. The HTML pages that use these applets define the parameters and values that define what the applet does or displays. The VRML Browser mentioned above is another example of a somewhat lightweight applet. An Application Dispatcher program is used to generate the input to the VRML Browser

SWAN IDE *(SWAN is an internal code name at the time this paper was written for a collection of new Java based technologies for scripting and exploiting the functionality of the SAS server environment)* is a project based Java visual development environment, which allows developers to easily build information delivery applets, applications, and re-useable components. Enterprise Data Access, Data Warehouse, and Decision Support functionality can be built into solutions by dragging remote server objects into the project. SWAN IDE is fully integrated with pre-built ROCF (discussed next) components which support seamless data access and viewing. It also facilitates a user building and seamlessly integrating their own SAS/AF® software models as well.

The **Remote Object Class Factory (ROCF -** also an internal code name**)** exposes SAS functionality to Java clients. ROCF objects are standard Java classes that are bean compliant. Users can exploit the data and compute power of SAS software by encapsulating them as SAS/AF models on the server and using the proxy Java classes (which the user can generate using the class editor in the IDE or the standalone tools) on the client. This allows the programmer in Java to treat these remote models as though there were local Java classes. A SAS/AF software model is defined as any object which contains methods to perform some type of work and is independent of a user interface. The most typical usage of a model is in the model-view paradigm; the model retrieves data from a database and the view formats and renders the data to the user. Using ROCF, you can effectively extend any existing or user written SAS/AF software model into the Java environment. The language boundaries are easily bridged: the client applets or applications are comprised of objects written in the Java language, which in turn make method calls to remote SAS/AF software objects which are written in SCL. The SCL then makes calls into or scripts the rest of the SAS System. To the Java object, the object written in SCL appears just like another Java object, and similarly the SAS/AF software models (objects written in SCL or C) have no notion of the language of their caller.

SWAN Business Viewer is a 100% pure Java technology for viewing data. It is designed to view data from the SAS Server. This data can be saved as MDDB's or tables, It requires either the EIS classes or HOLAP to accomplish the data gathering and summarization on the server. Its primary goal is to be able to view and navigate through business information while the user is within the friendly confines of a browser. It generates Java Beans and an HTML page that invokes a viewer of this bean. The bean can be built into other Java applications if required.

Industry Components

The World Wide Web is, by definition, an open environment. The use of other technologies is an integral part of web based solutions. Of course, Web servers and Web browsers are a given. There are a number of other technologies which should also be considered when using SAS/IntrNet software. A few of them are briefly described here.

Scripting Languages such as JavaScript and VBScript™ are lightweight interpreted programming languages with simple object-oriented capabilities. They operate at the client side (i.e., the browser) and allow for executable content to be included in web pages. This means that web pages can include dynamic programs that interact with the user, control the browser, and dynamically create the HTML content. JavaScript is more widely used and supported, and should be the scripting language of choice. But if development staff is well versed in Microsoft Visual Basic and the user community will be using Microsoft Internet Explorer as the browser, then VBScript may be a viable choice for a scripting language.

One of the more important capabilities of JavaScript is the ability to define code fragments that are to be executed when a particular event occurs. For example, JavaScript code could be used to:

- validate input (e.g., check for required fields, numeric values, etc.) before submitting a CGI request.
- automatically submit a CGI request when an item is selected from a list box (an HTML select tag).
- update a user's choices based on previous selections without requiring a round-trip to the server
- read and write properties of, and invoke methods of, Java applets and plug-ins.

```
function drill_order(from,to,label)

{ document.forms[0].elements[to].value = ' ';
  document.forms[0].elements[0].value = ' ';
  if (document.forms[0].elements[from].checked}
   { drlorder[drlorder.num] =
        document.forms[0].elements[from].value;
     drllabel[drlorder.num] = label;
     drlorder.num++;
   }
  else
   { for(i = 0; i < drlorder.num; i++)
      { if (drlorder[i] ==
           document.forms[0].elements[from].value)
        { for(j = i; j < drlorder.num; j++)
          { drlorder[j] = drlorder[j+1];
            drllabel[j] = drllabel[j+1];
          }
        }
      }
     drlorder.num--;
   }
  drlorder[drlorder.num] = ' ';
  if (drlorder.num > 0)
   { document.forms[0].elements[to].value =
          drllabel[0];
     document.forms[0].elements[0].value = drlorder[0];
   }
  for(i = 1; i < drlorder.num; i++)
   { document.forms[0].elements[to].value =
         document.forms[0].elements[to].value +
         '\r\n'+ drllabel[i];
     document.forms[0].elements[0].value =
         document.forms[0].elements[0].value +
         ' ' + drlorder[i];
   }
}
```

Figure 1. Sample JavaScript function.

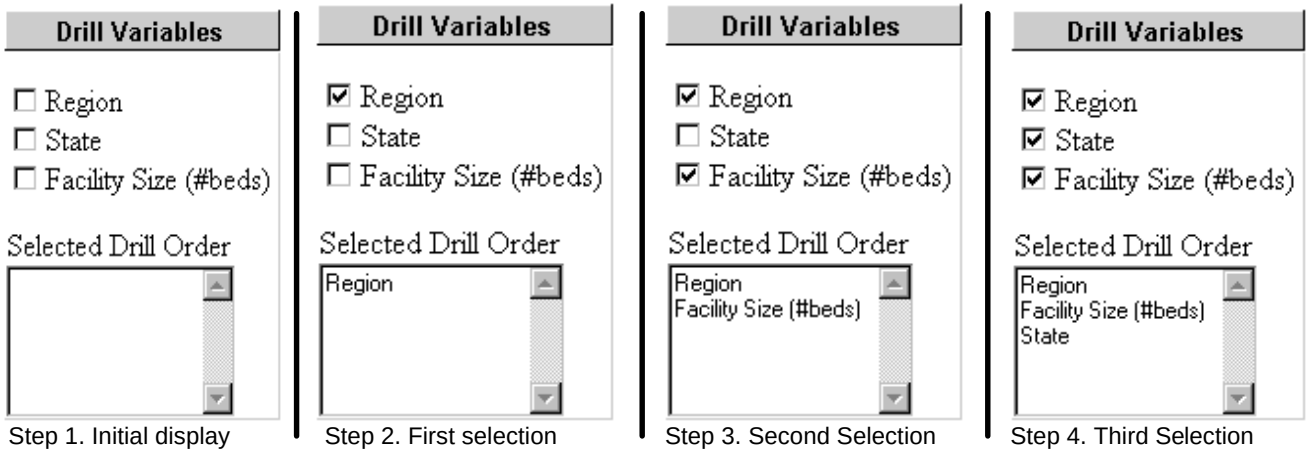

Figure 2. JavaScript sample showing dynamic update

For example, the JavaScript function in Figure 1 is a general purpose function that updates both a text display and a cumulative list of variables in drill-down order. It is used in the Application Dispatcher Xplore sample application. Sample output showing screen fragments as the user makes their selections is shown in Figure 2. Note that the HTML form/page is updated instantaneously with nothing run on the server.

Dynamic HTML adds even more capability to easily create dynamic Web pages. Both Internet Explorer 4 and Netscape Communicator 4 support an updated HTML object model that adds significant interactive capabilities. Unfortunately, as of the writing of this paper, the Netscape and Microsoft models are not fully compatible. Hopefully they will converge towards a common object model so Web developers can deploy applications utilizing these new capabilities without having to worry about what browser a user is using to access a web page. The Microsoft Dynamic HMTL object model (in the authors personal opinion) seems to be more robust, flexible and easier to implement interactivity with. For example, simple HTML lists built using the `<LI ...>` tag can be expanded and collapsed using a simple JavaScript function. The Xplore application (mentioned previously) has a Internet Explorer 4 (IE4) implementation of the library viewer that is automatically used whenever IE4 is the browser. Upon load, the contents of all libraries and catalogs is defined in the initial HTML. However, IE4's Dynamic HTML capability is used to expand/collapse library and catalogs without requiring any additional server processing (i.e., no round trip to the server). Figure 3 shows this output. The user can expand or collapse libraries or catalogs merely by clicking on them without the need for a round-trip to the server to update the HTML.

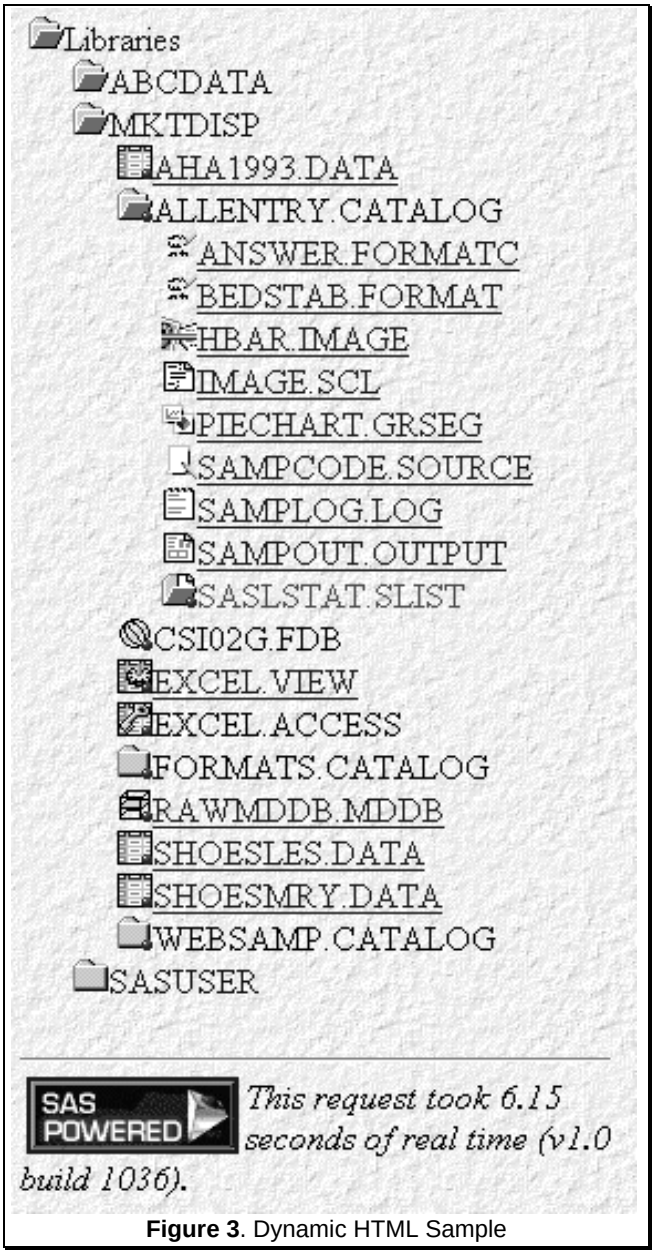

Figure 3. Dynamic HTML Sample

ODBC access to compliant data sources can be provided by many PC based Web Servers. Such Web Servers (e.g., Microsoft's IIS server™ can access SAS data using the ODBC capabilities included with SAS/IntrNet software.

HTML Editors can also make developing/implementing a dynamic Web application easier. HTML editors (e.g., Microsoft FrontPage™, Netscape Gold™) provide visual development environments for HTML pages. These pages can be the *front-end menus* or screens for Web

applications. It is possible to build very functional and attractive Web pages using such editors. SAS Institute is currently hoping to provide a SAS/IntrNet software SDK. This SDK will include the building blocks or *foundation classes* to be compatible with already established (and emerging) development environments to construct applications. Also included will be *design time controls* which will aid users in connecting the SDK components, their own code, and HTML to make an entire SAS/IntrNet application.

And of course, there are a number of **Java Integrated Development Environments** (IDEs) that can also be used with Java applets that utilize the SAS/IntrNet Java components. The ROCF components can be used to develop applets and applications using these third party IDEs (i.e., outside the Swan IDE framework) although it is easier to do this using Swan IDE

Component Based Architectures

The term *Intelligent Client* has been used to define thin-client applications where some of the processing and logic is implement on the client. When originally coined, such applications, by definition, needed to be implemented in Java. There was simply no other viable choice. CGI and HTML did not provide the interactivity needed to build such applications.

However, as mentioned above, CGI and HTML technology have evolved significantly over the last 6-12 months. In addition, Java has not yet matured at the rate many expected. Java is still a very important technology, but its role in dynamic web applications is evolving as the Web itself evolves. JavaBeans™ (and ActiveX, to a lesser extent, as the competing technology from Microsoft) will become important components in Web applications.

The *requirement* that truly GUI and/or user friendly applications be done in Java, while true just 6-12 months ago, just is no longer the case. Java is still a very important piece of the puzzle. But it is not the only piece. There will be many applications that are build solely with Java; just as there will be many applications built solely with HTML and CGI.

Now that CGI and HTML can now be augmented with Dynamic HTML and JavaScript to provide client side logic previously attainable only with Java and/or ActiveX, the term intelligent client can very legitimately be used for CGI/HTML based solutions. What technology to utilize depends on a variety of factors. The more important factors are discussed later in this paper.

The future is a component architecture where you may mix and match Java, JavaScript, CGI and HTML - using each at what they are best at. In addition to being used for complete applications, we will see Java used to implement small, quick to download, function specific applets. For example, graphics applets (as mentioned

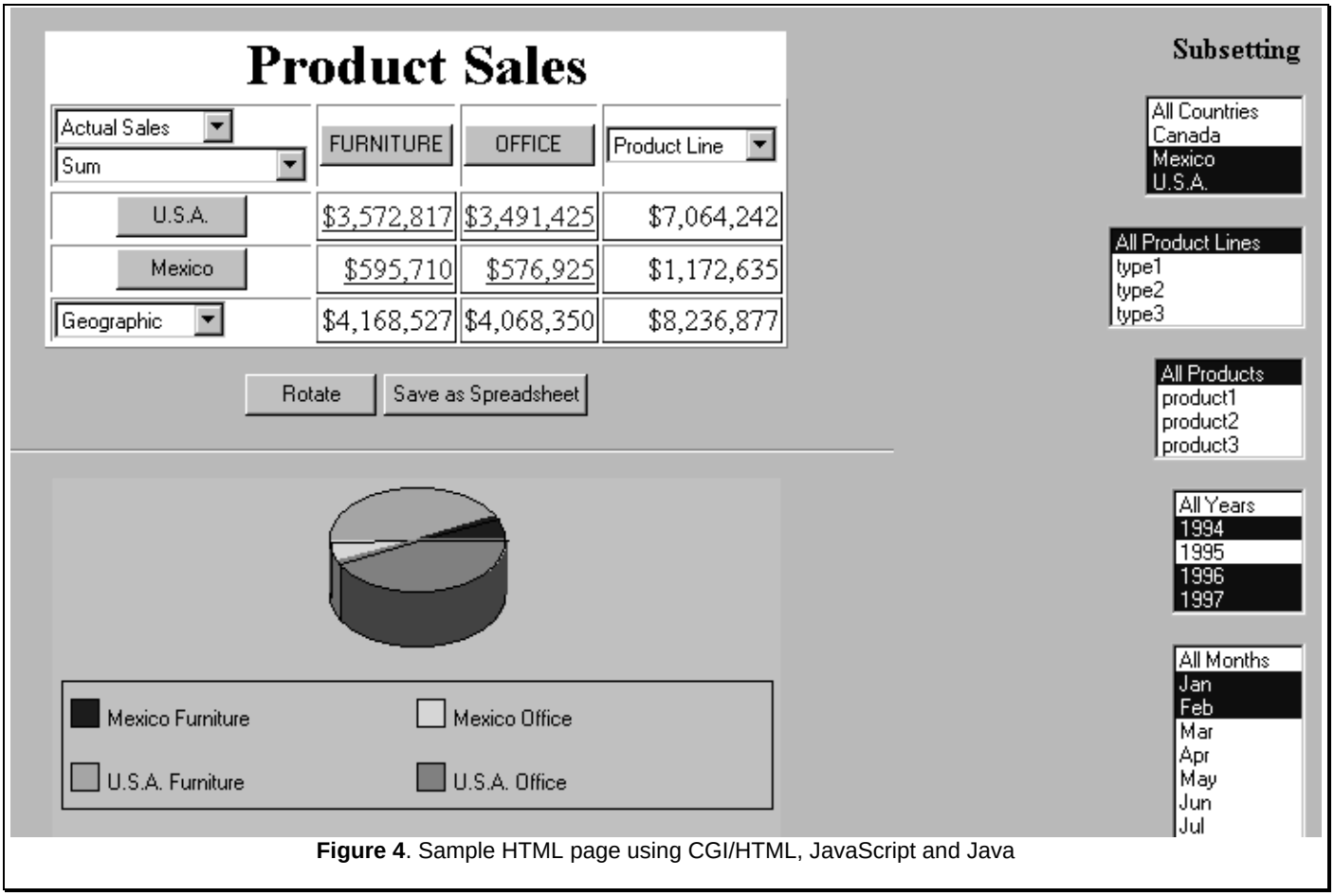

Figure 4. Sample HTML page using CGI/HTML, JavaScript and Java

above) or applets that can build WHERE clauses.

Figure 4 shows a mock up of an report which utilizes a CGI/HTML based Application Dispatcher program to access the data and generate all the HTML to generate the display, including JavaScript to automatically refresh the page when one of the dimensions is selected and a Java applet to draw the graphics

Product Dependencies

SAS/IntrNet software provides the basic *plumbing* or infrastructure to *Web-enable* the functionality of the SAS System. Using SAS/IntrNet software a user can deploy SAS functionality to any number of users via an organizations intranet, an extranet, or the world via the internet itself. SAS/IntrNet was specifically architected to make it easy for SAS users to deploy new and existing SAS applications via the Web. In order to provide licensing flexibility a site can license just the products they need in order to meet their needs. Thus it is not necessary to separately license Web-enablement for each product area (e.g., SAS/ETS software, SAS/QC software). Instead, a can site license SAS/IntrNet software on a server basis. And to provide even more flexibility, a site can license just the products they need based on their application needs.

Table 1 shows the product dependencies for what the user must license depending on the component being used. Of course, Base SAS software is always a requirement. SAS/GRAPH software is not included in the table; it is a requirement however if graphical output is to be generated.

Included in Table 1 are both the SAS/intrNet components discussed earlier as well as a variety of supported applications (e.g., the MDDB Viewer and the Data Warehouse Viewer).

The following points should be noted when reading the table:

- For some components, there may be multiple combinations of products that are sufficient to use the component. For example, both htmSQL and the SAS/SHARE*NET Driver for JDBC require a SAS Data Server. That server can be either SAS/SHARE software (which also requires that BASE and SAS/IntrNet be licensed) or the Scalable Performance Data Server.

- The SAS/SHARE*NET Driver for JDBC can query either a SAS/SHARE server or a SAS session started with the JConnect protocol. In either case, both SAS/IntrNet and BASE SAS software are required.

Web Tools Component	Required Server Products						
	Base SAS software	SAS/IntrNet software	SAS/SHARE software	SAS/CONNECT software	SAS/EIS software	SAS/Warehouse Administrator software	Scalable Performance Data Server
Publishing Tools	X						
Application Dispatcher	X	X					
MDDB Report Viewer	X	X			X		
Data Warehouse Viewer	X	X	X			X	
htmSQL	X_a	X_a	X_a				X_b
JConnect	X	X		X			
SAS/SHARE*NET Driver for JDBC	$X_{a,b}$	$X_{a,b}$	X_a	X_b			X_c
SWAN IDE	X	X		X			
Remote Object Class Factory	X	X		X			
SWAN Business Viewer	X	X		X	X		
Lightweight Applets							

LEGEND:
X means the product is required. When a subscript is present, products with a common subscript are sufficient to use the component

Table 1. Required products.

That is why either SAS/SHARE or SAS/CONNECT can be used. Which to use depends on where the data exist and whether a SAS/SHARE server is needed for other purposes.

- The Web Publishing Tools do not require that SAS/IntrNet be licensed. They are freely downloadable from the SAS Institute Web site, http://www.sas.com/web, and only require Base SAS software. SAS/GRAPH software is, of course, also required for graphics generation.

- The *Lightweight Applets* are standalone special purpose applets that will work as long as the HTML applet page contains the appropriate parameters and values. The HTML should be dynamically generated by a facility such as SAS/IntrNet software. However, they technically do not require SAS software.

- The Data Warehouse Viewer is a sample application that works with the metadata created and managed by the SAS/Warehouse Administrator. Licensing of the SAS/Warehouse Administrator is only required to create and manage that metadata.

- ROCF and SWAN/IDE will, in the future, support Corba and (perhaps) RMI (Remote Method Invocation). They currently support the JConnect protocol to talk to a back-end SAS server. When using the JConnect protocol, SAS/IntrNet is a required product. At the time this paper was written, the other licensing options for (e.g., for future implementation of Corba and RMI) had not yet been determined.

Which Component(s) to Use

One of the primary design goals for SAS/IntrNet software was to mask the complexity of other technologies such as CGI, HTML, Java, etc. that were needed to implement SAS software based Web solutions. Thus, it is possible for example to deploy an application using the SAS/IntrNet Application Dispatcher, for example, without requiring the user to know anything about CGI or HTML. Of course, a little knowledge of these technologies does give the developer more flexibility in implementing their web solution. However, the application and/or the data usage factors, should be the driving force behind determining what technologies to use. Implementing a CGI/HTML solution that does not provide the level of interaction needed should not be considered if it does not meet the user's needs. Likewise, implementing a sophisticated Java applet should not be the goal if the solution can be achieved with other (simpler) technologies. The technology to use depends on a number of factors.

Unfortunately there is no *end-all/be-all* checklist which can tell you specifically what technologies to consider for your application or reporting needs. Table 2 highlights some of the required skills necessary for the basic components of SAS/IntrNet software.

There are a number of different factors and criteria that need to be considered in deciding what components are most appropriate. You will need to consider not only the requirements and the skill sets available for the current application, but also the other web-based applications and skills available in your organization.

SAS/IntrNet Component	Required Skills
Application Dispatcher	• SAS programming skills - Data Step, Macro or SCL.
htmSQL	• HTML • SQL
SAS/SHARE*NET Driver for JDBC	• HTML (minimal - to build the applet HTML page) • SQL • Java
JConnect	• HTML (minimal - to build the applet HTML page) SAS programming skills - Data Step, Macro or SCL. • SQL skills may be required if making JDBC requests to the JConnect SAS session. • Java

Table 2. Minimal Required Skills

Some of the factors to be considered include:

- Are there existing SAS programs which need to make the output available to any/all users in your organization?

  If so, then the options to consider include:
  - using the Web Publishing Tools to statically generate those reports on a scheduled basis using batch processing.
  - using the Application Dispatcher to run those programs dynamically based on user input
  - using htmSQL for SQL based reports
  - if the front-end menus are complex and very interactive the use of Dynamic HTML, JavaScript or a Java applet should be considered.
  - if the current code makes use of SAS/AF software's object oriented capabilities, the packaging the code using the IDE and ROCF should be considered

  Note that in considering these options, you must keep in mind that while there are a number of simple uses of JavaScript, it is not an easy language to learn. And learning Java requires an even bigger time and training investment.

- Is this a new application or an extension to existing SAS applications?

If a new application, then the choice of technologies should be driven primarily by the available skill-sets within your organization.

If an existing application is to up be updated or expanded then the criteria above should be used in conjunction with the available skill-sets.

- Are the reports generated from static data or is the data changing frequently?

  If the data are static and the reports do not need to be customized based on user preferences, then Web Publishing may be the way to go. Otherwise a dynamic application may be necessary.

  If the report format is simple tables which can be generated with SQL, then either htmSQL or the SAS/SHARE*NET Driver for JDBC may be the best option. Typically, sending SQL requests to a Data Server will run faster than running a program using either the Application Dispatcher or JConnect.

- Are there a fixed set of output requirements or does each user need to ability to request just the output they need?

  This addresses two issues. First is whether the reports can be generated by a batch process and simply published to the Web site (in which the Publishing Tools are sufficient) or do the reports needs to be customized, in which case a dynamic SAS/IntrNet application is required.

  Second, are the details of the exact layout of the report fixed. Many dynamic reports can be easily generated using the Application Dispatcher and the Web Publishing Tools. If more exact control over the layout of the reports and output are needed, then some knowledge of HTML will be required when using either the Application Dispatcher (with custom Data steps using PUT statements) or htmSQL.

- What kind of processing is needed?

  For simple queries and tabular displays, the use of htmSQL should be the first choice.

  For applications that require complex data management or data analysis then one of the Compute Services (e.g., Application Dispatcher of JConnect) should be considered first.

- What are the skill sets available in your organization?

  If you have no (or limited) knowledge of CGI, HTML, Java, etc. then the Application Dispatcher using the Web Publishing Tools should probably be your first choice.

  If the majority of your staff is primarily skilled in SAS programming, the Application Dispatcher should be the option you investigate first.

  Alternatively, if you have SQL and HTML knowledge then htmSQL becomes a much more viable option.

  Note that HTML is fairly easy to learn. While JavaScript and Java are much harder to learn. Thus, a minimal investment in HTML training can greatly expand your options.

  Learning JavaScript is an order of magnitude more difficult. Programmers with some experience in object oriented programming will find JavaScript a bit easier to learn.

  Learning Java requires a larger investment of training resources and time.

  Of course, the greater interactivity and flexibility offered by both JavaScript and Java can make the investment in learning them (or hiring staff who already know them) more than worthwhile.

- What can you learn based on what you know?

  HTML is relatively easy to learn (e.g. in a few days or less) one can learn quite a bit about HTML.

  If you have programmers with experience is object oriented programming and syntax, learning JavaScript and Java are a bit easier. Programmers with C or C++ experience should be able to learn Java in a reasonable amount of time.

- Timeframe to deploy?

  If you have a lot of time, then you can give yourself the option of learning the technologies you need while perhaps using alternative technologies (e.g., the Application Dispatcher to web-enable your existing SAS programs) as an interim solution.

- Level of interactivity required?

  Pure CGI/HTML implementations will not provide instantaneous interactivity - each request will require a round trip to the server. If a lot of interactivity is required, then JavaScript, Dynamic HTML, and Java should be considered.

- Frequency of use?

  If the application is going to be run occasionally to deliver specific reports, then CGI/HTML solutions which are quick to download, and do the majority of the work on the server make more sense. If the application will be started first thing in the morning and used all day long (or for a significant length of time), then the extra download time to download a large and sophisticated Java applet is worthwhile.

- Where are the data stored?

  Of course the data must be located somewhere that your applications can access. In addition to providing multi-user update access, SAS/SHARE software can provide cross-platform access. Thus the Application Dispatcher and htmSQL, for example, can access any data as long as SAS/SHARE is installed on the box where the data are and TCP/IP is available.

If using Java applets, then the data must be either located on your web server or you must use the JTunnel version of either JConnect or the SAS/SHARE*NET Driver for JDBC to access your data on other platforms.

- How large are the data files?

  This is an important question because you want to make sure to perform as much data reduction as possible on the server. The download time for large files can have a significant impact on performance. Regardless of the technology chosen, data reduction should be done on the server.

  In addition to the download time itself for large files, there is also the issue that many browsers can not handle large HTML files - especially if formatted as HTML tables. The reason is that HTML tables must be completely parsed before they can be displayed. A large table (e.g., thousands of rows) could very well hang the user's browser. If a (not too) large data set must be displayed as an HTML table, then creating separate HTML tables for subsets of the data will circumvent the parsing problem mentioned above. The HTML Data Set Formatter (one of the Web Publishing Tools) supports *BY GROUP* processing to create separate HTML tables.

Conclusion

SAS software's Web enabling technology, and specifically SAS/Intrnet software, touches all areas of SAS software's capabilities, including the end-to-end data warehousing solution. From analysis tools like our data mining solution, to OLAP, query and reporting , and pre-packaged solutions, SAS/IntrNet software lets anyone with a desktop computer system and an industry standard browser access SAS software's acknowledged decision-support tools. This paper has highlighted the technologies included in SAS/IntrNet software and discussed some of the issues about how to determine which components of the technology are most appropriate based on the specific requirements of a user, organization or application.

References

Flanagan, David (1997), *JavaScript: The Definitive Guide, Second Edition*, Sebastopol, CA: O'Reilly & Associates, Inc.

Isaacs, Scott (1997), *Inside Dynamic HTML*, Redmond, WA: Microsoft Press.

SAS Institute Inc. (1998), *SAS/IntrNet Software: Delivering Web Solutions*, Cary, NC: SAS Institute Inc.

Acknowledgements

The author would like to acknowledge the contributions of the following individuals in the preparation of this paper. They either provided source material for inclusion in the text or provided valuable review comments.

Brian Garrett
Aaron Hill
Tim Mattson
Helena Pearce
Jack Wallace
Renee Harper
Deva Kumar
Rob Owen
Alan Pyne
Barbara Walters

The author can be contacted at:

1700 Rockville Pike
Suite 600
Rockville, MD 20852

(301) 881-8840 x3302
sasdjh@wnt.sas.com