Paper 128-28

SAS® and the Internet for Programmers

David Ward, InterNext, Inc., New Brunswick, NJ

ABSTRACT

The SAS programming community has been bombarded with demos, presentations, user group papers and white papers outlining why we should move our programs to the web. Even with so much emphasis placed on new and exciting ways to access SAS software, most programmers still sit in front of good 'ol Base SAS day in and day out. In this paper we will explore how you can use SAS with the Internet, whether or not you should actually do this, and why many programmers have avoided it altogether.

After thoroughly explaining how the web works (either on the Internet or an Intranet) you will learn about various ways to bring your SAS programs into this new world. This paper should give the reader a deeper understanding of the web and the various SAS products that work with it as well as provide a foundation for making decisions about their own problems.

INTRODUCTION

Many of the papers that have sprung forth from the fertile soil of user group conferences over the past few years have done a good job of introducing SAS programming techniques for the web. Experience has shown the author that although there are plenty of papers (shown by the existence of the Internet, Intranet, and the Web track at most conferences) and enthusiasm from the SAS Institute, most SAS programmers still have nothing to do with the web, or barely even understand it. This paper is for these – the "behind the scenes", nuts-and-bolts programmers who work with SAS code day in and day out. Most of their time may be spent creating, cleaning, or importing data into or out of other data management applications and they, possibly rightly so, do not see how the web can help.

DEMYSTIFYING THE WEB

Most of us use the web every day. Whether it is to view HTML-based email messages (usually the well-beloved unsolicited advertisements), read e-mail through a web browser, or check news, weather, or airline tickets. The web has even more deeply impacted the business intelligence world, with most reporting, analysis, or "enterprise" solutions now available for the web. Users know how to use a browser and often prefer to use it to access business knowledge in the form of reports, documents, or graphs and maps. If you are a SAS programmer who has not learned much about how the web works, it will be difficult for you to jump into SAS/IntrNet courses or books without "getting your feet wet" with Internet terminology first.

The term web stems from the usage of the larger term "world wide web" in the early days of the Internet. The acronym form of this term makes up the first portion of the URL of most large Internet sites today (www.google.com). Since the Internet consists of a collection of computers all around the world that are connected by wires of various kinds, a web is a fitting analogy. Over the years the Internet has grown to look more like a ball of yarn, but the term web has become a colloquialism of the English language so is here to stay.

The web really consists of two important components – web servers and web browsers.

WEB BROWSERS

Web browsers have two main functions – to retrieve web pages and display them. Over the years they have evolved to support a whole host of helper applications and functionality such as media display, but at their heart web browsers just display web pages. A "web page" usually refers to a text document that contains Hypertext Markup Language tags, better known as HTML. HTML is a fairly simple tag-based language that instructions web browsers in how to draw the page on the screen. HTML embeds text, data, and formatting marks all in one document. If you have never looked at HTML, you can usually view what is called the source code of a web page by choosing a menu item in your browser.

Today, browsers are able to render a wide variety of types of content thanks to the use of plug-ins – third party helper applications that can be embedded in the browser do display such documents as PDF files, Microsoft Word or office documents, or audio/visual files like MP3 songs.

One important aspect of how web browsers make requests for certain files is important to understand. What is called the Uniform Resource Locator (URL) is the "address" of a server (physical machine somewhere accessible to the browser) and page or resource on that server. We all know the now familiar syntax of the URL, seen in the example http://www.yahoo.com/dir/page.htm. The http:// portion denotes the protocol to use when communicating with the server (see below), while www.yahoo.com refers to the address of a computer and /dir/page.htm is a specific page on that server. If you begin developing web applications on your computer, you may use a URL that contains the server name "localhost" or "127.0.0.1". This is a special server name that refers to the local computer (i.e. http://localhost/cgi-bin/broker.exe).

WEB SERVERS

Web servers provide content in response to the requests of web browsers (or other applications that can display and manage content such as wireless phones). Web servers originally returned a web page to a browser that existed as a file on the server's file system. Thus, each web browser has a "document root" directory which corresponds to a URL (web address) requested by a user. If a user requests the URL http://myserver.com/man/home.htm and the web server's document root directory is c:\inetsrv\wwwroot, the server will return the file c:\inetsrv\wwwroot\man\home.htm to the browser. As the Internet matured, it became evident that users wanted access to more than just files that existed on the web server – they needed dynamic content of some kind.

CGI emerged as the standard method for allowing a web server to receive help from an external application installed on the server to handle a request that is more than just returning a file. For example, a user may enter a city and expect to receive a weather report for just that city. If the weather data is in a database that is constantly updated, a program needs to connect to the database and format the results into a web page that the browser can understand. CGI stands for common gateway interface and is just that – a definition of environment variables that the web server should set before calling the external application so that the "CGI program" will know what the user asked for. CGI programs can be either compiled applications (.exe files in Windows) or scripts that are interpreted (like Perl programs) and are not tied to any particular programming language. As of

SAS/IntrNet version 8.2, CGI is the method that SAS/IntrNet uses to communicate with a web server.

THE NETWORK OF THE INTERNET

The Internet is an amazing creation of modern man. It consists of a collection of eclectic computers throughout the world that are connected in some way, however tangled, by a series of wires, routers, and switches. Information flows around the globe on these wires at an amazing pace. Many often wonder how messages can travel back and forth to precisely and quickly from one particular computer to another. This is accomplished by using a network protocol called TCP/IP. The acronym stands for "transmission control protocol" and "internet protocol", and these two sets of networking instructions allow data to be broken down into packets and delivered to the appropriate destination by means of an IP address. The IP address is a four byte number that is used to uniquely identify a computer on a network. It looks like this:

216.25.199.27:80

The four digits (0-255) before the colon represent the IP address, while the last number after the colon is called the port number. When two computers agree to communicate, they must agree on a port number (usually 0 to around 65k). Using port numbers is the key to the same computers being able to talk about different things at the same time (web surfing, email, etc.). In the example above, port 80 is used, which is the default port for HTTP (or web). Each major internet service (like email – POP3) uses a "well-known" port number, making it easy for software developers to know which port number to check for particular kinds of services.

You may be thinking that this paper is getting a little thicker than you had hoped! Why, you ask, should I know about IP addresses and port numbers? If you want to set up and configure any kind of SAS server that can run on the web, you need to understand how other computers (including yourself for testing and development) can connect to it.

Once a web browser and web server establish communication via TCP/IP, they must know how to communicate using a particular set of instructions so that they can anticipate and understand each other's requests. The standard that has emerged over the years for this communication is the text-based protocol called hypertext transfer protocol (HTTP). A fitting analogy is to describe TCP/IP as a telephone and HTTP as a language like French. You can be connected to someone on the telephone but still not understand what they are saying unless you share a common language. You can see from the name that this protocol was designed to send and retrieve HTML based web pages, though it is used for all types of web content today. When a browser makes a request from a server it sends text like the following simple example:

```
GET /results.asp?user=sjones HTTP/1.1
Host: www.myhost.com
```

The server's response looks something like this:

```
HTTP/1.1 200 OK
Server: server-name
Content-Type: text/html

<HTML> - HTML document here
```

The lines of HTTP text are referred to as headers. It is important to understand headers as a programmer because headers contain advanced instructions for the browser that the users don't need to see. Headers allow you to do such things as send back an image, prompt the user to download a file or use a media player, set cookies, or control document caching. If you would like to see real HTTP headers that are being send from your web browser, try running the following simple SAS program:

```
filename web socket ':80' server
  termstr=crlf;
data _null_;
  infile web;
  input;
  if _infile_='' then do;
    file web;
    put 'HTTP/1.1 200 OK' /
        'Content-Type: text/html' / /
        'Hello from SAS!';
    stop;
  end;
  file log;
  put _infile_;
run;
```

Then try loading the following address in your web browser: http://localhost. You should see the HTTP headers in your SAS log!

INTRANET VERSUS INTERNET

The distinction between the Internet and Intranets can be difficult to grasp. The (one and only) Internet is the public system of routers, switches, and computers previously mentioned. In order for a computer to be "on" the Internet (able to communicate with other computers on the Internet) it must be assigned what is called a public IP address. In the real world of business, companies have entire offices or groups of computers that they want to be able to access the Internet but do not want the security problems of having their computers on the Internet just like the rest of the world. Thus, administrators create what is called a local Intranet, or a private network that can have its own web servers (often called internal servers) that only computers on the Intranet can access. This is typically how an internal company home page or portal is accessed – across the Intranet.

Many Intranets are set up to take advantage of a unique feature of the TCP/IP system of addressing – private IP addresses. Certain blocks of addresses have been set aside for use in private Intranets. The Internet knows not to try to find a "real" computer with these addresses, thus many companies can use the same addresses with no delivery problems. The most common examples of private IP addresses are 192.168.x.x and 10.0.x.x. If you have high speed Internet access at home and see these numbers in your computer's setup, chances are that you use an Internet gateway (like a router) to share the Internet connection among several computers. You have unknowingly set up your own private Intranet!

HOW SAS CAN WORK WITH THE WEB

SAS is one of the most flexible and powerful programming languages in the world, although it can be a bit esoteric at times (at least from the rest of the programming world). SAS can work with the web in two ways – reading from the web and writing to or for the web. We will first look at the simpler case – SAS reading from the web.

READING CONTENT FROM THE WEB

The first way that SAS can work with the web we will present is to have SAS function as a web browser, fetching content from web servers. Built into the SAS language are two access methods, the URL access method and the socket access method, that allow SAS programs to connect to web servers and retrieve content just like a browser can do. This technique can be useful for a number of things. The Internet is increasingly made up of databases that have been displayed in HTML format. Often times SAS programmers need to read and parse this data into their own SAS data sets or other RDBMS. This is a natural problem to solve by letting SAS read from the web server direction, in effect, pretending to be a web browser.

The URL Access method is the simplest way of reading data from a web page, as you can see in the following filename statement:

```
filename web url
'http://www.yahoo.com:80/dir/mypage.htm';
```

You need the port number (:80) because that is how SAS chose to wrote the URL access method! Once you have defined your filename correctly you can read from it like you would any other file, which means you can include it in data steps, SCL code, or any other place you would normally read from a file.

The socket access method is a more powerful but complicated way of reading data from web servers. This access method allows SAS to connect to any IP address and port number and communicate with a remote process on this address. To connect to web servers, you can just use the typical port 80. While the URL access method took care of submitting and stripping HTTP headers behind the scenes, with the URL access method you'll have to send them yourself, which gives you much more power in formatting a custom HTTP request. You can, for example, pretend to be a certain kind of web browser, send cookies, post form data, or do other things by creating your own HTTP headers using the socket access method. Please refer to the NESUG 2000 paper on the socket access method by the author for more detailed information and examples.

A third technique for reading data from web pages involves using named pipes (available on most operating systems). There are several good command line programs that let you fetch data from web servers, most notably CURL (http://curl.haxx.se), an open source, thus free, program that does just about anything you'll ever need with reading web pages, including support for SSL (secure communication). The following short example shows you how you could use named pipes to read from the result of a command line program that reads from the web:

```
filename web pipe 'curl command here';
data _null_;
  infile web;
  …
```

CREATING CONTENT FOR THE WEB WITH SAS

The most common way of using SAS with the web is to create content to display to web users. This is probably what you as a reader are most interested in as well. There are two ways to generate content for the web – statically and dynamically. Each is outlined below.

GENERATING STATIC CONTENT

The output delivery system (ODS) was introduced in version 7 and has become one of the integral components of the SAS system. The beauty of ODS lies in its power to natively write a number of different output formats including HTML, PDF, RTF, and Postscript. There is even coming support, by the use of the tagset feature, for XML and Microsoft Excel. Of key interest to this paper, of course, is the HTML output destination.

One of the great things about ODS is the fact that you can not only create a single HTML output file, but can create an entire set of files complete with a table of contents frame to easily navigate the results. This paper is not the venue to illustrate ODS code, as it is becoming quite a popular topic at SAS conferences and undoubtedly there will be many papers in these SUGI proceedings on the topic. What remains to be pointed out, however, is when using ODS to generate physical files (static content) is most useful.

When the number of classification levels of your report is relatively small (for example, year, gender, etc.) thus the total number of reports generated for a particular application is manageable, this technique works quite well for publishing output to the web. You do not have to bother with setting up a SAS server, learning how to configure web servers and CGI programs, or worry about maintaining or administering your own server to run SAS. You simply create the files on your local machine, then transfer them to the web server or a drive accessible to the web server. You can already imagine, though, that if you have thousands of reports or your data changes very frequently and it is important for your users to have updated reports, the static solution can turn into a nightmare.

GENERATING DYNAMIC CONTENT

Giving your users the power to choose their own input parameters, run a SAS program, and view/interact with the results all through a web browser without the need to have SAS installed on their computer is the amazing power of dynamic web-based SAS content. If you have hardware, money, and ultimately patience (to learn about networking, web servers, and how to configure them all together!) getting your own dynamic SAS environment up and running is well worth it. In fact, since the SAS language (both base SAS and SCL) are so powerful, you can do much more than just creating reports with ODS – you can create entire web-based applications all using SAS.

HOW SAS/INTRNET WORKS

Now that you have a little bit of network and internet terminology and concepts in the back of your head from the beginning of this paper, let's take a look at how SAS/IntrNet, SAS Institute's original product for web-enabling SAS, works. SAS/IntrNet works by connecting a web server to SAS sessions that are acting as servers via a CGI program. Recall that CGI is a standard for allowing web servers to call helper applications and that is just what the broker (the name of the SAS/IntrNet CGI program) program does – it passes the user's web information (such as form data) to SAS.

The broker takes care of finding a SAS session on the network that is free and ready to process the program that the user requested. The entire process happens very quickly, so that the user does not even have to know that SAS is being used to create the HTML page that they get in return. This is accomplished by maintaining a pool of SAS sessions that are ready and waiting for users instead of forcing the broker to start up a new SAS session for each request.

Once you have SAS/IntrNet installed and running, you can call your SAS programs with a URL similar to the following one (pardon the wrapping to fit in the column):

```
http://localhost/cgi-
bin/broker.exe?_program=mylib.myprog.sas&serv
ice=default
```

Consult your SAS/IntrNet documentation for more information about how the _program and _service parameters work. You can then create HTML forms that point directly to broker calls thus will pass the form data to your SAS program. Form data is available in your SAS program in the form of macro variables, which makes it very easy to process statements based on what the user selected and chose to pass to your program.

ALTERNATIVES TO SAS/INTRNET

If SAS/IntrNet is not an option for you, either because of price, or corporate red tape, you still have options for how to set up a dynamic SAS environment on your slice of the web. First, you can write your own CGI program in any language you choose (Perl is a pretty standard language) to either start up SAS or communicate with SAS sessions via TCP/IP socket communication (how SAS/IntrNet works). This will take time and learning, but the skills you acquire from going through this process will prove very useful in your web endeavors. For a step by step guide for writing a Perl program that starts up SAS for

each request consult the author's paper submitted at PharmaSUG 2001.

If you are not interested in writing your own CGI program and are interested in a fully tested commercial product to web-enable SAS, you can take a look at Axom Server Tools, from InterNext (the author's company). Axom includes a free learning edition that will work with Base SAS and the Learning Edition of SAS, and also has a robust commercial server that is built to be compatible with SAS/IntrNet. You may be surprised at how stable, advanced, feature packed, easy to use and install, and cost-effective Axom is. You can learn more about Axom, download the free and full versions, and contact InterNext by navigating to our website at http://www.internext-inc.com.

OTHER INSTITUTE PRODUCTS

There are a number of other SAS products that could potentially allow you as a programmer to publish dynamic SAS output to the web. These are briefly outlined below.

WebAF is part of the product called AppDev Studio. This is a visual development environment for writing Java applications that communicate with SAS to retrieve data and reports (like OLAP reporting). WebAF uses SAS/IntrNet and SAS/Connect behind the scenes to communicate with SAS code. It is important to note that with WebAF you are really writing predominantly Java code, thus your existing SAS programming expertise and investment will not go as far.

Java classes for SAS/Share and SAS/Connect are freely available. If you own either of these server products, you can write Java code to either submit SAS code and return the results, or read directly from a SAS/Share server. Because Java is so integrated with the web in the form of Java Servlets, Java Server Pages, and even Java Applets, it is a natural extension of this technology to use it with the web.

The final product of note in this paper is Integration Technologies. This is the Institute's big push to move a lot of server based functionality into one server. IT can "play well" with other programming languages and applications, helping SAS achieve one of its goals for version 9, interoperability. To access SAS from the web using IT, you can use Java Servlets or Java Server Pages, ISAPI applications (written in any programming language), CGI, or Microsoft standards like Active Server Pages. Unfortunately, as with WebAF, there is a heavy demand for non-SAS programming (whether it be VB or Java, etc.). Keep this in mind, as well as a reputed large price tag for a server, when making the decision as to what server technology will best suit your needs.

CONCLUSION

After learning about how the web works, how SAS can work with the web, and about several products that are available to help you use SAS on the web, you should have a clearer understanding of how SAS can be useful to you on the web. But alas, for some of you, you may still wonder how putting SAS on the web can really help you. Perhaps all of your output is in text-based form (like in the Pharmaceutical industry), or perhaps you have no output whatsoever (you only read and write data). In these cases, don't let an author like me try to pull you down a path that will only waste your time and money. While there are almost always innovative ways to revise a process to use the web, you or your group may not be ready to investigate such changes if a well-established process already exists and is fulfilling your business goals.

ACKNOWLEDGMENTS

SAS and SAS product names are registered trademarks of the SAS Institute. Other trademarks are the properties of their respective owners.

CONTACT INFORMATION

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

David L. Ward
InterNext, Inc.
354 George St. Suite 201
New Brunswick, NJ 08901
Work Phone: (732) 545-1035
Fax: (732) 875-0434
Email: dward@ffthinking.com
Web: www.ffthinking.com