



SAS/IntrNet® Software: A Roadmap

Table of Contents

Introduction	1
An Overview of SAS/IntrNet	1
Choosing a Web Technology	2
Terminology	2
Components of SAS/IntrNet.....	5
The Application Dispatcher, Compute Service.....	5
htmSQL, Data Service.....	6
SAS/CONNECT Driver for Java, Compute Service	7
SAS/SHARE Driver for JDBC, Data Service.....	7
Tunnel Feature	8
SAS Design-Time Controls (DTCs).....	8
Xplore Sample Application	8
Web Publishing Tools and the Output Delivery System.....	9
The Web Publishing Tools	9
The Output Delivery System (ODS)	9
Other SAS Web Components and Technologies.....	10
The SAS ODBC Driver, Data Service	10
Thin-Client Graphics.....	10
AppDev Studio, a SAS Applications Development Environment.....	11
The SAS Information Delivery Portal	12
Industry Components.....	12
Scripting Languages.....	12
DHTML	14
Component-based Architectures	15
Product Dependencies	17
Which Component(s) to Use.....	18
Conclusion.....	22

The content providers for *SAS/IntrNet: A Roadmap* are Mark Torr, Primary Content Provider, and Ingrid Hertel, Content Provider, in SAS Heidelberg, Germany.

Introduction

The rush to take mission-critical applications to the Web has changed the way companies compete and interact. This competition has produced a lower cost-per-desktop and has extended information access to just about anyone, whether through a Web browser, a personalized portal or a handheld device. SAS' Web-enablement technology brings companies the crucial ability to make knowledge-based decisions via the Web.

Since 1997, SAS' Web technology has provided companies the perfect vehicle for delivering global strategic solutions, such as customer relationship management, business performance management, financial analysis, e-intelligence and more, at the lowest possible cost with the highest rate of return.

This paper focuses on SAS/IntrNet software, a mature and proven technology for the deployment of Web solutions, which provides access to the power of SAS. You'll learn how you can use the various components of SAS/IntrNet in an information delivery infrastructure to help you turn data into strategically useful information, using a technology that is available to almost everyone and that most users already know: the Web. The paper may mention other Web technologies as applicable. Where possible, we include references to further reading.

An Overview of SAS/IntrNet

SAS/IntrNet opens the SAS System to the Internet, extranet or intranet. This software provides a solution for running ad-hoc reports and dynamic applications via the Web. SAS/IntrNet enables users to point-and-click on their Web browsers and get the information they need, thus bringing the power of SAS to the desktop through the medium of a Web browser. SAS/IntrNet provides all this capability without having to install SAS on everyone's desktop and without everyone needing to learn SAS in order to access the information required to make effective business decisions.

SAS/IntrNet comes equipped with a modern set of enabling tools and multiple "out-of-box" information delivery applications that you can point at data, immediately after you have installed the software. To make it easier to create SAS/IntrNet applications, SAS Design-Time Controls (DTCs) are also included with the software. DTCs provide a way to use a visual Hypertext Markup Language (HTML) editor as a point-and-click interface to perform complex, back-end SAS processing.

Broadly speaking, SAS/IntrNet is divided into three areas:

- **Data services** that let the user make SQL-type queries to the SAS server. Users can query, update and report data.
- **Compute services** that give the user full access to the analytical capabilities of the SAS server. Users can access and use any non-visual functionality provided by the SAS server.
- **Out-of-box applications** that use the SAS/IntrNet data and compute services to deliver access to MDDB cubes and to report the contents of SAS catalogs and libraries.

Choosing a Web Technology

Because organizations often use a diverse collection of technologies to store and manage information, you should consider many factors when developing a Web-based information delivery system, such as:

- User interface requirements.
- Data access and analysis requirements.
- Application performance.
- The costs of development, deployment and maintenance.

Terminology

Web technology has had a dramatic impact on information delivery and the applications development process. Today's technologies provide many benefits over the client/server model because it is no longer necessary to distribute and install applications on each desktop. This advantage makes information accessible from a Web browser.

The advent of the wireless industry has further extended this advantage to handheld clients, allowing for the delivery of information anywhere at anytime. The Web, as the main delivery mechanism, provides the infrastructure to ease the deployment of applications so that a client can request services, such as data access or processing, from another machine.

Web-enabled information delivery systems can be described in terms of the function they serve, how they work and the types of technology they use.

- **Web publishing** is the creation of static reports, which are uploaded to a Web server. Any user with a Web browser (such as Microsoft Internet Explorer or Netscape Navigator) 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 for text-based and GIF or JPEG for graphics are some examples of valid Internet content form.
- **Report distribution** means that reports can be customized for specific user needs. Simple queries constructed on the clients are passed through the Web server to a data repository, and then formatted as a report to be viewed on the client, usually a Web browser.
- **Application distribution** means that requests for decision support services can be initiated on the client, usually via a Web browser or a Java applet, and then executed by an application server with the results available to be viewed by the client.
- **Data services** lets you create Web-based applications that let the user view and query data from a Web browser. You create these applications through Web-enabled SQL. At the user's request, dynamic queries can be sent to a content server. SAS/IntrNet data services can be accessed via the Common Gateway Interface (CGI) or Java technology.

- **Compute services** provide the ability to not only query or view data, but to actually process the data on the back-end server. Doing so means it is possible to analyze the data, create custom reports with interactive graphics, do OLAP analysis and interact with data in the warehouse. SAS/IntrNet compute services can be accessed via the CGI or Java technology.
- **CGI** is a programming interface that allows a Web server to communicate with an external program. Typically, CGI programs are small, written in a script or a 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 compute services). When a Web browser accesses a Uniform Resource Locator (URL) for a CGI program, the Web server executes the CGI program. Usually, the CGI program invokes a session with a database or an application server, which processes the request from the client and returns the result to the Web browser via the Web server. Strictly CGI-based applications need repeated interaction with server(s) to provide interactivity with the user.
- **Dynamic HTML (DHTML)** was created to try to overcome the interaction limitations of standard HTML. Currently, Netscape Communicator and Internet Explorer support DHTML; however, DHTML suffers greatly due to different support levels and proprietary extensions that often make cross, Web-browser compatibility impossible. DHTML adds significant capability both in terms of interactivity and visual effects.
- **JavaScript** is a lightweight, interpreted programming language with simple object-oriented capabilities. JavaScript operates on the client side (i.e., the Web browser) and lets you include executable content in Web pages.
- **Java** is an object-oriented programming language developed and promoted by Sun Microsystems. This language is expressly designed for use in the distributed environment of the Internet. Java enables the creation of three types of applications.

The first type, and perhaps the most well known, is the **Java applet**. This type consists of two parts: an HTML page, which contains a reference to the applet, and a set of Java classes, which contains the program logic. The HTML page specifies which Java applet is to be invoked. (In fact, a single HTML page can include multiple applet references.) When a Web server returns an HTML page to a Web browser that contains a reference to an applet, the Web 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 input from any user. (Note that after leaving the HTML page that contains the applet, the applet is stopped and removed from memory.) For example, the applet can establish a connection to a database or application server, submit requests for processing, and receive and display the results. Because the Java applet is executing locally on the client side, it can provide a much richer interactive environment than a CGI program.

Another type of Java application, known as a **Java Server Page (JSP)**, allows the execution of Java code on the Web server. A JSP is an HTML page that contains intermixed Java code that is executed when the page is requested to produce dynamic content. This process reduces the need to download Java classes to the local machine, reduces the need to ensure that your clients all support the version of Java that you need (a major problem in the Java applet space) and speeds up execution. JSP provides greater benefits than CGI because JSP has the advantage of putting the power of Java at your fingertips.

Here's how a JSP works. A user requests a JSP from the Web server by requesting a specific URL. The Web server recognizes that the page is a JSP and passes the request to a servlet container that transforms the JSP into a Java servlet that is executed. (Think of a Java servlet as a Java applet without a face that runs on the Web server.) If the page has already been requested once and the JSP has not been modified, the previously created Java servlet is executed without the transformation taking place. Normally, the JSP will return Internet content (e.g., HTML, JavaScript, DHTML) or some other markup language to the Web browser that is responsible for rendering it.

The third type is a **Java application**, which executes the Java code outside of the Web server and the Web browser. In this sense, a Java application is similar to any executable you have on your machine that runs a program.

- **JavaBeans** is a component architecture framework for Java. This component defines a software Application Program Interface (API) that lets developers write re-usable components once and run them anywhere a Java virtual machine is available. The Java virtual machine interprets the bytecode into code that will run on the real computer hardware. JavaBeans bring to the Java environment what the ActiveX framework brings to the Microsoft Component Object Models (COM) environment.
- **ActiveX controls** are a Windows-only implementation that takes advantage of the Microsoft Windows implementation of COM technologies to provide interactivity and interoperability with other types of COM components and services. ActiveX controls are a next generation of Object Linking and Embedding (OLE) controls (OCX), which provides a number of enhancements specifically designed to facilitate distribution of components over the Web. ActiveX controls can be embedded in HTML pages (like Java applets) and then executed on the client in much the same way.

Java applets and ActiveX controls differ in that ActiveX controls only run on Windows, whereas Java applets are operating environment independent. When ActiveX controls are downloaded, they remain locally installed, can access the local PC file system and can communicate with any machine on the user's local network. For security reasons, Java applets have no access to the local file system by default. Java 2 (the latest version of Java) lifts these restrictions if the applet creator provides the required information and if the local user accepts that the applet is coming from the originator from whom it says its coming.

- **Active Server Pages (ASP)** can be compared to JSPs in that ASPs are programs that execute on the Web server and return Internet content to the client. ASP is a Microsoft standard and is only found on Microsoft Web servers.

Components of SAS/IntrNet

SAS/IntrNet contains a variety of components that you can use individually or in combination to address specific needs of a Web-enabled information delivery solution. You can use a single component only, or you can create a solution that involves multiple components, in which each component provides services that best meet your requirements. The power of SAS/IntrNet lies in the remarkable range of each component.

The Application Dispatcher, Compute Service

The Application Dispatcher is made up of two components: the Application Server and the Application Broker. The Application Dispatcher provides a CGI gateway between a Web browser and SAS. This gateway lets users build dynamic applications that can access the power of SAS from a Web browser. The gateway also provides the capability to run any SAS program that can be executed in batch mode. The users need only work with the details of the SAS program. All of the details relating to CGI, e.g., communicating the parameter values from the HTML page and returning the results to the user's Web browser, are handled by the Application Broker and require no knowledge of CGI. The Application Broker passes all incoming requests to the Application Server, which executes any SAS program that is can be executed in batch. The Application Dispatcher framework allows the deployment of SAS programs to multiple users (whether or not they have SAS installed).

Figure 1 shows the Application Dispatcher architecture.

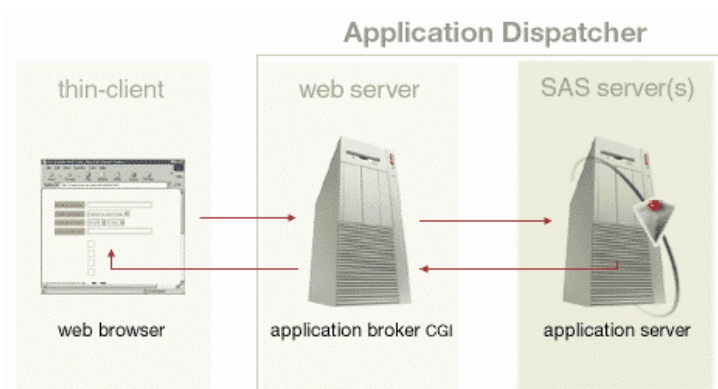


Figure 1: Application Dispatcher Architecture

SAS/IntrNet also delivers an intelligent load manager, which is another CGI component that can be used to intelligently route and handle a large volume of incoming requests. Based on its configuration, this load manager can start and stop remote servers based on demand, and route requests across multiple back-end application servers, regardless of the operating environment.

Figure 2 shows an example in which a new server is started because all others are busy. Of course, you can limit the number of servers that can be started at any one time to avoid a denial of service attack.

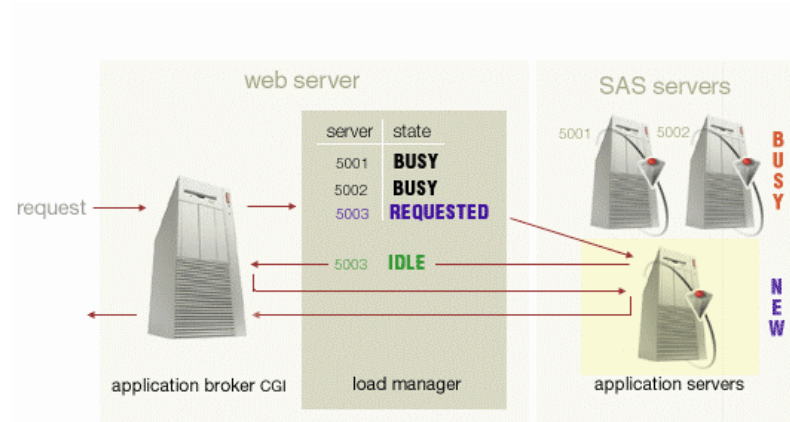


Figure 2: Load Manager for Scalability

SAS/IntrNet delivers complete end-to-end security. Figure 3 shows the security features you can use with SAS/IntrNet.

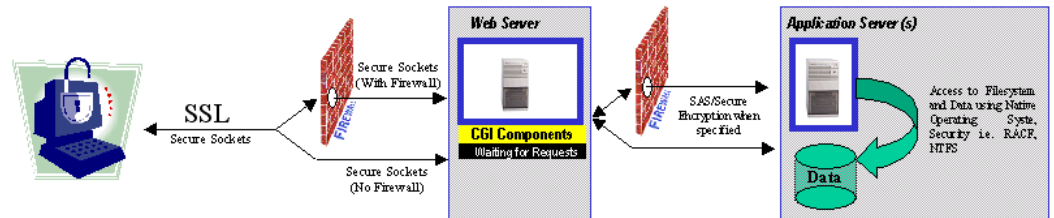


Figure 3: End-to-end Security

For security between the Web browser and the Web server, you can use any industry standard available on the market, such as Secure Sockets Layer (SSL), with or without a firewall. Starting with SAS Version 8, the SAS server also supports native operating environment authentication and authorization to protect access to the file system. With SAS/IntrNet Release 8.2, encryption and firewalls are enabled for data being passed between the Web server and the Application Server. SAS proprietary encryption is available with base SAS. Other encryption standards are available with SAS/Secure. This comprehensive security framework lets you customize your security, based on your organization's needs.

htmSQL, Data Service

htmSQL offers a gateway to SAS data from a Web browser, letting you build dynamic queries. htmSQL consists of a CGI program that resides on the Web server and can be used to access and update a SAS data set (including read access through views to an external DBMS). The user provides an input file (a .hsq file) containing SQL statements embedded in HTML, and htmSQL submits the statements to a SAS data server (either SAS/SHARE or the Scalable Performance Data Server). htmSQL retrieves and formats the results according to the HTML embedded in the

.hsql file. You can use htmSQL to create sophisticated, dynamic applications that let users manipulate report data to address their specific information requirements. htmSQL can also be used to generate any other type of markup language such as Extensible Markup Language (XML), Compact HTML (cHTML), Handheld Device Markup Language (HDML) and Wireless Markup Language (WML). Figure 4 shows the htmSQL architecture.

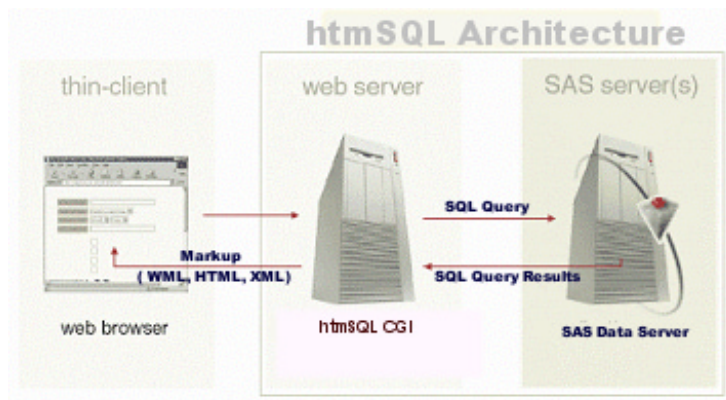


Figure 4: The htmSQL Architecture

SAS/CONNECT Driver for Java, Compute Service

The SAS/CONNECT driver for Java is a set of Java classes you can use to create Java applets, JSPs and Java applications that communicate with SAS software on a server, allowing you to take advantage of remote SAS computing resources. The driver provides functionality that is similar to what a SAS client can do with SAS/CONNECT, except that the functionality is available to any Java program and does not require SAS to be locally installed. The programs that you create using the SAS/CONNECT driver for Java 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. You can use the Java components in AppDev Studio to create applications that use the SAS/CONNECT driver for Java without your having extensive knowledge of Java and with much of the complexity hidden. AppDev Studio includes a copy of SAS/IntrNet for development purposes. See the section **AppDev Studio, SAS Application Development Environment** later in this paper for more information.

SAS/SHARE Driver for JDBC, Data Service

The SAS/SHARE driver for Java Database Connectivity (JDBC) is a set of Java classes you can use to create Java applets, JSPs and Java applications that communicate with a SAS data server (either SAS/SHARE or the Scalable Performance Data Server). The Java programs that you create let a user view and update data by submitting SQL queries and statements through a direct connection to the SAS server. You can use the Java components in AppDev Studio to create applications that use the SAS/SHARE driver for JDBC without your knowing Java and with much of the complexity hidden.

Tunnel Feature

The SAS/IntrNet tunnel feature employs HTTP tunneling to allow Java 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, unless you are using Java 2 and have explicitly allowed this. (Allowing connections to other machines is a security risk.)

When deploying Java applets, this security feature might 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.

The tunnel feature addresses both of these configuration problems. You can use the tunnel feature, with Java applets written using the Java components in SAS/IntrNet or the Java components in AppDev Studio, to eliminate the restriction on where your SAS server runs in relation to a Web server and firewall. JSP applications do not have any of these restrictions because they run as native operating environment applications and have access to all resources.

SAS Design-Time Controls (DTCs)

SAS DTCs make it easy to create SAS/IntrNet applications without your needing to know HTML or SAS programming. DTCs are add-in components for WYSIWYG HTML editors that support the Microsoft-defined design-time controls standard, including Microsoft FrontPage, Macromedia Drumbeat 2000, SoftQuad HoTMetaL PRO 5, webAF and more. DTCs are ActiveX controls that run inside the HTML editor and generate text based on the user's input into dialog boxes. In this way, you can use a favorite visual HTML editor to build SAS/IntrNet reports and applications via a point-and-click interface. With SAS DTCs, you can generate static reports and publish those reports to a Web server. You can also create dynamic reports using JSP and ASP technologies, retrieving and displaying the latest information when a user selects the page from the Web browser. (You don't need to know JSP or ASP coding to do this.)

Visit www.sas.com/rnd/web/index.html for more information on the SAS Design-Time controls and other SAS Web technologies.

Xplore Sample Application

SAS/IntrNet includes a sample Xplore application that lets you take your SAS environment into the world of the Web. Using the Application Dispatcher with the Web Publishing Tools, Xplore can dynamically access a variety of SAS data and file types for reporting, generating graphics and performing drill-down analysis on the Web.

See www.sas.com/rnd/web/intrnet/xplore.html for more information about the Xplore application.

Web Publishing Tools and the Output Delivery System

For users running SAS Version 6, the Web Publishing Tools are a collection of tools that let you generate static Web pages using SAS data and output. You can use these tools with SAS software to create Web content quickly and easily. Although, these tools continue to exist in Version 7 and Version 8 of SAS software, they have become largely redundant because of the Output Delivery System (ODS). The ODS provides an easier way to create static Web output and has much more functionality.

The Web Publishing Tools

The Web Publishing Tools are available with base SAS for many operating environments, including the mainframe, at no extra charge, and include the following:

- The **HTML Formatting Tools** are a collection of macros that enable you to format SAS data sets and procedure output into HTML pages that you can share with Web users. The current formatting tools are the:
 - **Output Formatter**, which saves output from any SAS procedure to an HTML file, making data available immediately from a Web browser.
 - **Data set Formatter**, which converts SAS data sets to HTML 3.x tables. This formatter supports WHERE clauses, BY-group processing and other data presentation capabilities.
 - **Tabulate Formatter**, which converts the output from the TABULATE procedure into HTML 3.x tables. After your tabular report is available in an HTML table, you can easily share it with other users on the Web.
- **GraphApplet HTML Generator** allows you to generate graphs and charts from SAS data. Then, you can use the GraphApplet or SAS/GRAPH Control for ActiveX to display and manipulate the graphics.
- **MetaView HTML Generator** generates the metadata that lets you view SAS/GRAPH output in a Java applet. To view the output as graphics, you must use the MetaViewApplet.
- **RangeView HTML Generator** generates a critical success factor (CSF) from a SAS data set. To view the generated CSF, you must have access to the RangeViewApplet.
- **TreeView HTML Generator** generates a hierarchical tree from a SAS data set. To view the generated tree, you must also have the TreeViewApplet.

The Output Delivery System (ODS)

Starting with Version 7 of SAS, ODS allows the direct creation of HTML files. ODS might be one of the most powerful additions to base SAS in the last 10 years. In general, ODS is a method of

delivering output in a variety of formats and of making the formatted output easy to access. Instead of taking the output from a procedure and creating HTML via post-processing, ODS creates the HTML during the initial creation of the procedure output. This method is more efficient and provides many new options for report layout. Through the addition of a few simple lines of code, developers can convert a standard program into one that creates HTML output without knowing HTML and without modifying existing SAS code.

In addition to HTML, ODS supports many other types of output formats, including Adobe Acrobat Portable Document Format (PDF), Rich Text Format (RTF) for use in Microsoft Word and other word processing programs, XML, WML for use with WAP-enabled handheld devices, ODS also produces GIF-, JPEG-, ActiveX- and Java-based graphics in conjunction with SAS/GRAPH. This capability lets developers create a wide variety of content that can be delivered over the Web with little or no extra effort.

Example

Standard SAS code:

```
PROC Print data=sasuser.class;
Run;
```

SAS Code with ODS to create HTML

```
ODS HTML BODY="C:\output.html";
PROC Print data=sasuser.class;
Run;
ODS HTML CLOSE;
```

Neither the Web Publishing Tools nor ODS is a part of SAS/IntrNet. However, you can use both in conjunction with SAS/IntrNet, and you can use both in batch mode to enable Web publishing.

Other SAS Web Components and Technologies

The following sections provide a brief overview of SAS technologies that you can use for Web enablement.

The SAS ODBC Driver, Data Service

The SAS ODBC driver provides ODBC-compliant Windows applications with read and write access to local and remote SAS data sets. This driver can be used with any programming language that supports the use of ODBC to provide open access to SAS data sets. Many Windows Web servers include functionality to access ODBC-compliant data sources. ODBC is widely used with ASP.

Thin-Client Graphics

Thin-client graphics are specialized, lightweight, visual Java applets or ActiveX controls. These applets and ActiveX controls are lightweight because they do not establish a persistent

connection to a server. Instead, they receive all the information they need from applet parameters. Lightweight applets and ActiveX controls are best used in conjunction with the Application Dispatcher or htmSQL, which can generate the applet and ActiveX HTML calls along with the custom data values that are used as input to the visual component. (Figure 8, later in this paper, shows a Java applet for graphics.) The HTML pages that use these components provide the parameters and values that define what the applet or ActiveX control does or displays.

AppDev Studio, a SAS Applications Development Environment

AppDev Studio is a complete applications development environment that provides easy access to the leading information delivery server from SAS. AppDev Studio provides a development suite that supports various styles of power- and Web-client applications for developing, deploying and maintaining information delivery applications. AppDev Studio gives you the choice of building Java-based applications on the client or on the server, flexible CGI and HTML applications, ASP applications, or traditional full-client applications, with ease and efficiency from within one stand-alone development environment. In addition, this software provides access to the proven, extensive server capabilities of SAS for enterprise applications development.

AppDev Studio includes the two Java components webAF and webEIS. webAF is a complete Java Integrated Development Environment (IDE) tailored for the rapid creation of Java applications, applets, servlets and JSPs that use the power of SAS.

AppDev Studio also includes for development purposes multiple SAS modules that are required to develop Web- and wireless-based applications as well as traditional SAS client applications. This capability greatly simplifies and speeds the development, deployment and maintenance of critical SAS applications.

You can license AppDev Studio in one of two ways, based on your needs and which software your developers already have available to them:

- License the AppDev Studio SAS System Edition if your developers already have a copy of SAS on their local machines and want to extend their ability to build any type of Web application that leverages SAS. This license includes (for development purposes only): SAS Integration Technologies, SAS/CONNECT, SAS/IntrNet, SAS/SHARE, webAF and webEIS.
- License the AppDev Studio Standard Edition if your developers do not currently have any SAS software on their machines and if you want to provide them with a complete suite of technologies so they can develop Web and wireless or traditional client/server applications that leverage SAS. This license includes (for development purposes only): base SAS, SAS/GRAPH, SAS/FSP, SAS/AF, SAS/EIS, SAS/CONNECT, SAS/IntrNet, SAS Integration Technologies, SAS/SHARE, webAF and webEIS.

Visit www.sas.com/products/appdev/index.html for more information on AppDev Studio.

The SAS Information Delivery Portal

The SAS Information Delivery Portal uses the power of SAS to deliver information to an entire enterprise. In addition, this software's ability to facilitate access to information and data analysis while providing a way for the user to tailor delivered information, increases a knowledge worker's efficiency and ability to make informed decisions.

The Information Delivery Portal has the capabilities to host many types of content. The content type being hosted can be static HTML-based reports, Web applications, Web links and much more. Regardless of its type, content in the Information Delivery Portal can be divided into tightly coupled or loosely coupled.

Loosely coupled content can be accessed by the Information Delivery Portal and accessed outside of the Information Delivery Portal. For loosely coupled content, the portal aggregates content and applications, but security, where required, is handled at the application level. Loosely coupled content requires knowledge of standard Web technologies, such as Java and HTML, and of the Java components (webAF and webEIS) in SAS/IntrNet or AppDev Studio.

Tightly coupled content can be accessed only by the Information Delivery Portal, after valid credentials have been provided. Tightly coupled content is very secure. It requires the greatest amount of knowledge about the Lightweight Directory Access Protocol (LDAP) and of SAS Integration Technologies, because you have to register metadata in the directory server.

See www.sas.com/products/portal/index.html for more information on the SAS Information Delivery Portal.

Industry Components

By definition, the Web is an open environment. The use of other technologies is an integral part of Web-based solutions. Of course, Web servers and Web browsers are standard. However, there are other technologies that you should consider when using SAS/IntrNet and other SAS Web technologies.

Scripting Languages

Scripting languages, such as JavaScript and VBScript, are lightweight interpreted programming languages with simple object-oriented capabilities. Scripting language programs operate at the client side (i.e., the Web browser), and they allow executable content to be included in Web pages. Executable content within the Web page means that Web pages can include dynamic programs that interact with the user, control the Web browser and create the HTML content dynamically.

If development staff is well versed in VBScript and the user community will be using Microsoft Internet Explorer as the Web browser, then VBScript might be a viable choice for a scripting language. JavaScript is more widely used and supported and should be the scripting language of

choice. One of the more important capabilities of JavaScript is the ability to define code fragments that are to be executed when a specific event occurs.

For example, you can use JavaScript code to

- Validate input (e.g., check for required fields, numeric values, etc.) before submitting a CGI request.
- Submit a CGI request automatically 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.

The JavaScript function in Figure 5 is a general-purpose function that updates both a text display and a cumulative list of variables in drill-down order. This function is used in the Application Dispatcher Xplore sample application.

```
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 5: Sample JavaScript Function

Figure 6 illustrates sample output that shows screen fragments as the user makes selections. Note that the HTML page is updated instantaneously without running on the server.

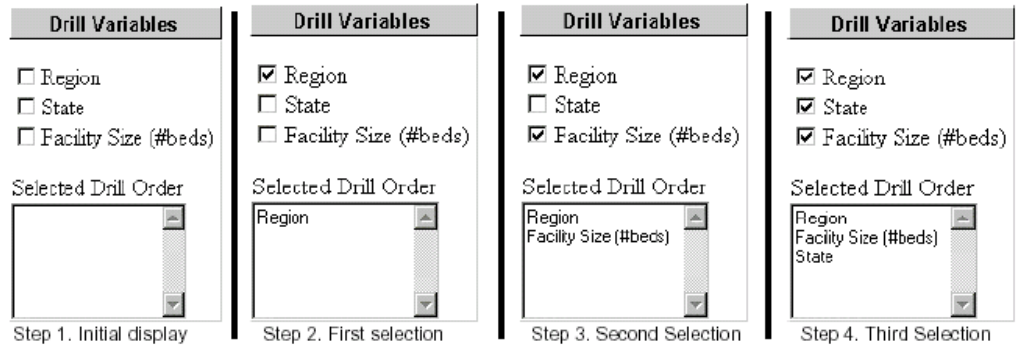


Figure 6: JavaScript Sample Showing Dynamic Update

DHTML

DHTML adds even more capability to create dynamic Web pages easily. Both Internet Explorer and Netscape Communicator support an updated HTML object model that adds significant interactive capabilities. However, at the time this paper was written, the Netscape and Microsoft models were not fully compatible. Thus, when deploying applications, developers must consider what Web browser is used to access a Web page. Hopefully, the two applications will converge toward a common object model so Web developers can use these new capabilities to deploy applications without having to think about which Web browser will be used to access a Web page.

In general, the Microsoft DHTML object model seems more robust and flexible, and it makes it easier to implement interactivity. For example, with Microsoft DHTML, you can use a simple JavaScript function to expand and collapse simple lists that were built using the `<LI . . >` tag. Xplore has an Internet Explorer 4 implementation of the library viewer that is automatically used whenever Internet Explorer 4 or later is the Web browser. While loading, the content of all libraries and catalogs is defined in the initial HTML. However, the DHTML capability in Internet Explorer 4 or later is used to expand and collapse libraries and catalogs without requiring any additional server processing. Figure 7 shows Xplore.

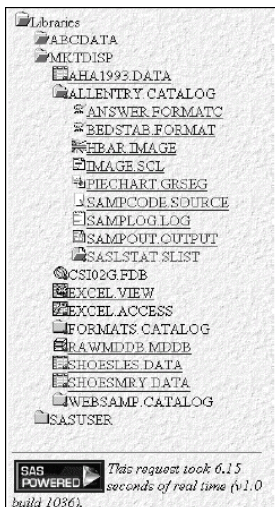


Figure 7: DHTML Sample

To see the DHTML and JavaScript function that delivers this functionality, you can run the Xplore sample available on the SAS Web site at www.sas.com and follow the links for demos to the Xplore sample. Then, view the source.

HTML editors can also make developing and implementing a dynamic Web application easier. You can choose between visual WYSIWYG or text-based HTML editors. WYSIWYG HTML editors (e.g., Microsoft FrontPage and Netscape Gold) provide visual development environments for HTML pages. These pages can be the front-end menus or screens for Web applications. The addition of the SAS DTCs makes the creation of SAS content within HTML editors very simple.

Component-based Architectures

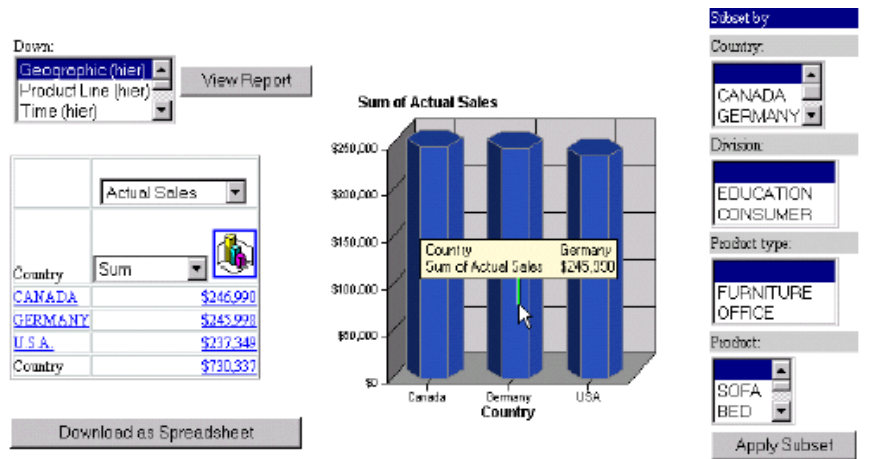
“Intelligent client” has been used to define thin-client applications in which some of the processing and logic is implemented on the client. When “intelligent client” was originally coined, by definition, thin-client applications needed to be implemented in Java as applets. There was no other viable choice available, because CGI and HTML did not provide the interactivity needed to build such applications. Given that CGI and HTML technology have evolved significantly over the last few years, the available technologies for building “intelligent clients” now includes CGI-, Java- and ActiveX-based solutions. The good news is that, no matter which architecture you choose, SAS has the technology to help you.

Today, developers can augment CGI and HTML with DHTML and JavaScript to provide client-side logic previously attainable only with Java and ActiveX. Thus, the term intelligent client can be used legitimately for CGI- and HTML-based solutions. Which technology developers use depends on a variety of factors. (See the section **Which Components to Use** later in this paper.) One might also say that the “intelligent client” technology has advanced to include simple browser-based wireless applications and might progress further as Personal Digital Assistants (PDAs) evolve.

The future of Web applications development consists of a component architecture in which developers can mix-and-match a variety of programming languages and concepts, such as Java,

JavaScript, CGI and HTML, using whichever is most appropriate for each application. In addition to being used for complete applications, we will see Java used to implement small, quick-to-download, function specific applets (e.g., graphics applets or applets that can build WHERE clauses). We will also see an increased use of Java on the server in the form of JSPs and Java servlets, which provide an alternative environment to the ASP technology from Microsoft that runs almost exclusively on Windows Web servers and requires the use of weak scripting languages.

Figure 8 illustrates a report that uses a CGI/HTML-based Application Dispatcher program to access the data and to generate the HTML that produces the display. The report also includes JavaScript to automatically refresh the page when one of the row or column dimensions is selected, and a Java applet to draw the graphics.



- JavaScript used to automatically submit when a list box item is selected
- Java used to generate the graph

Figure 8: Sample HTML Page Using CGI/HTML, JavaScript and Java

Product Dependencies

SAS/IntrNet provides infrastructure to Web-enable the functionality of SAS software. A second set is provided by SAS Integration Technologies. (This paper does not include information on SAS Integration Technologies. See www.sas.com/rnd/web/index.html for more information on this and other SAS Web technologies.)

Using SAS/IntrNet, you can deploy SAS functionality to an unlimited number of users via an intranet, an extranet, the Web or handheld devices. SAS/IntrNet is designed to make it easy for SAS users to deploy new and existing SAS applications via the Web and, as wireless devices are evolving, the flexibility of SAS/IntrNet allows it to exploit this new and exciting area as well. To provide licensing flexibility, you can license only the products you need to meet your requirements on the server to which you want to provide Web or wireless access. These products, because they live on the server on which SAS/IntrNet is installed, can then be exploited over the Web or via a wireless device.

Table 1 shows the SAS software you must license depending on the Web tools component being used. SAS/GRAPH is not included in the table; however, if you need to generate graphics, then SAS/GRAPH is a requirement.

Web Tools Component	Required Server Products
Web Publishing Tools and ODS	base SAS
Application Dispatcher	base SAS and SAS/IntrNet
htmSQL	base SAS, SAS/IntrNet, and SAS/SHARE or the Scalable Performance Data Server from SAS
SAS/CONNECT driver for Java	base SAS, SAS/IntrNet and SAS/CONNECT
SAS/SHARE driver for JDBC	base SAS and SAS/SHARE

Table 1: Product Dependencies

Please note the following:

- For some components, there might be multiple combinations of SAS that you can use. For example, htmSQL requires a SAS data server. That server can be either SAS/SHARE (which also requires that base SAS be licensed) or the Scalable Performance Data Server.
- The SAS/SHARE driver for JDBC can query a SAS/SHARE server or a SAS session started with the SAS/CONNECT driver for Java protocol. In either case, base SAS is required. Thus, you can use SAS/SHARE or SAS/CONNECT. Which driver you use depends on where the data exists and whether a SAS/SHARE server is needed for other purposes.
- The Web Publishing Tools and ODS do not require licensing SAS/IntrNet; however, you must license base SAS.

Which Component(s) to Use

One of the primary design goals for SAS/IntrNet was to mask the complexity of other technologies, such as CGI and HTML, that are needed to implement SAS Web technologies. The Java classes are provided in SAS/IntrNet to allow the exploitation of Java, as that technology emerges, in conjunction with SAS.

Subsequently, AppDev Studio was introduced along with its Java components, webAF and webEIS, to provide the ability to rapidly develop Java applications of all types (client or server side). Although AppDev Studio is not the focus of this paper, it is mentioned here because it provides a complete development environment for SAS with a particular emphasis on the Web and wireless worlds. We suggest that all customers who need to develop applications that exploit the power of SAS, license AppDev Studio for their developers. Of course, doing so is not a requirement.

Using SAS/IntrNet, you can deploy an application that uses the Application Dispatcher without your knowing anything about CGI or HTML. The advent of ODS makes this even easier. Of course, some knowledge of these technologies does give you more flexibility in implementing your Web solution. However, the application and the data usage factors should be the driving force behind determining which technologies to use. A CGI/HTML solution should provide the level of interaction needed by the user; otherwise, it should not be considered because more appropriate technologies probably exist.

Unfortunately, there's no checklist that can tell you specifically which technologies to use for your application or reporting needs. However, Table 2 highlights some of the required skills necessary for using the basic components of SAS/IntrNet.

SAS/IntrNet Component	Required Skills
Application Dispatcher	<ul style="list-style-type: none"> • SAS programming skills using the DATA Step, Macros or SCL • Minimal HTML programming skills for interactive applications (offset somewhat by the DTC)
htmSQL	<ul style="list-style-type: none"> • HTML (minimal knowledge) • SQL
SAS/SHARE driver for JDBC	<ul style="list-style-type: none"> • HTML (minimal knowledge) • SQL. • Java (the Java components, webAF and webEIS, in AppDev Studio can be used to minimize the need for Java knowledge)
SAS/CONNECT Driver for Java	<ul style="list-style-type: none"> • HTML (minimal knowledge) • SAS programming skills using Data Step, Macro or SCL • SQL skills might be required if making JDBC requests to the SAS/CONNECT session • Java (the Java components, webAF and webEIS, in AppDev Studio can be used to minimize the need for Java knowledge)

Table 2: Minimal Required Skills for Using SAS/IntrNet

There are many different factors and criteria you should consider when deciding which components are most appropriate for your development needs. In addition to the requirements

and the skills available for the current application, you will need to consider the other Web-based applications and skills available within your organization. For example:

1. Are there existing SAS programs that are needed to make the output available to some or all users in your organization? If so, then your choices include using:
 - The Web Publishing Tools or ODS to generate static reports on a scheduled basis using batch processing.
 - The Application Dispatcher to run those programs dynamically based on user input.
 - htmSQL for SQL-based reports.
 - DHTML, JavaScript or something that is Java-based if the front-end menus are complex and very interactive.
 - AppDev Studio and the Java component, webAF, if the current code uses object-orientated capabilities in SAS/AF and if the current code was designed using a model-viewer architecture.

When considering these choices, note that although there are a number of simple uses of JavaScript, it is not an easy language to learn.

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

If you are developing a new application, you should choose your technologies primarily by the available skill-sets within your organization.

If you are updating or expanding an existing application, you should use the criteria listed in number 1 above in conjunction with the available skills within your organization.

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

If the data is static and you do not need to customize the reports based on user preferences, then Web Publishing might be your best choice. Otherwise, you might want to consider a dynamic application.

If the report format consists of simple tables that can be generated with SQL, then either htmSQL or the SAS/SHARE driver for JDBC might be best.

Typically, sending SQL requests to a data server will run faster than running a program using either the Application Dispatcher or the SAS/CONNECT driver for Java.

4. Is there a fixed set of output requirements or does each user need the ability to request only the output they need?

This question addresses two issues. First, if the reports can be generated by a batch process and published to the Web site, then the Web Publishing Tools or ODS is sufficient. If you need customized reports, a dynamic SAS/IntrNet application is required.

Second, consider whether the details of the exact layout of the report are fixed. Many dynamic reports can be generated easily using the Application Dispatcher and the Web Publishing Tools or ODS. If you need more control over the layout of the reports and output, then some knowledge of HTML or the markup language you want to create will be required to use either the Application Dispatcher (with custom DATA steps that include PUT statements) or htmSQL.

5. What kind of processing is needed?

For simple queries and tabular displays, the use of htmSQL should be your first choice. For applications that require complex data management or data analysis, you should consider one of the compute services (e.g., Application Dispatcher or the SAS/CONNECT driver for Java).

6. What skill sets are available within your organization?

If your developers have limited or no knowledge of CGI, HTML or Java, then the Application Dispatcher using the Web Publishing Tools or ODS would be a good choice. If the majority of your staff is skilled primarily in SAS programming, the Application Dispatcher should be investigated first, because it will deliver the most rapid results and require the least amount of training for your staff.

Alternatively, if your staff has SQL and HTML knowledge, then htmSQL is a more viable choice. Java offers greater interactivity and flexibility, but often Java skills are not abundant. In addition, there might be some investment needed to teach Java to your staff, or you will need to hire people that already know Java. The Java components in AppDev Studio, webAF and webEIS, greatly simplify Java development.

7. Does knowing other programming languages help with using SAS/IntrNet?

You can learn quite a bit about HTML in a two-to-three day class. If you have programmers with experience in object-oriented programming and syntax, learning JavaScript and Java might be easier. Programmers with C or C++ experience should be able to learn Java in a relatively short period of time.

8. What is the timeframe to deploy?

If time is not a critical factor, you can let your staff learn the technologies they need while using alternative technologies (e.g., the Application Dispatcher to Web-enable your existing SAS programs) as an interim solution.

9. What level of interactivity is required?

CGI or HTML implementations do not provide instantaneous interactivity because each request requires a round trip to the server. If the application requires a lot of interactivity, you should consider DHTML, JavaScript or Java.

10. How often will the application be used?

If the application will be run occasionally to deliver specific reports, 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 (or for a significant length of time), then the extra time to download a Java applet is worthwhile. Another alternative here is to look to the use of JSP.

11. Where is the data stored?

The data must be stored where it can be accessed by your application. In addition to providing multi-user update access, SAS/SHARE provides cross-platform access. Thus, the Application Dispatcher and htmSQL can access any data if SAS/SHARE is installed on the computer where the data is stored and TCP/IP is available. If using Java applets, then the data must be located on the Web server, or you must use the tunnel version of either the SAS/CONNECT driver for Java or the SAS/SHARE driver for JDBC to access data on other operating environments. (If you are using Java 2, you can implement your own security model.) JSPs or Java servlets can use the SAS/CONNECT driver for Java or the SAS/SHARE driver for JDBC to access data on other operating environments without using tunneling.

12. How large are the data files?

The size of the data files is an important question because you should 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 for large files, there is also the issue that many Web browsers cannot handle large HTML files, especially those formatted as HTML tables. Thus, HTML tables must be completely parsed before they can be displayed. A large table might "hang" a Web browser. If a 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. The HTML data set Formatter (a component of the Web Publishing Tools) supports processing by group to create separate HTML tables.

13. Can you safely assume which Web browsers will be used to access your Web application?

A simple CGI/HTML application or JSP application that generates Internet content will most likely run in any Web browser. However, once you decide to use JavaScript, ActiveX, DHTML or Java applets, you need to make assumptions about which Web browsers will be used, or you need to ensure that you can install software on each user's machine to standardize the environment.

Conclusion

SAS' Web-enabling technology, and specifically SAS/IntrNet, touches all areas of SAS' powerful capabilities, including the end-to-end data warehousing solution. From analysis tools like the SAS data mining solution, to OLAP, query and reporting, and prepackaged solutions, SAS/IntrNet provides the perfect vehicle for bringing strategic solutions, such as customer relationship management, business performance management, financial analysis, business intelligence and more to a global audience via an industry-standard Web browser, at the lowest possible cost with the highest rate of return.

Visit www.sas.com for more information on SAS/IntrNet and other SAS solutions.



World Headquarters
and SAS Americas
SAS Campus Drive
Cary, NC 27513 USA
Tel: (919) 677 8000
Fax: (919) 677 4444
U.S. & Canada sales:
(800) 727 0025

SAS International
PO Box 10 53 40
Neuenheimer Landstr. 28-30
D-69043 Heidelberg, Germany
Tel: (49) 6221 4160
Fax: (49) 6221 474850

www.sas.com