Paper 181

Comparison of CGI and Java Technology Provided in SAS/IntrNet™ Software

Barbara Walters, SAS Institute Inc., Cary, NC

Don Chapman, SAS Institute Inc., Cary, NC

Abstract

Which technology to use? The components of SAS/IntrNet™ software utilize a variety of client/server architectures. There are data servers and compute servers, CGI (Common Gateway Interface) programs and Java™ applets. Together, these components provide the technology for delivering sophisticated applications through a Web browser. Deciding which components are best for your application depends upon a variety of factors, including security, user interface issues, your skill set, and Internet versus intranet delivery. We will explore each of these factors and explain how to choose the best components for your application.

Note: SAS Institute continues to provide new components that exploit Web technology. Please check the following URL for the most up to date information on SAS/IntrNet software:

http://www.sas.com/web

Introduction

SAS Institute provides a variety of servers: SAS/SHARE®, SAS/SPDS™, SAS/CONNECT®, SAS/IntrNet Application Server.

Some of these servers provide access to data only, some provide access to the full power of SAS® software. Web technology exploits these servers through both CGI and Java.

SAS/IntrNet CGI components:

- htmSQL
- Application Dispatcher

SAS/IntrNet Java components:

- SAS/SHARE driver for JDBC™
- SAS/CONNECT Driver for Java

This paper discusses the characteristics of each of the servers and the Web technologies that provide access to these servers.

Overview of Data Servers and Web Clients

The SAS/SHARE server and SPDS (Scalable Performance Data Server) provide SQL access to data and are therefore referred to as data servers. Multiple clients can access data simultaneously. Clients can read data, update data, and create new tables.Clients can be

- SAS clients
- Java clients
- htmSQL
- or other clients implemented using the SAS SQL Library for C.

An administrator is responsible for configuring, starting, and stopping a data server. The administrator configures the port number where the server will receive client requests.

The SAS/SHARE server allows the administrator to set the data access permissions for the server and the run mode of the server. The data access permissions determine what data on this server is accessible to clients that connect to the server. The run mode for this server can be set to either "secured" or "unsecured".

The default run mode for the SAS/SHARE server is "unsecured" mode. This mode allows any client to request services from the server. The client can request any data that is accessible to the server. Usually the administrator defines a set of libraries that are available from the server. However, a client can request that a particular library be defined for its use.

The server can also be run in "secured" mode which requires the clients to provide a user ID and password before the client can make requests of the server. The user ID must be a valid identity for the host where the SAS/SHARE server is running. With this configuration, the client can only access the data to which both the server and the client have permission.

In Version 6 of SAS software, the user ID and password can be encrypted before being sent to the server. Encryption support is only provided for user ID and password; the data is returned to the client unencrypted. SAS software, Version 7 provides encryption support for all information exchanged between the client and the server.

After it is started, the server continues to run whether or not any clients are actively using it. Because the server is already running, a client can connect quickly to the server and send requests. When the client is finished it disconnects from the server. Disconnecting from a server does not stop the server. It continues to run until the administrator shuts it down. When the SAS/SHARE server is no longer needed, the administrator can shut it down.

htmSQL and the SAS/SHARE driver for JDBC are the two components that exploit SAS/SHARE and SPDS servers.

htmSQL

htmSQL is a CGI program that can be used to query and update data on the server. A copy of htmSQL is run on the Web server for each request sent from a Web client. htmSQL requires nothing on the client, except a Web browser. All processing done by htmSQL is done on the Web server.

For each request, htmSQL accesses a .hsql file that contains information about what data to access and how the data should be formatted. The .hsql file contains both SQL and HTML directives. htmSQL uses the SQL directives to determine what requests to send to the data

server. It uses the HTML directives to determine how to format the information returned from the server.

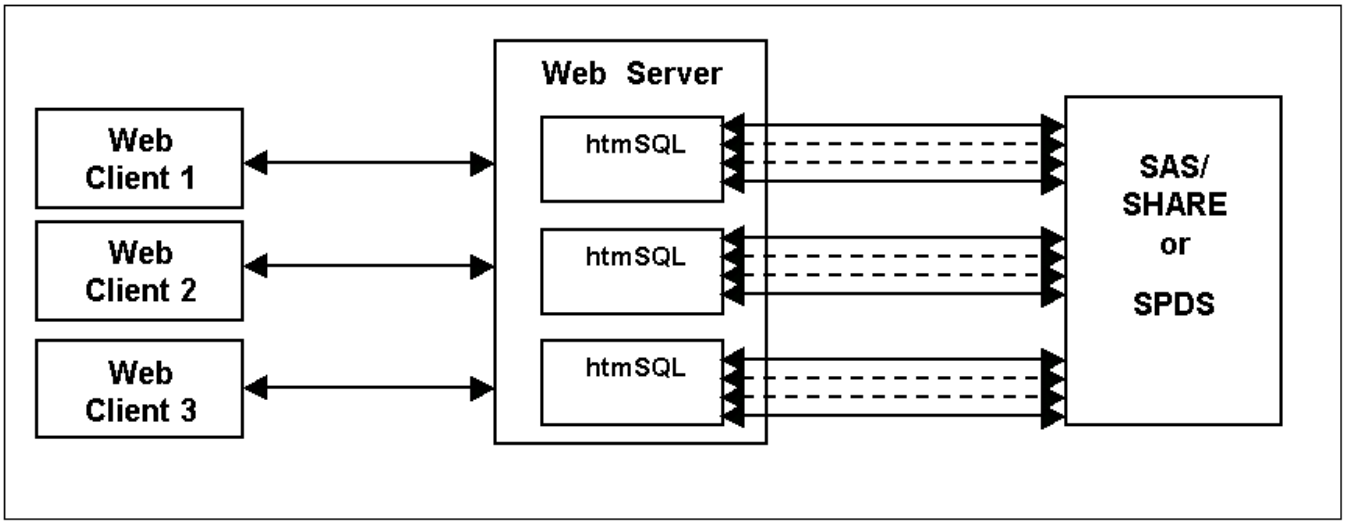

Figure 1

Figure 1 shows the relationship between the Web client, Web server, htmSQL and the data server. Each request from a client causes the Web server to start a copy of the htmSQL program. For each request, htmSQL reads the .hsql file, connects to the server, sends either single or multiple requests to the server, and disconnects from the server. htmSQL then formats the data and returns HTML content to browser.

htmSQL provides some support for security. In order to support a server is running in "secured" mode, htmSQL provides support for user ID's and passwords. This information is usually provided by the client through a fill-in form. This information must be sent to htmSQL for each request that accesses a secure server. This means the user must input this information with each request.

If the user is accessing the Web server by using a secure protocol, all communication between the client and the Web server is encrypted.

In addition, htmSQL will encrypt the user ID and password before it is sent to the SAS/SHARE server. Other communication between htmSQL and the SAS/SHARE server is not encrypted.

htmSQL is an excellent choice for simple applications that access data on the SAS/SHARE server and present it to the user. It can also accept data input by the user and use the SQL UPDATE statement to update the data on the server. Because it is a CGI application and returns information in HTML, it can be used by all users, either Internet or intranet.

In order to use htmSQL, the programmer must have HTML and SQL skills.

SAS/SHARE Driver for JDBC

The Java component that exploits the SAS/SHARE server is the SAS/SHARE driver for JDBC. It also uses SQL to access the SAS/SHARE server. Unlike htmSQL, the JDBC driver connects to the server and maintains that connection as long as necessary.

JDBC is a standard API that is distributed with all Java Virtual machines that support Java 1.1 and later. The SAS/SHARE driver for JDBC is an implementation of this API that accesses SAS/SHARE and SPDS servers.

JDBC provides the ability to access information about the data server itself, such as what libraries are assigned, what tables are available, what columns are contained in a table, what type of data is contained in a column. It allows the program to submit queries to access data or to update data. The latest release of JDBC provides support for accessing specific rows in a table and locking tables for update.

The SAS/SHARE driver for JDBC provides the same security features that are available to a SAS client. When the server is run in secure mode, the driver sends a user ID and password to the server. Unlike htmSQL, user ID's and passwords are passed directly from the JDBC program to the SAS/SHARE server.

This information is encrypted before being sent. An upcoming release of the driver will support encryption of all communication between the driver and the server. This support is dependent upon government approval for distribution.

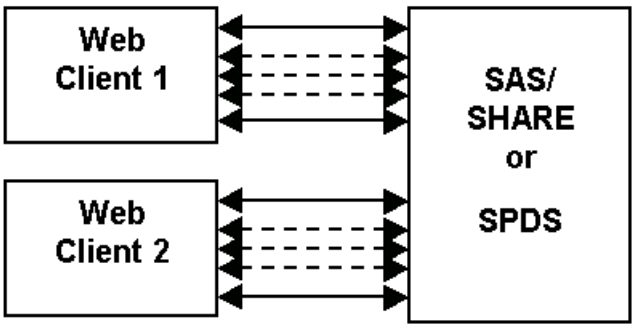

Figure 2

Figure 2 shows the relationship between the Web client and the data server. The Java program (applet) connects to the server, then sends requests to the server. It does not disconnect from the server until the applet closes the connection. This is usually done when the user leaves the page that contains the applet.

A JDBC program is the best choice if the user interface requires custom components, or if there are a number of requests sent to the server. It also supports a standard API, so other programs written to the JDBC API can use the SAS/SHARE driver.

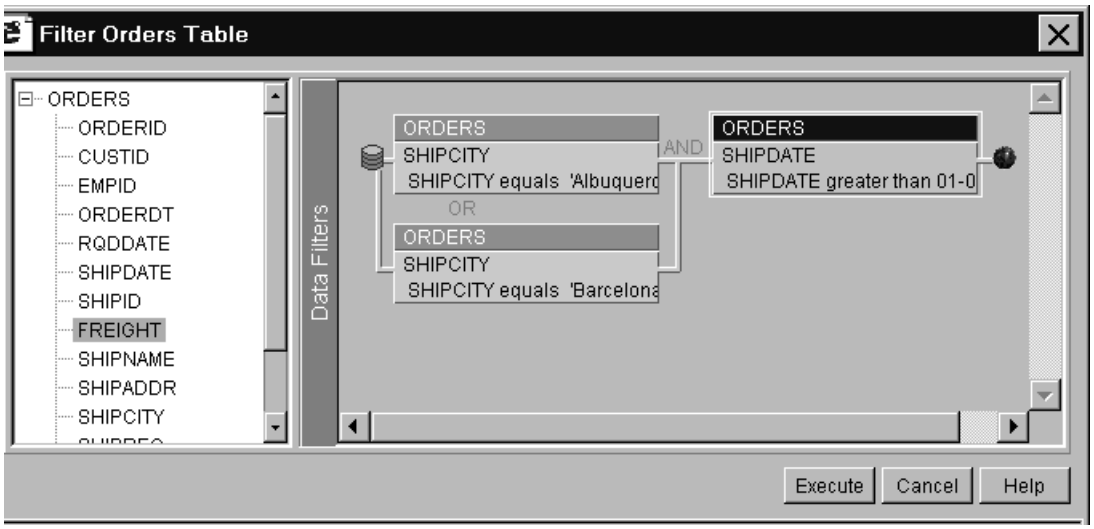

Figure 3

The filter tool included in MetaSpace Explorer is an excellent example of the use of JDBC and is shown in Figure 3. This tool constructs a where clause from an easy to use interface. This interface shows the flexibility of user interface components written in Java.

The filter tool uses JDBC to determine what columns are contained in a table and what distinct values are available in that column. It then creates a where clause which is used in an SQL query. The JDBC driver submits the query and returns the subsetted data.

In order to use the JDBC driver, the programmer must have Java and SQL programming skills.

Comparison of htmSQL and JDBC

Both of these components access the same data servers using SQL statements to access or update data. There are, however, architectural differences.

- htmSQL programs run on the Web server; JDBC programs run on the Web client.
- htmSQL requests are stateless. If there are a number of requests that rely on results from previous requests, the htmSQL programming may get very complicated. In a situation with multiple related requests, JDBC may be a better solution.
- htmSQL is a simpler programming model, the programmer needs HTML and SQL skills. In order to exploit JDBC, Java programming skills are required along with HTML and SQL skills.
- Java programs provide much more flexibility presenting information. Data retrieved through JDBC can be presented in tables, graphs, lists, or any visual component.
- JDBC provides a more complete security model, including full encryption.
- Deployment issues: these are discussed in a later section.

The majority of applications are probably best served by using htmSQL. It is a simpler component and provides excellent HTML presentation and requires nothing on the client except a Web browser (any Web browser).

However, if your application requires a number of requests to the server, a custom user interface, or has strict security requirements, JDBC is the best choice.

Overview of Compute Servers and Web Clients

Many applications require more than access to data. The application must perform computation or other transformations on the data. For applications like this, a compute server is required.

Unlike htmSQL and JDBC which both access the same types of servers, the compute components of SAS/IntrNet each communicate with a different server.

The two SAS/IntrNet components that access compute servers are the Application Dispatcher and the SAS/CONNECT driver for Java.

The Application Server (part of the Application Dispatcher) and SAS/CONNECT server are used as compute servers. The Application Server is accessed by the Application Broker (a CGI program) and the SAS/Connect server is accessed by the SAS/CONNECT driver for Java.

Application Dispatcher

The Application Dispatcher is a set of components that work together to provide compute services to a Web client. These components distribute work among servers and maintain session information. These components are the Application Broker (the CGI program), the Load Manager, and the Application Server. The Application Server is the SAS compute server.

Each time a client sends a request, a new copy of the Application Broker is started. The request is then routed to an Application Server. The Application Server runs a program associated with the requests and returns results back to the client. In SAS Software Version 6, the architecture allowed for a pool of Application Servers. In this version, every request handled by Application Server executed with the access permissions of the Application Server itself.

With SAS Software Version 7, the Load Manager was introduced. The Load Manager maintains information about the state of the Application Servers, whether they are busy handling a request or if they are available for work. The Application Broker requests an Application server from the Load Manager. The Load Manager responds with the a server that is available.

Figure 4 shows the relationships between Web clients and the components of the Application Dispatcher.

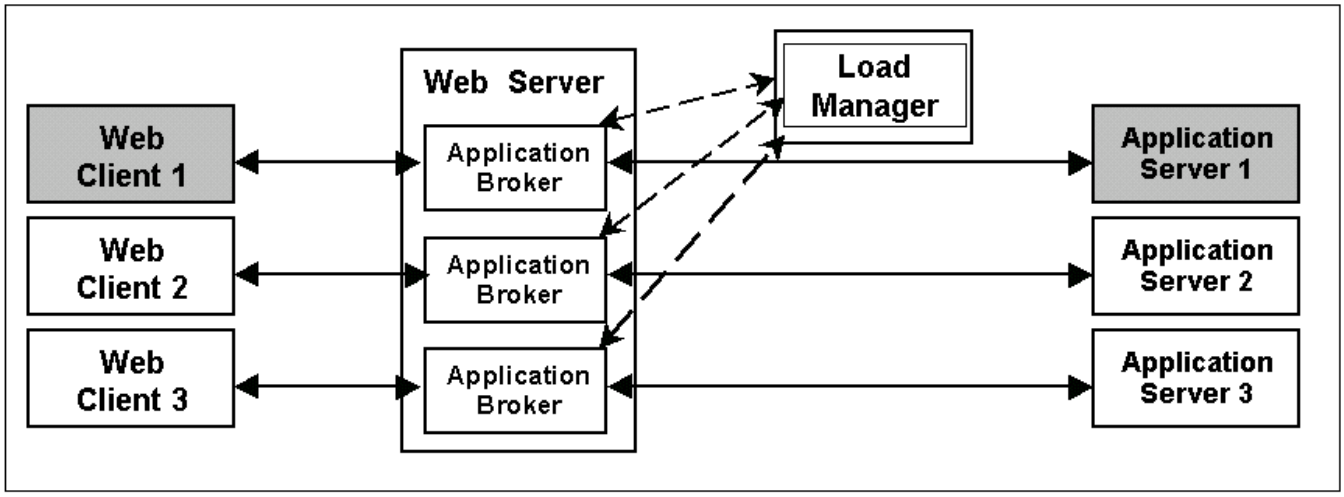

Figure 4

With SAS Software Version 8, two significant enhancements are available. These are the ability to create a security context and the ability to maintain a session.

The Application Server can create a security context similar to the "secured" mode of the SAS/SHARE server. The user is required to provide a user ID and password, which is used to construct a security context that restricts the user's access to only those resources to which they have authorization.

The user ID and password can be encrypted from the Web client to the Web server using secure protocols between these two components. The user ID and password is encrypted by the Application Broker before it sends this information to the Application Server. Other information exchanged between the Application Broker and Application Server is not encrypted.

The Version 8 implementation of the Application Dispatcher also allows a program to create a "session". A session allows multiple requests to be sent to the same server. A session allows the creation and use of a private library. This library is only available to subsequent requests using the same "session" identifier. Macro variables can be saved and used by subsequent requests using the same session identifier.

In figure 4, Web Client 1 is using a "session". All of the requests from Client 1 that utilize the session will be routed to Application Server 1. Only this server maintains the "session" information for this client. Web Client 2 and Web Client 3 are not using sessions. Their requests could be routed to any of the Application Servers.

Both the security context and session enhancements significantly improve the functionality and security of the Application Dispatcher.

In order to use the Application Dispatcher, the programmer must have HTML and SAS language programming skills.

SAS/CONNECT Server

A SAS/CONNECT server provides services to a single client. It allows a client to:

- send requests for remote execution of SAS Statements and procedures
- request data
- create remote SCL objects and make method calls.

A SAS/CONNECT server is started for a particular client, responds to requests from the client, then is shut down when the client is finished using the server.

Before a client can request services, it must first request that a SAS/CONNECT server be started. The server can be started by a telnet daemon or the SAS/CONNECT spawner.

The spawner provides several advantages over a telnet daemon. If the spawner is running in "secure" mode, the user ID and password are encrypted when sent from the client to the server. If a user ID is provided, the SAS/CONNECT server is started using that identity. This means that the server is running with the authorization privileges of that user and can only access resources available to that user.

With Version 8, the spawner offers additional security enhancements. With previous versions of SAS/CONNECT software, the server created a new port and waited for the client to connect to it. The client made two separate connections: one to the spawner or telnet daemon and one to the SAS/CONNECT server.

The version 8 spawner supports "socket inheritance". This means that the original connection between the spawner and the client is transferred to the SAS/CONNECT server. There is only one connection created and this is passed from the spawner to the SAS/CONNECT server.

The SAS/CONNECT server provides a "submit" interface which allows the client to send SAS statements to the server. The server will execute them, returning the SAS log and output to the client.

The server also provides the ability to upload or download files and information between the client and the server.

The server supports SQL access to data. The user can submit SQL queries or update data through this interface.

The client can request the server to create SCL objects for its use. The client can then make method calls on these objects, and destroy the objects when they are no longer needed.

SAS/CONNECT Driver for Java

The SAS/CONNECT driver for Java is a SAS/CONNECT client written entirely in Java. It provides support for much of the functionality provided by the SAS/CONNECT server.

In order to use the SAS/CONNECT driver, the programmer must have Java and either SAS language, SCL, or SQL programming skills.

The SAS/CONNECT driver for Java provides the same security features that are available to a SAS client. An upcoming release of the driver will support encryption of all communication between the driver and the server. This support is dependent upon government approval for distribution.

The driver provides classes that start the remote SAS/CONNECT server and connect to it. The SAS/CONNECT server has access to only those system resources that are available to the user identity that was used to create the server.

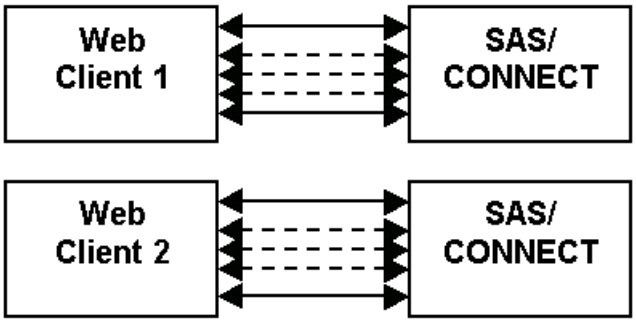

Figure 5

Figure 5 shows several Java clients and the relationship to the SAS/CONNECT servers.

Once the driver has connected to the server, the Java program can start submitting statements to the server and retrieving results. The results of these statements can be output (text), datasets or a file. All of these can be retrieved and displayed by the Java client.

Unlike the JDBC driver, the SAS/CONNECT driver is a proprietary API. The class ConnectClient provides the following methods:

- rsubmit()
- getListLines()
- getLogLines()
- getSharenet()
- getDownloadData()

After a set of SAS statements have been submitted to the SAS/CONNECT server by using the rsubmit method, the SAS log and output lines are returned to the client. The methods getListLines() and getLogLines() return the information sent from the server. Because the log and output are returned as ASCII text, they are be easily displayed in a text window.

A new data set can be created as a result of the statements executed on the SAS/CONNECT server. This data set can be accessed using the JDBC Connection object. The JDBC Connection object is retrieved using the getShareNet() method. This returns an instance of the JDBC Connection object and provides full JDBC support.

Graphics produced by the SAS/CONNECT server can be presented in a Java program by using graphics drivers that produce either GIF images or Java graphics. There are Web Publishing tools that produce either of these types of image.

The Web Publishing tools can be used to create a file on the server that contains the graphics. If the image is a GIF, the image can be downloaded to the client by submitting PROC DOWNLOAD. The downloaded file can be accessed through the getDownloadData() method and the GIF image can be displayed as a Java Image object.

The Graph Applet Generator (one of the Web Publishing tools) produces an HTML page which contains a reference to a graph applet. The HTML page can be displayed in the Web browser.

A new product, AppDevStudio™, contains a Java programming environment called webAF™. This product allows the programmer to create Java classes that access remote SCL objects. Please check the SAS Institute Web site for more information about this product.

Comparison of Application Dispatcher and SAS/CONNECT driver for Java

These two technologies access different servers which have different characteristics.

The Application Dispatcher utilizes a pool of servers. User requests are distributed among this pool. If the servers are very busy, the user will see longer response times.

The Java client has an individual server dedicated to it. The server can only be busy handling a request from one client.

The SAS/CONNECT Java client can provide full security services: the server will run only with the user's permissions and all information exchanged between the client and the server is encrypted.

The Application Server can provide a security context. However, the Application Server can only access data that both the Application Server and the user have permission to. If the user has permission to resources that are not available to the Application Server, these resources cannot be accessed. Also, communication between the Application Broker and Application Server is not encrypted, except for user ID's and passwords.

Just as htmSQL provided capabilities that meet the requirements of most applications compared to the Java implementation, the same can be said for the Application Dispatcher compared to a Java applet using the SAS/CONNECT driver for Java.

There are situations where a Java applet is more suitable. The programmer may not want to grant the Application Server access to all information that the user's have access to. The Java client will have exactly the user's permissions applied to it; an Application Dispatcher program may not.

The application may require a sophisticated user interface, or maintain state with multiple servers simultaneously. Again, a Java client would be the best choice. MetaSpace Explorer™ is an excellent example of an applet that exploits both the flexibility of Java components and the ability to access multiple servers.

Other Issues: Internet or Intranet Delivery

Are you going to provide your application on the Internet or is it used in an intranet? What software do you clients have?

Java technology is still fairly immature. Java 1.1 had its first production release in early 1997. At the time this paper was written (1/99), the current version is 1.1.7. Because of the rapid change in Java technology, applets that exploit new features of Java are difficult to deploy to Internet users. Internet users may have fairly old copies of browsers (in Internet terms, that would be software that is more than one year old). The Java Plug-in from Sun Microsystems, Inc., is a downloadable JVM that provides current Java support. There are both 1.1.7 and 1.2 versions of the Plug-in available. The plug-in can be downloaded and installed on a user's machine, but Internet users connected via a modem might be very irritated by this requirement and not install the software.

In the fourth quarter of 1998, Netscape and Microsoft both released a production release of the JVM that supported 1.1 technology. Neither of these vendors has announced plans of when they will provide a Java 1.2 release.

This does not mean that all Java applets are unsuitable for Internet use; only those that rely on new features or fixes may be problematic. An example of an applet that has been deployed very successfully is the Graph Applet. This applet produces high-quality graphics. Because it utilizes only stable features of Java, it can run in most (if not all) releases of Java 1.1, including previous versions shipped with Netscape and Microsoft browsers.

For Internet deployment, there are firewall issues as well. Both the SAS/SHARE driver for JDBC and SAS/CONNECT driver for Java usually use socket connections to communicate with SAS servers. Many Internet users are accessing the Web from behind a firewall and cannot create socket connections to machines outside of their firewall. The tunnel feature helps alleviate this problem. Rather than using a direct socket connection, the drivers communicate with a CGI program using HTTP protocol. The CGI program routes the request to another process that uses a socket to communicate with the server. While the tunnel feature helps with firewall issues, each transaction takes longer, because it is routed through several processes.

The CGI components of SAS/IntrNet use HTTP protocol to access the Web server. Most firewalls allow HTTP protocol through to the Internet. The CGI components usually return HTML to the client. The CGI programs require nothing more than a Web browser on the client and are more likely to provide consistent and more stable support for Internet clients.

New Servers

In Version 8 there is a new product introduced called SAS/INTEGRATION TECHNOLOGIES. This product provides access to SAS software through DCOM, CORBA and other protocols. Included with this product are new Java components that access this server.

There is a new JDBC 2.0 driver that exploits advanced functionality provided by the server. This driver provides result sets that are formatted using SAS or user-defined formats, scrollable result sets, and record-level locking.

Scrollable result sets mean that a program can request accesses to a table and request a specific row in the table. For example, a program could request rows 1 through 10, then rows 31 to 40, and then back to 21 to 20. The program can navigate to any row in the table.

In contrast, the SAS/SHARE driver provides forward-only scrolling. This means the program can read the tables from the first row to the last row but cannot navigate back to a previous row.

This new driver also provides record level locking. This means that when a row is read, it can be locked. This prevents other users from updating the row. Other users may read the row, but not update it.

This server provides full encryption for all communication. The Java components will provide encryption pending government approval for distribution.

Summary

This paper provided an overview of the components of SAS/IntrNet and the SAS servers. It compared the CGI components, Java components and the corresponding servers. It described some of the problems with deploying applications over the Internet.

Hopefully, the reader will be able to use this information to select the most suitable technology for their application.

References

http://java.sun.com/products/jdbc/ - Use this URL to review JDBC information.

http://www.sas.com/rnd/web – This is the home page for SAS/IntrNet software.

Authors

Barbara Walters
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 677-8000 x6668
sasbbw@sas.com

Don Chapman
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 677-8000 x6707
sasdnc@sas.com

Acknowledgements

The authors would like to thank the following reviewers:

- Renee Harper