Paper 217-30

# Configuring J2EE Application Servers for Use with the SAS®9 BI Platform

John Roth, SAS Institute Inc., Cary, NC

## ABSTRACT

Using planning files, the SAS Software Navigator, and the SAS Configuration Wizard, you are guided through a series of steps that enable you to quickly install and configure the SAS®9 Business Intelligence Platform. This process includes installing and configuring third-party J2EE application servers and deploying SAS Web applications. While this process gets SAS software up-and-running quickly, there are many additional configuration considerations needed in order to provide a robust, scalable, and secure middle-tier environment. There are many best practices recommended in vendors' product documentation, in books, and from online resources, but how do you know which recommendation is best for configuring application servers for use with the SAS BI Platform? This presentation explores some steps that you will probably want to take to be able to provide a robust, scalable, and secure middle-tier environment. I will discuss specific issues for Apache Jakarta Tomcat, BEA WebLogic Server, and IBM WebSphere Application Server.

## INTRODUCTION

The first time you install the SAS®9 BI Platform will probably be in a test or development environment. Your initial objectives will be to get the software up-and-running, integrate your company's data, integrate your SAS procedures, and create custom reports for your company. If you are working in an isolated test lab, then the need for security, robustness, and scalability might not be critical to achieving your initial objectives, In fact, turning on tight security early in the setup process might impede rapid development. However, before you put your application into production, you will need to take some additional steps. This paper discusses the general areas of software management that you need to understand in order to provide a robust, scalable, and secure middle tier, including recommendations for identifying production system requirements, designing the infrastructure topology, performance tuning, enhancing security, and configuring appropriate software maintenance levels.

## KNOW YOUR REQUIREMENTS FOR SECURITY, SCALABILITY, AVAILABILITY, AND MAINTAINABILITY

Before you design an appropriate network topology for your application, you should have a clear understanding of your requirements for security, scalability, availability, and maintainability. These requirements drive many of your topology design decisions. In some cases, you might have to make trade-offs where conflicting requirements indicate different directions. Having a clear sense of priority about potentially competing objectives will facilitate your decision-making process. The following subsections identify questions about your application that you should consider as you gather your requirements.

### SECURITY

How will clients physically access your application?

- Will they need external access through the Internet, that is, from outside your company's firewall?
- Will access come from workstations and other client devices that reside on your internal intranet?
- How does your company use firewalls to isolate internal network components?

Do you have any authorization requirements?

- Do you need to limit access to certain members of your organization?
- Can anyone in your organization look at the data and the reports that are generated?
- What is the business cost if your data gets into the wrong hands?
- Do you have more than one type of user?

What threats are of most concern to you?

- External unauthorized entry?
- Internal unauthorized access to sensitive data?
- Packet sniffing to gain unauthorized information?
- Denial of service attacks?
- Modification or falsification of sensitive data?

Tight security can be expensive to maintain and might restrict the scalability of the system.

#### SCALABILITY

How scalable must your application be?

- How many users do you expect your application to handle?
- What kind of response time is acceptable?
- Characterize access: Will there be a steady stream of requests or will you have a burst of traffic at certain points in the business cycle? For example, a financial application requires monthly activity that produces unusually high activity on a specific day of the month.
- If your system experiences an uneven load, can you tolerate an occasional slow response?
- What is the business cost associated with delays?

How quickly must your production application environment adapt to change?

- Do you anticipate growth in use over time?
- How quickly must you respond to higher demands if they arise?

Highly scalable systems can be more complex and expensive to implement, but if they are well-designed, they can be more easily maintained as load increases.

#### AVAILABILITY

How much downtime can your application tolerate?

- Will you have planned periodic outages or will the system be expected to run 24/7?
- How fast must the system be brought back up after an unplanned outage?
- What is the business cost associated with unplanned failures?

Highly available systems can be expensive to implement and maintain. Therefore, only a high business cost associated with downtime would justify the huge commitment of resources necessary to provide high availability.

#### MAINTAINABILITY

Once placed into a production environment, any server requires continued monitoring and on-going maintenance. How maintainable is your configuration?

- Who will maintain your system?
- Will the maintenance staff consist of one person or a team of people?
- What level of monitoring will be required?
- Will automated processes be required for deploying and configuring components of the system?
- How will new components and updates be introduced?
- Will there be a test environment for staging modifications and updates?

As security, scalability, and availability requirements increase, maintainability becomes more complicated and demands more attention from those who will maintain your application.

## KNOW YOUR BUDGET AND INFRASTRUCTURE CONSTRAINTS

You've identified all the requirements for your application. However, there will probably be additional requirements and constraints imposed on your application that are based on corporate policies and budgetary limitations. Here are some questions you should ask as you complete your requirements gathering process.

#### BUDGETARY CONSTRAINTS

Do you have a budget that will allow you to meet your application's requirements?

- Are sufficient resources allocated or budgeted in accordance with the anticipated system complexity?
- For applications that are only used occasionally or that generate light traffic, can you use existing hardware?
- Do you have sufficient software, hardware, and human resources allocated to support the testing and production systems?
- What happens if you do not have a sufficient budget to achieve your applications' production goals?
- How long will it take to procure any new hardware or software?

You might have to make some configuration trade-offs based on resource limitations.

#### INFRASTRUCTURE CONSTRAINTS

Whether you are building a small department-level application or a large enterprise-wide solution, your application must conform to your corporate standards.

- What are the security standards at your company? Even the smallest projects must conform to the security standards. Even a small security breach can make the entire corporate infrastructure vulnerable to attack.
- What other corporate standards must you support?
- How will you integrate the supporting servers into your corporate network topology?

#### THE BOTTOM LINE

Designing the topology will require that you balance priorities between the budget and management costs versus security, scalability, and availability. Once you have a good understanding of these requirements and constraints and have established a clear set of priorities, you are ready to begin designing your topology.

## DESIGNING YOUR SYSTEM TOPOLOGY—ONE SIZE DOES NOT FIT ALL

This section discusses potential topologies and explains how they can help you meet your requirements for security, scalability, availability, manageability, and budget. The topologies are introduced from simplest to most complex. All the configurations that are presented in this paper show the same SAS Foundation Server tier, which consists of a SAS Metadata Server that resides on a dedicated machine along with an unspecified number of additional compute servers that run various SAS Foundation Servers including SAS Workspace Servers, SAS Stored Process Servers, and SAS OLAP Servers. There are many issues that must be considered when designing the Foundation Server tier; however that discussion is beyond the scope of this paper.

#### TOPOLOGY 1 - A BASIC SETUP

Let us start with a relatively basic configuration of the SAS Enterprise BI Server that can be installed, deployed, and configured using the SAS Software Navigator and the SAS Configuration Wizard. This scenario is depicted in Figure 1.

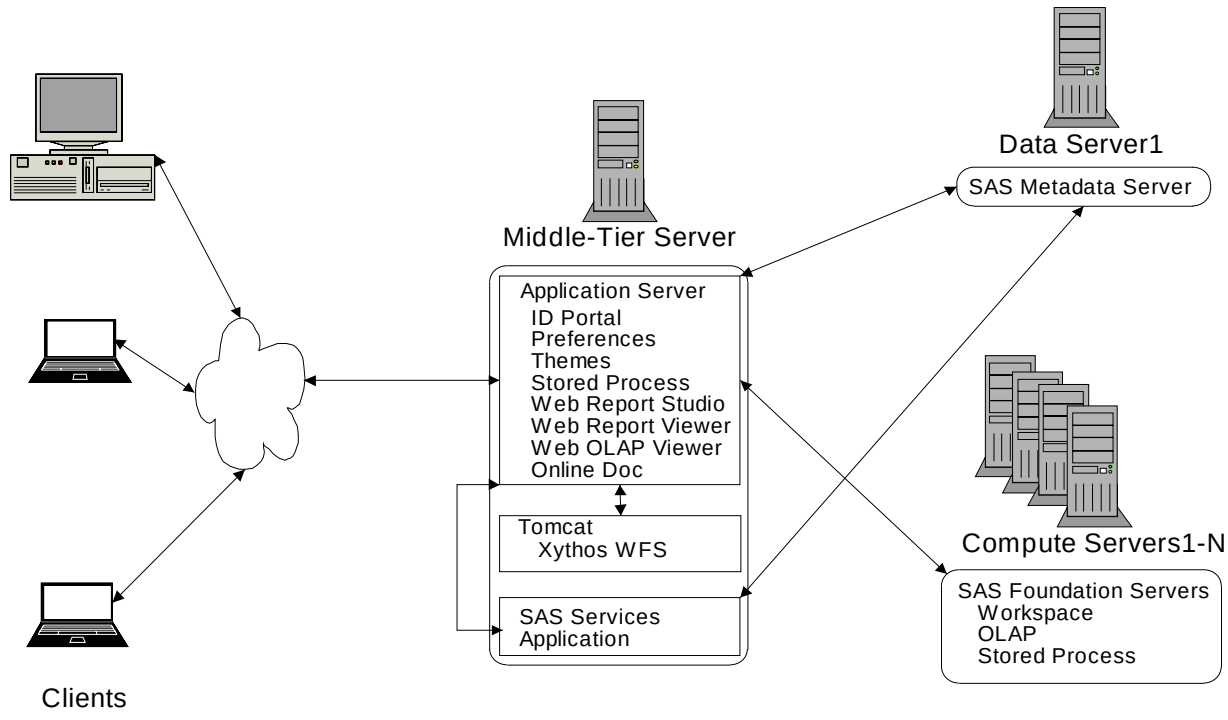


Figure 1. Topology 1 - A Basic Topology

If you start with an appropriately designed SAS planning file, then little additional configuration or tuning is required. All the SAS middle-tier components are installed on a single server. These components include the following:

- SAS Web Applications that run in a Tomcat, WebLogic, or WebSphere servlet container:
  - Portal Web Application
  - SAS Preferences Web Application
  - SAS Themes Web Application
  - SAS Stored Process Web Application
  - SAS Web Report Studio
  - SAS Web Report Viewer
  - SAS Web OLAP Viewer
  - SAS Documentation Web Application
- Xythos WebFile Server (WFS) WebDAV content services, which runs in a separate Tomcat container
- The SAS Services Application, which runs in a separate Java Virtual Machine (JVM) process.

The Xythos WFS is a WebDAV server that is configured, by default, in its own separate Tomcat servlet container. It is generally recommended that Xythos WFS run under Tomcat, but it can alternatively be configured to run under WebLogic or WebSphere if desired. Xythos WFS also requires a database, but this detail is not shown in these topology diagrams. Xythos WFS can be configured with PostgreSQL, IBM DB2, Oracle, or Microsoft SQL Server.

The key advantage to this basic topology is that it is easy to set up and configure. It is ideal for an initial development or test environment in which frequent modifications are likely, and less rigorous system administration procedures will be mandated. The down side to using this topology is that it provides less security and scalability than other topologies.

#### TOPOLOGY 2 – PLACING THEMES STATIC CONTENT IN A WEB SERVER

A frequently recommended industry best practice is to deliver static content from a Web (HTTP) server rather than from an application server. While all supported application servers provide special servlets that deliver static content, they are generally less efficient than dedicated HTTP servers such as Apache HTTP Server, IBM HTTP Server (IHS), and Microsoft Internet Information Services (IIS), which are designed specifically for this purpose.

The usual recommendation is to place a Web server in front of the application server as a proxy. In this configuration, the static content is placed in the Web server's file space. A module or plug-in is provided by the application server vendor that runs as part of the Web server process. The module is configured to pass along all requests for dynamic content (JSPs and servlets) to the application server. In this configuration, the client sends all requests to and receives all replies from the front-end HTTP server and has no direct exposure to the application server.

Figure 2 illustrates the flow of requests through this type of configuration. In order for this configuration to work correctly, the static content must be extracted from the Web application's WAR file, placed on the HTTP server, and the application server plug-in must be correctly configured to pass requests for dynamic content back to the application server. SAS does not currently provide tools, scripts, or documentation on how to separate static content, nor has this configuration been fully tested. However, someone who is experienced with Web applications might be able to pull apart the WAR file and set up this configuration. Topology 3 illustrates a similar but more maintainable configuration that can achieve comparable performance objectives.

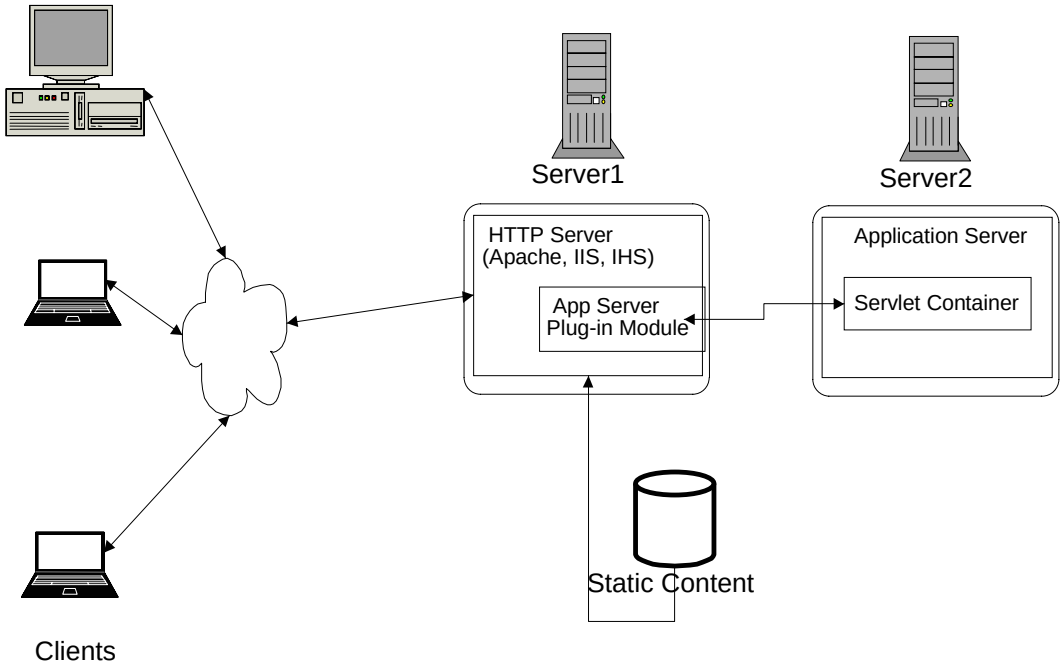


Figure 2. Using an HTTP Server as a Proxy

There is an alternative recommendation for splitting out a portion of SAS static content and placing it under the auspices of a Web server. The SAS Themes Web Application consists entirely of static files, including images, cascading style sheets, and HTML fragments. The *SAS Web Infrastructure Kit 1.0 Administration Guide* provides instructions for deploying SAS Default Themes and your custom themes into a Web server. This configuration is a bit different from what is shown in Figure 2. The HTTP server is not acting as a proxy—client requests are directed to the application server but the HTML that is dynamically generated and delivered to the client has URIs for images and cascading style sheets that point to the Themes HTTP Server. The client is directly connecting to both the application server and the Web server. Figure 3 illustrates Topology 2, which is a configuration in which the HTTP server that delivers Themes content is running on a separate server machine. Although the HTTP server can run on the same physical machine as the application server, this might result in undesirable competition for system resources.

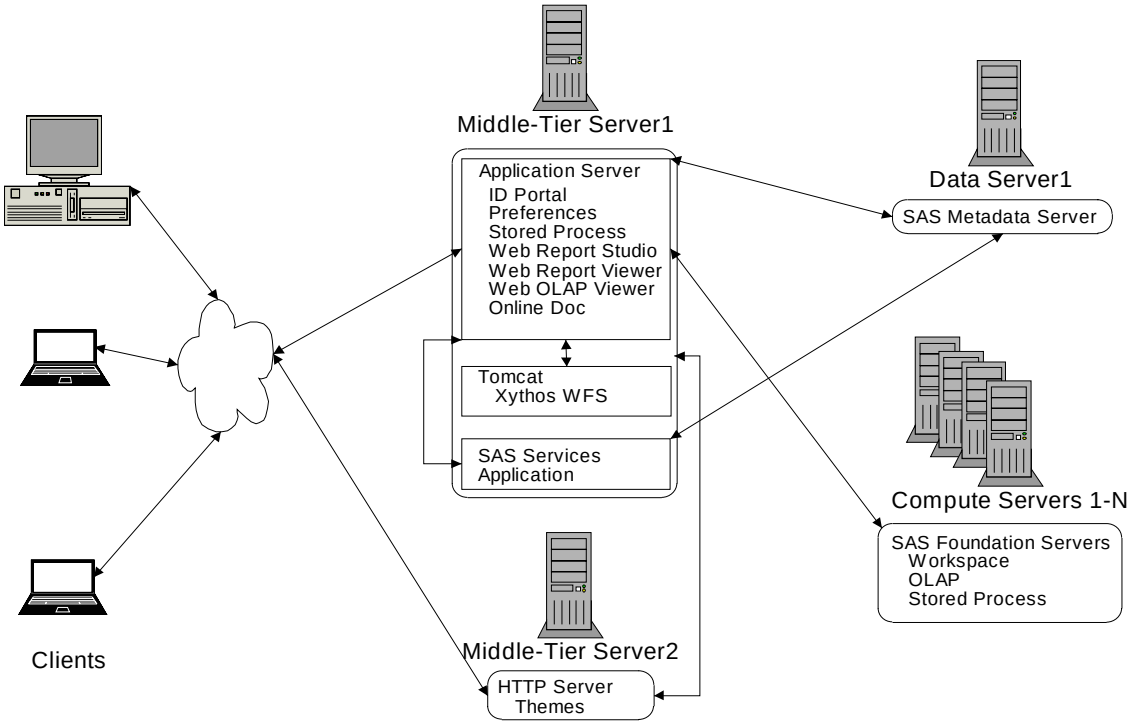


Figure 3. Topology 2 – Placing SAS Themes on an HTTP Server

The primary advantage to Topology 2 over the basic configuration (Topology 1) is the potential for increased performance. However, here are a few disadvantages.

- The configuration is a little more complicated to set up and, consequently, will incur higher maintenance costs.
- An additional server is introduced, which increases hardware and software costs.
- Two servers must now be exposed to the client. This can be perceived as a security issue. (This issue is discussed later in the paper.)

There is also a performance advantage to Topology 2 over a configuration of the HTTP server acting as a proxy. In the configuration shown in Figure 2, all client requests and server responses for dynamic content must now go through an additional network hop and an additional layer of processing when they are routed through the HTTP proxy.

#### TOPOLOGY 3 – CACHE STATIC CONTENT ON A PROXY SERVER

In Topology 2, only the Themes static content is placed on the HTTP server. With this configuration, there is still a significant amount of static content in the remaining SAS WAR files that must be delivered by the application server. By using caching proxy servers, it is possible to offload the majority of static content from the application server without having to split up the WAR files and explicitly deploy content on each server. With a caching proxy, the first request for a static resource is passed from the proxy to the application server. The application server sends the response back through the proxy, which keeps a copy of the resource in its cache. On subsequent client requests, the proxy finds the resource in its cache and delivers the content directly, circumventing the application server. Only a few initial requests are received by the application server.

There are several types of servers that can provide caching. An HTTP server that acts as a proxy can cache the content that it receives from back-end servers while it delivers its own managed content. Apache HTTP Server can handle caching through the configuration of mod_proxy and mod_cache. Notice that mod_cache is currently an experimental feature. There are also specialized applications such as the open source Squid Web Proxy Cache and IBM's Edge Component Caching Proxy, which is bundled with WebSphere Network Deployment (ND). These products are dedicated to the proxy function and are not designed to be a direct source of HTTP content. They are

tuned specifically to broker requests on behalf of back-end servers. IBM's WebSphere Plug-in can also cache static content and some dynamic content. IBM recommends using the WebSphere Plug-in over other caching mechanisms.

Figure 4 illustrates Topology 3, which is a SAS BI Server configuration with cached static content being served from a caching HTTP proxy server. In this configuration, the proxy server handles all client requests and hides the application server addresses from the client. For this configuration, the Themes have been placed on the HTTP server along with the cached static content. As with the previous configuration, the SAS metadata for the Themes should be updated so that the generated URLs for Themes content point to the proxy server and not to the application server.

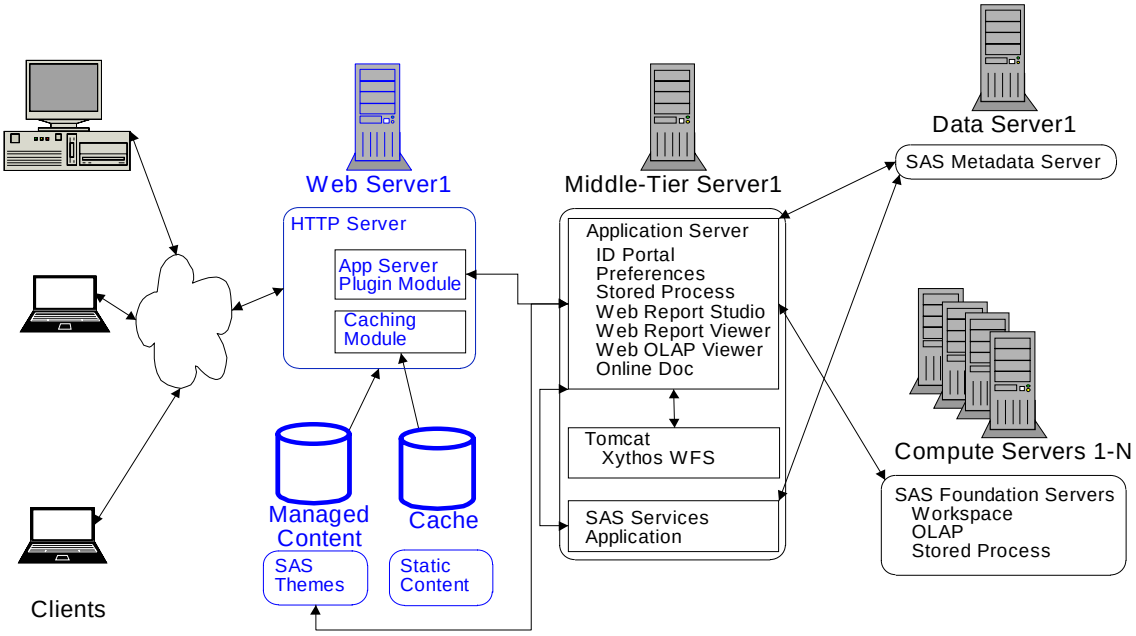


Figure 4. Topology 3 – Using a Caching Proxy Server

There are several advantages to using Topology 3 in addition to the previously mentioned ability to offload some of the workload from the application server.

- Applications can easily be moved to other machines without having to modify the SAS metadata or inform the clients of any change in location.
- This configuration offers potentially better security because it does not directly expose the application server to the client. It can work well with a demilitarized zone (DMZ) as we will see in the next section.
- This configuration can easily be scaled to support clustering if the workload increases.

There are also some disadvantages to using Topology 3.

- Typically the HTTP server will reside on a separate machine, which adds to overall cost.
- Maintenance costs increase, because there is yet another server to manage and configure.
- It is important to make sure that cached content is cleared if any changes are made to the original copies on the back-end server.
- As with any proxy model, there are multiple network hops and multiple servers handling a single request and response for dynamic content.

**TOPOLOGY 4 – USE YOUR CORPORATE DEMILITARIZED ZONE (DMZ)**

Many corporations will use a series of firewalls to create a DMZ between the servers that form the first line of contact with client applications that reside on the Internet and the corporate computing infrastructure. The outer firewall that connects to the public network is called the Domain Firewall. Only a few ports are typically allowed to be opened through this firewall. Often, only HTTP and HTTPS connections using the standard ports 80 and 443, respectively, can be opened. Servers that reside directly behind this firewall are exposed to a wide range of clients through these limited ports and are not fully trusted. An additional firewall, called the Protocol Firewall sits between these machines and the secure server network. The protocol firewall might allow a few additional ports to be open, but the range of machines that are allowed to make connections is typically restricted to the servers that reside in the DMZ. Important business logic and data should not be placed on servers that reside in the DMZ. Usually, HTTP servers, proxy servers, and load balancers (but not application servers) reside in this space. If your application must be accessed by clients through the Internet, you will probably have to consider using a DMZ. Often, user workstations and other client devices on a corporate intranet will have to connect to secure servers through an internal DMZ. Before implementing any topology work, meet with your internal security experts to determine what will be required for your application.

Topologies 1 and 2 enable clients to talk directly to the application server; these topologies cannot work with a DMZ. Because it uses a proxy server as an intermediary, Topology 3 can easily work with a corporate DMZ. Figure 5 shows Topology 4, which is a modified form of Topology 3 with the HTTP proxy server residing in a DMZ.

The primary advantage of Topology 4 is the additional security that it provides over the previous topologies. If your application must reside in a secure corporate computing environment or must communicate with external clients over the Internet, Topology 4 might be the simplest configuration that you can realistically consider. If your clients and your middle-tier application can live in a common internal network domain and are not isolated by a DMZ, you might be able to consider a more basic topology such as Topologies 1 and 2.

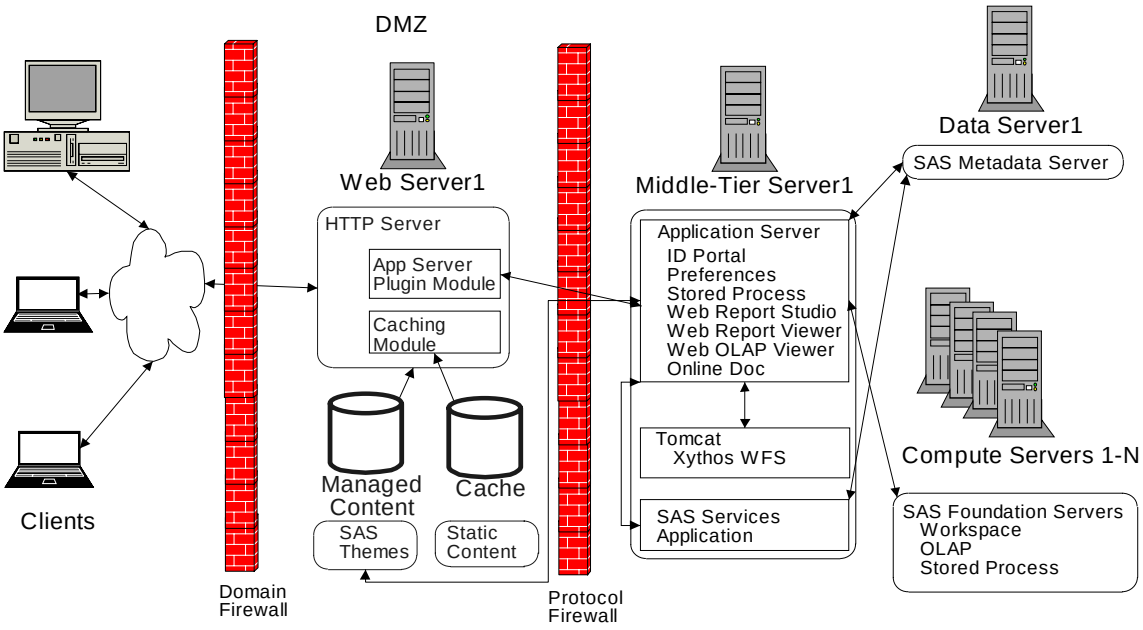


Figure 5. Topology 4 – Integrating with a DMZ

**TOPOLOGY 5 – USE AN APPLICATION SERVER CLUSTER**

In order to provide greater scalability, availability, and robustness, most application servers support some form of clustering. Multiple application server instances can be clustered to handle client requests for a set of deployed applications. Workload distribution is usually managed by the application server plug-in module that runs in the HTTP proxy server.

Individual application server instances can co-exist on the same machine or can be spread across a group of server machines. Clustering multiple application server instances on the same machine is called *vertical clustering.*

Spreading the application server load across multiple machines is called *horizontal clustering*. There are advantages to distributing a set of applications with either clustering strategy. If you have a fairly large machine with four or more CPUs and a large multi-gigabyte address space, it might be more cost-effective to use vertical clustering rather than a single application server instance. Notice that the latest available versions of WebLogic and WebSphere only support 32-bit JVMs. Even where 64-bit JVMs can be used, garbage collection becomes less efficient when the heap-size approaches 1.5 gigabytes. Thus to optimally use a large machine, multiple JVMs might be the best configuration. With horizontal clustering, the workload can be distributed evenly across a number of available servers that each house one or more application server instances. Availability is increased because, if one server crashes, the other servers in the cluster can still handle new requests. If you are budgeting for new hardware, you might find that it's cheaper to design a horizontal cluster that uses multiple, less expensive machines rather than a vertical cluster that uses fewer large servers.

SAS Web applications can be deployed into a clustered environment as long as all requests that are associated with a single session remain with the same server. IBM refers to this strategy as *session affinity*; BEA documentation refers to it as *server affinity* and calls these sessions *sticky sessions*. This is the default behavior for both WebSphere and WebLogic. Note that both of these application servers also provide the ability to migrate HTTP sessions from one server to another in order to support fine-grained load balancing at the individual request level. By allowing session migration, application servers allow sessions to continue in spite of individual server failure. In order for application servers to provide session migration, all objects that are stored in an HTTP session must be serializable and lightweight. This model works well for transaction-oriented applications such as online shopping carts where session recovery is critical and session content is fairly lightweight. Business intelligence sessions tend to contain large data elements such as results sets from ad hoc queries, reporting, and analytical tasks and typically do not involve transactional operations. SAS Web applications store some non-serializable objects in HTTP sessions. By configuring an application server cluster to use session affinity, SAS applications can live in and benefit from a clustered environment. New SAS sessions can be distributed across available servers by using round-robin or weighted scheduling algorithms. Figure 6 shows a configuration with SAS Web applications deployed across a horizontal cluster of machines. In this configuration, the SAS Services Application is not distributed. Xythos WFS also remains undistributed in a Tomcat container, however, because it is a Web application, it can also be distributed with this topology.

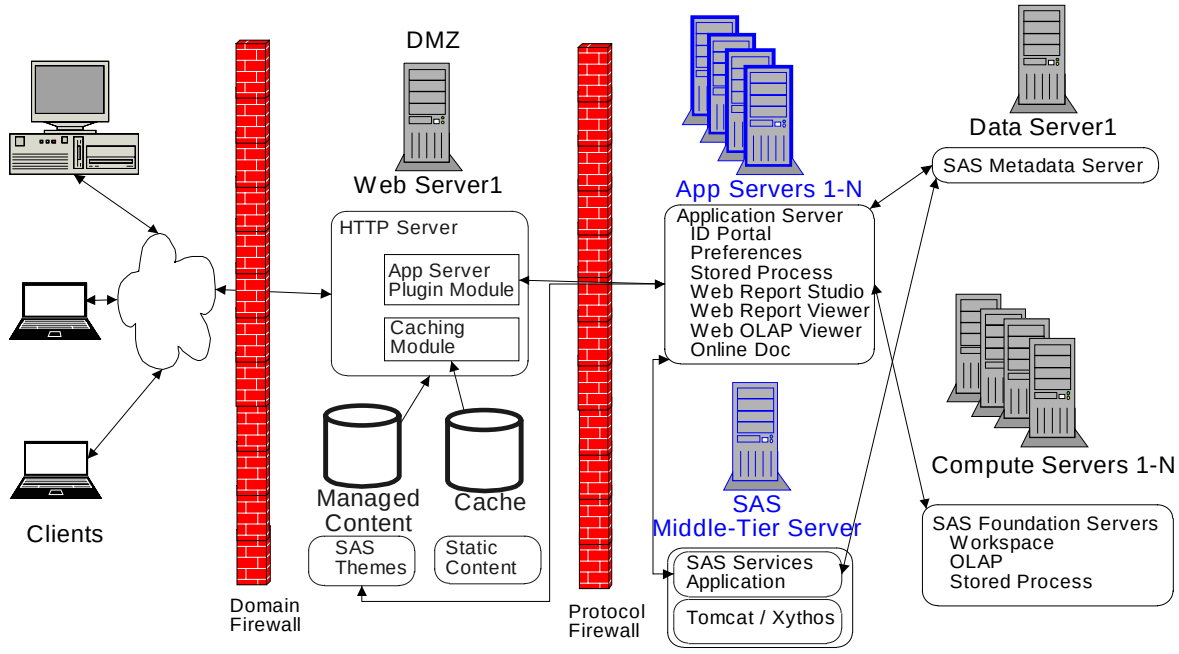


Figure 6. Topology 5 – Clustered Application Servers

At the time of this writing, the *SAS Web Infrastructure Kit 1.0 Administration Guide* describes a different approach to load distribution. Several "Best Practices" scenarios are presented. Instead of clustering, it is recommended that you move individual Web applications to their own servers. These scenarios are recommended to reduce the memory usage that can occur when all Web applications must run in a common application server. A clustering strategy has the following advantages over the "static" distribution approach that is recommended by the guide.

- Deployment is easier to manage with clustering. All machines and server instances are identically configured. Application servers provide deployment management services that make management of a cluster easier.
- A cluster is more scalable. It is easy to add additional nodes to a cluster and further distribute the load. The static approach is limited after each application is placed on its own server.

Clustering can significantly increase your costs and maintenance overhead relative to the previous topologies. You should only add application server clustering if your performance or availability needs require it. If you are using WebSphere 5.1, you will need the Network Deployment (ND) bundle rather than the basic product. WebLogic Server 8.1 supports clustering. The version of Tomcat shipped with SAS, 4.1.18, does not fully support a clustered environment. While later versions of Tomcat do support clustering, they have not been tested by SAS. If you are planning to use Tomcat but need clustering support, you might want to consider using WebSphere ND or WebLogic instead.

#### TOPOLOGY 6 – USE HARDWARE/SOFTWARE LOAD BALANCERS

With Topology 5, the application servers are clustered to balance the load and increase availability. While this provides redundancy for the application servers, the Web servers and other back-end components remain as potential bottlenecks and single points of failure. HTTP traffic can also be distributed across multiple Web servers by placing load balancers in front of them. Load balancing can be performed by software servers such as IBM's Edge Components Load Balancer or by specialized hardware such as Cisco CSS Controller, Nortel Alteon Controller, or F5 Networks BIG-IP Controller. In Topology 6 (Figure 7), a single load balancer handles all client requests and distributes HTTP requests across a cluster of HTTP servers.

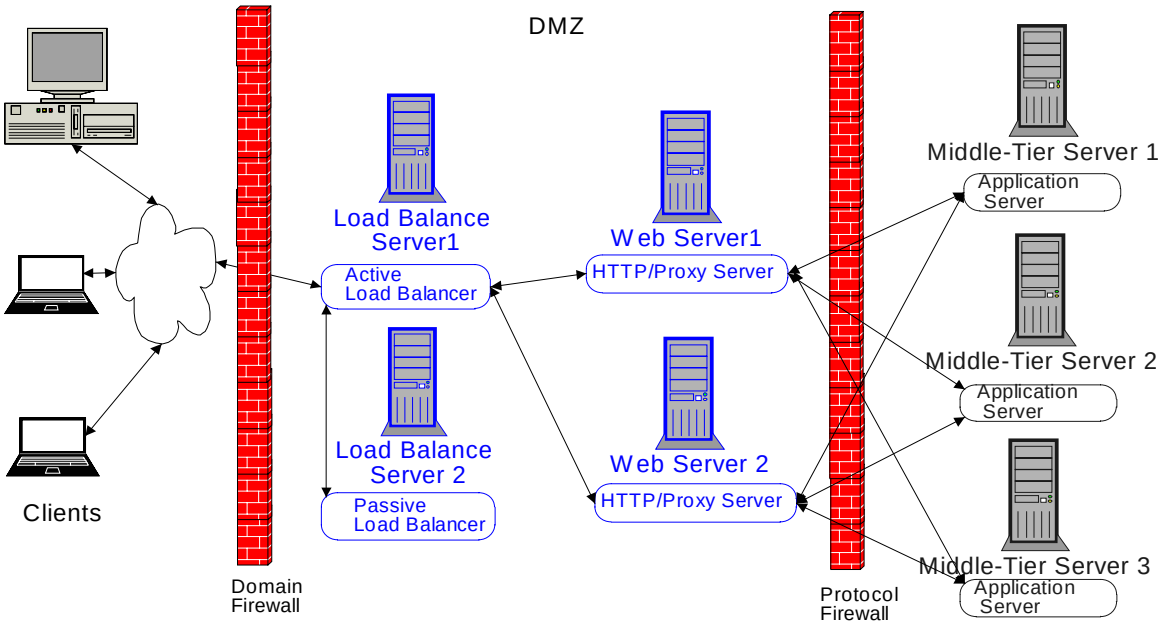


Figure 7. Topology 6 – Load Balancing and a Fully Redundant Web Tier

In order to maintain high availability, a back-up "passive" load balancer is often used to monitor the health of the primary "active" server. If the secondary server detects a failure, it will take over as the primary server. IBM touts this configuration as the "golden standard" topology—achieving very high availability. Keep in mind that, even though all the Web-tier components are now redundant, there are still single points of failure in the back tier; particularly with the SAS Services application and the SAS Metadata Server. If you have such a configuration already in place, SAS applications could benefit from the higher availability. If you find that your HTTP servers are a bottleneck, it might be useful to load balance your Web traffic. You might also find that your corporate infrastructure already provides a similar infrastructure into which your SAS applications must be configured. Remember that the SAS Themes metadata must always point to the address that is exposed to the clients.

Six topologies have been presented from the simplest to the most complex. If you know your corporate standards and your application's requirements, you should be able to use one of the topologies to balance all your objectives. After you have selected the topology that you are going to use, there are still additional security and tuning issues to consider.

## SECURITY

In the previous section, one security consideration was discussed, that is, firewalls. Here are additional security considerations that you should evaluate.

- Do you need to secure your communications links from unauthorized viewing and potential alteration? If so, which links need to be protected?
- Do you need to authenticate your users and provide restricted access? If so, where should authentication occur?
- What additional steps must be taken to lock down the system before it is placed in production?

### SECURING LINKS WITH SSL

If you are moving sensitive information across the Internet, you will probably want to use HTTPS and Secure Sockets Layer (SSL) to encrypt your communications links. SSL uses Public Key Cryptography, based on a public/private key pair. Each of your servers acts as an end-point for an encrypted communications link and will have to manage certificates that contain the securely hidden private key and certificates that contain the public key. A description of how Public Key Cryptography and SSL work is beyond the scope of this paper. However, there are many good sources for that information, some of which are listed in the "Resources" section at the end of this paper. Here are some of the factors you might want to consider when deciding where and how to use these tools.

- Which links do you want to encrypt? If you look at any topology diagram in the previous section, you see that each arrow represents a potential communications link that could be encrypted. You probably want to encrypt any data that might move across the public Internet. If connections to your site use VPN, they are already encrypted. Otherwise, traffic to-and-from your site is open to packet sniffing by just about anyone. Credit card numbers, social security numbers, and any other sensitive information should always be encrypted on the Internet. At a minimum, traffic between the client and your proxy server endpoint in the corporate DMZ should be encrypted. If you want a higher-level of security, you can encrypt links all the way to the application server, keeping all data that's transferred in the DMZ confidential. If you are concerned about internal packet sniffing you could encrypt everything, but doing this is costly, as you can see from the remaining considerations in this section.
- Some load balancing schemes might rely on packet content for routing. SSL might introduce some restrictions on what can be accomplished with a load balancer.
- Cryptography is computationally intensive. It will reduce the amount of traffic that your servers can handle because more CPU time will be dedicated to encryption and decryption.
- Certificates expire. If you want a highly available system, you will want to prepare for certificate renewal in advance to avoid unexpected downtime.
- You will have to decide whether you need certificates that are generated by a Certification Authority (CA) or if self-signed certificates are adequate for your application. If you use self-signed certificates, you can save some money, but you have to distribute the certificates to your clients yourself.

### AUTHENTICATION

The SAS®9 BI Platform enables Web authentication. In this authentication model, either a Web server or a servlet container is responsible for authenticating a user. The Portal application establishes a trust relationship with the SAS Metadata Server by using the SAS Trusted User. In order to access other SAS application servers (SAS Workspace Server, SAS Stored Process, SAS OLAP Server), credentials for the authenticated server must be found in the SAS Metadata Repository. This section briefly discusses issues with the servers that run in the DMZ and the middle-tier. There are other papers that give details about SAS authentication and security.

There are several places where Web authentication can occur: the servlet container, a Web server, or a proxy server. Usually, performing authentication in the DMZ, as close to the client as possible, provides a higher level of security. It's better not to let unauthorized requests get into the secure zone if it is not necessary. If a proxy or a proxy Web server is used, it can perform the authentication and pass the user credentials to the servlet container as part of the HTTP request packet. Another advantage to using a front-end server to authenticate users is single sign-on. Probably a number of corporate applications will be supported by a common set of proxy and Web servers. By

having a common server handle authentication, users will not have to repeat authentication for each back-end application.

There are some limitations when you have the HTTP server perform authentication. The Portal Public Kiosk will not be available unless the application server performs the authentication. Also, it will not be possible to use customized authentication based on your own business logic. If either of these are important capabilities, then you might want to allow the application server to perform authentication.

There are several standard methods for HTTP authentication:

- **Basic Authentication** is the simplest and least secure. The credentials (user name and password) are passed as part of the request. The password is sent as plain text. The Apache HTTP server supports basic authentication by using mod_auth. With this module, a simple password file is managed as part of the Apache installation. This mechanism can be slow if the password file is very large. If security is important, then basic authentication should only be used with an encrypted connection (HTTPS). Apache provides the module `mod_auth_dbm` for storing credentials in a database.
- **Digest Authentication** encrypts the password. The Apache module `mod_auth_digest` is still experimental. As with basic authentication, a simple password file is used for a database.
- **Client Certificate Authentication** uses public key cryptography to authenticate the client. Usually, when SSL is used, the server must have a private key and public certificate to authenticate to the client. There are no passwords involved with this model. With Client Certificate Authentication the client must have a private key that is kept secret and a public certificate that is available to the server. This mechanism is ideal for use with Smart-Cards, which store the client private keys.

Of these three authentication models, SAS Web applications have been tested only with basic authentication. With Web authentication, the SAS BI Platform needs the credentials that are provided to the servlet in order to correctly map the user to a SAS Metadata Identity.

#### LOCK IT DOWN!

There are many other basic steps that you can take to tighten security and reduce the risk of infiltration. In a development or initial test environment, you probably do not want to lock things down too much, but in the final stages of testing and in a production environment, security should be fully enabled. Although the books recommended in the “Resources” section have numerous suggestions, a few of the most critical suggestions are summarized here:

- **Install and run the application server as a special user account—never as the root user.** The application server process should not have any special user privileges. It is best to compartmentalize and limit the system-level permissions that any specific application can have. In the event that infiltrators (hackers) gain control of an application server or the application server account, they wouldn’t be able to do as much damage as if they had access to a system administrator’s account.
- **Enable Java 2 Security.** SAS Web applications provide several policy files that restrict the permissions allowed to the SAS code base. When first installing SAS, you will probably want to demonstrate that things work with the security turned off. When you first turn the security on, you will want to use the “all permissions” versions of the policy files. If this works well, you can use the policy files with explicit security capabilities turned on.
- **Run applications in separate containers.** While application servers enable many unrelated Web applications and Enterprise JavaBeans (EJB) to execute in the same Java Virtual Machine (JVM), it might be a good idea to isolate products from different vendors by running them in their own JVM instances. Remember that everything that runs in a single JVM shares a common address space and does not fully benefit from the isolation that is provided by distinct-process virtual address spaces.
- **Limit access to application server configuration files.** Make sure that configuration files can be read and written only by those who need access. For example, the Tomcat directory `conf/` contains sensitive configuration files such as `server.xml`. If Tomcat basic authentication is used and the file `conf/tomcat-users.xml` is not protected, it could be modified to gain unauthorized access to Web applications.

## TUNING AND CONFIGURATION

Designing the right network and server topology in the beginning, will take care of many large scale performance and scalability considerations. Next, we will focus on tuning each component in the configuration: the operating system of each server machine, the JVMs, and finally the application servers. This section provides examples of tuning parameters that can be modified. This list is not comprehensive, but the references and resources included at the end of this paper discuss more parameters and provide much more detail.

### OPERATING SYSTEM TUNING

Check both your application server and your operating system tuning guides for appropriate operating system parameters. There might be some specific recommendations for tuning your operating system when running Java applications, particularly application servers that are not the usual default values. Here are some examples:

- Be sure that your TCP sockets are optimally configured. The default values might not always be optimal. As an example, BEA recommends reducing the keep-alive value for closed sockets to 60 seconds from the default value of 4 minutes for Solaris. BEA recommends a number of additional TCP settings for Solaris in the *WebLogic Server Tuning Hardware, Operating Systems, Network, and Tuning* guide.

- You will probably have to increase the number of open file descriptors on UNIX operating environments in order to handle a large number of open socket connections. In the WebLogic Server Tuning guide, BEA recommends setting `rlim_fd_cur` and `rlim_fd_max` to 8192 as a starting point.

- On UNIX, the maximum number of file descriptors that can be opened for use by the current process can be increased by using the `ulimit` command that is built into the shell. If you expect any more than a few concurrent sessions to be supported by your application, you will probably want to increase this value to about 5000 open connections before launching the application server.

- Hewlet Packard provides a tool called Java Out-of-Box (OOB) that will re-configure the HP-UX kernel with parameters that provide better behavior for Java applications.

### JAVA RUNTIME TUNING

There are several configuration options available for tuning the Java memory management system. The *SAS 9.1.3 Intelligence Platform: Planning and Administration Guide* recommends a number of options. See the section, “Tuning the Java Virtual Machine under Configuring Your Servers for Better Performance.” Always be sure that you are specifying options for the specific vendor’s Java Runtime. Many of the heap management configuration parameters that apply to Sun’s or Hewlet Packard’s runtimes will not be supported with IBM’s JVM. *SAS 9.1.3 Intelligence Platform: Planning and Administration Guide* suggests a number of “Quick Start Settings”. These provide a starting point. You will want to monitor system performance and adjust these to meet your needs based on performance testing and production system experience. There are a number of free tools available for monitoring and evaluating JVM memory usage and performing problem analysis. Some of these include `jstack`, `HPJtune`, `jvmstat`, `GCViewer`. `Jprobe` is another excellent tool; a free version is available.

Anytime that you upgrade your Java Runtime, you should review your memory management settings. In recent updates, Sun has been making a number of modifications to the memory management system to fix bugs and improve performance. These changes might affect the appropriateness of some options.

You should also be aware of an important issue that can occur with Sun’s and Hewlet Packard’s JVMs. The RMI distributed garbage collection (DGC) facility has a problem that can result in frequent unnecessary garbage collection. Long pauses to the running application will ensue as a result. Look for evidence of excessive garbage collection in a log file generated by -XX:+PrintGCDetails. You can work around this issue by setting the server JVM options `sun.rmi.dgc.server.gcInterval` and `sun.rmi.dgc.client.gcInterval` to a high value such as 3600000. Given that SAS Web applications use RMI to communicate with the SAS remote services application, it is recommended that you set this option for both components.

### APPLICATION SERVER TUNING

There are a number of ways to enhance application server performance. Parameters sometimes default to values that are more appropriate for development purposes than for use in a production environment. The references below suggest many application server-specific ways to do this. Here are some general configuration parameters to look for:

- **Turn off the servlet auto-reloading feature in a production environment**. In a development environment, it can be useful for the servlet container to check for updated servlet code and automatically reload the application as needed. This requires regular monitoring of the file system for updates; Tomcat and WebLogic perform this check, by default. The *SAS 9.1.3 Intelligence Platform Administration and Planning Guide* provides directions for turning this feature off for Tomcat. (See the section "Tuning the J2EE Server or Servlet Container.") SAS Web applications are distributed with a weblogic.xml file that turns off servlet auto-reloading. By default, Servlet checking is off for WebSphere.

- **Disable Domain Name System (DNS) Lookups**. The servlet specification provides an API for accessing the name of the remote client or the most recent proxy that forwarded the request. Supporting this feature requires a DNS lookup for every request. Turning on this feature is generally not that useful in a configuration with proxy servers and results in increased network traffic and potential delayed responses. By default, Tomcat does the DNS lookup. This option should be turned off. By default, WebLogic returns the IP address rather than the host name.

- **Tune the number of threads based on the number of CPUs and current demand**. Each application server has some configuration options to manage threads that perform various functions. You will want to configure these options based on the amount of traffic your servers receive and on the number of CPUs that your server has at its disposal. Tomcat Connectors listen for incoming requests. A pool of threads is available to handle these requests. The number of threads available can be managed with the `minProcessors` and `maxProcessers` parameters in the `server.xml` configuration file. WebLogic uses Execution Queues to process requests. New queues can be created and a pool of threads can be configured. See "Tuning the J2EE Server or Servlet Container" in the *SAS 9.1.3 Intelligence Platform Administration and Planning Guide* for guidelines on tuning these parameters for the Information Delivery Portal Tomcat and WebLogic. WebSphere also provides configuration parameters for managing the minimum and maximum numbers of threads available for the Web container.

- **Precompile JavaServer Pages (JSP)**. By default, JSPs are compiled the first time they are accessed, which results in very slow performance for the client that makes the first access. WebLogic and WebSphere provide options that enable JSPs to be compiled when the application server is started. Be aware that this step can take time and will cause startup to take a little longer. Tomcat does not provide a facility to pre-compile JSPs. You can create an automated client application that causes an initial hit for each page.

## SUPPORTED APPLICATION SERVER PRODUCTS, VERSIONS, AND FIXES

There are many J2EE application servers that are available on the market today. While the J2EE standard provides common programming interfaces that are used by all compliant application servers, there is a lot of room for variation; especially in the areas of deployment, configuration, and administration. Before SAS can claim to support an application server, it must thoroughly test its Web applications in the vendor's container, develop installation and deployment tools, and train its technical support staff on the use of these application servers.

SAS knows that many of its customers have also invested in application server training for their employees, and their employees have developed expertise with a specific vendor's products. Many would prefer to leverage that experience with SAS products if possible. At SAS, the priorities for selecting these application servers are based on customer feedback. Due to testing requirements and limited resources, the SAS BI Platform currently only supports three servlet containers: Apache Jakarta Tomcat 4.1, BEA WebLogic Server 8.1, and IBM WebSphere Application Server 5.1. A copy of Tomcat 4.1.18 is provided on the Third- Party CD and can be included in the installation plan. SAS does some testing of Web applications with these application servers under each supported operating environment. Specific operating environments are supported for each application server.

The Third-Party Software Downloads Web page on the SAS Customer Support Web site (http://support.sas.com/thirdpartysupport/) identifies the most current set of operating systems, application servers, and Java Development Kits (JDK) that are supported. At this site you will find SAS recommendations for specific application server service packs and patches along with any special notes for installation. These are the configurations that SAS has tested during the development process. These might not always be the latest patches or service packs available from the application server vendor. Often the vendor will release a new service pack or patch very soon after SAS releases its own update. In an ideal world, you should be able to use the latest patch level that is available from a vendor, but historically SAS has encountered problems with some upgrades from every supported vendor. A best practice recommendation is to use the exact version that is specified on the Third-Party Support site.

If you have questions or concerns regarding support of a particular service pack or patch, contact SAS Technical Support. SAS has developed a strategy for delivering SAS service packs on top of the SAS 9.1.3 Platform. Application server support levels are reviewed before SAS service pack testing begins.

#### VENDOR SUPPORT POLICIES AND PRACTICES

Like SAS, all application server providers have their own strategy for supporting their customers. Unless you use the copy of Tomcat that is provided on the Third Party CD, you will probably purchase your application servers directly from one of the supported vendors, and the vendor will handle your technical support for the application server. SAS recommends that you have your application server licenses in place before beginning the SAS installation process.

SAS does provide the option of purchasing an OEM version of application servers from BEA and IBM when customers purchase solutions such as SAS Marketing Automation, SAS Financial Management, and SAS Strategic Performance Management. If you purchase your application server directly from SAS as part of one of these solutions, SAS will be your first-level technical support provider.

#### APACHE JAKARTA TOMCAT SUPPORT

Tomcat is an open-source servlet container that is available through the Apache Software Foundation. You can search for known Tomcat bugs in the Apache Software Foundation bug database, Bugzilla. Information is available from The Apache Jakarta Project home page (http://jakarta.apache.org/tomcat/index.html.)

#### BEA WEBLOGIC SERVER SUPPORT

BEA provides patches for non-security related issues directly to their customers. These patches can be downloaded but are not publicly available on their Web site. If you buy WebLogic directly from BEA, contact BEA to gain access to these patches. If you purchased a copy of WebLogic Server as part of a bundled SAS solution such as SAS Marketing Automation, the required patches will be automatically installed as part of your SAS installation process. If SAS needs to recommend additional patches after release, those patches will be made available to you through SAS Technical Support.

BEA handles security patches in a different manner. Security patches are made available on their public Web site at http://dev2dev.bea.com/resourcelibrary/advisoriesnotifications/index.jsp. SAS documents any security fixes that we have installed during our testing process on the Third-Party support site.

#### IBM WEBSPHERE APPLICATION SERVER SUPPORT

IBM provides *fixes* and *cumulative fix packs* on their public Web site. As the name implies, fix packs accumulate all previously released fixes since the initial product shipment. If you purchase your product directly from IBM, you can download these SAS recommended updates from IBM's Web site and install them on top of the base product before deploying any SAS Web applications. If you license WebSphere Application Server Network Deployment bundled with a SAS solution, it will come with a CD that contains the recommended fixes and cumulative fixes that should be applied after installation of the base product.

## CONCLUSION

When designing an appropriate middle-tier infrastructure for your SAS Business Intelligence application, topology design, security issues, performance and tuning considerations, in addition to scalability, maintainability, and availability, all need to be carefully balanced against budgetary and infrastructure constraints. By carefully reviewing your production system requirements and corporate policies and priorities, you will be able to implement a robust, secure, scalable middle-tier that fits your company's needs. For details about implementing these recommendations, see the reference and resource materials below.

## REFERENCES

BEA Systems, Inc. 2004. *Product Documentation, WebLogic Server Performance and Tuning*. Available http://e-docs.bea.com/wls/docs81/perform/HWTuning.html.

SAS Institute Inc. 2004. *SAS® 9.1.3 Intelligence Platform: Planning and Administration Guide.* Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2004. *SAS® Web Infrastructure Kit 1.0 Administration Guide.* Cary, NC: SAS Institute Inc.

## RESOURCES

Barcia, Roland and Hines, Bill, et al. 2005. *IBM WebSphere Deployment and Advanced Configuration.* Upper Saddle River, NJ: Prentice Hall.

Brittain, Jason and Darwin, Ian F. 2003. *Tomcat: The Definitive Guide*, Sebastopol. CA: O'Reilly and Associates.

Chopra, Vivek, et al. 2004. *Professional Apache Tomcat 5*, Sebastopol CA: Wrox.

Francis, Tim, et al. 2003. *Professional IBM WebSphere 5.0 Application Server, A Guide to Building J2EE Applications from IBM WebSphere Architects*, Wiley and Sons.

Kovari, Peter. 2005, *WebSphere Security Fundamentals,* International Business Machines Corporation.

Nyberg, Gregory and Patrick, Robert, et al. 2003. *Mastering BEA WebLogic Server Best Practices for Building and Deploying J2EE Applications,* Indianapolis, IN: Wiley.

Ramanujam, Vinohda and Braswell, Byron. 2003. "IBM WebSphere V5 Edge of Network Patterns". Available http://www.redbooks.ibm.com/redbooks/pdfs/sg246896.pdf.

Roehm, Birgit and Csepregi-Horvath, Balazs, et al. 2004, *IBM WebSphere V5.1 Performance, Scalability, and High Availability, WebSphere Handbook Series,* International Business Machines Corporation.

Williamson, Leigh et al. 2005. *IBM WebSphere System Administration*, Upper Saddle River, NJ: Prentice Hall.

Wunderlich, Holger et al. 2003. "Building Multi-Tier Scenarios for WebSphere Enterprise Applications", Available http://www.redbooks.ibm.com/redbooks/pdfs/sg246956.pdf.

## ACKNOWLEDGEMENTS

I would like to thank my colleagues at SAS who have reviewed this paper and have contributed many ideas and suggestions. In particular, I would like to thank Don Asper, Cliff Bills, Erik Burgdorf, Don Chapman, Gordon Hirsch, Keith Holdaway, Dharmita Lutz, Steve Jenisch, Doug Melzer, Larry Noe, Rob Owen, Kevin Smith, Brian Thorstad, Susanna Wallenberger, George Wilder, and Lew Woods for their assistance.

## CONTACT INFORMATION

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

John Roth
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: 919-531-6868
Fax: 919-677-4444
E-mail: John.Roth@sas.com