

FSA Integration Partner

United States Department of Education

Federal Student Aid



FAFSA 8.0 - WAS 5.0 Configuration Change Report

Task Order #145

Deliverable 145.1.3

Version 1.2

December 22, 2003



Executive Summary

Introduction

The U.S. Department of Education's Office of Federal Student Aid (FSA) administers and operates the Free Application for Federal Student Aid (FAFSA). While available in paper form, FSA also provides this service over the Web. College bound students seeking federal financial aid make use of the FAFSA program. Over 7 million students utilized the Web Site to apply for federal financial aid during the academic year 2002-2003. FSA anticipates that the number of users/applicants will increase by 30% during the 2003-2004 academic year, and will continue to rise in future years as the number of paper submissions decreases. This anticipated growth makes it imperative for FSA to maximize the capacity and availability of the FAFSA Web infrastructure while concurrently minimizing the amount of support FSA's representatives provide for questions from students or those experiencing difficulties with completing the form. The FAFSA Web product is commonly referred to as 'FAFSA on the Web' and incorporates all requirements related to the paper FAFSA for each school year.

The primary objectives for redesigning FAFSA on the Web are to leverage industry best practices to improve usability and accessibility by customers, performance of the Web application during peak periods of FAFSA submissions, and create the foundation for efficient enhancements, as necessary.

Background

In the 2002-2003 academic year, there was a 50% increase in FAFSA Web users and application submissions. As the number of FAFSA on the Web users increases, the application must scale to accommodate the additional users. FSA anticipates that the number of applicants will increase by 30% this year. To account for the expected increase, a N-Tiered architecture has been implemented for horizontal scaling.

Objective

The purpose of the Configuration Change Report is to recommend a production configuration to handle the peak load for 2003-2004.



Amendment History

DATE	SECTION / PAGE	DESCRIPTION	Submitted By
12/09/03	All	Performance test configuration document	Roshani Bhatt
12/17/03	2.6	Updated JVM Heap Size - correctly listed as 1 GB in paragraph - updated in breakout table to be 1024. Updated Peak WAS Servers to updated server names	Matt Portolese
12/22/03	2.1	Updated section to include a one line description of each eFix for WAS.	Matt Portolese



Table of Contents

- 1 INTRODUCTION5**
- 1.1 OVERVIEW5
- 1.2 FAFSA SOFTWARE MATRIX5
- 1.3 COMPONENT ARCHITECTURE.....6
 - 1.3.1 Load Balancer6
 - 1.3.2 Web Server.....6
 - 1.3.3 Application Server6
 - 1.3.4 Deployment Manager6
 - 1.3.5 Messaging.....7
 - 1.3.6 Operating System.....7
- 1.4 FAFSA ON THE WEB CURRENT CONFIGURATION.....9
- 2 WEBSHERE APPLICATION SERVER SETTINGS10**
- 2.1 EFIXES10
 - 2.1.1 PQ77056.....10
 - 2.1.2 PQ71950.....10
 - 2.1.3 PQ72597.....10
- 2.2 SECURITY.....10
- 2.3 SSL VERSUS NON SSL.....10
- 2.4 WEB CONTAINER THREAD COUNT.....11
- 2.5 WEB CONTAINER MAXKEEPALIVECONNECTIONS AND MAXKEEPALIVEREQUESTS11
- 2.6 HEAP SIZE.....12
- 2.7 SESSION SIZE12
- 2.8 SESSION CACHE13
- 2.9 WEB APPLICATION - RELOAD INTERVAL AND ENABLE13
- 2.10 SESSION AFFINITY13
- 3 IBM HTTP SERVER SETTINGS.....14**
- 3.1 EFIXES14
- 3.2 MAX CLIENTS14
- 3.3 MAXKEEPALIVEREQUESTS14
- 3.4 KEEPALIVETIMEOUT14
- 4 PERFORMANCE ENVIROMENT CONFIGURATION.....14**
- 4.1 WEB SERVERS.....14
- 4.2 APPLICATION SERVERS15
 - 4.2.1 Template Application Server parameters15
- 4.3 OTHER SCRIPTS.....16
- 4.4 OTHER SOFTWARE (RCS, WILY, WSADMIN)17
- 4.5 SHARED LIBRARIES.....17
 - 4.5.1 JDBC Providers.....18
 - 4.5.2 Oracle JDBC18
 - 4.5.3 Shadow Direct JDBC.....19
- 5 FAFSA RECOMMENDED PRODUCTION ARCHITECTURE20**
- 5.1 PERFORMANCE TEST RECOMMENDATION FOR FAFSA PEAK20
- 5.2 PROPOSED FAFSA 8.0 PRODUCTION ENVIRONMENT – GO LIVE21
- 5.3 PROPOSED FAFSA 8.0 PEAK ENVIRONMENT.....22



1 Introduction

1.1 Overview

This document outlines the configuration of the performance environment for the *FAFSA on the Web 8.0* application. This environment was used to stress test the performance environment infrastructure as well as the FAFSA application code. The environment is composed of two Cisco Content Services Switch (CSS) servers that are used as Load Balancers and a Solaris server that is used for WebSphere Deployment Manager. Two HP-UX 11i N-class servers serve as IBM HTTP Server (IHS) Web Servers and 2 HP-UX 11i N-class servers serve as WebSphere Application Servers (WAS). The following table specifies the ITA software versions.

1.2 FAFSA Software Matrix

The following matrix captures the major technology components of FAFSA and the changes for the application’s January 2004 release to August 2004.

No.	Component	Previous Version FAFSA 7.0	New version FAFSA 8.0
1	Akamai	Yes	Yes
2	Application Code Changes	FAFSA 7.0	FAFSA 8.0
3	DB2	Version 6.0	Version 7.0
4	Hardware Refresh in Performance Environment	Web Servers(4x450 CPU, 8 GB memory); Application Servers (8X360 CPU, 8 GB memory and 8X750 CPU, 16 GB memory)	Web Servers (8X750 CPU, 8 GB memory); Application Servers (8X750 CPU, 8 GB memory and 8X750 CPU, 16 GB memory)
5	HP Operating System	HP-UX 11	HP-UX 11i
6	HTML vs. SSL images	Yes	Yes
7	IBM HTTP Server	Version 1.3.2.6	Version 1.3.2.6
8	IBM WebSphere Application Server	Version 3.5.6	Version 5.01 + PQ77056 + PQ71950 +PQ72597
9	Load Balancing	eNetwork Dispatcher (4.0.2.25)	CSS
10	Mainframe OS	ZOS 1.2	ZOS 1.4
11	Oracle Database	Version 8.1.72	Version 8.1.74
12	CICS	Version 1.2	Version 2.2
13	Shared vs. Dedicated Environments	Dedicated Web and Application Servers Shared Database and Mainframe	Dedicated Web and Application Servers Shared Database and Mainframe
14	Web Trends	4.0	5.0
15	WebSphere MQ	Version 5.2	Version 5.3.1



16	Wily Interscope	Not installed in performance test environment	Activated in performance environment
----	-----------------	---	--------------------------------------

In production, FAFSA 8.0 will have the following components:

- Akamai
- CSS
- WebSphere Deployment Manager
- Web Server (IHS)
- Application Server (WAS)
- WebSphere MQ
- Oracle Databases
- Mainframe
- DB2

1.3 Component Architecture

1.3.1 Load Balancer

CSS is used to load balance HTTP requests to the Web Servers. CSS includes a proprietary algorithm for checking the health of each server and redistributes the load accordingly if a Web Server is experiencing problems. The CSS machine is configured with Box-To-Box redundancy and Advanced SSL turned off. Sticky IP is turned off. Should the primary CSS go down or is taken down, the backup CSS immediately comes online.

1.3.2 Web Server

IHS is an Apache based Web Server that provides HTTP server functionality. Any static HTTP documents can be served via the Web Server. The IHS Web Server plugin provides connectivity from the Web Server to the Application Server. The plugin provides load balancing and failover for the Application Server. Each web server can connect to both Application Servers.

1.3.3 Application Server

WAS is a Java Based Servlet and JSP Engine that conforms to Sun Microsystems’s J2EE 1.3 specification. WebSphere is built upon the Java Development Kit 1.3.1.08, which includes servlet specification 2.3 and JSP specification 1.2.

1.3.4 Deployment Manager

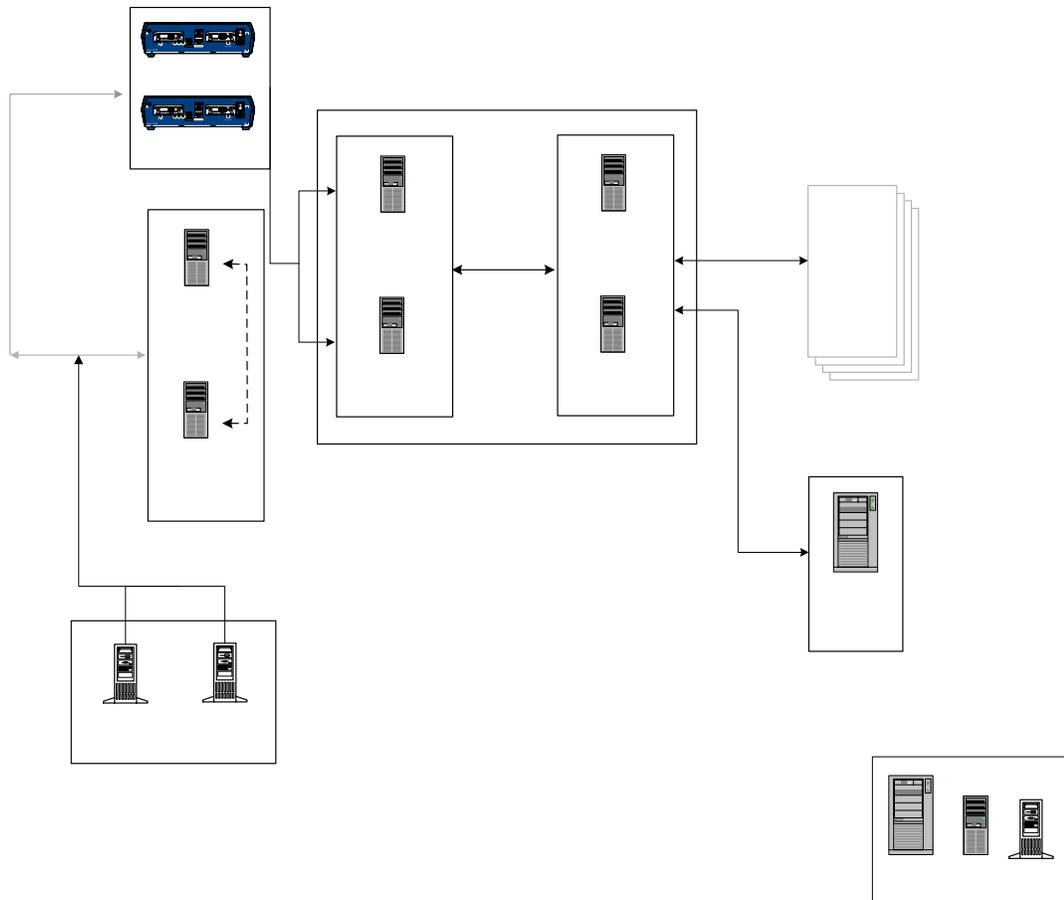


WebSphere Deployment Manager allows for single point management for the entire production FAFSA WebSphere cell, which would include administrative configuration and control over the FAFSA cluster.

1.3.5 Messaging

An MQSeries cluster is configured on the Application Servers so that the FAFSA application can drop messages to the EAI cluster, which immediately transports it to the destination.

The following diagram shows a pictorial representation of what the Performance Environment looks like.



1.3.6 Operating System

HP-UX Version 11i on the Web and Application Servers
December 2002 Quality Patch



Several HP-UX 11i tcp-ip parameters need to be tuned so that IHS and WebSphere may function correctly. These tcp-ip parameters are controlled via the NDD interface.

Ndd parameters - /etc/rc.config.d/nddconf

tcp_conn_request_max 1024

Specifies the maximum number of of connection requests that the operating system can queue when the server does not have any available threads.

fin_wait_2_timeout 1200000

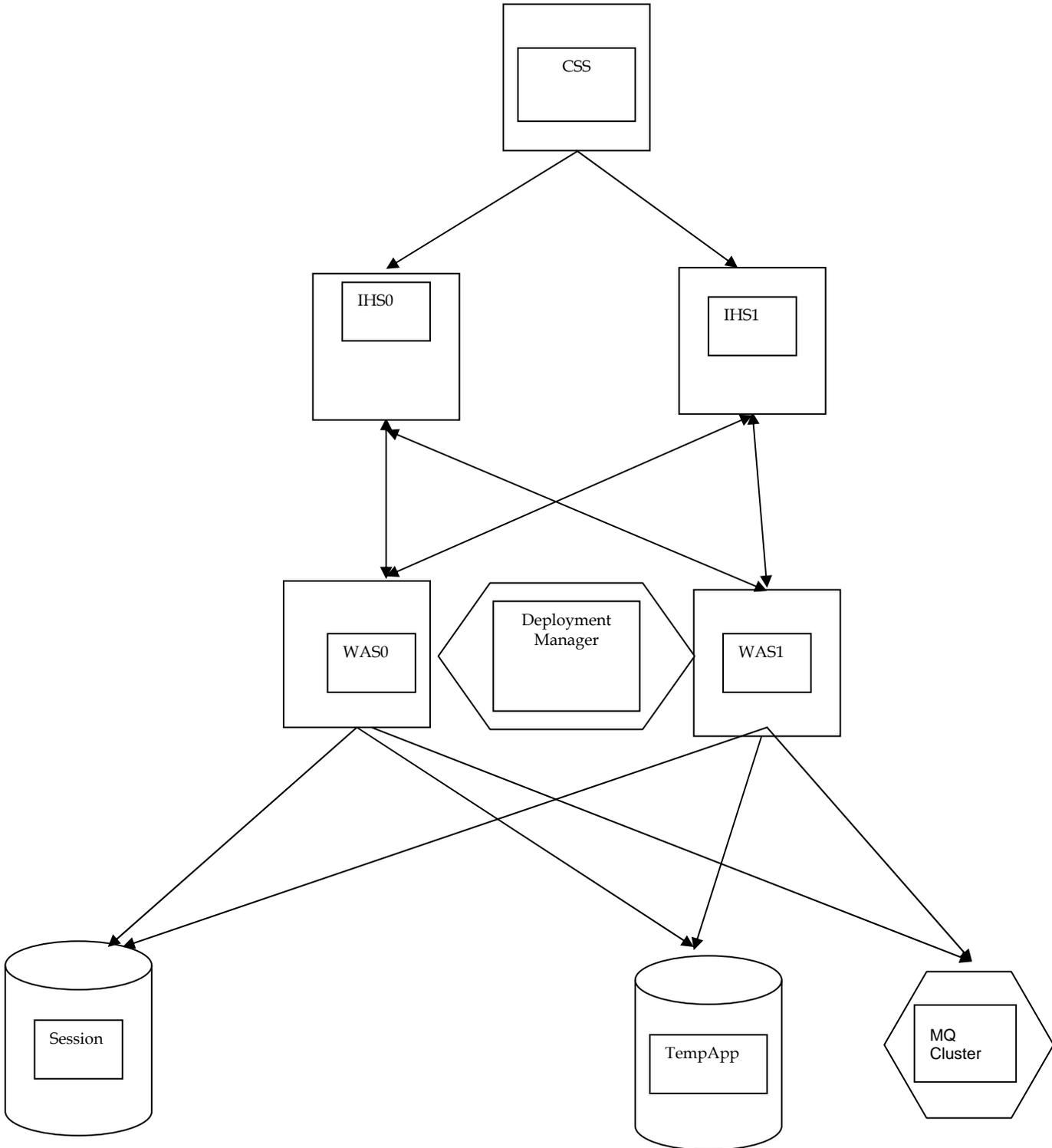
Specifies a timeout when a tcp-ip port enters the FIN_WAIT_2 state. HP-UX 11i by default specifies no timeout with a value of 0. HP support recommended a timeout of 1200000 milliseconds which equates to 20 minutes.

Chatr +pd64M +pi64M /opt/WebSphere/AppServer/java/bin/PA_RISC2.0/native_threads/java

Setting the Java Virtual Machine page to 64M. This is specified within the IBM WebSphere InfoCenter as a best practice for HP-UX. Running this command will error if a Java daemon is executing.



1.4 FAFSA on the Web Current Configuration





2 WebSphere Application Server Settings

2.1 EFixes

2.1.1 PQ77056

This Efix corrects a problem with WebSphere 5.0.1 where if Session Persistence is enabled and the session is greater than 4 KB then the session is stored properly into Oracle.

2.1.2 PQ71950

This Efix corrects a problem with WebSphere 5.0.1 for a First Failure Data Collection (FFDC) memory leak.

2.1.3 PQ72597

This Efix corrects a problem with WebSphere 5.0.1 for a memory leak associated with Performance Monitoring Infrastructure (PMI) data use.

2.2 Security

Admin Console Security was enabled for the performance test. The use of Computing Base Security by CSC on the HP-UX systems made the local OS registry incompatible with WebSphere security. Also since no LDAP registry is available, the only other option is to use a custom registry that comes with WebSphere. Basically User IDs and passwords are stored in root owned files and WebSphere verifies User ID and Passwords using built in custom registry support. The password and User ID files are located at:

```
/opt/WebSphere/security/users.props
/opt/WebSphere/security/groups.props
```

2.3 SSL versus Non SSL

By default the WebSphere plugin passes on the protocol that the Web Server request came in as to the application Server. This means that a HTTPS request from the server would result in a HTTPS request between the Web Server and the Application Server. An HTTP request would pass on as a http request.

Client Request	Protocol Enabled in Plugin	AppServer Protocol
HTTP	HTTP,HTTPS	HTTP
HTTPS	HTTP,HTTPS	HTTPS
HTTPS	HTTP	HTTP
HTTP	HTTPS	HTTPS



It is common knowledge that SSL encryption uses more CPU than non SSL traffic. Documentation indicates that the handshake done to determine the cipher specification and set up the primary and private key is the big CPU user. Once the cipher and key is determined, encryption for the bulk data transfer uses much less CPU.

Performance testing showed that by only allowing http traffic in the plugin, there is a substantial difference in performance. With SSL enabled, average CPU utilization increased by 20%/1000 users.

<u>Users with SSL Enabled</u>	<u>Average CPU Utilization</u>
1000	20%
2000	40%
3000	60%

With SSL disabled, the CPU utilization increased by 10%/1000 users

<u>Users with SSL Disabled</u>	<u>Average CPU Utilization</u>
1000	10%
2000	20%
3000	30%

SSL can be disabled in the WebSphere plugin by deleting the HTTPS transport within the Web Container.

Application Server-><Server name>->Web Container->HTTP Transport-><Port number>

2.4 Web Container Thread Count

WebSphere 5.x introduced an HTTP Server as the transport for the WebSphere plugin that lives between the Web Server and the Application Server. A HTTP client (WebSphere plugin) connects to WebSphere’s embedded WebServer to ensure communication between the Web Server and the Application Server. By specifying a maximum number of threads on the Application Server, the WebSphere Admin is able to limit the number of threads into the WebSphere Web Container.

[Application Server](#)-><[Server name](#)>-><[Web Container](#)>->Thread Pool

Minimum Size	80
Maximum Size	100
Thread inactivity timeout	3500

2.5 Web Container MaxKeepAliveConnections and MaxKeepAliveRequests

Since WebSphere 5.0 has introduced HTTP 1.1 protocol between the Web Server and the Application Server, the Web Server plugin now can keep persistent HTTP connections open to the application server for a defined number of requests. This enhances performance due to



fact that the pre 5.0 plugin had to open a socket port for every Application Server request. To enable this performance enhancement, the following custom properties must be added to the HTTP Transport configuration. This should be added for the SSL and non SSL port. The MaxKeepAliveConnections for SSL and non-SSL should add to 80% of the maximum thread count (Specified in section 2.1) allowed for the Application Server.

If the maximum thread count for the web container is set at 100, then 80% of the threads can be allocated for the persistent connections. In this case 80 connections were specified for the http transport. The MaxKeepAliveRequests parameter specifies how long the persistent connection lives before it is recycled. Since the Web Server thread will initiate the Application Server thread, it is best to ensure that the Web Server MaxKeepAliveRequests parameters matches the App Server parameter. In this case both the Web Server and Application server parameter is set to 10000000.

[Application Server](#)-><[Server name](#)>-><[Web Container](#)>-><[HTTP Transport](#)>-><[Port number](#)>->Custom Properties

MaxKeepAliveConnections	80
MaxKeepAliveRequests	10,000,000

2.6 Heap Size

Heap size of a JVM is very important to an Application’s performance. An application will build a certain number of objects within the heap and how the application terminates these objects determines how big a heap may need to get. Performance testing has shown that FAFSA needs a large memory map to handle 3000 users. This memory map may vary depending on a user’s session length or whether a large number of runtime errors are occurring. Performance testing has shown that a heap size of 1gigabyte is sufficient to process 3000 users with a session lifetime of 900 seconds. In fact in testing 3000 users for 1 hour using the Fill Out FAFSA business process, 1 full garbage collection was performed. On the average, scavenger garbage collections were able to maintain the heap space with 9 scavengers/ minute.

[Application Server](#)-><[Server name](#)>-><[Process Definition](#)>-><[Java Virtual Machine](#)>

Initial Heap Size	1024
Maximum Heap Size	1024

2.7 Session Size

The eight business processes that were used during the performance test have been checked to ensure that persisted HTTP servlet session size is less then 4K. The FAFSA business processes and their respective session size appear in the table below.

Fill Out a FAFSA	2.9 Kbytes
Corrections	4.1 Kbytes



2.8 *Session Cache*

Since WebSphere plugin guarantees session affinity, WebSphere maintains a Session memory cache that holds the latest session updates. Its important that this cache is sized correctly or if the Cache fills up, the Application Server will be forced to persist the sessions that have not been used recently. This can lead to severe thrashing between the Application Server and the database if the cache is completely overrun. The cache was sized to 4000 records so that a minimum of 3000 users were interacting with the Application Server and the cache still had 25% spare space for runtime errors and session timeouts.

[Application Server](#)-><[Server name](#)>->[Web Container](#)->[Maximum in-memory session count](#)

2.9 *Web application - Reload interval and enable*

Reload enable specifies whether the Web application's file system is scanned for updated files periodically on the Application Server. If files such as servlet class files or JSPs are found to be updated since the last scan then they can be reloaded. Reload enable should be turned off for production.

[Applications](#)-> [Enterprise Applications](#)-> [FOTW App](#) -> Reload Enabled

2.10 *Session Affinity*

Due to the fact that Session Affinity is guaranteed by the Web Server plugin on the client's first request, a Session database is not considered critical to ensuring the client's request goes to the same application server every time. This simple enhancement turns the session database into a failover device and allows WebSphere to reduce the number of session database writes and reads. In fact, with WebSphere 5.0 the session management configuration has reduced session database writes from every servlet call (3.5.6) to a time based system. The Application Server session management needs to be configured for the custom setting (default). This allows only updated attributes to be saved and the session is saved every 120 seconds.

[Application Server](#)-><[Server name](#)>->[Web Container](#)->[Session Management](#)->Distributed Environments Settings



3 IBM HTTP Server Settings

3.1 *Efixes*

WAS_Plugin_09_03_2003_5.0.x_cumulative (Cumulative Plugin efix)

3.2 *Max Clients*

The value of the IHS max client parameter can significantly affect the performance of an application. If this value is too high, it is possible to overwhelm the application server and cause it to stall. If this value is too low then it is possible to induce a bottleneck with client requests before they reach the application server.

MaxClients	768
------------	-----

3.3 *MaxKeepAliveRequests*

This parameter specifies the maximum number of requests to allow during the Keep Alive persistent connection. Once the number of requests on a single persistent connection passes this number the connection is recycled.

MaxKeepAliveRequests	10,000,000
----------------------	------------

3.4 *KeepAliveTimeout*

This parameter represents the number of seconds the persistent connection will wait before closing the connection for lack of traffic.

KeepAliveTimeout	5
------------------	---

4 Performance Environment Configuration

This is how the Performance Test environment was configured for our testing.

4.1 *Web Servers*

IBM HTTP Server 1.3.26.1 + WAS_Plugin_09_03_2003_5.0.x_cumulative (Cumulative Plugin efix)

The original plugin did not pass SSL attribute data correctly to the Application Server. When adding additional plugin efixes or fix packs, special care needs to be taken to ensure that this efix is included.



The following Web Server Configuration is maintained in the /opt/HTTPServer/conf/httpd.conf file on both Web Servers.

MaxClients	768
MinSpareServers	5
MaxSpareServers	10
StartServers	10
MaxRequestPerChild	10000
MaxKeepAliveRequests	10,000,000
KeepAliveTimeout	5

Non-SSL Name Based Virtual Hosts

Url	HTTP://perf.fotw.ed.gov
Doc root	/www/fotw/htdocs
IP address	4.20.18.111
Port	80

SSL Name Based Virtual Host

Url	HTTPS://perf.fotw.ed.gov
Doc root	/www/fotw/htdocs
IP address	4.20.18.111
Port	443
Certificate	fotwperf-2

4.2 Application Servers

Version 5.01 + PQ77056 + PQ71950 +PQ72597

The Performance Test used a cluster of four Application Servers. Two Application Servers resided on HPN8 following the naming convention of FOTWn8c1 and FOTWn8c2. Two Application Servers resided on RP3 following the naming convention FOTWrp3c1 and FOTWrp3c2.

4.2.1 Template Application Server parameters

The current version of WAS employs a web-based administrative client that may be viewed in a browser, but the previous WAS version (3.5.6) utilized a Java client. The path to the "Java Virtual Machine" is included here as it appears in the WAS 5.0 administrative client to illustrate this new capability.

[Application Servers](#) > [FOTWServer](#) > [Process Definition](#) > Java Virtual machine



```
Classpath
BootClasspath
Initial Heap Size =1024
Maximum Heap Size =1024
Verbose garbage collection=not checked (Gc data goes to native_stdout.log)
```

[Application Servers](#) > [FOTWServer](#) > [Process Definition](#) > [Java Virtual Machine](#) >Custom Properties

```
APP_PROPERTY_PATH = ${FOTW_ROOT}/properties
Autonomy.config.xml = ${FOTW_ROOT}/properties/autonomy.properties
EAI_CONNECTION_MODEL = POOL
EAI_MQ_PROP_FILE = /www/eai/eaicodr1/mqipool.properties
LOGGING_CONFIG_FILE = www/eai/eaicodr1/EAICONF.INI
Syslog.config.xml = ${FOTW_ROOT}/properties/rcs.xml
awt.toolkit = com.eteks.awt.PJAToolkit
com.ibm.websphere.sendredirect.compatibility = true
java.awt.fonts = ${JAVA_HOME}/jre/lib/fonts
java.awt.graphicsenv = com.eteks.java2d.PJAGraphicsEnvironment
java2d.font.usePlatformFont = false
log4j.configuration = file:${FOTW_ROOT}/properties/log4j.properties
user.home = ${FOTW_ROOT}/lib/pja
ws.ext.dirs = ${FOTW_ROOT}/lib/logger
```

[Application Server](#)-><[Server name](#)>->[Web Container](#)->Session Management

```
Enable Cookies - Check
Enable URLRewriting - Check
Allow Overflow – Check
Session Timeout = 30 Minutes
Maximum in memory Session Count = 4000
```

Sessions were set up via the persistent session configuration to store sessions to a database. Upon startup of the different Application Servers

4.3 Other Scripts

NodeSM – Nanny Startup for the node agent that was written at FSA. This script monitors the Node Agent Java process and restarts it up to 3 times if the node agent dies with a non-zero return code. The Vanilla (out-of-the-box) WebSphere product does not have a monitoring nanny for the node agent. This script calls startNode_mod.sh which must reside in the <WAS_ROOT>/bin directory. NodeSM only needs to reside on nodes that have node agents in the /opt/WebSphere/AppServer/bin directory

An example of starting this would be:

```
cd /opt/WebSphere/AppServer/bin
./NodeSM &
```



ManagerSM - Nanny Startup for the Deployment Manager agent that was written at FSA. This script monitors the Deployment Manager Java process and restarts it up to 3 times if the Deployment Manager dies with a non-zero return code. The Vanilla WebSphere product does not come with a monitoring nanny for the Deployment Manager. This script calls startManager_mod.sh which must reside in the <WAS_DEPLOYMENTMANAGER_ROOT>/bin directory. ManagerSM only needs to reside on the Deployment Manager in the /opt/WebSphere/DeploymentManager/bin directory

For example:

```
cd /opt/WebSphere/DeploymentManager/bin  
./ManagerSM &
```

4.4 Other Software (RCS, Wily, wsadmin)

```
/opt/WebSphere/AppServer/lib/ITA/protomatter-1.1.8.jar  
/opt/WebSphere/AppServer/lib/ITA/rcs.exception.5.0.jar  
/opt/WebSphere/AppServer/lib/ITA/rcs.logging.5.0.jar  
/opt/WebSphere/AppServer/lib/ITA/rcs.search.5.0.jar  
/opt/WebSphere/AppServer/lib/ITA/StartupRcs.jar  
/opt/WebSphere/AppServer/lib/ITA/rcs.persistence.5.0.jar  
/opt/WebSphere/AppServer/lib/ITA/commons-logging.jar  
/opt/WebSphere/AppServer/lib/ITA/commons-logging-api.jar  
/opt/WebSphere/AppServer/lib/ITA/commons-httpclient-2.0-beta1.jar  
/opt/WebSphere/AppServer/lib/ITA/jakarta-oro-2.0.5.jar  
/opt/WebSphere/AppServer/lib/ITA/RCSjars.tar  
/opt/WebSphere/AppServer/lib/ITA/backup  
/opt/WebSphere/AppServer/lib/ITA/rcs.email.5.0.jar  
/opt/WebSphere/AppServer/lib/ITA/rcs.logging.5.0.jar  
/opt/WebSphere/AppServer/lib/ext/ProbeBuilder.jar  
/opt/WebSphere/AppServer/lib/ext/Agent.jar
```

4.5 Shared Libraries

[Environment](#)->Shared Libraries

Shared libraries are useful when different versions of a common framework are to be associated to different applications. An example of this is when Application A makes use of logging framework 1.2 and Application B makes use of logging framework 1.0. The following Jar files are defined as Shared Libraries:



CMNJLOG.jar
 EAI.jar
 StartupRcs.jar
 com.ibm.mq.jar
 connector.jar
 Jakarta-oro-2.0.5.jar
 mqm java lib
 protomatter-1.1.8.jar
 rcs.exception.5.0.jar
 rcs.logging.5.0.jar
 rcs.persistence.5.0.jar
 rcs.search.5.0.jar
 commons-logging.jar
 commons-logging-api.jar
 commons-httpclient-2.0-beta1.jar

Within each Application Server, these jars were added to the Application Classloader

[Application Servers](#)-><[Server Name](#)-><[ClassLoader](#)->ClassLoader_1

4.5.1 JDBC Providers

The Oracle data sources are used in saving Temp Application data, PIN data, and Session data. The Shadow JDBC providers just need to be created so that the JNDI can find their names. They are not used in any real database transaction.

Name	Oracle JDBC Thin Driver	ShadowDirect JDBC Driver
Description	Oracle JDBC Thin Driver	Custom JDBC2.0-compliant provider configuration
Classpath	\${ORACLE_JDBC_DRIVER_PATH}/classes12.zip	\${SHADOW_JDBC_DRIVER_PATH}/scjd12.jar
Native Library Path	\${ORACLE_HOME}/lib	\${SHADOW_HOME}/lib
Implementation Classname	oracle.jdbc.pool.OracleConnectionPoolDataSource	com.neon.jdbc.DataSource

4.5.2 Oracle JDBC

For Oracle JDBC Thin Driver, create the following data sources (V5). Please note that these are the custom properties for the Performance Test . The custom properties would be different for production.

Name	JNDI Name	Custom Properties	Description	Connect Info
------	-----------	-------------------	-------------	--------------



FAFSA 8.0 – WAS 5.0 Configuration Change Report

FOTW34TempAppDS	jdbc/FOTW34TempAppDS	<ul style="list-style-type: none"> • URL=jdbc:oracle:thin:@4.20.14.15:1672:FAFSASTG • dataSourceName=FOTW34TempAppDS • databaseName=FAFSASTG • enableMultithreadedAccessDetection=false • portNumber=1672 	FOTW 0304 Temp Save JDBC Datasource	Min=1 Max=10 Conn Timeout=1800 Aged Timeout=0 Reap time=180
FOTWPin	jdbc/FOTWPin	<ul style="list-style-type: none"> • URL=jdbc:oracle:thin:@4.20.14.15:1675:EACSTG • dataSourceName= FOTWPin • databaseName= EACSTG • enableMultithreadedAccessDetection=false • portNumber=1675 	PIN site Oracle JDBC Datasource	Min=0 Max=10 Conn Timeout=1800 Aged Timeout=1800 Reap time=180
FOTWTempAppDS	jdbc/FOTWTempAppDS	<ul style="list-style-type: none"> • URL=jdbc:oracle:thin:@4.20.14.15:1672:FAFSASTG • dataSourceName=FOTW34TempAppDS • databaseName=FAFSASTG • enableMultithreadedAccessDetection=false • portNumber=1672 	FOTW 0203 Temp Save JDBC Datasource	Min=1 Max=20 Conn Timeout=1800 Aged Timeout=1800 Reap time=180
FOTWSessions	Jdbc/FOTWSessions	<ul style="list-style-type: none"> • URL=jdbc:oracle:thin:@4.20.14.15:1676:SESSST2 • dataSourceName=FOTW34TempAppDS • databaseName=SESSST2 • enableMultithreadedAccessDetection=false • portNumber=1676 	FOTW Sessions DataBase	Min=1 Max=20 Conn Timeout=1800 Aged Timeout=1800 Reap time=180

The data source “Connect Info” should be used in production.

4.5.3 Shadow Direct JDBC

For ShadowDirect JDBC Driver, create the following data sources (V5). Please note that these are the custom properties for the Performance Test . The custom properties would be different for production.

Name	JNDI Name	Custom Properties	Description
FOTW12DataSource	jdbc/FOTW12DataSource	<ul style="list-style-type: none"> • dataSourceName=FOTWMF12D • enable2Phase=false 	0102 FOTW DB2 datasource (used by 0203 servlets)
FOTW34DataSource	jdbc/FOTW34DataSource	<ul style="list-style-type: none"> • dataSourceName=FOTWMF34D • enable2Phase=false 	FOTW 0304 DB2 DataSource
FOTWDataSource	jdbc/FOTWDataSource	<ul style="list-style-type: none"> • dataSourceName=FOTWMF23D • enable2Phase=false 	0203 FOTW DB2 datasource
T4WANDDataSource	jdbc/T4WANDDataSource	<ul style="list-style-type: none"> • dataSourceName= T4WANT • enable2Phase=false 	T4Wan DB2 DataSource for FAA Access



5 FAFSA Recommended Production Architecture

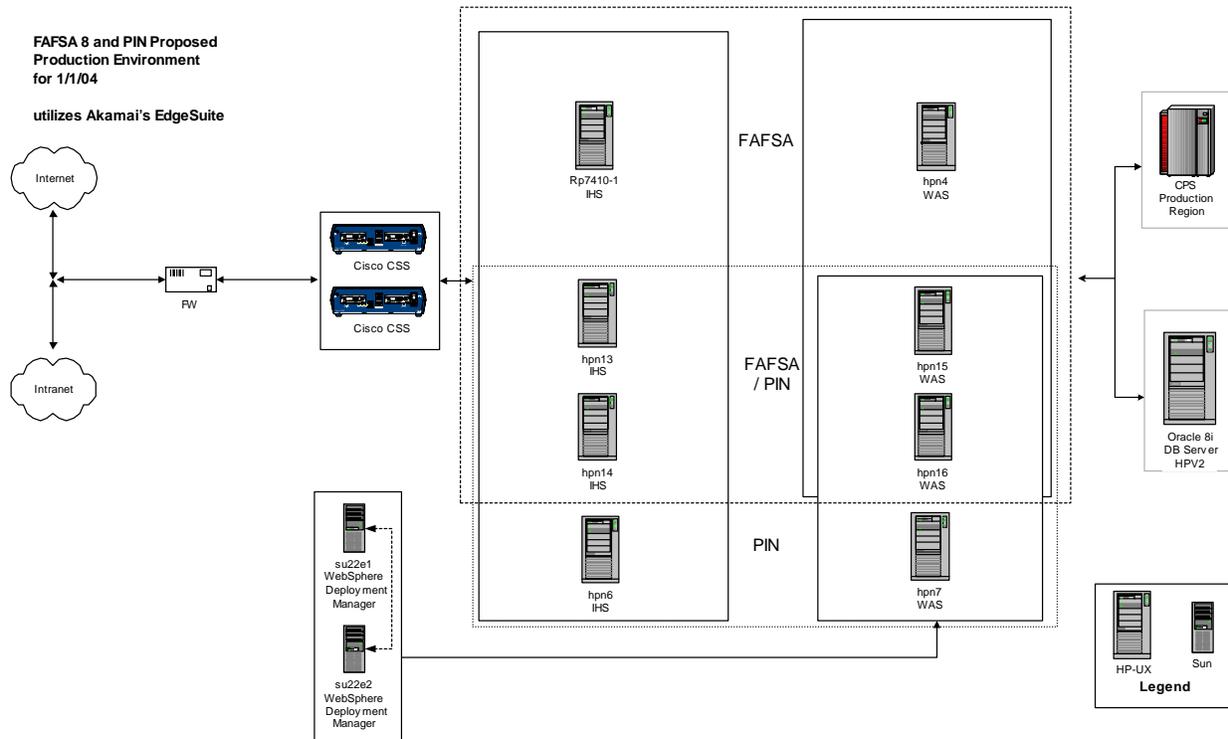
5.1 Performance Test Recommendation for FAFSA Peak

- 11 machines are needed for FAFSA 8.0 and PIN Peak
 - 4 machines are shared – FAFSA / ED PIN
 - 2 machines dedicated for ED PIN
 - 5 machines dedicated for FAFSA

Application	IBM HTTP Server	WebSphere Application Server
FAFSA	HPN? RP7410-1	HPN4 HPN? HPN?
FAFSA & PIN	HPN13 HPN14	HPN15 HPN16
PIN	HPN6	HPN7



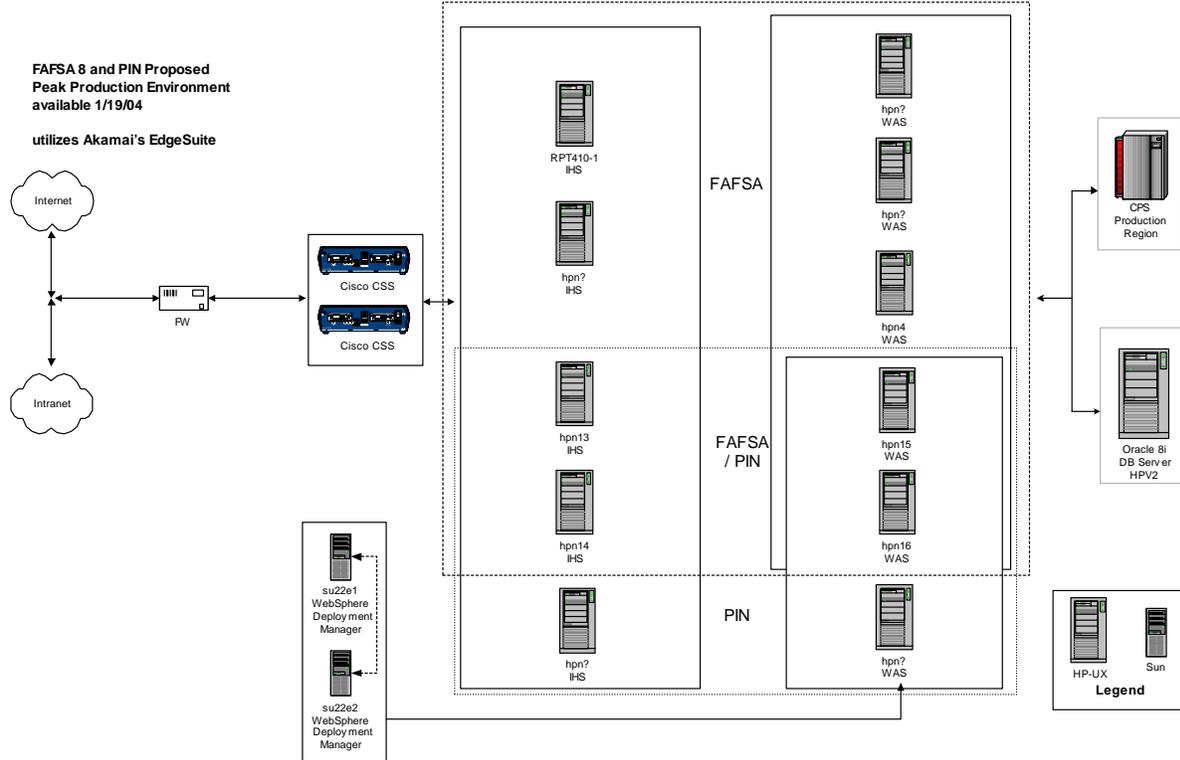
5.2 Proposed FAFSA 8.0 Production Environment – Go Live



- WebSphere Application Server – HPN15
 - 2 clones – FAFSA
 - 1 clone – FAFSA Demo
 - 1 clone – PIN
- WebSphere Application Server – HPN16
 - 2 clones – FAFSA
 - 1 clone – FAFSA Demo
 - 1 clone – PIN
- WebSphere Application Server – HPN7
 - 2 clones – PIN
- WebSphere Application Server – HPN4
 - 2 clones – FAFSA



5.3 Proposed FAFSA 8.0 Peak Environment



- WebSphere Application Server – HPN15
 - 2 clones – FAFSA
 - 1 clone – FAFSA Demo
 - 1 clone – PIN
- WebSphere Application Server – HPN16
 - 2 clones – FAFSA
 - 1 clone – FAFSA Demo
 - 1 clone – PIN
- WebSphere Application Server – HPN7
 - 2 clones – PIN
- WebSphere Application Server – HPN4
 - 2 clones – FAFSA
- WebSphere Application Server – HPN?
 - 2 clones – FAFSA
- WebSphere Application Server – HPN?
 - 2 clones – FAFSA