OS/390 e-business Infrastructure: IBM WebSphere Application Server 1.2

Customization and Usage

- Configure, set up, maintain, troubleshoot the Application Server
- Enable DB2, CICS, IMS, MQ, Oracle databases
- Sample programs and servlets included

Roland Trauner
Denis Gaebler
Bruce Smith

Redbooks

ibm.com/redbooks

SG24-5604-01

International Technical Support Organization

OS/390 e-business Infrastructure: IBM Websphere Application Server 1.2 Customization and Usage

July 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix E, “Special notices” on page 149.

Second Edition (July 2000)

This edition applies to IBM WebSphere Application Server 1.2 for OS/390.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Contents

Figures

Tables

Preface

This redbook will help you understand, configure, and use the IBM WebSphere Application Server 1.2 for OS/390. The material for the book was developed using OS/390 R8 and the IBM HTTP Server 5.2 for OS/390.

This book is for webmasters and system programmers who install or customize the IBM WebSphere Application Server for OS/390. We describe how to configure the Application Server and enable subsystems to use it. We also provide sample programs and servlets, and give information on more advanced WebAS (WAS) configuration and performance tuning.

In Appendix C, we include a relevant portion of the softcopy document WebSphere Application Planning, Installing and Using Version 1.2, GC34-4757 to help you perform both basic and advanced servlet administration tasks.

This redbook should be used in conjunction with OS/390 e-business Infrastructure: IBM HTTP Server 5.1 - Customization and Usage, SG24-5603. It is an update to OS/390 e-business Infrastructure: IBM WebSphere Application Server 1.1 - Customization and Usage, SG24-5604 (level 00).

In addition to the material in this book, we provide samples and other material on the Redbooks home page at:

`http://www.redbooks.ibm.com`

You may access the FTP server directly at:

`ftp://www.redbooks.ibm.com/redbooks/SG245604/`

We update the samples there as necessary.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Poughkeepsie Center.

Roland Trauner is an IT consultant and professional expert at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of OS/390 e-business and network computing. Before joining the ITSO in 1998, Roland worked on the IBM S/390 technical support team in Germany for IBM EMEA Central Region. Roland joined IBM in 1979. If you have questions or

comments regarding this book, feel free to contact him by e-mail at trauner@us.ibm.com.

Denis Gaebler is an IT specialist at IBM Germany. He holds a degree in business with special field computer science from the Berufsakademie Dresden. He has two years experience in IMS/ESA and its connectors. During his study he worked extensively with Net.Commerce and Internet technologies. If you have questions regarding this book or IMS and Internet connectivity on OS/390 and cross platforms, feel free to contact him by e-mail at gaebler@de.ibm.com.

Bruce Smith is an IT Specialist at IBM UK. He has been an OS/390 systems programmer for the last ten years, since graduating with a BEng in Engineering Electronics from The University of Warwick. If you have questions regarding this book, or Internet connectivity on OS/390, feel free to contact him via e-mail at smithbj@uk.ibm.com.

Thanks to the following people for their contributions to this project:

Rich Conway
International Technical Support Organization, Poughkeepsie Center, NY

Alex Louwe-Kooijmans
International Technical Support Organization, Poughkeepsie Center, NY

Viviane Anavi-Chaput
International Technical Support Organization, Poughkeepsie Center, NY

Rick Long
International Technical Support Organization, San Jose Center, CA

Ida Strickland-Wood
IBM Research Triangle Park, NC

Curt Cotner
IBM Santa Teresa Laboratory, San Jose, CA

Thomas L. Ramey
IBM Santa Teresa Laboratory, San Jose, CA

The team that wrote the first edition of this redbook

Roland Trauner, Fernando N.A. Ferreira, Young Chung Park, Uwe Sager, Minesh Vallabh, Andre Watanabe

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 165 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com`
- Send your comments in an internet note to `redbook@us.ibm.com`

Chapter 1. IBM WebSphere Application Server introduction

In June 1998, IBM published the IBM WebSphere Strategy. The strategy, and the products related to it, have been enhanced since that time. The most current information about this topic can be found on the IBM Web site at:

`http://www.ibm.com/software/webservers/appserv/os390.html`

The IBM WebSphere product portfolio contains such products as Web servers, Web application servers, networking infrastructure tools, and application development tools such as Component Broker (CB).

IBM WebSphere Application Server 1.2 is a Java engine (servlet engine) that works as an application server plug-in to Domino Go Webserver (DGW) 5.0 for OS/390 R5 and R6, or to IBM HTTP Server for OS/390.

IBM WebSphere Application Server 1.2 replaces IBM WebSphere Application Server 1.1, which succeeded the Java application server plug-in ServletExpress shipped with Domino Go Webserver 5.0.

1.1 IBM WebSphere Application Server 1.2 functions

IBM WebSphere Application Server 1.2 supports Java servlets (servlet level 2.0.1) and Java Server Pages (JSP level 0.91). The Java (JDK) level to run the IBM WebSphere Application Server should be 1.1.8.

WebAS 1.2 replaces WebAS 1.1. It runs on OS/390 R5 and higher.

- For R5 and R6 it will be plugged to Domino Go Webserver 5.0.
- For R7 it will be plugged to IBM HTTP Server 5.1.
- For R8 and R9 it will be plugged to IBM HTTP Server 5.2.

IBM WebSphere Application Server 1.2 provides several new functions:

- Connection Manager support for JDBC
- New servlet management functions
- New Application Server Manager interface
- Improved property file update process
- Support for the WLM-managed (scalable) Web server
- Support for retrieving session tracking information from DB2 databases

1.2 IBM WebSphere Application Server project playground

This is a short introduction to our "playground," the system setup we used for the items we describe in the book. We include it here because you will find some names and/or settings we used mentioned in the screen shots or configuration samples.

1.2.1 Project playground description

To develop this book, we ran several WebSphere Application Server environments in parallel. We used started procedures to run the servers.

Following is the procedure to run IBM HTTP Server 5.2 for OS/390. We called it WEBAPPLE.

```
//APPLESRV PROC P1='-vv',
// P2='-r /web/apple/httpd.conf',
// P3='-p 7101',
// LEPARM='ENVAR("_CEE_ENVFILE=/web/apple/httpd.envvars")'
//*********************************************************************
//*
//WEBSRV   EXEC PGM=IMWHTTPD,REGION=0K,TIME=NOLIMIT,
// PARM=('&LEPARM/&P1 &P2 &P3')
//*PARM=('HEAPP(ON) ALL31(ON) POS(ON) STAC(200K) &LEPARM/&P1 &P2 &P3')
//*
//STEPLIB  DD DSN=DSN610.SDSNLOAD,DISP=SHR
//         DD DSN=DSN610.SDSNEXIT,DISP=SHR
//DSNAOINI DD DSN=DB2V61S3.DB2CLI.CLIINI.RRS,DISP=SHR
//*
//SYSIN    DD DUMMY
//OUTDSC   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
```

Figure 1. Sample Web server procedure

There is no deep meaning behind the names we used; we just named the servers from A to Z, and in order not to be tedious, we used names associated with food and such.

All servers ran IBM HTTP Server 5.2 for OS/390 and IBM WebSphere Application Server 1.2 on OS/390 R8.

The configuration files for the various servers were placed in HFS data sets named /web/apple/, /web/bean/, /web/candy/, and so on.

We also configured the ServerRoot and PID file directive to point to the respective directories. Refer to *OS/390 e-business Infrastructure: IBM HTTP Server 5.1 - Customization and Usage*, SG24-5603 for details.

The environments for the IBM WebSphere Application Server engines were placed in the following separate HFS data sets:

/was/apple/AppServer/
/was/bean/AppServer/
/was/candy/AppServer/

Chapter 2. IBM WebSphere Application Server configuration

This chapter guides you through the post-installation steps required to configure the IBM WebSphere Application Server (WebAS or WAS).

2.1 Prerequisites

There are a number of prerequisites to WebAS. Before installing it, you must have a working HTTP server and also have Java installed.

First complete the IBM HTTP Server installation and customization as described in *HTTP Server Planning, Installing, and Using Version 5.2,* SC31-8690. It is important with all documentation in this area to always have the latest. For the most current information and updates, go to:
`http://www.ibm.com/software/webservers/httpservers/doc52.html`

If you are installing your HTTP server from scratch, we suggest reading through 2.2.4, “Directory structure naming convention” on page 8 first, so you can plan the names you will use for your directory structure.

Note: We strongly recommend that you first customize and start the IBM HTTP Server without WebAS to make sure that you can successfully serve HTML pages before you start the WebAS customization.

For installing WebAS, and for migration considerations from previous releases of WebAS, refer to the following documents for details:

- *WebSphere Application Server for OS/390 Application Server Planning, Installing, and Using Version 1.2*, GC34-4757
- *WebSphere Application Server for OS/390 Release 2, Modification Level 0 Program Directory*, GI10-6780

For the most current information and updates, go to:

`http://www.ibm.com/software/webservers/appserv/library.html`

In order to have Java available to WebAS, you must install the Java Development Kit (JDK). At the time of writing, the latest generally available version of the Java Development Toolkit for OS/390 was JDK 1.1.8 (January 2000 update), which is downloadable from the following URL:

`http://www.ibm.com/s390/java`

In addition, this URL documents the latest prerequisites and instructions required to install and customize the JDK environment. For additional information, refer to the following redbooks:

- *Integrating Java with existing Data and Applications on OS/390*, SG24-5142
- *e-business Application Solutions Using Java: Volume I,* SG24-5342
- *e-business Application Solutions on OS/390 Using Java: Samples,* SG24-5365
- *Java Programming Guide for OS/390*, SG24-5619

2.2 Basic setup

This section gives a brief overview of the procedure to follow to get WebAS up and running in its most basic configuration. It then goes on to describe the steps of the process in enough detail so you can get the setup ready for starting your Web server with WebAS.

2.2.1 Overview of the setup process

Once you have a working Web server (HTTP Server) and Java installed, you need to create and populate the WAS (FMID HEJS120) directories as described in *OS/390 V2R8.0 Program Directory*. This set of directories is known as the *applicationserver_root* or simply the *install image*, and by default is mounted at /usr/lpp/WebSphere/AppServer. WebSphere 1.2 requires an Application server model in order to run, and you build as many of these from your single applicationserver_root as you require. An Application server model is a set of directories, similar in structure to applicationserver_root, but called *server_model_root*, containing the configuration properties of an individual Application server. To create the Application server model, you run the makeserver.sh shell script.

The makeserver.sh shell script requires three parameters to be passed to it with: the name of the server_model_root, which httpd.conf is used by the Web server associated with this Application server, and which jdk_root to use. It then creates a WebSphere configuration file, *was.conf*, and sets up Application server properties files.

The file was.conf is a documented listing of all the variables used by the shell script to build the Application server properties files. When the Application server is initialized and running, it uses the definitions in the properties files, not the ones in was.conf. This needs to be borne in mind when maintaining

the Application server. Subsequent changes to the properties files are carried out by modifying was.conf, and running the updateproperties utility.

So, in summary, you create a single applicationserver_root, by default in /usr/lpp/WebSphere/AppServer. You then build as many Application server models (IBM WebSphere Application servers) as you require by running makeserver.sh and telling it where to create server_model_root, which httpd.conf it corresponds to, and which jdk_root to use. It would be very unusual to have more than one Application server for a Web server. Conversely, installations may have a number of Web servers and associated Application servers running on a given OS/390 system.

2.2.2 Preparing the environment

1. Define the Java environment, especially the $JAVA_HOME variable, in your .profile file. You can issue the `echo $JAVA_HOME` command in the OMVS shell to verify that the variable points to the JDK directory.
2. Make sure you are running under a superuser ID or UID=0.
3. Give the superuser ID read access to the profile BPX.FILEATTR.PROGCTL in the RACF FACILITY class to enable this superuser ID to update the HFS file attributes using the `extattr` command:

```
RDEFINE FACILITY BPX.FILEATTR.PROGCTL UACC(NONE)
PERMIT BPX.FILEATTR.PROGCTL CLASS(FACILITY) ID(superuserID) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

Figure 2. extattr enablement

2.2.3 Creating applicationserver_root

The creation of applicationserver_root (the install image) is achieved by working through the installation instructions detailed in the *Program Directory for WebSphere Application Server for OS/390 Release 2, Modification Level 0*, GI10-6780, the latest version of which can be found at:

`http://www.ibm.com/software/webservers/appserv/doc/os390/v12/hejs120.htm`

It is worth going through the entire Program Directory carefully, paying particular attention to the PSP bucket. Note that the Program Directory does not mention applicationserver_root by name, but talks about “Installing the Application Server” instead.

The PSP bucket, and a lot of extra problem determination and troubleshooting information can be found in *WebSphere Troubleshooter for OS/390* at the URL

http://www.ibm.com/software/webservers/appserv/troubleshooter.html

2.2.4 Directory structure naming convention

The directory structure that you choose to implement governs the future ease of maintenance of your Application server.

We suggest having a directory structure for your Application server similarly named as the one for your Web server. Although you may initially have a single Web server and Application server, you would be wise to use a naming convention for your directories that allows for painless expansion.

During our work on WebSphere, we were running a number of Web servers and Application servers concurrently on the same OS/390 system, and we adhered to the convention outlined in Table 1.

Table 1. Directory structure names

Directory content	Directory name
First HTTP server	/web/<server_name_1>/
*n*th HTTP server	/web/<server_name_*n*>/
Applicationserver_root	/usr/lpp/WebSphere/AppServer
First server_model_root	/was/<app_server_1>/AppServer/
*n*th server_model_root	/was/<app_server_*n*>/AppServer/

Note: The text enclosed in angled brackets is for you to decide upon. So, /web/<server_name_1>/ could be /web/serverA/. For ease of typing, you may want to keep names as short as possible. You could replace AppServer with A, for instance. The drawback of shortening the names is a reduction in the clarity of the directories' contents.

It makes sense, for ease of problem diagnostics and future maintenance, to keep <server_name_*n*> and <app_server_*n*> the same for a given Web server and Application server pair. Note also that you should only perform mkdir's as far as /was/<app_server_n>/, as the makeserver shell script will only write to directories that do not already exist.

2.2.5 Building the Application Server model (server_model_root)

As outlined in 2.2, "Basic setup" on page 6, you need to create an Application server model (server_model_root) for each Application server. This is done by running the makeserver.sh script.

When you run makeserver.sh, you must specify three parameters: the server_model_root you want to create, the httpd.conf that applies to the Application server, and the java_root that the Application server will use. You also need to make sure that you are running as superuser when you issue the command.

The syntax of the command is:

```
makeserver.sh <server_model_root> <httpd.conf> <java_root>
```

When you run the makeserver.sh script, you get some brief details about the parameters used for the server_model_root you are building, and an indication of whether your command executed successfully.

```
ITSO3B:/usr/lpp/WebSphere/AppServer/config: >makeserver.sh /was3/itso3b/AppServe
r /web3/itso3b/httpd.conf /usr/lpp/java18p/J1.1

Start - OS/390 WebSphereAS 1.2 makeserver: Tuesday 03/14/00 03:16:16 PM

Starting utility 'updateproperties' ....   - 15:16:55 on 03-14-2000
App Server Root Directory : /was3/itso3b/AppServer/
App Server Log File        : /was3/itso3b/AppServer/logs/updateproperties.log
Properties in all Files successfully updated - 15:16:55 on 03-14-2000

Ended Successfully - OS/390 WebSphereAS 1.2 makeserver: Tuesday 03/14/00 03:16:5
6 PM
ITSO3B:/usr/lpp/WebSphere/AppServer/config:
```

Figure 3. makeserver.sh invocation and output

You will notice that makeserver.sh also runs updateproperties, which takes the contents of was.conf created by makeserver.sh, and creates the configuration files that the Application server runs with. This is discussed in more detail in 2.4, “Application Server maintenance” on page 12.

The makeserver script creates the initial WebSphere configuration file was.conf, which can be found in server_model_root/properties/was.conf.

2.2.6 Updating httpd.conf

Once you have created your server_model_root, you need to make a number of changes to your Web server’s httpd.conf in order to use it, as shown in Figure 4 on page 10. All references to our sample server_model_root /was3/itso3b/AppServer will need to be modified to the name of your server_model_root.

```
#   ================================
#   *** CAServlet & WAS directives ***
#   ================================

ServerInit    /was3/itso3b/AppServer/lib/libadpter.so:AdapterInit 1
/was3/itso3b/AppServer
ServerTerm    /was3/itso3b/AppServer/lib/libadpter.so:AdapterExit

Service /servlet/* /was3/itso3b/AppServer/lib/libadpter.so:AdapterService
Service /*.jsp     /was3/itso3b/AppServer/lib/libadpter.so:AdapterService
Service /*.jhtml   /was3/itso3b/AppServer/lib/libadpter.so:AdapterService
Service /*.shtml   /was3/itso3b/AppServer/lib/libadpter.so:AdapterService
```

Figure 4. Required updates to httpd.conf

Note:

1 This line has been split for redbook printing purposes. In the actual file, it is on a single line.

You also need to add some Pass statements in order for requests for html files associated with servlets to search the appropriate directories, something similar to the following (which we use for our examples):

```
Pass    /IBMWebAS/samples/*        /was3/itso3b/AppServer/samples/*
Pass    /IBMWebAS/*                /was3/itso3b/AppServer/web/*
```

2.2.7 Turning on the program control flags

The program control flags need to be turned on for a number of modules if you are using UNIX System Services level security, that is, the RACF BPX.DAEMON facility class is active.

WebAS modules

The program control flags need to be set for the following WebAS modules:

/applicationserver_root/lib/libadpter.so
/applicationserver_root/lib/libicsnativ.so

Go to the OMVS shell using the superuser ID and do the following:

```
cd /applicationserver_root/lib
extattr +p *.so
ls -E

The result follows:

-rwxr-x--x  -ps  2 WEBADM   IMWEB     221184 May 14 09:55 libadpter.so
-rwxr-x--x  -ps  2 WEBADM   IMWEB     114688 May 14 09:55 libicsnativ.so
```

Figure 5. Turning on program control for WebAS modules

Alternatively, you can change the program control flags using the ISPF shell (ISHELL) using Option **a**, choose **Edit**, then **Extended attributes**, and set the program control flag to 1 for each module.

Note that because these modules reside in applicationserver_root, you need only perform this once, and not once for each server_model_root.

JDK DLLs

WebAS depends on a proper Java installation. If you install the JDK, retain a “proper” environment for the Application server by setting the extended program control flags for the modules in:

/usr/lpp/java18p/J1.1/lib/mvs/native_threads

Go to the OMVS shell and do the following:

```
cd /usr/lpp/java18p/J1.1/lib/mvs/native_threads
extattr +p *.*

You can check the attributes using:
ls -E
```

Figure 6. Turning on program control for Java modules

2.3 IBM WebSphere Application Server startup

Once you have completed the entire process outlined in 2.2, “Basic setup” on page 6, you can start your Web server, which will automatically start the Application server plug-in. The startup will take longer because the WebAS plug-in needs more system resources. The messages shown in Figure 7 on page 12 confirm that the server has initialized successfully and can process Internet requests.

```
T$HASP373 WEB3B    STARTED
IMW3534I PID: 83886231 SERVER STARTING
IMW3536I SA 83886231 0.0.0.0:7102 * * READY
```

Figure 7. HTTP server startup messages written to OS/390 SYSLOG

You should confirm that both the Web server and Application server are started by looking at the output written to the SYSPRINT DD for a message containing the *smiley face*, like the following:

```
IMW0234I Starting.. httpd
........... This is IBM WebSphere Application Server V1R2M3
........... Built on Oct 18 1999 at 08:05:49.
........... Started at Thu Mar 16 11:55:28 2000
........... IBM WebSphere Application Server is ready :-)
IMW0235I Server is ready.
```

Figure 8. Web server startup messages with successful instantiation of WebAS

A single sample servlet (shown in Table 2) is shipped with WebAS and automatically loaded at Web server startup. Try to execute it, to verify that everything is configured correctly. To execute the servlet, you just need to open its URL.

Table 2. Sample servlets shipped with WebAS

Servlet	URL to open	Expected result
Hello	http://*your.server.name*/servlet/HelloWorldServlet	Displays the string, `Hello World`.

For additional samples, refer to Chapter 4, “How to run the samples” on page 31.

2.4 Application Server maintenance

Modifications to the Application server’s properties are done by editing was.conf, and then running the updateproperties utility. You must run updateproperties from applicationserver_root/config (usually /usr/lpp/WebSphere/AppServer/config). This should only be done when the Web server is down.

When you run updateproperties you get a brief response indicating the result of your action:

```
ITSO3B:/usr/lpp/WebSphere/AppServer/config: >./updateproperties /was3/itso3b/App
Server
Starting utility 'updateproperties' ....   - 13:40:54 on 03-17-2000
App Server Root Directory : /was3/itso3b/AppServer/
App Server Log File        : /was3/itso3b/AppServer/logs/updateproperties.log
Properties in all Files successfully updated - 13:40:55 on 03-17-2000
ITSO3B:/usr/lpp/WebSphere/AppServer/config: >
```

Figure 9. Updateproperties invocation and output

You will notice that an updateproperties.log file is created, in sever_model_root/logs/updateproperties.log, which details all of the parameters processed from was.conf in the latest run of updateproperties.

2.5 Multiple IBM WebSphere Application Servers setup

With WebAS 1.1 it was a cumbersome process to install multiple Application server instances. You needed to make clones of your server_model_root and make changes to a number of hard-coded parameters. Now you just use makeserver.sh to create as many instances as you need.

We suggest that you adhere to a directory naming scheme similar to that in 2.2.4, “Directory structure naming convention” on page 8.

2.6 Troubleshooting

In this section we identify some common installation problems that you may encounter. Some of these were drawn from experiences with WebAS 1.1, but have been retained in this section as they might still give pointers to possible problem resolutions.

More hints and tips on how to configure WebAS can be found in *WebSphere Troubleshooter for OS/390* at the URL:

`http://www.ibm.com/software/webservers/appserv/troubleshooter.html`

Enable native DLL plug-in logging and JVM logging for diagnostic messages generated by WebAS by setting the appropriate parameters in was.conf similar to those in the parameter file fragment shown in Figure 10 on page 14, and running updateproperties. The paths for the log files will be /server_model_root/logs/ by default.

```
ncf.native.logison=true
ncf.native.logfile=/was3/itso3b/AppServer/logs/native.log
ncf.jvm.stdoutlog.enabled=false
ncf.jvm.stdoutlog.file=true
ncf.jvm.stdoutlog.filename=/was3/itso3b/AppServer/logs/ncf.log
```

Figure 10. was.conf parameters for enabling diagnostic message logging

Here are some things to analyze:

2.6.1 Application Server does not initialize

See ncf_native.log:

```
StoreTranslationinVM done
StoreTranslationinVM done
StoreTranslationinVM done
StoreTranslationinVM done
After InitDirectoryTranslationBefore CallNCFInit /web/egg/h1
```

Figure 11. Application Server failing to initialize

Possible causes:

- The server.properties file does not have the servletservice option in the autostart options:

  `/was/egg/properties/server/servlet/server.properties`
  `server.service.autostart=adminservice` **`servletservice`**

- Missing or incorrect JVM install library.
- Missing or incorrect IBM WebSphere Application Server install library.

2.6.2 Clients may receive ERROR 500 with message

The message is:

```
IMW0241E Access denied - surrogate user setup error.
```

Review the IBM HTTP Server trace log to determine if the following messages were issued for the particular client request:

```
Failed access as Surrogate: <surrogate ID>, Errno: 139,

Errno2: 0be802af, Error: EDC5139I Operation not permitted.

IMW0241E Access denied - surrogateuser setup error.
```

or

```
Failed access as Surrogate: <surrogate ID>, Errno: 139,
```

```
Errno2: 090c02af, Error: EDC5139I Operation not permitted.

IMW0241E Access denied - surrogateuser setup error.
```

If these messages are generated for the failing client request, you must turn on program control. See 2.2.7, “Turning on the program control flags” on page 10 for instructions.

Error 500 also occurs when calling a servlet and the Service handler is not activated. This is often caused by typos in the ServerInit, ServerTerm and Service configuration directives in httpd.conf. An indication of this case is if you turned on the Application server logs but nothing is written to them.

Another Error 500 condition is if there are typos in the setup for the ncf.jvm.classpath property in was.conf. Turn on the Application server logs and verify the initialization messages in ncf.jvm.log. At the beginning of this log file, the WebSphere Application Server shows a resolution of the classpath and libpath used.

2.6.3 Forever sleeping JVM

If you customized the IBM WebSphere Application Server, it sometimes never initializes. There are also cases where it initialized at first, but when additional support, like the JDBC support, was added, the problem occurred.

The following is an example of the NCF native log file somebody sent the team that wrote the WebAS 1.1 book. You can see that it is not one of our examples because it does not follow our file setup rules.

```
Opened log file /usr/lpp/WebSphere/AppServer/logs/native.log.

Try to create a rule base out of
/usr/lpp/WebSphere/AppServer/properties/server/servlet/servletservice/rules.
properties
Rule base was created

/servlet=invoker

*.jsp=pageCompile

*.jhtml=pageCompile

*.shtml=pageCompile

PATH=/usr/lpp/java/J1.1/bin:/bin:.:/usr/sbin:/usr/lpp/internet/bin:/usr/lpp/
internet/sbin:/usr/lpp/ldap/bin:/usr/lpp/java/J1.1/bin
LIBPATH=/usr/lpp/java/J1.1/lib:/usr/lpp/java/J1.1/lib/mvs/native_threads:/us
r/lpp/WebSphere/AppServer/lib:/usr/lib:/usr/lpp/internet
Attempting to load Java library: libjava.a

Loaded libjava.a and function pointers successfully.

ncf.jvm.classpath =
```

```
/usr/lpp/WebSphere/AppServer/lib/ibmwebas.jar:/usr/lpp/WebSphere/AppServer/lib/jst.j
ar:/usr/lpp/WebSphere/AppServer/..........
Opened java VM, classpath =
/usr/lpp/WebSphere/AppServer/lib/ibmwebas.jar:/usr/lpp/WebSphere/AppServer/lib/jst.j
ar:/usr/lpp/WebSphere/Appserver/..........
createJVM: About create thread for JVM...

createJVM: pthread_create...errno = 132

createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
createJVM: Sleeping..Zzzzz
.........
```

In most of these cases, the problem is solved by adjusting the storage values for the Web server. See 6.1.8, “Language Environment (LE)” on page 91.

Chapter 3. Enable subsystems for the Application Server

This chapter discusses the customization needed to enable communication between the subsystems and IBM WebSphere Application Server.

See the following sections for descriptions to enable specific subsystems:

- 3.1, “Enable DB2 for the IBM WebSphere Application Server” on page 17
- 3.2, “Enable CICS for the IBM WebSphere Application Server” on page 24
- 3.3, “Enable IMS for the IBM WebSphere Application Server” on page 25
- 3.4, “Enable MQSeries for the IBM WebSphere Application Server” on page 27
- 3.5, “Enable Oracle for the IBM WebSphere Application Server” on page 28

For information about DB2, IMS, MQSeries and CICS installation and customization refer to the following documentation:

- For CICS, *Installation Guide,* GC33-1681*, CICS Customization Guide,* SC33-1683*, Planning for Installation,* GC33-1789 and *CICS Program Directory.*
- For DB2, *UDB for OS/390 V6 Installation Guide,* GC26-9008*,* and *Program Directory for DB2 for OS/390,* GT02-0571*.*
- For IMS, *IMS/ESA V6 Installation Volume I: Installation and Verification,* GC26-8736*, IMS/ESA V6 Installation Volume II: System Definition and Tailoring,* GC26-8737 and the *IMS/ESA Release Planning Guide V6,* GC26-8744*.*
- For MQSeries, *MQSeries Planning Guide,* GC33-1349*,* and *MQSeries for OS/390 V2 Release 1 Program Directory,* GI10-2501*.*

3.1 Enable DB2 for the IBM WebSphere Application Server

In order for the Web servers and their Java applications to have access to DB2, you need JDBC and/or SQLJ support on OS/390.

For JDBC and SQLJ support, you need DB2 for OS/390 V5.1 or higher. Always apply the latest maintenance level available.

We discuss only the JDBC implementation here. For more information about SQLJ, see *UDB for OS/390 V6 Application Programming Guide and Reference for Java,* SC26-9018, and the URL:

http://www.ibm.com/software/data/db2/os390/sqlj.html

3.1.1 IBM WebSphere Application Server settings for JDBC

In the file was.conf, do the following:

- Add to ncf.jvm.classpath[1]:
  /usr/lpp/db2/db2610/classes/db2jdbcclasses.zip
- Add to ncf.jvm.libpath:
  /usr/lpp/db2/db2610/lib
- In the IBM HTTP Server 5.2 for OS/390 startup procedure add the steplib:
  //DSNAOINI DD DSN=DB2V610.DB2CLI.CLIINI,DISP=SHR

  Note: You also need a steplib to SDSNLOAD and SDSNEXIT if they are not already in the linklist (usually they are). The steplib can be specified either in the Web server's started task procedure or in the httpd.envvars file.

  For more information about the CLI initialization data set, see 3.1.2.2, "DB2 Call Level Interface (CLI) initialization file" on page 19.

3.1.2 DB2 JDBC support enablement

- The following DB2 components must be installed and enabled:
  - DB2 DRDA with DDF
  - DB2 Call Level Interface (CLI)
  - DB2 JDBC Driver

    For information about DRDA, CLI and JDBC Driver installation refer to *DB2 for OS/390 V6 Application Programming Guide and Reference for Java(TM),* SC26-9018 and *Program Directory for DB2 for OS/390,* GT02-0571.
- GRANT the servlet owners to the DSNACLI plan (see 3.1.2.2, "DB2 Call Level Interface (CLI) initialization file" on page 19 for the DSNACLI name).

  Note: If you rebind plan DSNACLI at DB2, you have to redo your GRANTs to this plan.
- Make sure your servlets refer to the right DB2 subprotocol, DB2 Driver name, and the location name. See an example in 4.5.1, "JDBCServlet" on page 44.

  The JDBC driver name for DB2 for OS/390 is *ibm.sql.DB2Driver*.

  The DB2 subprotocol for OS/390 is *db2os390*.

[1] We put that path information right before the Java classes.zip file which is the last entry in the classpath.

For information about the location name see 3.1.2.3, “DB2 location-name” on page 21.

3.1.2.1 JDBC overview

JDBC is a Java application programming interface (API) that Java applications use to access any relational database. DB2 for OS/390 support for JDBC enables you to write Java applications that access local DB2 data or remote relational data on a server that supports DRDA.

To enable JDBC for OS/390, a Java Development Kit (JDK) for OS/390 is required. We suggest that you use the most current version (we used 1.1.8). The contents of the JDK include a Java compiler, Java Virtual Machine (JVM), and Java Debugger. You can learn more about the JDK and obtain the code from the Java for OS/390 Web site at:

`http://www.ibm.com/s390/java`

A Java application executes under the JVM. The Java application first loads the JDBC driver, in this case the DB2 for OS/390 JDBC driver, and subsequently connects to the local DB2 subsystem or a remote DRDA application server by invoking the DriverManager.getConnection method.

3.1.2.2 DB2 Call Level Interface (CLI) initialization file

A set of optional keywords can be specified in a DB2 CLI initialization file, an EBCDIC file that stores default values for various DB2 CLI configuration options. Because the initialization file has EBCDIC text, it can be updated using a file editor, such as the TSO editor.

The DB2 CLI initialization file is read at application runtime. The file can be specified by either a *DSNAOINI* DD card or by defining a *DSNAOINI* UNIX System Services environmental variable. The initialization file specified can be either a tradition MVS data set or a UNIX HFS file.

For MVS data sets, the record format of the initialization file can be either fixed or variable length. See the following example of a CLI ini file.

```
; This is a comment line...
; Example COMMON stanza
[COMMON]
CONNECTTYPE=1
; STMTS added for WEBAS samples
MULTICONTEXT=1
MAXCONN=0
MVSDEFAULTSSID=DBS3
; TRACE is for IBM DEBUGGING
; CLITRACE is for application debugging
;be sure trace is not started before the ini is read, else this is
;  missed
;be sure that you put this ini out just before the job you want to
; trace, that is, if you have a setup insert job. run that before.
; turn diagnosis trace on and increase trace buffer size
TRACE=0
TRACE_NO_WRAP=1
TRACE_BUFFER_SIZE=2000000
; turn user application trace on and direct to DD name CLITRACE
CLITRACE=0
TRACEFILENAME=DD:CLITRACE

; Example SUBSYSTEM stanza for DBS3 subsystem
[DBS3]
MVSATTACHTYPE=RRSAF
PLANNAME=DSNACLI

; Example DATA SOURCE stanza for data source
[DSGC]
AUTOCOMMIT=1
```

Figure 12. CLI INI sample file

MVSDEFAULTSSID = ssid

The MVSDEFAULTSSID keyword specifies the default DB2 subsystem to which the application is connected when invoking the SQLAllocEnv function. You must specify a 4-character name of an installed DB2 subsystem.

MVSATTACHTYPE = CAF / RRSAF

The MVSATTACHTYPE keyword is used to specify the DB2 for OS/390 attachment type that DB2 CLI uses to connect to the DB2 for OS/390 address space. This parameter is ignored if the DB2 CLI application is running as a DB2 for OS/390 stored procedure. In that case, DB2 CLI uses the attachment type that was defined for the stored procedure.

CAF: DB2 CLI uses the DB2 for OS/390 call attachment facility (CAF).

RRSAF: DB2 CLI uses the DB2 for OS/390 Recoverable Resource Manager Services attachment facility (RRSAF).

In our testing of the IBM WebSphere Application Server servlets samples, we used RRSAF. That is a prerequisite for the session manager facility of IBM WebSphere Application Server, that some of the samples are used.

PLANNAME = planname
The PLANNAME keyword specifies the name of the DB2 for OS/390 PLAN that was created during installation. A PLAN name is required when initializing the application connection to the DB2 for OS/390 subsystem, which occurs during the processing of the SQLAllocEnv call.

If no PLANNAME is specified, the default value DSNACLI is used.

3.1.2.3 DB2 location-name
The Java application identifies the target data source it wants to connect to by passing a database Uniform Resource Locator (URL) to the DriverManager.

The URL values for a DB2 for OS/390 data source are specified as follows:

jdbc:db2os390:<location>

Note: db2os390 is the subprotocol name for DB2 for OS/390.

If location-name is not the local site, it must be defined in SYSIBM.LOCATIONS. If location-name is the local site, it must have been specified in the field DB2 LOCATION NAME of the DISTRIBUTED DATA FACILITY panel during DB2 installation. In our case the subsystem ID was used.

To see the location name at your DB2 subsystem, open the DB2 Master address space sysout. Find "DDF". Look at the keyword "Location". It will be displayed during DB2 startup.

In the following example, the location name is DBS3.

```
  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY DB61MSTR STC01882  DSID     2 LINE  CHARS 'DDF' FOUND
 COMMAND INPUT ===>                                          SCROLL ===> CSR
12.22.22 STC01882  DSNL003I @ DDF IS STARTING
12.22.25 STC01882  DSNL519I @ DSNLILNR TCP/IP SERVICES AVAILABLE
                              FOR DOMAIN wtsc58oe AND PORT 33380
12.22.25 STC01882  DSNL519I @ DSNLIRSY TCP/IP SERVICES AVAILABLE
                              FOR DOMAIN wtsc58oe AND PORT 33381
12.22.25 STC01882  DSNL004I @ DDF START COMPLETE
                              LOCATION  DBS3
                              LU        USIBMSC.SC58DB2
                              GENERICLU -NONE
                              DOMAIN    wtsc58oe
                              TCPPORT   33380
                              RESPORT   33381
12.22.25 STC01882  DSN9022I @ DSNYASCP 'STA DB2' NORMAL COMPLETION
```

3.1.3 IBM WebSphere Application Server settings for SQLJ

SQLJ is the JAVA equivalent of embedded SQL. It performs better than JDBC but requires additional maintenance. To add the SQLJ support in IBM WebSphere Application Server, do the following:

1. In the file *was.conf*:
   Add to ncf.jvm.classpath:
   `/usr/lpp/db2/db2610/classes/db2sqljruntime.zip`
2. In the file *httpd.envvars:*
   Add the following environment variables:

   DB2 SQLJ subsystem ID: `DB2SQLJSSID=DBS3`
   DB2 SQLJ plan name: `DB2SQLJPLANNAME=SQLJSAMP`
   DB2 SQLJ attach type: `DB2SQLJATTACHTYPE=RRSAF`

   There are more settings available, depending on your environment's needs.

   Alternatively, you might specify the `DB2SQLJPROPERTIES` environment variable, which points to a file containing all the necessary SQLJ settings. By default it points to *db2sqljjdbc.properties* in the present working directory. Here is an example of such a profile:

```
DB2SQLJSSID=DBS3
DB2SQLJ_TRACE_FILENAME=/ITSO/db2/jdbctrace
DB2SQLJPLANNAME=DSNJDBC
DB2SQLJMULTICONTEXT=yes
DB2SQLJATTACHTYPE=RRSAF
DB2SQLJDBRMLIB="//USER.DBRMLIB.DATA"
```

We used RRS as attachtype. The SQLJPLAN needs to be created by the application programmer and bound to DB2. On how to prepare a servlet using SQLJ in WebSphere, refer to 4.5.6, “SQLJ servlet sample” on page 52. You can specify only one SQLJPLAN for your or the Web server’s UNIX System Services environment variable. Make sure that all DBRMs from servlets using SQLJ are bound in that single plan. We recommend binding the DBRM member belonging to one servlet into one package and binding all the servlet packages to one plan. This will make the maintenance of changes easier. While creating that plan, don’t forget to include the default package DSNJDBC.* in that plan. Make sure that the Web server user ID has execute rights granted on the SQLJPLAN.

A more detailed description is in Chapter 20 of *JAVA Programming Guide for OS/390,* SG24-5619.

Also refer to *DB2 for OS/390 V6 Application Programming Guide and Reference for Java(TM),* SC26-9018, and the IBM DB2 SQLJ Web page at:

`http://www.ibm.com/software/data/db2/java/sqlj`

3.1.4 Settings for the new Type 2 SQLJ/JDBC Driver

If you plan to use JDBC and SQLJ concurrently from different servlets, then make sure that you have applied APARFIX CQ19814 (V5) or APARFIX AQ36011 (V6) and installed and customized the new Type 2 Driver. The old driver delivered with DB2 uses different connect types for JDBC and SQLJ resulting in SQLSTATE=-808 and making the environment virtually unusable.

The new driver only works with the db2sqljjdbc.properties file and the DB2SQLJPROPERTIES environment variable specified in your httpd.envvars file as described in 3.1.3, “IBM WebSphere Application Server settings for SQLJ” on page 22. You need to have the serialized profile created by running the db2genJDBC utility (ask your DBA; it is run once during driver install) in your Application Server classpath as well. Only one plan for both SQLJ and JDBC is possible (DB2SQLJPLANNAME). Therefore, you need to include the default JDBC packages and the SQLJ packages for your servlets in one plan. For more details, refer to the README file delivered with the new driver.

Note: The old driver delivered with DB2 was not JDBC-certified! This may mean that the new driver may in some cases exhibit behavior different from the old one.

After applying the APARFIX, the names for the new combined JDBC and SQLJ driver are as follows (used in the class.for.name method):

For JDBC: ibm.sql.DB2Driver

For SQLJ: COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver

We found only this scenario working in our environment, although the documentation states that the SQLJ driver name could be used for JDBC as well. Make sure that your applications comply with JDBC specification Version 1.0, or you might get strange results. For example: We experienced problems when using the executeQuery() method for updates and inserts with the new driver, which worked with the old driver.

The driver distinguishes between SQLJ and DB2 using the subprotocol name, which is either db2os390 or db2os390sqlj. The URLs for accessing DB2 are as follows:

– JDBC: `jdbc:db2os390:<location>`

– SQLJ: `jdbc:db2os390sqlj:<location>`

3.2 Enable CICS for the IBM WebSphere Application Server

The Java interface into CICS used to be via the IBM CICS Gateway for Java (MVS). With the CICS Transaction Server, this is now the CICS Transaction Gateway for OS/390 (CTG). While working on CICS for this book, we used CICS Transaction Server 1.3.0.

Before you begin interfacing your Application server to CICS, you must have your CICS transaction server working properly, and also have EXCI correctly configured. Refer to *CICS External Interfaces Guide*, SC33-1944 for details.

More information about the CICS Gateway and Java can be found at:

`http://www.ibm.com/software/ts/cics/about/modern/cicsjava.html`

To download the CTG code, go to:

`http://www.ibm.com/software/ts/cics/download`

When we downloaded the code, we received a tar file called ctg-310m.tar. We unpacked this file into /usr/lpp/cicsts . It created a subdirectory called ctg. The install path for our CTG is now /usr/lpp/cicsts/ctg .

We used a pax command to unpack the file:

```
pax -rv < ctg-310m.tar
```

3.2.1 Preparing the Application Server for CICS TS

A number of changes need to be made to the Application server, the Web server, and CICS in order to get Java servlets talking to the CICS transaction server:

1. In was.conf:
   - Add the following jar files to ncf.jvm.classpath:

     /usr/lpp/cicsts/ctg/classes/ctgclient.jar

     /usr/lpp/cicsts/ctg/classes/ctgserver.jar
   - Add the following directory to ncf.jvm.libpath:

     /usr/lpp/cicsts/ctg/bin

   Note: We found it necessary to add these three definitions at the beginning of the properties definitions lists.
2. In the IBM HTTP Server 5.2 for OS/390 startup procedure, add the CICS Transaction Server SDFHEXCI library as a steplib to the Web server started procedure.
3. Define the Language Environment resources to CICS:
   - Define data set CEE.SCEECICS as a steplib to your CICS procedure.
   - Define CEE.SCEERUN and CEE.SCEECICS to the DFHRPL concatenation in your CICS procedure.
4. Define session and connection resources to CICS by installing the DFH$EXCI group.

3.3 Enable IMS for the IBM WebSphere Application Server

In order for the Web servers and their Java applications to have access to IMS, you need the IMS TCP/IP OTMA Connector (ITOC) installed and customized. In IMS Version 7, this will be the IMS Connect Feature. Open Transaction Manager Access (OTMA) is the interface to access IMS transactions and was introduced with IMS V5. The ITOC is a so-called OTMA client and is the interface between TCP/IP and OTMA. For OTMA communication a protocol called Cross Memory Communications Facility (XCF) is used.

More information about the ITOC can be found at:

http://www.ibm.com/software/data/ims/imstoc.html

The ITOC is available for IMS Version 5 and higher. For using ITOC, the OTMA needs to be set up. For further information, refer to the *IMS/ESA V6 OTMA Guide and Reference,* SC26-8743.

3.3.1 Enabling JITOC (Java ITOC) interface

The JITOC is the preferred tool to develop Java servlets accessing IMS. It uses the IBM Enterprise Access Builder library, the IBM Java Record Library, the IBM Common Connector Framework and the IBM JITOC class library. The JITOC uses the ITOC for accessing IMS.

In order to run servlets developed with VisualAge for Java Enterprise Edition, the following jar files need to be uploaded to the host, stored in your HFS, and added to ncf.jvm.classpath:

- /your_path/eablib.jar
- /your_path/recjava.jar
- /your_path/ccf.jar
- /your_path/imstoc.jar

You will find these files in the classes directory where the IBM Connectors have been installed on your workstation running Visual Age for Java Enterprise Edition.

In addition, the jar files containing the Enterprise Access Bean (EAB) commands your servlets executes need to be in the classpath as well.

3.3.2 Enabling the OTMA Callable Interface sample classes

The OTMA Callable Interface is an API that hides the complexity of the XCF protocol that is used to communicate between OTMA and its clients. OTMA CI is available with APAR PQ17203 for IMS V6.1.

There is a Java class sample available that makes the OTMA Callable Interface API available on OS/390 using the Java Native Interface (JNI). The JAVAOTMA package can be downloaded at:

http://www.s390.ibm.com/nc/sntc/

In order to use this package, you need to adjust your environment as follows:

In the file was.conf:

- Add to ncf.jvm.classpath: /your_path/otmaclass.jar

- Add to ncf.jvm.libpath the path where *libjotmaNative.so* is stored.

A more detailed description is in Chapter 20 of *Java Programming Guide for OS/390,* SG24-5619.

Further information about the setup and configuration of the OTMA CI API can be found in Appendix D of *IMS/ESA V6 OTMA Guide and Reference,* SC26-8743. Check the IMS OTMA CI Web page as well:

`http://www.ibm.com/software/data/ims/otmaci.html`

3.4 Enable MQSeries for the IBM WebSphere Application Server

In order for the Web servers and their Java applications to have access to MQSeries, you need to have the MQSeries subsystem configured and customized and the Queue manager up and running.

3.4.1 IBM WebSphere Application Server settings for using MQSeries

The Java classes for the OS/390 bindings connection (not using the common connector framework) can be found and downloaded at the MQSeries txppacs homepage (package name MA1G):

`http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html`

The classes used here are different from those that are used by Visual Age for Java V3 and included in the Common Connector Framework.

An API description is on the download page as well, including the API and class descriptions to use MQSeries.

In order to get the MQSeries classes for OS/390 running, you need to adjust your environment as follows:

1. In the file was.conf:
   - Add to ncf.jvm.classpath: `<mq_dir>/java/lib`
   - Add to ncf.jvm.classpath: `<mq_dir>/java/lib/com.ibm.mq.jar`
   - Add to ncf.jvm.libpath: `<mq_dir>/java/lib`
2. Add to STEPLIB the following data sets, if not already in linklist concatenation:
   - *MQHLQ*.SCSQAUTH
   - *MQHLQ*.SCSQANLE

STEPLIB entries could be done either in httpd.envvars or in the Web server's started task procedure.

A more detailed description about the usage of the MQSeries classes for OS/390 is in Chapter 19 of *Java Programming Guide for OS/390,* SG24-5619.

3.5 Enable Oracle for the IBM WebSphere Application Server

This section describes how to enable the IBM WebSphere Application Server to access Oracle data using JDBC.

3.5.1 Obtain Oracle JDBC drivers

In order for the Web servers and their Java servlets to have access to Oracle, you need the Oracle JDBC type 4 (*thin*, Java Sockets) driver class files.

The Oracle drivers are comprised of two class files: one is for National Language Support (NLS) and the other for base JDBC driver requirements.

You may obtain copies of these files directly from Oracle Corporation, at the following URL:

`http://www.oracle.com`

Follow the links to the *Oracle Technology Network*. There you need to register in order to obtain the Oracle JDBC drivers.

When downloading JDBC driver files, be sure to obtain the *JDBC-Thin, 100% Java* and *NLS-zip, 100% Java* versions of the class files, and make sure you select them for the Oracle version you are using.

We tested the Oracle8i 8.1.6 JDBC "thin" driver files specifically developed for version 1.1.x of the JDK.

Note that, at the time of this writing, there are two sets of Oracle JDBC drivers: One is for use with JDK 1.1.x and the other for the J2EE (JDK 1.2.x) releases. Make sure you download the 1.1.x version of appropriate Oracle JDBC type 4 drivers, since WAS 1.2 only supports JDK 1.1.x.

We downloaded and used the drivers in the following directory:

/usr/lpp/oracle/jdbc/classes

3.5.2 Configure and install Oracle for OS/390

You will also need to have Oracle for OS/390 version 8.04 or later (8.1.5) installed and configured for access via TCP/IP. Consult the Oracle for OS/390 Installation, Administration and User's guides.

3.5.3 IBM WebSphere Application Server settings for JDBC

Add the Oracle JDBC drivers to ncf.jvm.classpath in the IBM WebSphere Application Server configuration file was.conf.

These consist of two zip files:

/usr/lpp/oracle/jdbc/classes/classes12.zip

/usr/lpp/oracle/jdbc/classes/nls_charset12.zip

Run updateproperties to activate the changes to was.conf.

This is all the setup required. In order to test your configuration, you may use the sample servlet for JDBC provided by Oracle. You may obtain the sample from the additional material repository of the redbooks Web server at:

`ftp://www.redbooks.ibm.com/redbooks/sg245604.`

It is called OracleTestServlet.java. It must be compiled and moved into the appropriate directory where you store servlets, like /web/apple/servlets, and then invoked from your browser at:

`http://your_server/servlets/OracleTestServlet`

Chapter 4. How to run the samples

In this chapter we describe the steps you need to perform in order to be able to use the samples provided with this redbook. These samples are based on the samples that are delivered with IBM WebSphere Application Server V1.1. They were improved to suit our environment and to make them easier to maintain. Other samples are new and were added to make a complete collection for this redbook.

It is assumed that your IBM HTTP Server and WebAS are up and running on your system.

We tested and used all these samples using Netscape Communicator V4.7. We have not tested them using any earlier version or other browsers.

Be sure to have Javascript enabled. Some samples use Javascript.

4.1 Download and install the samples

You may obtain the samples from the additional material repository of the redbooks Web server at:

`ftp://www.redbooks.ibm.com/redbooks/sg245604.`

All the samples are packaged together into an approximately 6.5 MB pax file, Was12.Samples.ITSO.pax.

Download the file into the OS/390 UNIX HFS directory structure, as we did into /u/trauner in our example. Use the OMVS shell. Change the directory to your server_model_root. Unpack the file using the `pax -rv` command. Be sure to be authorized to create directories and have write access, else you would need to use superuser mode.

Example:

```
TRAUNER:/u/trauner: >cd /was/helen/AppServer
TRAUNER:/was/helen/AppServer: >
 ===> pax -rv < /u/trauner/Was12.Samples.ITSO.pax
```

Unpacking of this archive file creates three subdirectories, /samples, /doc and /web, and will also put files into the /servlet directory.

4.2 Configuring the environment

In order to run the samples, some key additions need to be made to your configuration files and WebAS environment. Although most of the updates should already be in place, we make reference to them here.

4.2.1 Updating the httpd.conf file

Add the following PASS rules required to access the samples from the Web browser:

```
Pass /IBMWebAS/samples/*     /server_model_root/samples/*
Pass /IBMWebAS/doc/*         /server_model_root/doc/*
Pass /IBMWebAS/*             /server_model_root/web/*.
```

where *server_model_root* is the directory structure defined by your WebSphere Application Server environment. In our examples we used /was/helen/AppServer.

4.2.2 Updating the httpd.envvars file

The httpd.envvars file might need some updates to run the samples, as shown in Figure 13.

```
PATH=/bin:.:/usr/sbin:/usr/lpp/internet/bin:/usr/lpp/internet/sbin:
1 /usr/lpp/ldap/bin:/usr/lpp/java18p/J1.1/bin
SHELL=/bin/sh
TZ=EST5EDT
LANG=C
LC_ALL=en_US.IBM-1047
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:
1 /usr/lpp/ldap/lib/nls/msg/%L/%N
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/sbin:/usr/lpp/ldap/lib:
1 /usr/lpp/java16p/J1.1/lib/mvs/native_threads:
1 /usr/lpp/db2/db2610/lib                                  2
LD_LIBRARY_PATH=/usr/lpp/db2/db2610/lib                    3
JAVA_HOME=/usr/lpp/java18p/J1.1
CLASSPATH=.:/usr/lpp/internet/server_root/CAServlet:
1 /usr/lpp/java18p/J1.1/lib/classes.zip:
1 /usr/lpp/db2/db2610/classes/db2jdbcclasses.zip           4
STEPLIB=DB2V610.SDSNLOAD:DB2V610.SDSNEXIT                  5
DSNAOINI=DB2V610S3.DB2CLI.CLIINI.RRS                       6
```

Figure 13. Updates to httpd.envvars to run the ITSO samples

Notes for the httpd.envvars file:

1 These lines have been split for redbook printing purposes. In the actual file, they are on a single line.

2 Optional - the path is only required for the DB2 database samples. If the path is already defined in the ncf.jvm.libpath statement in the was.conf file (which results in the jvm.properties file), it can be ignored. We recommend that you define it in was.conf only, if you will not use DB2 outside the WAS environment.

3 Optional - we found that the DB2 database samples worked without the LD_LIBRARY_PATH specified.

4 Optional - if ncf.jvm.use.system.classpath=true in was.conf, the CLASSPATH will be concatenated to ncf.jvm.classpath during WAS initialization. If set to false, the CLASSPATH setting in httpd.envvars will be ignored and only the CLASSPATH specified in was.conf is used.

We recommend that you define it in was.conf only, if you will not use DB2 outside the WAS environment and set ncf.jvm.system.classpath=false.

5 Optional - if the DB2 data sets are in your MVS LNKLST concatenation, or defined in the Web server started task procedure as a STEPLIB, ignore this statement. We recommend that you define it in the Web server started task procedure.

6 Optional - if the DSNAOINI ddname is specified in your Web server started task procedure, this line can be ignored. We recommend that you define it in the Web server started task procedure.

4.2.3 Updating the was.conf file

The following classes need to be added to the ncf.jvm.classpath property, in addition to the default WebAS class libraries:

```
/usr/lpp/java18p/J1.1/lib/classes.zip:                ---> contains JDK1.1.8 classes
/usr/lpp/db2/db2610/classes/db2jdbcclasses.zip:       ---> required for DB2 samples
```

The following classes need to be added to the ncf.jvm.libpath property, in addition to the default WebAS executable libraries:

```
/usr/lpp/java18p/J1.1/lib:/usr/lpp/java18p/J1.1/lib/mvs/native_threads:  --->JDK DLL's
/usr/lpp/db2/db2610/lib          ---> required for DB2 samples(JDBC drivers)
```

The ncf.jvm.path property should contain the Java path /usr/lpp/java18p/J1.1/bin (default setting defined by the makeserver script).

4.2.4 Updating the personal .profile file

The following minimum paths need to be added to your personal (user) .profile file to allow you to compile most of the Java source files (refer to the samples and subsystem setup in order to set up the environment variables):

```
#==========================================================================
# Additions to your .profile file
#==========================================================================
JAVA_HOME=/usr/lpp/java18p/J1.1/
PATH=$PATH:/usr/lpp/java18p/J1.1/bin:$HOME
# ========================================================================
# Java JDBC section - only required for running DB2 samples within your shell
# ========================================================================
export LIBPATH=/usr/lpp/db2/db2610/lib:$LIBPATH
export CLASSPATH=$CLASSPATH:/usr/lpp/db2/db2610/classes/db2jdbcclasses.zip
export CLASSPATH=$CLASSPATH:/was/egg/lib/ibmwebas.jar:/was/egg/lib/jst.jar
export CLASSPATH=$CLASSPATH:/was/egg/lib/jsdk.jar:/was/egg/lib/x509v1.jar
export LD_LIBRARY_PATH=/usr/lpp/db2/db2610/lib
export DSNAOINI=DB2V610S3.DB2CLI.CLIINI.RRS
export STEPLIB=$STEPLIB:DB2V610.SDSNEXIT:DB2V610.SDSNLOAD
#This line exports the variable settings so that they are known to the
# system.
export PATH JAVA_HOME
```

To verify that Java is correctly installed, type:

```
java -fullversion
```

The Java command displays the version and build date of the JDK.

4.2.5 Customizing the userprofile properties

Two of the ITSO database samples (WOMBank and TicketCental) require the use of the UserProfile class. The UserProfile class is part of the com.ibm.servlet.personalization.userprofile package and holds basic information about the user.

To successfully use the UserProfile class, you need access to DB2 OS/390 to store the data, and IBM HTTP Server must have access to it. Follow these steps to configure the class:

Step1. Create the database.

The UserProfile class stores its data in a DB2 table called userprofile (by default). You'll need to create the table as shown in A.1, "User profile" on page 98.

Step2. Enable server access to the UserProfile class.

To enable the UserProfile class, configure the class manually. Update the userprofile.properties file in the /*server_model_root*/properties/server/servlet/servletservice/ directory as shown in Figure 14. We found that this file is not updated using was.conf and running the updateproperties utility located in the /*server_model_root*/config directory. Create a backup of this file before making any modifications.

```
# @(#)sessionstate.properties
#
# User Profiles Configuration Screen
#
#ncf.userprofile=false -------> default
ncf.userprofile=true                                1
#ncf.userprofile.db.used=db2 ------> default
ncf.userprofile.db.used=db2os390                    2
#ncf.userprofile.db.jdbcdriver=COM.ibm.db2.jdbc.app.DB2Driver-> default
ncf.userprofile.db.jdbcdriver=ibm.sql.DB2Driver     3
ncf.userprofile.db.instance=DBS3                    4
ncf.userprofile.db.owner=user_name                  5
ncf.userprofile.tablename=userprofile
ncf.userprofile.userid=
ncf.userprofile.password=
ncf.userprofile.classname=com.ibm.servlet.personalization.
6 userprofile.UserProfile
```

Figure 14. Sample userprofile.properties file

Notes to Figure 14 on page 35:

1 Enable the user profile class; by default it is disabled.

2 The name of the database product on OS/390.

3 The name of the JDBC driver on OS/390.

4 This is the LOCATION name as specified in DDF in DB2 or the local name. The name can be obtained from the DB2MSTR started task log when DDF initializes, for example:

```
STC01145  DSNL004I @ DDF START COMPLETE
                     LOCATION  DBS3
                     LU        USIBMSC.SC58DB2
                     GENERICLU -NONE
                     DOMAIN    wtsc58
                     TCPPORT   33380
                     RESPORT   33381
STC01145  DSN9022I @ DSNYASCP 'STA DB2' NORMAL COMPLETION.
```

Refer to Chapter 3, “Enable subsystems for the Application Server” on page 17 for more details.

5 The name of the owner of the user profile table. See A.1, “User profile” on page 98.

6 This line has been split for redbook printing purposes. In the actual file the text appears on one line.

Step3. Restart WebAS to activate the changes.

4.2.6 Configuring the session tracker

Two of the database samples (WOMBank and TicketCental) require the use of the session tracker.

WebAS can maintain a series of requests in a session, originating from the same user, at the same workstation. The class that coordinates the session tracking is:

`com.ibm.servlet.personalization.sessiontracking.IBMSessionContextImpl`

It is called the session tracker. Refer to *WebSphere Application Server Planning, Installing and Using Version 1.2,* GC34-4757 for more information. By default, the session tracker is disabled. Refer to the session.properties file in the /*server_model_root*/properties/server/servlet/servletservice/ directory as shown in Figure 15 on page 37. Any modifications to this file should be made using the was.conf file in /*server_model_root*/properties/ directory. The

updateproperties utility will overwrite the settings made manually with those defined in was.conf.

```
# @(#)session.properties.1.4 97/09/14
#
# Properties for session tracking
#
enable.sessions=true
enable.urlrewriting=false
enable.cookies=true
enable.protocolswitchrewriting=false

session.invalidationinterval=10000
session.swapinterval=10000
session.persistence=true
# if nothing is entered for session.swapdirectory,
# Session Tracking will use its default of
# <instal root>/logs/sessSwap
session.swapdirectory=
session.maxresidents=1024
session.invalidationtime=1800000

session.standalone.host=true
session.server.host=
session.cluster.host=

session.cookie.name=sesessionid
session.cookie.comment=servlet Session Support
session.cookie.domain=
session.cookie.maxage=
session.cookie.path=
session.cookie.secure=
```

Figure 15. Sample default session.properties file

Note to Figure 15:

If the browser does not accept cookies, this implementation will not work.

Important

Shut down and restart the HTTP Server for the changes to take effect. We found this to be true for all servlet changes in WebAS.

4.3 Accessing the samples

All the samples provided with WebAS can be found in /*server_model_root*/samples.

For quick access to the samples, use your browser to open the samples index page `http://your.server.name/IBMWebAS/samples` where *your.server.name* is the host name of the machine on which the IBM WebSphere Application Server is installed, for example `http://wtsc58oe.itso.ibm.com:7101/`.

The source code (.java files) and HTML files for a given sample are in the respective subdirectories under the /*server_model_root*/samples path. We also found some of the source files in the /*server_model_root*/servlets directory.

4.4 Running the no-database samples

This section describes the samples that do not require a database.

4.4.1 ReqInfoServlet

This servlet requires no modifications to the source. Use your browser to open the samples index page and select the servlet under Basic Servlets. The servlet extracts information about the servlet request and returns it to the client.

The sample displays output as shown in Figure 16 on page 39.

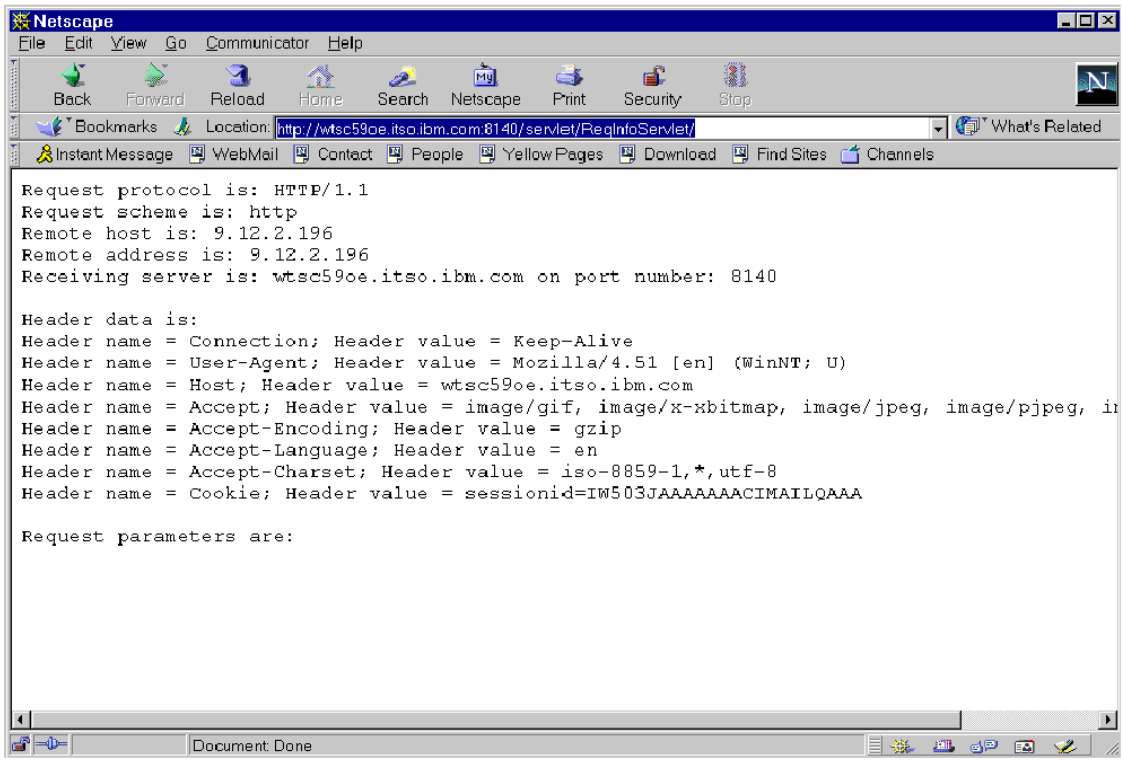


Figure 16. Sample output from ReqInfoServlet

4.4.2 FormDisplayServlet

This servlet requires no modifications to the source. Use your browser to open the samples index page and select the servlet under Basic Servlets. The servlet reads input from an HTML form and returns it in a Web page.

The sample displays output as shown in Figure 17 on page 40.

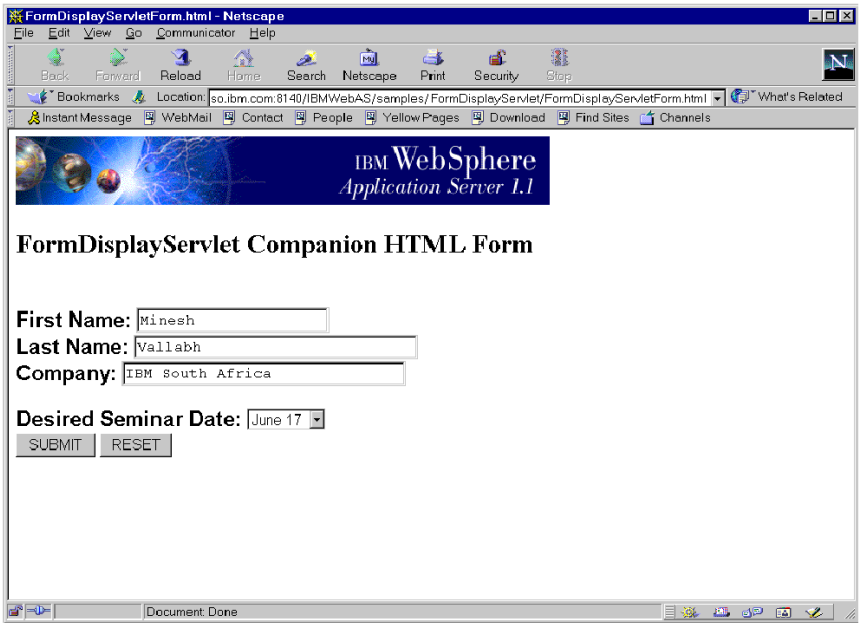


Figure 17. Sample output from FormDisplayServlet (part 1 of 2)

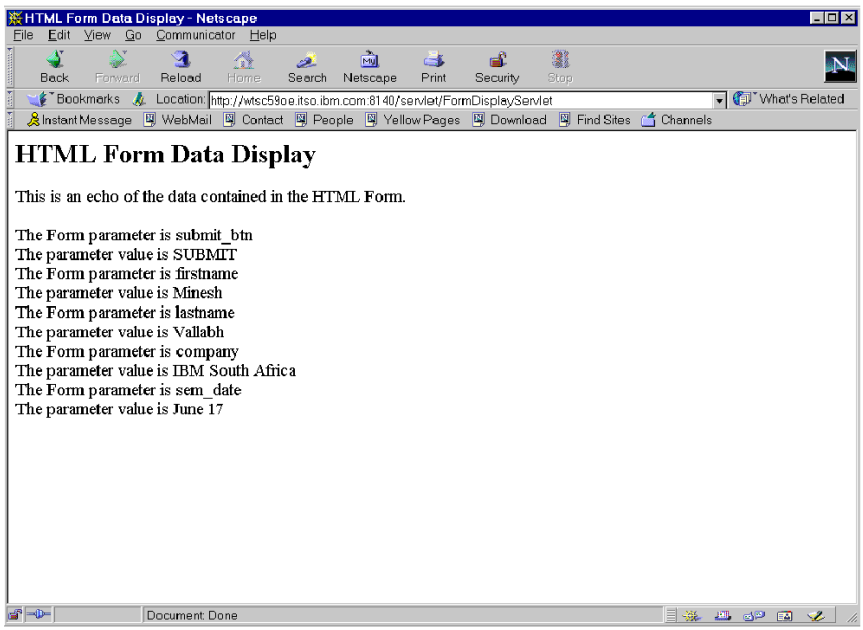


Figure 18. Sample output from FormDisplayServlet (part 2 of 2)

4.4.3 FormProcessingServlet

This servlet requires no modifications to the source. Copy the *seminar.txt* file from the *server_model_root*/samples/FormProcessingServlet directory to the Web content root directory as specified in the httpd.conf file, for example:

```
Pass      /*          /web/apple/pub/*
```

Make your Web server authorization user ID the owner of the /web/apple/pub/seminar.txt file and set the permission bits to 766 in order to have access to the file. If the user ID statement in httpd.conf is %CLIENT%, make sure that all the client user IDs accessing this sample have read and write rights to this file.

Use your browser to open the samples index page and select the servlet under Basic Servlets. The servlet reads and processes registration data from an HTML form, returns a customized page, and writes registration data to a file.

The sample displays output as shown in Figure 19.

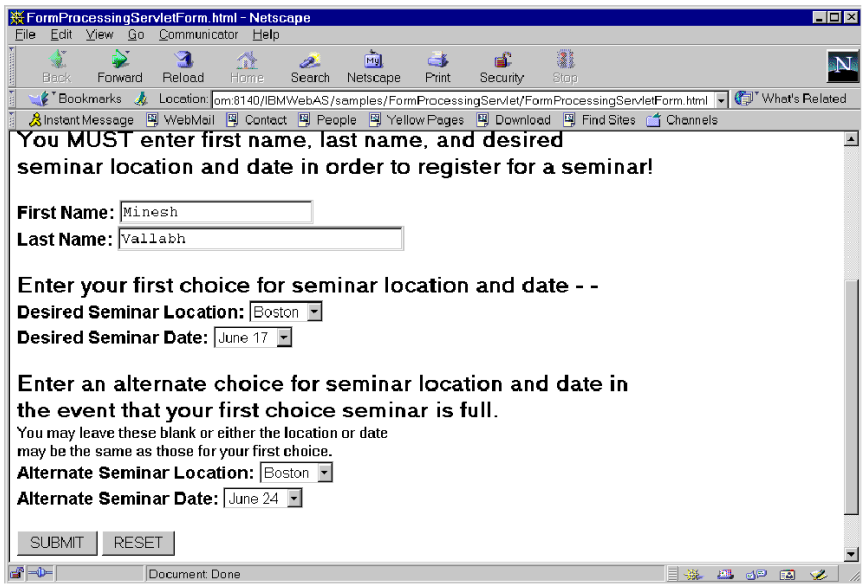


Figure 19. Sample output from FormProcessingServlet (part 1 of 2)

```
June+3|Boston|Larry|Bird|
June+3|Boston|Kevin|McHale|
June+10|Dallas|Emmitt|Smith|
June+10|Dallas|Troy|Aikman|
June+17|Miami|Dan|Marino|
June+17|Miami|Don|Shula|
June+24|Seattle|Ken|Griffey|
June+24|Seattle|Randy|Johnson|
June+17|Boston|m|v|                    1
June+17|Boston|m|v|                    1
June+24|Boston|Helen|H.|
```

Figure 20. Sample contents of the seminar.txt file

Notes to Figure 20:

1 The servlet stores the values in an 8x2 string array (Venue and Date combinations). Boston already has two entries for June 17; hence the class is full.

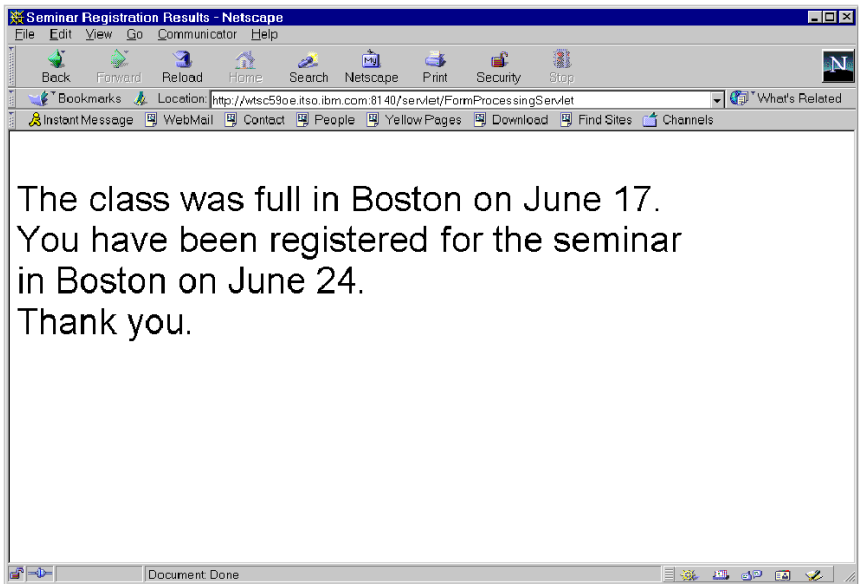


Figure 21. Sample output from FormProcessingServlet (part 2 of 2)

4.4.4 XtremeTravel

The XtremeTravel sample uses the XTBean.class and SendMessage.class files. We found the XTBean.class file in the /*server_model_root*/web/admin/classes directory. This directory had to be placed in ncf.jvm.classpath in was.conf. When the sample is called and the Java server page (JSP) is served, the pagecompile directory gets created by the pagecompile servlet. This directory is used to keep track of JSP-created Java files that have been compiled into Class files. Ensure that you have the correct owner and permission bits set to this directory (corresponding to the Web server's authorization user ID set with the UserID statement in httpd.conf).

We found that the ibmwebas.jar file in the /server_model_root/lib directory did not contain the correct com.ibm.servlet.servlets.personalization.util package, hence the SendMessage class could not be found together with the other classes, namely, CheckMessage, GetMessage, GetVariableText and SetVariableText.

We corrected the "hard links" for the missing classes to exist in the directory /server_model_root/servlets/com/ibm/servlet/servlets/personalization/util.

4.5 Running the database samples

Important

If you plan to make changes to the Java source files, we recommend doing the following:

- Copy the respective source files to your own user directory.
- Make the changes as outlined in this chapter.
- Make sure that all needed classes and class libraries are in the CLASSPATH.
- Compile the code using the *javac filename.java* command.
- Copy the generated class files to the *server_model_root*/servlets directory.

  We put the sample in the server_model_root/servlets directory. You should put your own servlets into a directory defined by the servlets.classpath definition or in a directory belonging to your WebAS instance.

This ensures that the original source files stay intact for future reference.

Before running the samples:

Make sure that the subsystems are identified to WebAS and customized to accept requests, including the sample databases and applications that are delivered with them.

Note: All samples are described for using the old JDBC/SQLJ driver that is delivered with DB2. For running the samples with the new type 2 driver described in 3.1.4, “Settings for the new Type 2 SQLJ/JDBC Driver” on page 23, refer to 4.5.11, “Run the samples with the new Type 2 Driver” on page 65.

4.5.1 JDBCServlet

The JDBCServlet reads input from an HTML form to build an SQL query, makes a JDBC connection to the IBM DB2 sample database, processes the query result set, and returns it to the Web page.

To run the sample, complete the following steps:

Step 1. Create the sample database provided by DB2 for OS/390.

This sample uses the EMPLOYEE sample database provided with DB2 for OS/390. Consult with your database administrator (DBA) for information on how to create this database.

Step 2. Copy and edit the login.properties file.

Copy the login.properties file from the *applicationserver_root*/samples directory to the *applicationserver_root*/servlets directory, where *applicationserver_root* is the root directory of your WebAS installation.

Edit the applicationserver_root/servlets/login.properties file. We had the following values specified:

1. JDBCServlet.dbOwner=*dsn8610*, the owner of the employee database (Consult your DBA.)
2. JDBCServlet.dbUserid=null
3. JDBCServlet.dbPassword=null
4. JDBCServlet.dbName=*DBS3*, the location name in DDF (Refer to Figure 14 on page 35 for an explanation of what the location name is.)
5. JDBCServlet.dbProtocol=db2os390, JDBC subprotocol
6. JDBCServlet.dbPoolName=JdbcDb2, the JDBC pool name (Ask your webmaster.)
7. JDBCServlet.JDBCDriver=ibm.sql.DB2Driver, the JDBC Driver Name
8. JDBCServlet.BackGround= /IBMWebAS/samples/images/background.jpg (The background picture for the output page could be optionally specified.)

The sample is changed so that all necessary parameters could be changed in the login.properties file. You can change them without changing and recompiling the sample source code.

Step 3. Run the sample.

Use your browser to open the samples index page and select the servlet under Basic Servlets.

Known Problems:

- Not all views or SELECT combinations are available. For example, SELECT STAFF.* is a table that does not exist in the sample employee database in DB2 OS/390. This operation results in an SQL exception, which is displayed on your browser.
- After three successful queries we got the following message on our browser: `***Connection is already in use . . . try later!***`. We had to

restart the Web server in order to run this sample again. We found this true when running multiple IBM WebSphere Application Server instances using the same JDBCPool name and accessing the same tables. Make sure that the JDBCPools are unique across multiple Web servers.

4.5.2 IBMConnMgrTest

The IBMConnMgrTest servlet shows how to use the connection manager to manage JDBC connections to the IBM DB2 sample database.

To run the sample, complete the following steps:

Step 1. Create the sample database provided by DB2 for OS/390.

This sample uses the EMPLOYEE sample database provided with DB2 OS/390. Consult with your database administrator for information on how to create this database.

Step 2. Copy and edit the login.properties file.

Copy the login.properties file from the server_*model_root*/samples directory to the *server_model_root*/servlets directory.

Edit the *server_model_root*/servlets/login.properties file. We had the following values specified:

1. JDBCServlet.dbOwner=*dsn8610*, the owner of the employee database (Consult your DBA.)
2. JDBCServlet.dbUserid=null
3. JDBCServlet.dbPassword=null
4. JDBCServlet.dbName=*DBS3*, the location name in DDF (Refer to Figure 14 on page 35 for an explanation.)
5. JDBCServlet.dbProtocol=db2os390, JDBC subprotocol
6. JDBCServlet.dbPoolName=JdbcDb2, the JDBC pool name (Ask your webmaster.)
7. JDBCServlet.JDBCDriver=ibm.sql.DB2Driver, the JDBC Driver Name
8. JDBCServlet.BackGround= /IBMWebAS/samples/images/background.jpg (The background picture for the output page could be optionally specified.)

Note: Some samples use the same login.properties values, like the IBMConnMgrTest and the JDBCServlet sample. This is true for other samples as well. In the section we listed all values for the properties, even though that special sample doesn't use all of them. For details about what properties are

used by a certain sample, refer to the source code of the sample. The login.properties were used for easier maintenance of the samples in a test environment, where it might happen that subsystem IDs, owners of the databases, and database qualifiers change frequently.

Step 3. Run the sample.

Use your browser to open the samples index page and select the servlet at the database servlets page. The sample displays the following result: Hello JOHN Parker.

4.5.3 IBMDataAccessTest

The IBMDataAccessTest servlet shows how to use the IBM VisualAge for Java data access beans to connect to the IBM DB2 sample database. This sample uses a class library that is only delivered with VisualAge for Java Enterprise Edition V2 or V3.

To run the sample, complete the following steps:

Step 1. Create the sample database provided by DB2 for OS/390, if not already installed.

This sample uses the EMPLOYEE sample database provided with DB2 OS/390. Consult with your database administrator for information on how to create this database.

Step2. Copy and edit the login.properties file.

Copy the login.properties file from the *applicationserver_root*/samples directory to the *applicationserver_root*/servlets directory, where *applicationserver_root* is the root directory of your WebAS installation.

Edit the *applicationserver_root*/servlets/login.properties file. We had the following values specified:

1. JDBCServlet.dbOwner=*dsn8610*, the owner of the employee database (Consult your DBA.)
2. JDBCServlet.dbUserid=null
3. JDBCServlet.dbPassword=null
4. JDBCServlet.dbName=*DBS3*, the location name in DDF (Refer to Figure 14 on page 35 for an explanation.)
5. JDBCServlet.dbProtocol=db2os390, JDBC subprotocol
6. JDBCServlet.dbPoolName=JdbcDb2, the JDBC pool name (Ask your webmaster.)

7. JDBCServlet.JDBCDriver=ibm.sql.DB2Driver, the JDBC Driver Name
8. JDBCServlet.BackGround= /IBMWebAS/samples/images/background.jpg (The background picture for the output page could be optionally specified.)

Step 3. Update your was.conf file.

We needed to include the data access beans class library. We found it in the EAB/runtime20 directory of VisualAge for Java V3.

Edit the *server_model_root/*properties/was.conf file. Add the following class to ncf.jvm.classpath in order to run the sample:

- `/your_path/ivjdab.jar`

Activate the changes.

Step 4. Run the sample.

Use your browser to open the samples index page and select the servlet under Database Servlets. The sample displays two result rows. When reloading the servlet you will see an increasing value in the second row. This is intended; it is due to the use of caching. Check the source code of the sample for details.

4.5.4 WOMBank

This sample uses a JDBC connection to a DB2 database to maintain customer IDs, passwords, and account information. It does its own user authentication, allowing customers to register or log in, open accounts, deposit, withdraw, or transfer funds, and view a log of their transactions. WOMBank uses user profiles and session tracking to personalize the Web pages and track the application state. To run the sample, complete the following steps:

Step 1. Create the sample database.

WOMBank provides a womdb.bat file to populate the database. The file is found in the directory *server_model_root*/samples/WomBank/. We created the equivalent SQL statements for DB2 OS/390 based on this file; see A.2, “WOMBank” on page 99 for the SQL statements.

Step 2. Copy and edit the login.properties file.

Copy the login.properties file from the *applicationserver_root*/samples directory to the *applicationserver_root*/servlets directory, where *applicationserver_root* is the root directory of your WebAS installation.

Edit the *applicationserver_root*/servlets/login.properties file. We had the following values specified:

1. WomBank.dbOwner=dsn8610**,** the owner of the womdb database as defined in A.2, “WOMBank” on page 99 (Consult your DBA for more information.)
2. WomBank.dbUserid=null
3. WomBank.dbPassword=null
4. WomBank.JDBCLocation=DBS3 (For the LOCATION name in DB2 DDF, refer to Figure 14 on page 35.)
5. WomBank.JDBCDriver=ibm.sql.DB2Driver, the JDBC Driver
6. WomBank.JDBCProtocol=db2os390, JDBC subprotocol

Step3. Configure the user profile class.

See 4.2.5, “Customizing the userprofile properties” on page 35 for details.

Step4. Configure the session tracker.

See 4.2.6, “Configuring the session tracker” on page 36 for details.

Step5. Update the permission bits.

The sample creates a directory called *server_model_root*/logs/upobjects. Make sure that the Web server authentication user ID specified in httpd.conf has access to create this directory (WEBADM in our case). We set the permission bits for the logs directory to 775, since the group ID for the user WEBADM and the group owner of the logs directory were both IMWEB. If the UserID statement in httpd.conf is %CLIENT%, make sure that all the client user IDs accessing this sample have read, write and execute rights to this directory.

Step6. Run the sample.

Use your browser to open the samples index page and select WOMBank.

4.5.5 TicketCentral

This sample uses a JDBC connection to a DB2 database. It searches several preloaded database tables to find ticket information, and reads and writes other tables to authenticate customers and maintain their data. Registered customers can log in, specify ticket search information, select tickets to purchase, and use a credit card to pay for them. TicketCentral uses user profile and session tracking to personalize the Web pages and track the application state.

To run the sample, complete the following steps:

Step 1. Create the sample database.

TicketCentral provides a tcdb.bat file to populate the database. The file is found in the directory *server_model_root*/servlets/. We created the equivalent SQL statements for DB2 OS/390 based on this file. See A.3, “TicketCentral” on page 100 for the SQL statements.

Step 2. Copy and edit the login.properties file.

Copy the login.properties file from the *applicationserver_root*/samples directory to the *applicationserver_root*/servlets directory, where *applicationserver_root* is the root directory of your WebAS installation.

Edit the *applicationserver_root*/servlets/login.properties file. We had the following values specified:

1. TicketCentral.dbOwner=*owner_name***,** the owner of the TCDB database as defined in A.3, “TicketCentral” on page 100 (Consult your DBA.)
2. TicketCentral.dbUserid=null
3. TicketCentral.dbPassword=null
4. TicketCentral.JDBCLocation=DBS3 (For the LOCATION name in DB2 DDF, refer to Figure 14 on page 35.)
5. TicketCentral.JDBCProtocol=db2os390

You might wonder why there is no JDBC Driver specified as in the WomBank sample. The JDBC Driver in all TicketCentral source files is ibm.sql.DB2Driver and is hard-coded.

Step 3. Populate the TCDB database.

From the OMVS shell, issue the following commands to populate the database:

```
cd /applicationserver_root/servlets

java PopulateTCData tcevent   tcevent.txt
java PopulateTCData tcprice   tcprice.txt
java PopulateTCData tcvenue   tcvenue.txt
java PopulateTCData tcsession tcsession.txt.
```

Step 4. Configure the user profile class.

Refer to 4.2.5, “Customizing the userprofile properties” on page 35 for details.

Step 5. Configure the session tracker.

Refer to 4.2.6, “Configuring the session tracker” on page 36 for details.

Step 6. Run the sample.

Use your browser to open the samples index page and select Ticket Server.

4.5.6 SQLJ servlet sample

This sample uses SQLJ to connect to a DB2 database.

To run the sample, complete the following steps:

Step 1. Create the sample database provided by DB2 for OS/390 (if not already done).

This sample uses the EMPLOYEE sample database provided with DB2 OS/390. Consult with your database administrator for information on how to create this database.

Step 2. Edit the login.properties file (if not already done).

Edit the *server_model_root*/servlets/login.properties file. This sample uses the same values as the JDBCServlet Sample. We had the following values specified:

6. SQLJServlet.SQLJDriver= COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver, the SQLJ driver name
7. SQLJServlet.SQLJProtocol=db2os390sqlj, SQLJ subprotocol
8. SQLJServlet.SQLJLocation=DBS3, the location name in DDF (Refer to Figure 14 on page 35 for an explanation.)
9. SQLJServlet.BackGround= /IBMWebAS/samples/images/background.jpg (The background picture for the output page could be optionally specified.)

Step 3. Update your personal profile file.

Edit the /u/*user_name/*.profile file. Make sure that your PATH statement includes the following in order to run the DBRM utility:

```
/usr/lpp/db2/db2610/bin
```

Make sure that your LIBPATH statement includes the following:

```
/usr/lpp/db2/db2610/lib
```

Activate the changes.

Step 4. Run the DBRM utility.

It creates and stores a member in your DBRM library using the .ser file that was created by the SQLJ preprocessor utility (*db2profc*). This member is used to create the SQLJPLAN later. The default data set name is <*USERNAME*>.DBRMLIB.DATA. You might change this using the environment variable DB2SQLJDBRMLIB:

```
export DB2SQLJDBRMLIB="//'HLQ.DBRMLIB.DATA'"
```

Go to the OMVS shell and run the utility using the following command (the .ser file that is delivered with the sample is in the *server_model_root*/servlets directory):

```
db2profc -pgmname=SQLJSAMP SQLJSampleServlet_SJProfile0.ser
```

Change the pgmname as appropriate for your environment. We used SQLJSAMP.

Step 5. Bind the DBRM member and create the SQLJPLAN.

Ask your database administrator for support. In order to run this sample, bind the DBRM member directly to a plan. In 4.5.8, “SQLJ manufacturer servlet” on page 57 we describe how to run multiple SQLJ servlets using packages for each servlet and creating a plan for all packages. We used the DB2 ISPF panels to create the SQLJSAMP plan.

```
.
                         BIND PLAN                              SSID: DBS3
COMMAND ===>

Enter DBRM data set name(s):
 1  MEMBER ......... ===> SQLJSAMP
 2  PASSWORD ....... ===>
 3  LIBRARY ........ ===> 'USER.DBRMLIB.DATA'
 4  ADDITIONAL DBRMS? ......... ===> NO        (YES to include more DBRMs)

Enter options as desired:
 5  PLAN NAME ................. ===> SQLJSAMP  (Required to create a plan)
 6  CHANGE CURRENT DEFAULTS? .. ===> YES       (NO or YES)
 7  ENABLE/DISABLE CONNECTIONS? ===> NO        (NO or YES)
 8  INCLUDE PACKAGE LIST?...... ===> NO        (NO or YES)
 9  OWNER OF PLAN (AUTHID)..... ===> WEBADM    (Leave blank for your primaryID)
10  QUALIFIER ................. ===> USERID   (For tables, views, and aliases)
11  CACHESIZE ................. ===>           (Blank, or value 0-4096)
12  ACTION ON PLAN ............ ===> REPLACE   (REPLACE or ADD)
13  RETAIN EXECUTION AUTHORITY. ===> NO        (YES to retain user list)
14  CURRENT SERVER ............ ===> DBS3                (Location name)
15  INCLUDE PATH?.............. ===> NO        (NO or YES)

PRESS:  ENTER to process    END to save and exit     HELP for more information .
```

This servlet accesses the EMP table that is delivered with the DB2 sample databases. The SELECT statement in the sample does not reference a qualifier for the table. In order to run the sample, make sure to specify a qualifier and that the table *USERID*.EMP exists and is authorized to access (GRANT).

Known Problems:

- If you experience an SQLSTATE=FFFFF (an internal SQLJ processing error), then it might be that you have an authorization problem, or the SQLJPLAN environment variable is missing or wrong, or your plan is

missing a package. Maybe you forgot to grant access to the plan after rebinding the plan.

- SQLState=50003 means that the timestamps of the servlet and the package do not match. This might be because the .ser file is not in the classpath or the bind options are not appropriate.

  In order to run the sample, we used the following BIND options:

```
.
                        DEFAULTS FOR BIND PLAN                SSID: DBS3
COMMAND ===>

Change default options as necessary:

 1  ISOLATION LEVEL ......... ===> CS          (RR, RS, CS, or UR)
 2  VALIDATION TIME ......... ===> RUN         (RUN or BIND)
 3  RESOURCE RELEASE TIME ... ===> DEALLOCATE  (COMMIT or DEALLOCATE)
 4  EXPLAIN PATH SELECTION .. ===> NO          (NO or YES)
 5  DATA CURRENCY ........... ===> NO          (NO or YES)
 6  PARALLEL DEGREE ......... ===> 1           (1 or ANY)
 7  RESOURCE ACQUISITION TIME ===> ALLOCATE    (USE or ALLOCATE)
 8  REOPTIMIZE FOR INPUT VARS ===> NO          (NO or YES)
 9  DEFER PREPARE ........... ===> NO          (NO or YES)
10  KEEP DYN SQL PAST COMMIT. ===> NO          (NO or YES)
11  DBPROTOCOL .............. ===>             (Blank, DRDA or PRIVATE)
12  OPTIMIZATION HINT ....... ===>             (Blank or 'hint-id')
13  DYNAMIC RULES ........... ===> RUN         (RUN or BIND)
14  SQLRULES ................ ===> DB2         (DB2 or STD)
15  DISCONNECT .............. ===> EXPLICIT    (EXPLICIT, AUTOMATIC,
                                                 or CONDITIONAL)

PRESS:  ENTER to process    END to save and exit     HELP for more information
.
```

  Don't forget to grant execute rights to the authorization user ID of the Web server (WEBADM in our sample) in order to be able to access the plan; you might use SPUFI to run the SQL command:

```
GRANT EXECUTE ON PLAN SQLJSAMP TO WEBADM;
```

  Remember that the authorization user ID is specified in http.conf with the userid statement. If userid %CLIENT% is used, every Web user is subject to authentication with user ID and password. Then this userid is used to access DB2. The rights granted to the plan override the rights granted to the packages that are included in the plan.

Step 6. Change the environment variables in your httpd.envvars.

Specify the PLAN that includes the DBRM you created in Step 8, as described in 3.1.3, “IBM WebSphere Application Server settings for SQLJ” on page 22.

```
.
DB2SQLJPLANNAME=SQLJSAMP
DB2SQLJATTACHTYPE=RRSAF
DB2SQLJSSID=DBS3
.
```

Step 7. Run the sample.

Use your browser to open the samples index page and select the servlet under Database Servlets. The sample displays all people stored in the Employee sample database.

Additional hints:

- Make sure that the IBM WebSphere Application Server is not active when rebinding or binding the plans/packages while testing. You might experience hung threads in DB2, leading to the inability to rebind.
- You might experience a java.sql.SQLException: No suitable driver. SQLSTATE=08001 SQLCODE=0 error, when the environment variables are not specified in httpd.envvars, or the wrong driver is specified in login.properties.

4.5.7 Steps for compiling and running an SQLJ servlet

Here is a description of how to prepare, compile and run .sqlj files that contain embedded SQL and the environment changes that are necessary. SQLJ sources end with .sqlj and not with .java, even though they contain Java code. The description is made using the SQLJ Sample Servlet example. The basis is the driver delivered with DB2.

Step 1. Update your personal profile file.

Edit the /u/*user_name/*.profile file. Add the following classes to the classpath in order to be able to compile the SQLJ sample sources:

- `/usr/lpp/db2/db2610/classes/db2sqljclasses.zip`
- `/usr/lpp/db2/db2610/classes/db2sqljruntime.zip` (only necessary if you plan to run Java applications in your OMVS shell)

Make sure that your PATH statement includes the following:

```
/usr/lpp/db2/db2610/bin
```

Make sure that your LIBPATH statement includes the following:

```
/usr/lpp/db2/db2610/lib
```

Activate the changes.

Step 2. Run the SQLJ preprocessor.

It translates the embedded SQL statements marked by #sql in the source code to classes and creates the profile that is later used for binding the plan or package.

Go to the OMVS shell and run the translator using the following command:

```
sqlj SQLJSampleServlet.sqlj
```

The utility generates the SQLJSampleServlet.java source file and the serialized profile SQLJSampleServlet_SJProfile0.ser.

Step 3. Compile the Java source.

Go to the OMVS shell and compile the source code using the following command:

```
javac SQLJSampleServlet.java
```

The compile generates the class files necessary to run the samples:

```
SQLJSampleServlet.class
SQLJSampleServletIter.class
SQLJSampleServlet_SJProfileKeys.class
SQLJSampleServletctx.class
```

Copy these files to the *server_model_root*/servlets directory using the following command:

```
cp *.class server_model_root/servlets
```

Step 4. Run the DBRM utility.

It customizes the .ser file for DB2 for OS/390 and stores a member in your DBRM library. This member is later used to create the SQLJPLAN. The default data set name is <*USERNAME*>.DBRMLIB.DATA. You might change this using the environment variable DB2SQLJDBRMLIB:

```
export DB2SQLJDBRMLIB="//USERNAME.DBRMLIB.DATA"
```

Go to the OMVS shell and run the utility using the following command:

```
db2profc -pgmname=SQLJSAMP SQLJSampleServlet_SJProfile0.ser
```

Change the pgmname as appropriate for your environment; e.g., we used SQLJSAMP.

It is essential that the .ser file is in a directory that is included in the classpath; e.g., copy it to the *server_model_root*/servlets directory using the following command:

```
cp *.ser server_model_root/servlets
```

Step 5. Bind the DBRM to be included in the SQLJPLAN of your IBM WebSphere Application Server.

4.5.8 SQLJ manufacturer servlet

This sample uses SQLJ to read and update a DB2 database and provides a sample for a manufacturer database.

To run the sample, complete the following steps:

Step 1. Create the sample manufacturer database provided with this sample.

Consult with your database administrator for information on how to create this database. We used the DB2 SPUFI panels to create it. In your *server_model_root*/samples/sqlj directory is a file called db2.data.xmit. It contains all the SQL statements necessary to create the sample manufacturer database.

Create an MVS data set with the following space allocation settings:

```
.
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120
1st extent cylinders: 1
Secondary cylinders : 1
```

Copy file db2.data.xmit to that data set:

```
cp db2.data.xmit "//'USER.DB2.XMIT'"
```

Change to TSO and receive the data set:

```
receive INDSN('USER.DB2.XMIT')
```

Specify DSNAME('*OUTPUTDATASETNAME*') when prompted. It will create a PDS containing one member called INSERT2. Adapt this member to suit your environment's needs; e.g., name of the database, owner, storage group, and run SPUFI to create the database and insert the records containing the sample data. Make sure that the output data set, if used, is large enough (the output will be around 8 cyls); otherwise you will get B37 abends).

Step 2. Edit the login.properties file (if not already done).

Edit the *server_model_root*/servlets/login.properties file. This sample uses the same values as the JDBCServlet sample. We had the following values specified (both SQLJ servlets use the same SQLJ settings):

1. SQLJServlet.SQLJDriver= COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver, the SQLJ driver name
2. SQLJServlet.SQLJProtocol=db2os390sqlj, SQLJ subprotocol
3. SQLJServlet.SQLJLocation=DBS3, the location name in DDF (refer to Figure 14 on page 35 for an explanation)
4. SQLJServlet.BackGround= /IBMWebAS/samples/images/background.jpg (the background picture for the output page could be optionally specified)

Step 3. Update your personal profile file (if not already done).

Edit the /u/*user_name/*.profile file. Make sure that your PATH statement includes the following in order to run the DBRM utility:

```
/usr/lpp/db2/db2610/bin
```

Make sure that your LIBPATH statement includes the following:

```
/usr/lpp/db2/db2610/lib
```

Activate the changes.

Step 4. Run the DBRM utility.

It creates and stores a member in your DBRM library using the .ser file that was created by the SQLJ preprocessor utility (*db2profc*). This member is later used to create the SQLJPLAN. The default data set name is <*USERNAME*>.DBRMLIB.DATA. You might change this using the environment variable DB2SQLJDBRMLIB:

```
export DB2SQLJDBRMLIB="//'HLQ.DBRMLIB.DATA'"
```

Go to the OMVS shell and run the utility using the following command (the .ser file that is delivered with the sample is in the *server_model_root*/servlets directory):

```
db2profc -pgmname=SQLJMANU SQLJServlet_SJProfile0.ser
```

Change the pgmname as appropriate for your environment; e.g., we used SQLJMANU. If you plan to use the SQL Sample Servlet described in 4.5.6, “SQLJ servlet sample” on page 52 as well, make sure you run the utility for that sample as well:

```
db2profc -pgmname=SQLJSERV SQLJSampleServlet_SJProfile0.ser
```

Step 5. Bind the DBRM members and create the SQLJPLAN.

Ask your database administrator for support. In order to run this sample, bind the DBRM member directly to a plan. We assume that you plan to run both the SQLJ Servlet Sample and the SQLJ Manufacturer Servlet. So here is the description of how to create one package for each servlet and a plan containing those packages to run the servlets. We used the DB2 ISPF panels for this.

A DBRM member is bound to a package. Several packages (usually if they belong to the same application) are collected under the same collection ID. So a single package is referenced with *collectionID.PackageName*. Multiple packages might be bound into one PLAN. This is done for easier maintenance (remember, only one SQLJ plan could be specified per Application Server instance). If one application changes, only the corresponding package needs to be rebound, not the whole PLAN, which might include 100 or more packages.

Create a package for the SQLJ servlet sample:

```
.
                          BIND PACKAGE                        SSID: DBS3
COMMAND ===>

Specify output location and collection names:
 1  LOCATION NAME ............. ===>                     (Defaults to local)
 2  COLLECTION-ID ............. ===> SQLJSERV            (Required)
Specify package source (DBRM or COPY):
 3   DBRM:         COPY:        ===> DBRM       (Specify DBRM or COPY)
 4  MEMBER    or  COLLECTION-ID ===> SQLJSERV
 5  PASSWORD  or  PACKAGE-ID .. ===>
 6  LIBRARY   or  VERSION ..... ===> DBRMLIB.DATA
                                          (Blank, or COPY version-id)
 7  ........  --  OPTIONS ..... ===>      (COMPOSITE or COMMAND)
Enter options as desired:
 8  CHANGE CURRENT DEFAULTS? .. ===> YES        (NO or YES)
 9  ENABLE/DISABLE CONNECTIONS? ===> NO         (NO or YES)
10  OWNER OF PACKAGE (AUTHID).. ===> WEBADM     (Leave blank for primary ID)
11  QUALIFIER ................. ===> WEBHELEN  (Leave blank for OWNER)
12  ACTION ON PACKAGE .... .... ===> REPLACE    (ADD or REPLACE)
13  INCLUDE PATH?............. ===> NO         (NO or YES)
14  REPLACE VERSION ........... ===>
                                          (Replacement version-id)
PRESS:  ENTER to process    END to save and exit     HELP for more information
```

Make sure that the database QUALIFIER.EMP exists. This is the place to specify the owner of the database, not the servlet source code. The BIND will only succeed if the database exists and authorizations are set properly (remember GRANT). Note that the collection ID is specified and the package ID will be equal to the member name.

Create a package for the SQLJ manufacturer servlet:

```
.
                          BIND PACKAGE                        SSID: DBS3
COMMAND ===>

Specify output location and collection names:
 1  LOCATION NAME ............. ===>                    (Defaults to local)
 2  COLLECTION-ID ............. ===> SQLJMANU           (Required)
Specify package source (DBRM or COPY):
 3   DBRM:         COPY:        ===> DBRM       (Specify DBRM or COPY)
 4  MEMBER    or  COLLECTION-ID ===> SQLJMANU
 5  PASSWORD  or  PACKAGE-ID .. ===>
 6  LIBRARY   or  VERSION ..... ===> DBRMLIB.DATA
                                          (Blank, or COPY version-id)
 7  ........  --  OPTIONS ..... ===>      (COMPOSITE or COMMAND)
Enter options as desired:
 8  CHANGE CURRENT DEFAULTS? .. ===> YES        (NO or YES)
 9  ENABLE/DISABLE CONNECTIONS? ===> NO         (NO or YES)
10  OWNER OF PACKAGE (AUTHID).. ===> WEBADM     (Leave blank for primary ID)
11  QUALIFIER ................. ===> WEBHELEN  (Leave blank for OWNER)
12  ACTION ON PACKAGE .... .... ===> REPLACE    (ADD or REPLACE)
13  INCLUDE PATH?............. ===> NO         (NO or YES)
14  REPLACE VERSION ........... ===>
                                          (Replacement version-id)
PRESS:  ENTER to process    END to save and exit     HELP for more information
```

Make sure that the database QUALIFIER.MANUFACTURER exists. This is the place to specify the owner of the database, not the servlet source code. The BIND will only succeed if the database exists and authorizations are set properly (remember GRANT).

Ask your DBA about the default JDBC package. In our environment it was stored under collection-id DSNJDBC with names DSNJDBC1, DSNJDBC2, DSNJDBC3, and DSNJDBC4. Make sure that execute authorizations are granted properly (we had execute rights granted to the Web server user ID).

Create the PLAN including the packages necessary to run both the SQLJ Sample Servlet and the SQLJ manufacturer servlet:

```
.
                         BIND PLAN                          SSID: DBS3
COMMAND ===>

Enter DBRM data set name(s):
 1  MEMBER ......... ===>
 2  PASSWORD ....... ===>
 3  LIBRARY ........ ===>
 4  ADDITIONAL DBRMS? ......... ===> NO        (YES to include more DBRMs)

Enter options as desired:
 5  PLAN NAME ................. ===> WEBHELEN  (Required to create a plan)
 6  CHANGE CURRENT DEFAULTS? .. ===> YES       (NO or YES)
 7  ENABLE/DISABLE CONNECTIONS? ===> NO        (NO or YES)
 8  INCLUDE PACKAGE LIST?...... ===> YES       (NO or YES)
 9  OWNER OF PLAN (AUTHID)..... ===> WEBADM    (Leave blank for your primaryID)
10  QUALIFIER ................. ===>           (For tables, views, and aliases)
11  CACHESIZE ................. ===>           (Blank, or value 0-4096)
12  ACTION ON PLAN ............ ===> REPLACE   (REPLACE or ADD)
13  RETAIN EXECUTION AUTHORITY. ===> NO        (YES to retain user list)
14  CURRENT SERVER ............ ===>                    (Location name)
15  INCLUDE PATH?.............. ===> NO        (NO or YES)

PRESS:  ENTER to process    END to save and exit     HELP for more information
```

These are the options we used for the PLAN:

```
.
                       DEFAULTS FOR BIND PLAN               SSID: DBS3
COMMAND ===>

Change default options as necessary:

 1  ISOLATION LEVEL ......... ===> CS          (RR, RS, CS, or UR)
 2  VALIDATION TIME ......... ===> RUN         (RUN or BIND)
 3  RESOURCE RELEASE TIME ... ===> DEALLOCATE  (COMMIT or DEALLOCATE)
 4  EXPLAIN PATH SELECTION .. ===> NO          (NO or YES)
 5  DATA CURRENCY ........... ===> NO          (NO or YES)
 6  PARALLEL DEGREE ......... ===> 1           (1 or ANY)
 7  RESOURCE ACQUISITION TIME ===> ALLOCATE    (USE or ALLOCATE)
 8  REOPTIMIZE FOR INPUT VARS ===> NO          (NO or YES)
 9  DEFER PREPARE ........... ===> NO          (NO or YES)
10  KEEP DYN SQL PAST COMMIT. ===> NO          (NO or YES)
11  DBPROTOCOL .............. ===>             (Blank, DRDA or PRIVATE)
12  OPTIMIZATION HINT ....... ===>             (Blank or 'hint-id')
13  DYNAMIC RULES ........... ===> RUN         (RUN or BIND)
14  SQLRULES ................ ===> DB2         (DB2 or STD)
15  DISCONNECT .............. ===> EXPLICIT    (EXPLICIT, AUTOMATIC,
                                                 or CONDITIONAL)

PRESS:  ENTER to process    END to save and exit     HELP for more information
```

The following panel is used to specify the packages that need to be included in the PLAN. Remember you need to include the default packages and the packages for the two servlets just created. For the packages it is enough to specify the collection ID and an asterisk to include all packages:

```
.
                    PACKAGE LIST FOR BIND PLAN                SSID: DBS3
COMMAND ===>                                            SCROLL ===> PAGE

Enter names to be included in PACKAGE list for: WEBHELEN

CMD      LOCATION          COLLECTION          PACKAGE-ID

""""                       DSNJDBC             *
""""                       SQLJSERV            SQLJSERV
""""                       SQLJMANU            SQLJMANU
```

The PLAN will only be bound successfully if the specified tables and plans exist and the authorization is specified properly.

Don't forget to GRANT execute rights on the plan you just created to the Web server's authorization user ID (which is PUBLIC in our environment):

```
GRANT EXECUTE ON PLAN WEBHELEN TO PUBLIC;
```

Step 6. Change the environment variables in your httpd.envvars.

Specify the PLAN that includes the DBRM you created in Step 8, as described in 3.1.3, "IBM WebSphere Application Server settings for SQLJ" on page 22.

```
.
DB2SQLJPLANNAME=WEBHELEN
DB2SQLJATTACHTYPE=RRSAF
DB2SQLJSSID=DBS3
.
```

Step 7. Run the sample.

Use your browser to open the samples index page and select the servlet under Database Servlets. The sample lets you display, add and update records in the manufacturer database.

4.5.9 JDBC manufacturer servlet

This sample uses JDBC to read and update a DB2 database and provides a sample for a manufacturer database.

To run the sample, complete the following steps:

Step 1. Create the sample manufacturer database provided with this sample (if not already done).

Refer to Step1 in 4.5.8, “SQLJ manufacturer servlet” on page 57 for details.

Step 2. Edit the login.properties file (if not already done).

Edit the *server_model_root*/servlets/login.properties file. This sample uses the same values as the JDBCServlet sample. We had the following values specified (both JDBC manufacturer servlet and JDBC manufacturer JSP sample use the same JDBC settings):

1. Manufacturer.dbOwner=WEBHELEN
2. Manufacturer.dbUserid=null
3. Manufacturer.dbPassword=null
4. Manufacturer.dbName=DBS3, the location name in DDF (refer to Figure 14 on page 35 for an explanation)
5. Manufacturer.dbProtocol=db2os390, JDBC subprotocol
6. Manufacturer.dbPoolName=JdbcDb2
7. Manufacturer.JDBCDriver=ibm.sql.DB2Driver, the JDBC driver name
8. Manufacturer.BackGround= /IBMWebAS/samples/images/background.jpg (the background picture for the output page could be optionally specified)

Step 3. Run the sample.

Use your browser to open the samples index page and select the servlet under Basic Servlets, Query a database (JDBC). The sample lets you display, add and update records in the manufacturer database.

4.5.10 JDBC manufacturer JSP sample

This sample uses Java Server Pages (JSPs) and Java Beans to read and update a DB2 database and provides a sample for a manufacturer database.

JSPs have the advantage that business logic (Java Beans) and presentation (HTML in the JSPs) are separated. With this technology you are able to use standard tools to create JSPs for your HTML output without recompiling the Java code if the presentation requires changes.

To run the sample, complete the following steps:

Step 1. Create the sample manufacturer database provided with this sample (if not already done).

Refer to Step 1 in 4.5.8, “SQLJ manufacturer servlet” on page 57 for details.

Step 2. Edit the login.properties file (if not already done).

Edit the *server_model_root*/servlets/login.properties file. This sample uses the same values as the JDBCServlet sample. We had the following values specified (both JDBC manufacturer servlet and JDBC manufacturer JSP sample use the same JDBC settings):

1. Manufacturer.dbOwner=WEBHELEN
2. Manufacturer.dbUserid=null
3. Manufacturer.dbPassword=null
4. Manufacturer.dbName=DBS3, the location name in DDF (Refer to Figure 14 on page 35 for an explanation.)
5. Manufacturer.dbProtocol=db2os390, JDBC subprotocol
6. Manufacturer.dbPoolName=JdbcDb2
7. Manufacturer.JDBCDriver=ibm.sql.DB2Driver, the JDBC driver name
8. Manufacturer.BackGround= /IBMWebAS/samples/images/background.jpg (The background picture for the output page could be optionally specified.)

Step 3. Run the sample.

Use your browser to open the samples index page and select the servlet under Basic Servlets, Query a database (JDBC). The sample lets you display, add and update records in the manufacturer database.

4.5.11 Run the samples with the new Type 2 Driver

All previous samples used the driver that was delivered with DB2 for OS/390. This chapter explains the steps to implement the new driver.

With the new Type 2 Driver that is introduced in 3.1.4, “Settings for the new Type 2 SQLJ/JDBC Driver” on page 23 the use of the *db2profc* utility changed. The pgmname specified is now restricted to 7 characters. The reason for this is that four DBRM members are created for all possible isolation levels (which is automatically done for the DSNJDBC package).

The new *sqlj* utility now does all the steps necessary to create the serialized profile, the Java code, and the executables (Steps 2 and 3 in 4.5.7, “Steps for compiling and running an SQLJ servlet” on page 55) out of the *.sqlj source.

In order to run all described database samples in this book with the new driver, the following steps are required:

Step 1. Rerun the db2profc utility for all your SQLJ servlets and bind the packages. This will create the following four DBRM members for each isolation level:

- *THENAME*1 for isolation level UR
- *THENAME*2 for isolation level CS
- *THENAME*3 for isolation level RS
- *THENAME*4 for isolation level RR.

Bind the package THENAME to include all four DBRMs with the specified isolation levels. Make sure that the execute rights for this package have been set to suit your environment’s needs.

Step 2. Make sure that the package DSNJDBC created by the DBRMs from the new driver exists.

Check if the package includes the following DBRMs bound with the right isolation level (ask your DBA for support):

- *DSNJDBC*1 for isolation level UR
- *DSNJDBC*2 for isolation level CS
- *DSNJDBC*3 for isolation level RS
- *DSNJDBC*4 for isolation level RR.

Create the package if it is not already bound. If you are not sure whether the package is from the new or from the old driver, run the db2genJDBC

utility from the new driver and bind the package with the DBRMs and isolation levels mentioned in this step.

Step 3. Bind all packages into one plan.

Bind the DSNJDBC package and all SQLJ packages created for your servlets into one plan. Make sure that you do not override special qualifiers that your packages might contain while binding the plan. Do not forget to grant execution rights for that plan to meet your environment's needs.

Note: Previously granted rights are lost when replacing or freeing packages or plans.

Step 4. Update your *db2sqljjdbc.properties* file.

Change the SQLJPLANNAME environment variable to the name of the plan that contains all SQLJ packages and the default DSNJDBC package.

Step 5. Check PATH, LIBPATH and CLASSPATH in your environment (was.conf, your personal profile).

Change the PATH, LIBPATH and CLASSPATH environment variables, if the new driver is in a different directory than the old or default driver. Your system programmer may have chosen the option to use both drivers concurrently or to install the new driver in a different directory for fallback reasons.

Step 6. Restart the Web server to activate the changes.

Important: If you experience strange behavior of your servlets, check to make sure that they meet the JDBC 1.0 specification.

4.6 IMS samples

IMS servlet samples using ITOC and OTMA Callable Interface assume IMS, OTMA and ITOC (IMS Connect Feature in Version 7) to be started and customized as described in 3.3, "Enable IMS for the IBM WebSphere Application Server" on page 25.

4.6.1 IMS JITOC servlet sample

This sample allows the user to access the IMS IVP (Installation Verification Procedure) phone book transactions. These transactions are delivered with IMS. Check that the corresponding transactions (IVTNV, IVTNO) and database (IVPDB2) are up and running. The ITOC or IMS Connect Feature

needs to be configured and running as well. This sample uses class libraries that are only delivered with VisualAge for Java Enterprise Edition V3.

The IMS JITOC Servlet Sample uses the VisualAge for Java classes as described in 3.3.1, “Enabling JITOC (Java ITOC) interface” on page 26. These class libraries need to be added to the ncf.jvm.classpath statement in was.conf.

The jar file that contains the EAB commands that the sample uses is located in the directory *server_model_root*/samples/JTCV83. It must be added to the ncf.jvm.classpath in was.conf as well.

Restart the server in order to get the changes applied.

For running the sample you need to know the following information, which needs to be typed in the mask that invokes the servlet:

- ITOC host name or IP address
- ITOC port number
- IMS datastore ID
- Valid RACF user ID and password to access IMS (not needed if OTMA security is set to NONE)

Known Problems:

The servlet is not perfect in catching all errors that might occur -- for example, if the transaction is stopped, or in some security violation scenarios when IMS sends messages, including NACK.

4.6.2 IMS OTMA Callable Interface command sample

This sample allows the user to test with OTMA CI (OTMA Callable Interface), which directly connects to OTMA not using the ITOC.

It is assumed that the OTMA CI sample classes are configured as described in Appendix D of *IMS/ESA V6 OTMA Guide and Reference,* SC26-8743.

To run the sample you need to know the following information, which needs to be typed in the Formula that invokes the servlet (for the XCF information, refer to your system programmer or issue a /DIS OTMA command):

- XCF group name, where your IMS is server
- Your IMS XCF member name
- Your unique XCF client member name

- The IMS command and its keywords

We recommend that you use this interface with some (not many) OTMA clients that have multiple sessions. XCF supports up to 256 clients in one group and up to 999 sessions per client. This means that session management should be implemented. However, our samples don't have a session management included.

Known problems:

- You need to apply an APAR, otherwise you will get abends while freeing the session in the sample. Refer to PMR PQ35669.
- The command cannot be longer than 8 characters (including the slash), due to a restriction in the OTMA_allocate call. So the short commands should be used (like DBR instead of DBRECOVERY). Keywords should be specified in the keywords field. There are no restrictions.
- Check your IMS security in order to be able to issue IMS commands using the OTMA interface. IMS default security prohibits issuing commands.
- Some commands, such as /DIS TRAN ALL, return segments that are too long for OTMA CI to handle, and you will get errors instead of a result. Sessions will be shown with the /DIS TMEMBER TPIPE ALL command.
- The Java classes do not provide security parameters to be passed to IMS (although the OTMA CI API does). The Web server started task user ID is used and should be authorized in IMS.

You might display a successful connect to IMS by issuing the /DIS OTMA command in IMS. It will show a list with all XCF clients, even if they are inactive now (IMS XCF information is highlighted in bold):

```
.
R 059,/DIS OTMA
IEE600I REPLY TO 059 IS;/DIS OTMA
DFS000I     GROUP/MEMBER      XCF-STATUS   USER-STATUS    SECURITY
IMSA
DFS000I     IMSXCF
IMSA
DFS000I     -IMSAV7           ACTIVE       SERVER         NONE
IMSA
DFS000I     -HWSMEM           ACTIVE       ACCEPT TRAFFIC
IMSA
DFS000I     -WEBHELEN         NOT DEFINED  DISCONNECTED
IMSA
DFS000I     -OTMACLXX         NOT DEFINED  DISCONNECTED
IMSA
DFS000I     -OTMACLX1         ACTIVE       ACCEPT TRAFFIC
IMSA
DFS000I     -OTMACL7          NOT DEFINED  DISCONNECTED
IMSA
DFS000I     *00068/211036*   IMSA
061 DFS996I *IMS READY*  IMSA
.
```

4.6.3 IMS OTMA CI servlet sample

This sample allows the user to test to OTMA CI , which directly connects to OTMA not using the ITOC. This servlet includes the code to generate the HTML. A better way is to separate the business logic from the presentation. This can be done using an HTML editor to create Java Server Pages (JSPs) and to use Java beans to handle the communication (e.g., a data bean and an OTMA CI communication bean). Otherwise, as in this example, you need to change the Java source code every time the HTML needs to be updated or changed.

It is assumed that the OTMA CI sample classes are configured as described in Appendix D of *IMS/ESA V6 OTMA Guide and Reference,* SC26-8743.

Step 1. Make sure the necessary transactions and database are available.

This sample uses the DSPALLI, CLOSE and DISBURSE IVP transactions, accessing the DI21PART database, that are delivered with IMS. Make sure that the transactions and PSBs are started and the database DI21PART is available.

Step 2. Edit the login.properties file.

Edit the *server_model_root*/servlets/login.properties file. You have to specify the OTMA connection properties. We had the following values specified:

1. OTMAServlet.XCFGroupName=IMSXCF
2. OTMAServlet.IMSMemberName=IMSAV7
3. OTMAServlet.ClientMemberName=WEBHELEN

Step 3. Run the sample.

4.7 MQSeries sample

MQSeries samples assume MQSeries to be configured and the queue manager to be started and customized as described in 3.4, “Enable MQSeries for the IBM WebSphere Application Server” on page 27.

4.7.1 MQSeries servlet sample

The MQSeries sample uses the OS/390 bindings connection to directly connect to the queue manager. The sample allows you to specify the queue manager name to connect to, the queue to use, and the message to be sent and received on the input form HTML page.

In order to run the sample, you’ll need a queue that is get enabled, put enabled, and shared. We used the following settings to define the queue and to run the sample successfully. The most important values for creating the queue are specified here (the output was generated using the MQSeries CSQOREXX panels):

```
.
Queue name  . . . . . . . . . CSQ2.TESTQUEUE
Description . . . . . . . . : Testqueue
Put enabled . . . . . . . . : Y  Y=Yes,N=No
Get enabled . . . . . . . . : Y  Y=Yes,N=No
Usage . . . . . . . . . . . : N  N=Normal,X=XmitQ
Storage class . . . . . . . : DEFAULT
Permit shared access  . . . : Y  Y=Yes,N=No
Default share option  . . . : S  E=Exclusive,S=Shared
Index type  . . . . . . . . : C  N=None,M=MsgId,C=CorrelId
```

This sample uses the correl ID and the message ID internally to put and get the message and only your message back. It will work in a concurrent user environment as well. For further details refer to the source code.

4.8 CICS Transaction Gateway sample

The CICS Transaction Gateway (CTG) sample assumes you have set up the CICS Transaction Server, the Web server, and also the Application server in line with the guidance in 3.2, “Enable CICS for the IBM WebSphere Application Server” on page 24.

4.8.1 CICS TG servlet sample

The CICS sample uses the CTG class ECIRequest. This is documented in *CICS Transaction Server for OS/390 CICS Internet Guide Release 3* , SC34-5445.

The sample is trivial. It takes three input fields from the Web page that invokes it and passes the name of a CICS program and the contents of a communication area to the CICS region specified on the Web page.

The return codes and return string from the ECIRequest are given as response.

By default, the program called is READCO. This should be modified to any locally available CICS program.

4.9 More samples

More samples can be found on a CD attached to *e-business Application Solutions on OS/390 Using Java: Samples,* SG24-5365 or from our FTP server directly at:

`ftp://www.redbooks.ibm.com/redbooks/SG245365/`

Additional samples can be found on a CD attached to *Java Programming Guide for OS/390,* SG24-5619 or from our FTP server directly at:

`ftp://www.redbooks.ibm.com/redbooks/SG245619/`

Chapter 5. WAS advanced configuration

This chapter contains more advanced WebAS configuration information that is not covered in the basic setup in Chapter 2, “IBM WebSphere Application Server configuration” on page 5. We recommend that you work through that chapter before going on with this one. The bulk of this chapter deals with configuration useful in development environments.

5.1 Automatic servlet reloading

Turning on automatic servlet reloading is only recommended for development environments, as it requires considerable resource overhead. The reason for this is that as soon as a servlet in one of the reload directories is modified, all of the servlets that have been loaded from a reload directory are also reloaded.

You turn on automatic servlet reloading by setting the servlets.reload property to “true” in was.conf. You can then list all the directories from which you want servlets to be automatically reloaded using the servlets.reload.directories property.

The search sequence used by WebAS for servlets is as follows:

1. The directories listed in the ncf.jvm.classpath property
2. The directories listed in the servlets.reload.directories property
3. The contents of the server_model_root/servlets directory

You must be aware, though, that automatic reload only works for 2. and 3. If any of your reload directories are also in your ncf.jvm.classpath, they will not be subject to automatic reloads. There is no need to place your /servlets directory in servlets.reload.directories, because it is always searched by default and always subject to automatic reloads if servlets.reload is set to true.

Automatic reloading takes place immediately when a servlet loaded from a reload directory is modified. This typically happens when a servlet is recompiled. However, it could also happen when a different version of a servlet is copied into the reload directory from elsewhere. You do not have to reinvoke the servlet to initiate the reload.

You can see, in Figure 22 on page 74, the activity that takes place when the reload starts, servlets are destroyed and reinitialized. This log is found in file event_log.<server_start_date>, in

server_model_root/logs/servlet/servletservice/ . For more information about Application server logging and tracing, see 5.2, “Logging Application Server events and errors” on page 75.

```
 BROWSE -- event_log.Mar202000                        Line 00000019 Col 033 112
 Command ===>                                                  Scroll ===> CSR
[March 22, 2000 1:43:09 PM EST] ServletManager.instantiateServlet: Loaded local class c
CICStest.
[March 22, 2000 1:43:09 PM EST] CICStest: init
[March 22, 2000 1:43:09 PM EST] ServletManager.loadServlet CICStest: class = CICStest c
=<none> arguments = <none>
[March 22, 2000 1:43:09 PM EST] HelloWorldServlet: destroy
[March 22, 2000 1:43:09 PM EST] ServletManager.instantiateServlet: Loaded local class c
HelloWorldServlet.
[March 22, 2000 1:43:09 PM EST] HelloWorldServlet: init
[March 22, 2000 1:43:09 PM EST] ServletManager.loadServlet HelloWorldServlet: class =
HelloWorldServlet class URL =<none> arguments = <none>
[March 22, 2000 1:43:09 PM EST] com.sun.server.http.InvokerServlet: destroy
[March 22, 2000 1:43:09 PM EST] ServletManager.instantiateServlet: Loaded local class c
com.sun.server.http.InvokerServlet.
[March 22, 2000 1:43:10 PM EST] com.sun.server.http.InvokerServlet: init
[March 22, 2000 1:43:10 PM EST] ServletManager.loadServlet invoker: class =
com.sun.server.http.InvokerServlet class URL =<none> arguments = <none>
[March 22, 2000 1:43:10 PM EST] Invalidated sessions due to classloader reload
```

Figure 22. A fragment of servletservice/event_log showing servlets reloading

5.1.1 Reloading servlets from JAR files

We tested the automatic reloading of servlets contained in JAR files. Our setup was simple, consisting of a single JAR file containing two servlets, as follows:

```
ITSO3B:/was3/itso3b/AppServer/servlets/CICS: >jar -tf ../CICStestJ.jar
META-INF/MANIFEST.MF
CICStest.class
CICStestZ.class
```

The JAR file resided in our server_model_root/servlets directory, and we had servlets.reload set to true.

We found that once a servlet from the JAR file was loaded by the Application server, then if a change was made to the JAR file, the expected reload processing took place.

So, when we recompiled CICStestZ and rebuilt the JAR file, reload processing started. The Application server did not reload the servlets from the new JAR file until they were called for.

5.2 Logging Application Server events and errors

There are two sections in *WebSphere Application Server for OS/390 Application Server Planning, Installing, and Using V1.2*, GC34-4757 that give details on how to turn on tracing and logging for the Application server. The sections are “Configuration settings: Logging settings for error and event messages” and Appendix B, “Producing error logs for IBM support personnel”.

We republished some chapters of this publication in Appendix D of this book. See C.4, “Producing error logs for IBM support personnel” on page 145.

These should be used in conjunction with the tracing available within the Web server.

Web server tracing options are detailed in *IBM HTTP Server for OS/390 HTTP Server Planning, Installing, and Using Version 5.2,* SC31-8690, which is available on the Web at:

`http://www.ibm.com/software/webservers/httpservers/doc/v52/icswgmst.html`

There are also some details in the comments at the top of the default HTTP server started task.

Another important source of information about Application server log and trace settings is in the comments was.conf file, primarily in sections ”Log settings” and ”Trace settings”.

It is important to remember, once again, that WebAS logging options are controlled by properties in was.conf. So after you have made changes you must stop the Web server, run updateproperties, and restart the Web server.

The two logging options we found most useful when modifying and testing our samples were native DLL logging and Java Standard Out logging.

Native DLL logging is controlled by the boolean ncf.native.logison, and the Java Standard Out is controlled by two booleans. The first sets logging on or off, ncf.jvm.stdoutlog.enabled, the second specifies whether it is written to a file, or to the Java console, ncf.jvm.stdoutlog.file.

All log path and file names are customizable.

5.3 Java Server Pages (JSP) support

In order for your Web server to support Java Server Pages (JSPs are a freely available specification for extending the Java Servlet API to generate dynamic Web pages on a Web server) you need to have the following parameter coded in httpd.conf:

```
Service /*.jsp     server_model_root/lib/libadpter.so:AdapterService
```

You almost certainly added this during your Application server setup, described in 2.2.6, “Updating httpd.conf” on page 9.

The first time a JSP is accessed, WAS will compile it to a servlet. WAS is using the server_model_root/servlets/pagecompile directory as a working directory.

Make sure to allow WAS to create subdirectories and write into pagecompile. This can be done by either setting the pagecompile directory permission bits to 777, or by setting the owner of the directory to the anonymous user ID of the Web server (usually PUBLIC) and the permission bits to 775 or 755. This implies that JSPs are called using the user ID of PUBLIC.

Write access is only needed the first time a JSP is called, to compile the JSP. You may use the method to change the permission bits to 777 for this occasion, and you are then referencing all JSPs. After that is completed, you might change the permission bits back to the original value.

If you change a JSP, you first need to have write access again (because it will be recompiled). In some circumstances we also found that if you change a JSP, you might need to delete the appropriate files in the pagecompile directory manually before accessing the changed JSP (which will then start a compile again).

Chapter 6. Performance tuning

This chapter describes performance information related to the IBM WebSphere Application Server (WAS or WebAS).

In general, WAS performance is affected by end-to-end components and subsystems that provide application services. These can be software, hardware, and the networking environment.

In the WAS environment, the application services are mainly affected by external components and subsystems, since WAS is constructed merely by adding a servlet engine to the Web server.

This also implies that resources of server processor cycles and storage can be important to WAS performance, as can Java servlet codes that can cause a long-wait or deadlock situation due to wrong handling of resource protection mechanisms of the Java servlet engine.

This book is designed to be used in conjunction with *OS/390 e-business Infrastructure: IBM HTTP Server 5.1 - Customization & Usage*, SG24-5603. Refer to that book for Web server performance-related items.

This chapter is a presentation guide that uses a presentation about IBM WebSphere Application Server 1.2 tuning by Roland Trauner. You may obtain the latest version of this presentation over the Internet using a Web browser from the following FTP site:

`ftp://www.redbooks.ibm.com/redbooks/ebiz390/como.html`

6.1 IBM WebSphere Application Server tuning

We list here the topics of the presentation and add some comments if needed.

Components to tune:

- OS/390
- OS/390 UNIX
- DFSMS
- WLM, SRM
- VTAM, TCP/IP
- Java
- IBM HTTP Server
- IBM WebSphere Application Server
- LE environment

Objectives:

- OS/390 R8
- IBM HTTP Server 5.2
- IBM WebSphere Application Server 1.2
- Java /390 1.1.8
- WAS is the only business application on the system
  - No mixed workload, i.e. WAS & Domino /390
- Hardware (CPU and I/O) is not discussed
  - MIPS, Memory, Network Adapters, I/O cache etc.

This is the environment for which the tuning recommendations are targeted.

6.1.1 Basic OS/390 tuning

Language environment runtime

- Put SCEERUN in LINKLIST (SYS1.PARMLIB)
- Put SCEELPA in LPALST (SYS1.PARMLIB)
- Include the following modules in Dyn. LPA:

```
INCLUDE LIBRARY(CEE.SCEERUN) MODULES(CEEBINIT CEEBLIBM CEEEV003 EDCZV
EDCZ24 )
```

- Include the following modules in MLPA (SYS1.PARMLIB(IEALPAxx)):

```
INCLUDE LIBRARY(SYS1.LINKLIB) MODULES(IEFIB600 IEFXB603)
```

Source: *OS/390 V2R8 UNIX System Services Planning*, SC28-1890

Check the following performance Web site:

```
http://www.s390.ibm.com/oe/perform/bpxpftgt.html
```

Make the Web server non-swappable

Customize SCHEDxx (SYS1.PARMLIB):

```
PPT PGMNAME(IMWHTTPD) NOSWAP
```

Keep in mind that if you change the module name for the Web server (see LE setup), you need to add that module name here.

UID and GID caching

Use VLF caching for OE UIDs and GIDs (SYS1.PARMLIB(COFVLFxx)):

```
CLASS NAME(CSVLLA)EMAJ(LLA)
CLASS NAME(IRRUMAP) EMAJ(UMAP)
CLASS NAME(IRRGMAP)      EMAJ(GMAP)
CLASS NAME(IRRGTS)       EMAJ(GTS)
CLASS NAME(IRRACEE)      EMAJ(ACEE)
```

```
CLASS NAME(IRRSMAP) EMAJ(SMAP)
```

Make sure VLF is started.

Source: *OS/390 V2R8 UNIX System Services Planning*, SC28-1890

Check the following performance Web site:

http://www.s390.ibm.com/oe/perform/bpxpftgt.html

6.1.2 OS/390 UNIX tuning

Basic setup

Customize BPXPRMxx (SYS1.PARMLIB):

```
MAXPROCSYS(5000)    /* Allow 5000 processes to be active concurrently */
MAXPROCUSER(500)    /* Allow each user (same UID) to have 500   */
                    /* concurrent processes active              */
MAXUIDS(200)        /* 200 concurrent users                     */
MAXFILEPROC(10000)  /* 10000 open files per user                */
MAXTHREADTASKS(3000)/* Allow 3000 threads tasks to be active    */
                    /* concurrently in a single process         */
MAXTHREADS(6000)    /* Allow 6000 threads to be active          */
                    /* concurrently in a single process         */
```

Use the RMF Kernel activity report to adjust the options.

These parameters influence the entire UNIX environment. The values should be set to meet the requirements of the combined peak demand for all UNIX applications.

More BPXPRMxx customization (SYS1.PARMLIB)

Change the values of the XPG4 Interprocess Communications definitions:

```
IPCMSGNIDS    (20000)     /* default 500 */
IPCMSGQBYTES  (262144)    /* default     */
IPCMSGQMNUM   (10000)     /* default     */
IPCSHMNIDS    (20000)     /* default 500 */
IPCSHMSPAGES  (2621440)   /* default     */
IPCSHMMPAGES  (25600)     /* default 256 */
IPCSHMNSEGS   (1000)      /* default 10  */
IPCSEMNIDS    (20000)     /* default 50  */
IPCSEMNSEMS   (32767)     /* default 25  */
IPCSEMNOPS    (32767)     /* default 25  */
```

Disable all tracing

Customize CTIBPXxx (SYS1.PARMLIB):

- Turn TRACEOPTS OFF

- OS/390 UNIX CTRACE options

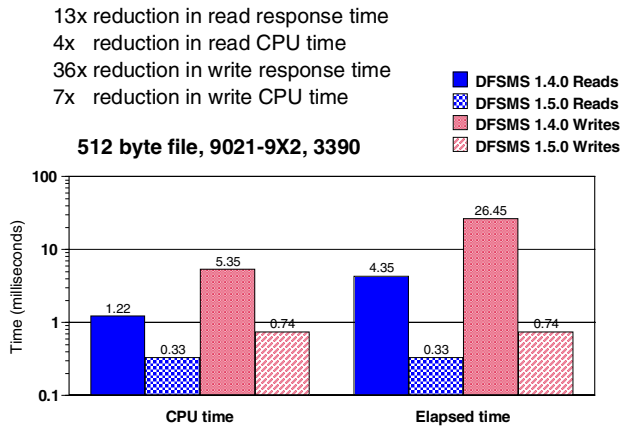
DFSMS 1.5 HFS setup

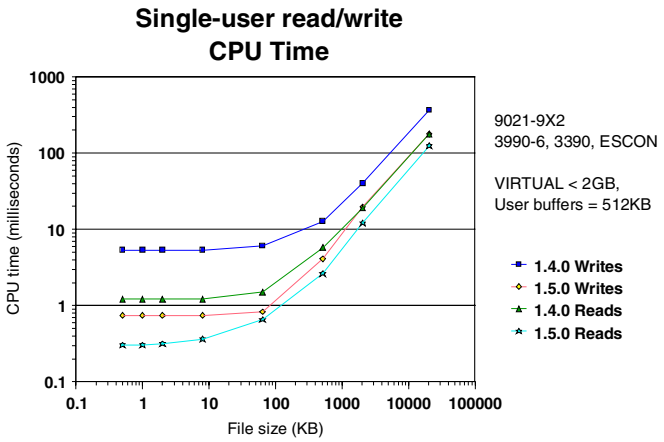
Customize BPXPRMxx (SYS1.PARMLIB) or use the confighfs command.

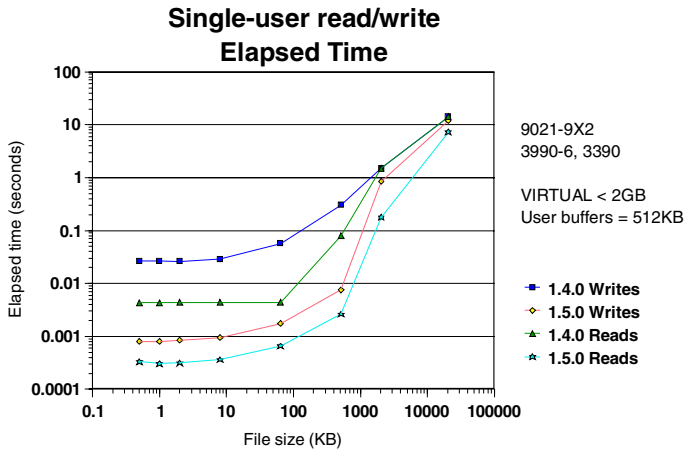
- Set VIRTUAL and FIXED values.
  - Determine the values using the RMF - HFS Postprocessor report.
  - See *Hierarchical File System Usage Guide*, SG24-5482 for recommendations.
  - The default for FIXED is 0. Fix at least 10 MB. Max is 50% of real storage.
- Define an in-storage file system (TFS) for /tmp.

Note: A Web server/WebSphere production environment mostly reads data.

The deferred-write capability of DFSMS 1.5 is valuable for log files if they are needed.







Source: *Hierarchical File System Usage Guide*, SG24-5482

http://www.redbooks.ibm.com/abstracts/sg245482.html

DFSMS 1.5; result of confighfs query:

```
TRAUNER:/u/trauner: >confighfs -q
HFS Statistics
( 03/16/00 2:33pm )
 Virtual storage:     _____41984(pages)
                      _______164(MB)
 Fixed storage:       ______2423(pages)
                      _9.4648438(MB)
 Lookup cache hit:    _____________3580652
```

```
Lookup cache miss:  ________________52937
1st data page hit:  _______________8608559
1st data page miss: ________________69826

Pool  Size #DS   BP_pages      Fixed   Already_fixed  Not_already_fixed
   1 ____1 __1 _____12868 ______2027 _____99261 ___2904923
   2 ____4 __1 ______4844 _______108 _____18022 ____316902
   3 ___16 __1 ______7312 _______288 _______238 ______7083
   4 ___64 __1 _____16960 _________0 _____44112 _____12468
```

RACF Setup

Enable the RACF UNIXMAP class:

```
SETROPTS CLASSACT(UNIXMAP)
```

- UNIXMAP reduces the problem of "invalid" UIDs and GIDs.
- Check if all HFS files are owned by a valid UID and GID.
  - Valid means: having a RACF user ID assigned.
  - Command to find these files:

    ```
    find / -nouser -nogroup
    ```

Enable SAFFASTPATH:

```
RDEFINE FACILITY BPX.SAFFASTPATH UACC(NONE)
```

This limits RACF checking for HFS access.

Source: *OS/390 V2R8 UNIX System Services Planning*, SC28-1890

SMF recording

Limit SMF recording as much as possible.

Turn off SMF type 92 recording (HFS access recording).

Customize BPXPRMxx (SYS1.PARMLIB).

6.1.3 Workload Manager

Adjust the dispatching priorities

Set up the appropriate policies for WLM goal mode.

Set up IEAIPSxx and IEAICSxx for compatibility mode.

Recommendations:

- Keep in mind what the "service" chain is:

  1) VTAM

2) TCP/IP

3) OMVS

4) Web server

6.1.4 VTAM and TCP/IP tuning

Communication Storage Manager (CSM)
Customize IVTPRMxx (SYS1.PARMLIB)

Typical values are:

```
FIXED MAX(120M)
ECSA MAX(120M)
POOL (4K,ECSA,200,20,100)
POOL (4K,DSPACE,200,20,100)
POOL (16K,DSPACE,50,10,50)
POOL (32K,ECSA,20,10,20)
POOL (60K,DSPACE,1,1,6)
POOL (180K,DSPACE,1,1,2)
```

Source: *CSM Guide*, SC31-8575

TCP/IP tuning for WebSphere
Customize TCPIP.PROFILE:

- Increase the Send Buffer Pool Size depending on the average data you send from your Web environment:

  ```
  TCPCONFIG RESTRICTLOWPORTS
  TCPSENDBFRSIZE   65535
  ```

- Disable the SNMP subagent:

  ```
  SACONFIG DISABLE
  ```

- Increase the connection request queue (default 10):

  ```
  SOMAXCONN 1000
  ```

Source: *IP Configuration Guide*, SC31-8513

6.1.5 Java tuning

Java I/O tuning
Most Java functions can only be tuned in the IBM WebSphere Application Server.

The only thing you can tune is to explicitly cache Java classes and executables.

- Explicitly means: if you want to do special caching besides the DFSMS HFS caching.
- Use the filecache command. The following definitions are part of the /etc/rc script:

```
/usr/sbin/filecache -p
/usr/sbin/filecache -a /usr/lpp/java18/J1.1/bin/*
/usr/sbin/filecache -a /usr/lpp/java18/J1.1/lib/classes.zip
/usr/sbin/filecache -a /usr/lpp/db2/db2610/lib/*
/usr/sbin/filecache -a /usr/lpp/db2/db2610/classes/*
```

You may also cache critical JAR files, application "bins," etc.

Java for S/390 source:

```
http://www.s390.ibm.com/java
```

6.1.6 IBM HTTP Server tuning

General setup

Configure httpd.conf:

```
DNS-Lookup off                  DNS resolution for the log
imbeds off                      Server Side Includes
# AccessLog  /web/apple/logs/httpd-log Disable all logging
Welcome  index.html      Check the order of the welcome files
UseMetaFiles off         Don't use meta files
UseACLs never            Don't use access control lists
SMF none                 Don't write SMF records
SNMP off                 Disable SNMP
ServerPriority -19       20 is highest
MaxActiveThreads 120     adjust the threads to your demands
```

Source: *IBM HTTP Server Customization and Usage*, SG24-5603.

```
http://www.redbooks.ibm.com/abstracts/sg245603.html
```

Enable the Fast Response Cache (FRCA)

Configure httpd.conf:

```
EnableFRCA on            Enables FRCA
FRCACacheSize 25000      # of 4k blocks , ca 100MB
FRCACacheEntries 1000    Max amount of files in cache
FRCAMaxFileSize 200k     Max File Size to be cached
FRCAStackName TCPIP      Name of TCP/IP stack; Match BPXPRMxx
FRCAVirtualHost auto     Serve multiple IP adresses
```

FRCA caches only static, unprotected resources.

For protected resources, use *CacheLocalFile*.

Display an FRCA cache example:

```
D TCPIP,TCPIPOE,NET,CACH
EZZ2500I NETSTAT CS V2R8 TCPIPOE 673
CLIENT: WEB3C           LISTENING SOCKET:  0.0.0.0..7103
  MAXCACHESIZE:       0000025000  CURRCACHESIZE:      0000000035
  MAXNUMOBJECTS:      0000001000  CURRNUMOBJECTS:     0000000017
  NUMCONNS:           0000000011  CONNSPROCESSED:     0000000000
  CONNSDEFERRED:      0000000011  CONNSTIMEDOUT:      0000000000
  REQUESTSPROCESSED:  0000000000  INCOMPLETEREQUESTS: 0000000000
  NUMCACHEHITS:       0000000000  NUMCACHEMISSES:     0000000011
  NUMUNPRODCACHEHITS: 0000000000
CLIENT: WEBHELEN        LISTENING SOCKET:  0.0.0.0..8890
  MAXCACHESIZE:       0000025000  CURRCACHESIZE:      0000000131
  MAXNUMOBJECTS:      0000001000  CURRNUMOBJECTS:     0000000051
  NUMCONNS:           0000000086  CONNSPROCESSED:     0000000035
  CONNSDEFERRED:      0000000051  CONNSTIMEDOUT:      0000000000
  REQUESTSPROCESSED:  0000000091  INCOMPLETEREQUESTS: 0000000000
  NUMCACHEHITS:       0000000091  NUMCACHEMISSES:     0000000044
  NUMUNPRODCACHEHITS: 0000000000
2 OF 2 RECORDS DISPLAYED
```

TCP/IP parameters

Configure httpd.conf:

```
ListenBacklog 250        Listen queue < than TCP/IP SOMAXCONN
InputTimeout 20 secs     Receive
OutputTimeout 90 secs    Send
ScriptTimeout 30 secs    Server Scripts
PersistTimeout 2 secs    Persistent connections. Start low
MaxPersistRequests 2     Amount of objects within persistent connection
```

Streamline the configuration

Remove all definitions that are not needed for your Web application:

- Disable all Proxy items; configure httpd.conf:

```
#   ====================================================
#   *** OS/390 Web Traffic Express Support directives ***
#   ====================================================

# ServerInit          /usr/lpp/internet/bin/Jav_dll.so:Javelin_init
# Service /cgi-bin/dogc.icapi /usr/lpp/internet/bin/Jav_dll.so:doGC
# PreExit             /usr/lpp/internet/bin/Jav_dll.so:Javelin_preFilter
# Enable  ICSERRORLOG /usr/lpp/internet/bin/Jav_dll.so:Javelin_errorLog
# Pass /reports/javelin/*   /usr/lpp/....
```

- Disable service functions; configure httpd.conf:

```
# service /cgi-bin/htimage*        INTERNAL:HTImage*
# service /cgi-bin/imagemap*       INTERNAL:HTImage*

# service /Usage*                  INTERNAL:UsageFn
# service /admin-bin/trace*        INTERNAL:TraceFn
```

- Disable the Web admin and reporting; configure httpd.conf:

```
# Pass /icons/*            /usr/lpp/internet/server_ro...
# Pass /Admin/*.jpg        /usr/lpp/internet/server_ro...
# Pass /Admin/*.gif        /usr/lpp/internet/server_ro...
# Pass /Admin/*.html       /usr/lpp/internet/server_ro...
# Pass /Docs/*             /usr/lpp/internet/server_ro...
# Pass /reports/java/*     /usr/lpp/internet/server_ro...
# Pass /reports/*          /web3/itso3c/reports/*
# Pass /img-bin/*          /usr/lpp/internet/server_ro...
```

- Disable the Web admin and reporting protection; configure httpd.conf:

```
# Protection  IMW_Admin  {
#      ServerID     IMWEBSRV_Administration
#      AuthType     Basic
#      PasswdFile   %%SAF%%
#      ACLOverride  Off
#      Mask         WEBADM,webadm
# }
#
# Protect /admin-bin/* IMW_Admin WEBADM
# Protect /reports/*   IMW_Admin WEBADM
# Protect /Usage*      IMW_Admin WEBADM
```

Check statement order

- If you have protected resources, put the PASS and EXEC statements for the unprotected resources before the PROTECT statement.
- Put the PASS and EXEC statements for the protected resources after the PROTECT statement.
- Caution: PASS /* must always be last!

Environment variables

Optimize the path library concatenation; configure httpd.envvars:

- PATH

```
1) /usr/lpp/internet/bin
```

```
2) /usr/lpp/internet/sbin
3) /usr/lpp/java/J1.1/bin
4) /usr/sbin
5) /bin
6) /usr/lpp/ldap/bin (if needed, check /bin for symlinks)
```

- NLSPATH

```
1) /usr/lpp/internet/%L/%N
2) /usr/lib/nls/msg/%L/%N
3) /usr/lpp/ldap/lib/nls/msg/%L/%N  (if needed,
                                    check /usr/lib/nls/msg for symlinks)
```

- LIBPATH
  - Define the libraries used by the Web server only. All WAS directories should be defined in was.conf.

```
1) /usr/lpp/internet/bin
2) /usr/lpp/internet/sbin
3) /usr/lpp/java/J1.1/mvs/native_threads
4) /usr/lpp/ldap/lib
```

- CLASSPATH
  - Define the libraries used by the Web server only. All WAS directories should be defined in was.conf.

```
1) /web/apple/mycgi
2) /usr/lpp/java/J1.1/lib/classes.zip
```

- Other environment variables:

```
GSK_SSL_HW_DETECT_MESSAGE=1   To indicate if the HW
                             crypto feature is used
_EDC_IP_CACHE_ENTRIES=50      Web server DNS resolution cache
```

Related discussions

Make sure the Web server is not started using a trace option (i.e. -vv).

Optimize your Web page design:

- Avoid excessive use of "dummy" GIFs to place the content.
- If you need to use SSL, try to avoid graphics in the SSL page.
  - Use Frames to separate encrypted from non-encrypted content.

Store Web pages in ASCII.

Use dedicated Application Web servers:

- Use a Portal server and Application servers such as an SSL server, WAS server DB2, WAS server IMS, etc.
- Use Redirect or "hidden" Proxy to connect these servers.
- Each Web server can be tuned separately.
- WLM characteristics for each Web server can be different.

6.1.7 WAS tuning

Main WAS configuration file:
...server_model_root.../properties/was.conf

Notes to was.conf:

- It is only a "front-end" file.
- It is not used for the server operation.
- Configuration changes done in was.conf require a utility program (updateproperties) to be "activated."
- Updateproperties analyzes was.conf, does some checking, and spreads the configuration parameters into several properties files (see next page).
- Check .../logs/updateproperties.log for details.

Updateproperties changes the following properties files:

- In ...server_model_root/properties/server/servlet/servletservice:

  jvm.properties

  session.properties

  connmgr.properties

  systemDefaults.properties

  servlets.properties

  httpd.properties

  rules.properties
- In ...server_model_root/properties/server/servlet:

  debug.properties

WAS configuration
FAQ:

- Do I need to use WAS.CONF or can I update these properties files manually?

Answer:

- You can update the properties files manually.
- For some configuration changes, you need to update them.

Caution:

- Never update properties in the properties files that are updated by updateproperties, or, in other words, that are defined in was.conf.
- Next time you run updateproperties, your updates are lost.

Configure was.conf:

- Enable functions (true) only if needed

  java.compiler=jitc Enable the JITC (default)

  session.enable=false Just enable the session tracker if needed

  log.error.level=1 Log major problems. That should not occur too often. If so, then you have serious problems that should be fixed rather than turning off this log completely.

  log.event.level=0 Turn off the event log.

  servlets.startup=invoker HelloWorldServlet. You may specify your servlets here. Then they will be loaded at server startup time. If not specified, they will be loaded the first time they are used. Server startup time is faster if no servlets are specified.

  servlets.reload=false Don't enable automatic servlet reloading. Stop/Start the Web server to do so

  ncf.native.logison=false Native DLL logging. Disable the log. This log is very valuable to develop servlets and to set up the server. Once done and working, there should be no need to enable this log.

  ncf.jvm.stdoutlog.enabled=false Java standard outlogging. Same condition as native log. Depends on the amount of messages your servlets produce.

  ncf.native.os390.debug=0 OS/390 UNIX native tracing. Tracing always inhibits performance.

  trace.enable=false WAS internal Java tracing.

Configure server.properties:

...server_model_root.../properties/server/servlet

server.security=false Servlet security manager. Depends on the security requirements.

Configure httpd.properties:

...server_model_root.../properties/server/servlet/servletservice

enable.acls=false Don't use Access Control Lists.

Configure jvm.properties:

...server_model_root.../properties/server/servlet/servletservice

ncf.jvm.mx=67108864 Java maximum heap size (default). Don't limit the heap size here. The heap size is limited by LE parameters linked with the Web server module.

CLASSPATH discussion

The more applications and subsystems you add to WAS, the longer the classpath will be.

Use dedicated function Web servers to reduce the length.

Put the application directories first.

Put the Java classes.zip file last.

Put subsystem jar and zip files after the WAS-provided files (jst.jar etc.). Order: most frequently used first.

If you define ncf.use.system.classpath=true in was.conf, then the CLASSPATH defined in httpd.envvars will be concatenated to ncf.jvm.classpath.

CLASSPATH verification

The native.log.mmmddyyy file displays the “effective” CLASSPATH after the resolution.

```
Opened log file /was3/itso3z/AppServer/logs/native.log.Apr102000.
Try to create a rule base out of /was3/itso3z/AppServer/properties/se
Rule base was created
...
PATH=/usr/lpp/java18p/J1.1/bin:/bin:.:/usr/sbin:/usr/lpp/internet/bin
LIBPATH=/usr/lpp/java18p/J1.1/lib:/was3/itso3z/AppServer/lib:/usr/lib
Attempting to load Java library: libjava.a
Loaded libjava.a and function pointers successfully.
ncf.jvm.classpath = /web3/itso3z/servlets:/was3/itso3z/AppServer/serv
```

```
Opened java VM, classpath = /web3/itso3z/servlets:/was3/itso3z/AppSer
createJVM: About create thread for JVM...
createJVM: pthread_create...errno = 0
createJVM: Sleeping..Zzzzz
After JNI_GetDefaultJavaVMInitArgs /web3/itso3z/servlets:/was3/itso3z
launchCreateJVM: About to loop forever...
createJVM: Out of loop - jvm_ready=1
createJVM: JVM creation was successful!
Run: Successfully created JVM!
Started java VM.
Initialization is done.
```

LIBPATH discussion

The more applications and subsystems you add to WAS, the longer the LIBPATH might become.

Use dedicated Application Web servers to reduce the length.

Be aware that the LIBPATH of httpd.envvars is concatenated to ncf.jvm.libpath. Every library defined in the httpd.envvars is valid for the whole Web server environment and can be omitted in ncf.jvm.libpath.

6.1.8 Language Environment (LE)

Runtime options for WAS

Recommended WAS runtime options:

```
HEAPP(ON)
ALL31(ON)
ANYHEAP(,,,KEEP)
POS(ON)
STACK(200k,16k,ANY,FREE)
STOR(,,,0K)
LIBS(1k,1k,FREE)
BE(400k,50k,FREE)
RPTOPTS(ON)
```

Relink the Web server module with the appropriate LE parameters.

The procedure is described in *IBM HTTP Server Customization and Usage*, SG24-5603.

```
http://www.redbooks.ibm.com/abstracts/sg245603.html
```

Conclusion

Overcome JCL parameter limitations.

Load modules with different LE parameters for different needs are possible:

LMOD WEBAPPLE, WEBBILL etc.

Display the LE options

Use RPTOPTS(ON) to display all the effective LE options.

RPTOPTS sample:

```
Options Report for Enclave main 04/03/00 4:36:20 PM
Language Environment V2 R8.0

LAST WHERE SET              OPTION
-------------------------------------------------------------------
Programmer default            ABPERC(NONE)
Programmer default            ABTERMENC(RETCODE)
Programmer default           NOAIXBLD
Programmer default            ALL31(ON)
Programmer default          ANYHEAP(4194304,1048576,ANYWHERE,FREE)
Programmer default           NOAUTOTASK
Programmer default           BELOWHEAP(409600,51200,FREE)
Programmer default            CBLOPTS(ON)
Programmer default            CBLPSHPOP(ON)
Programmer default            CBLQDA(OFF)
Programmer default            CHECK(ON)
Programmer default            COUNTRY(US)
Programmer default           NODEBUG
Programmer default            DEPTHCONDLMT(10)
Installation default           ENVAR("")
Programmer default    ENVAR("_CEE_ENVFILE=/web3/itso3a/httpd.envvars"
Programmer default               ERRCOUNT(0)
Programmer default               ERRUNIT(6)
Programmer default            FILEHIST
Default setting             NOFLOW
Programmer default      HEAP(4194304,1048576,ANYWHERE,KEEP,4096,4096)
Programmer default            HEAPCHK(OFF,1,0)
Programmer default
              HEAPPOOLS(ON,16,10,48,10,232,10,616,10,1040,10,2048,10)
Programmer default            INFOMSGFILTER(OFF,,,,)
Programmer default            INQPCOPN
Programmer default            INTERRUPT(OFF)
Programmer default            LIBRARY(SYSCEE)
Programmer default           LIBSTACK(1024,1024,FREE)
Programmer default           MSGFILE(SYSOUT,FBA,121,0,NOENQ)
Programmer default            MSGQ(15)
Programmer default            NATLANG(ENU)
Programmer default         NONONIPTSTACK(4096,4096,BELOW,KEEP)
Programmer default            OCSTATUS
Programmer default           NOPC
```

```
Programmer default          PLITASKCOUNT(20)
Programmer default           POSIX(ON)
Programmer default          PROFILE(OFF,"")
Programmer default           PRTUNIT(6)
Programmer default           PUNUNIT(7)
Programmer default           RDRUNIT(5)
Programmer default           RECPAD(OFF)
Programmer default           RPTOPTS(ON)
Invocation command             RPTSTG(ON)
Programmer default           NORTEREUS
Programmer default             RTLS(OFF)
Programmer default           NOSIMVRD
Programmer default             STACK(204800,16384,ANYWHERE,FREE)
Programmer default             STORAGE(NONE,NONE,NONE,8192)
Programmer default             TERMTHDACT(TRACE)
Programmer default           NOTEST(ALL,"*","PROMPT","INSPPREF")
Programmer default             THREADHEAP(4096,4096,ANYWHERE,KEEP)
Programmer default             TRACE(OFF,4096,DUMP,LE=0)
Programmer default             TRAP(ON,SPIE)
Programmer default             UPSI(00000000)
Programmer default           NOUSRHDLR()
Programmer default             VCTRSAVE(OFF)
Programmer default             VERSION()
Programmer default             XUFLOW(AUTO)
```

Display the Storage options

Use RPTSTG(ON) to find out about the storage used in certain areas (heappools etc.). Use the output to adjust the LE storage settings.

Use RPTSTG(ON) only for short-term measurements, then turn it off again. RPTSTG(ON) degrades performance.

RPTSTG sample:

```
Storage Report for Enclave main 04/03/00 4:36:20 PM
Language Environment V2 R8.0

    STACK statistics:
      Initial size:                                      204800
      Increment size:                                     16384
      Maximum used by all concurrent threads:            259876
      Largest used by any thread:                         45704
      Number of segments allocated:                          58
      Number of segments freed:                               0
    NONIPTSTACK statistics:
      Initial size:                                           0
      Increment size:                                         0
```

```
  Maximum used by all concurrent threads:                 0
  Largest used by any thread:                             0
  Number of segments allocated:                           0
  Number of segments freed:                               0
LIBSTACK statistics:
  Initial size:                                        1024
  Increment size:                                      1024
  Maximum used by all concurrent threads:             37536
  Largest used by any thread:                           816
  Number of segments allocated:                           1
  Number of segments freed:                               0
THREADHEAP statistics:
  Initial size:                                        4096
  Increment size:                                      4096
  Maximum used by all concurrent threads:                 0
  Largest used by any thread:                             0
  Successful Get Heap requests:                           0
  Successful Free Heap requests:                          0
  Number of segments allocated:                           0
  Number of segments freed:                               0
HEAP statistics:
  Initial size:                                     4194304
  Increment size:                                   1048576
  Total heap storage used (sugg. initial size):    84067032
  Successful Get Heap requests:                       68473
  Successful Free Heap requests:                      67660
  Number of segments allocated:                          12
  Number of segments freed:                               0
ANYHEAP statistics:
  Initial size:                                     4194304
  Increment size:                                   1048576
  Total heap storage used (sugg. initial size):    13156200
  Successful Get Heap requests:                        1672
  Successful Free Heap requests:                        719
  Number of segments allocated:                          12
  Number of segments freed:                               1
BELOWHEAP statistics:
  Initial size:                                      409600
  Increment size:                                     51200
  Total heap storage used (sugg. initial size):      560456
  Successful Get Heap requests:                          63
  Successful Free Heap requests:                         56
  Number of segments allocated:                           5
  Number of segments freed:                               1
Additional Heap statistics:
  Successful Create Heap requests:                        0
  Successful Discard Heap requests:                       0
```

```
  Total heap storage used:                                  0
  Successful Get Heap requests:                             0
  Successful Free Heap requests:                            0
  Number of segments allocated:                             0
  Number of segments freed:                                 0

HeapPools Statistics:
  Pool 1 size:   16
    Successful Get Heap requests:    1-   8            13147
    Successful Get Heap requests:    9-  16            26149
  Pool 2 size:   48
    Successful Get Heap requests:   17-  24            23903
    Successful Get Heap requests:   25-  32            16618
    Successful Get Heap requests:   33-  40             8912
    Successful Get Heap requests:   41-  48             7307
  Pool 3 size:  232
    Successful Get Heap requests:   49-  56             3955
    Successful Get Heap requests:   57-  64             2284
    Successful Get Heap requests:   65-  72             1644
    Successful Get Heap requests:   73-  80             1661
    Successful Get Heap requests:   81-  88              951
    Successful Get Heap requests:   89-  96              807
    Successful Get Heap requests:   97- 104             5628
    Successful Get Heap requests:  105- 112             5879
    Successful Get Heap requests:  113- 120             7960
    Successful Get Heap requests:  121- 128             6932
    Successful Get Heap requests:  129- 136             5104
    Successful Get Heap requests:  137- 144             4585
    Successful Get Heap requests:  145- 152              699
    Successful Get Heap requests:  153- 160              469
    Successful Get Heap requests:  161- 168              808
    .....
    Successful Get Heap requests: 1977-1984                2
    Successful Get Heap requests: 1985-1992                3
    Successful Get Heap requests: 1993-2000               15
  Requests greater than the largest cell size:          7480
HeapPools Summary:
  Cell  Extent   Cells Per  Extents    Maximum     Cells In
  Size  Percent  Extent     Allocated  Cells Used  Use
  ---------------------------------------------------------
    16    10       17476           1        9272        9168
    48    10        7489           2        9333        9280
   232    10        1747           2        2278        2111
   616    10         672           2         964         617
  1040    10         400           1         334         333
  2000    10         208           1         200         196
  ---------------------------------------------------------
```

```
      Suggested Percentages for current Cell Sizes:
        HEAPP(ON,16,6,48,13,232,14,616,15,1040,9,2000,10)
      Suggested Cell Sizes:
        HEAPP(ON,32,,64,,152,,400,,1040,,2048,)
    Largest number of threads concurrently active:           58
End of Storage Report
```

6.1.9 More tuning information

OS/390 e-business performance page:

http://www.s390.ibm.com/nc/perform.html

Domino Go Webserver for OS/390: Capacity Planning:

http://www.s390.ibm.com/perform/dgwperf.html

Tuning your Web server for better performance:

http://www.ibm.com/software/webservers/httpservers/doc/v51/ttun.htm

Language Environment performance considerations:

http://www.s390.ibm.com/le/perform/index.html

WebSphere Troubleshooter

http://www.ibm.com/software/webservers/httpservers/troubleshooter.html

Tune-up and perform:

http://www.s390.ibm.com/oe/bpxa1tun.html

Porting (link to the Porting Guide):

http://www.s390.ibm.com/oe/bpxa1por.html

Java performance

http://www.s390.ibm.com/java/perform.html

Appendix A. Sample JCL and source code

This appendix contains the sample JCL and source code that we used to run the samples in Chapter 4, “How to run the samples” on page 31.

Also refer to the following FTP site for the source files:

ftp://www.redbooks.ibm.com/redbooks/SG245604/

Note

Consult your database administrator (DBA) before running these jobs.

To run the jobs you need to have the relevant DB2 authority. In our installation, we used the job shown in Figure 23 to execute the SQL statements:

```
//<JOBCARD> JOB ............................
//STEP1    EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*,DCB=BLKSIZE=133
//SYSUDUMP DD SYSOUT=*
//SYSTSIN  DD *
  DSN SYSTEM(DBS3)
    RUN PROGRAM(DSNTIAD)  PLAN(DSNTIA2T) -
        LIBRARY('DB2V61S3.RUNLIB.LOAD')
  END
//SYSIN    DD  DSN=USER.JCL.CNTL(sql_input_deck),DISP=SHR
//
```

Figure 23. Sample JCL to execute SQL statements

Note:

- Customize the SQL statements to reflect your DB2 environment. All names and user IDs in the SQL source that could be changed are highlighted in italics.
- It is important that the GRANT statements are executed.

A.1 User profile

The userprofile database is required by the user profile class to maintain persistent information about your Web site visitors and to use that information to customize your Web pages. Create the userprofile table using the sample SQL statements in Figure 24.

```
CREATE STOGROUP WEBASTOR VOLUMES (OP1DB2) VCAT DB2V510U;

CREATE DATABASE WEBASDB STOGROUP WEBASTOR BUFFERPOOL BP32K;

CREATE TABLESPACE USRPRFDB IN WEBASDB
  USING STOGROUP WEBASTOR
  PRIQTY 20 SECQTY 20 ERASE NO BUFFERPOOL BP0 CLOSE NO;
COMMIT;

CREATE TABLE USERPROFILE (
              USERNAME           VARCHAR(20) NOT NULL,
              LANGUAGE           VARCHAR(20),
              ADDRESS1           VARCHAR(20),
              ADDRESS2           VARCHAR(20),
              TOWN               VARCHAR(20),
              STATE              VARCHAR(20),
              COUNTRY            VARCHAR(10),
              ZIPCODE            VARCHAR(10),
              EMPLOYER           VARCHAR(20),
              EMAIL              VARCHAR(40),
              DAYPHONE           VARCHAR(20),
              NIGHTPHONE         VARCHAR(20),
              FAX                VARCHAR(20),
              FIRSTNAME          VARCHAR(20),
              MIDNAME            VARCHAR(20),
              LASTNAME           VARCHAR(20),
              GROUPS             VARCHAR(26),
            PRIMARY KEY (USERNAME)
) IN WEBASDB.USRPRFDB;
CREATE UNIQUE INDEX XC_USERPROFILE
                 ON USERPROFILE (USERNAME)
                 CLOSE NO;
COMMIT;
GRANT ALL ON TABLE OWNER.USERPROFILE TO PUBLIC;
COMMIT;
```

Figure 24. SQL statements to create the userprofile database

A.2 WOMBank

```
CREATE TABLESPACE WOMDB IN WEBASDB
  USING STOGROUP WEBASTOR
  PRIQTY 20 SECQTY 20 ERASE NO BUFFERPOOL BP0 CLOSE NO;
COMMIT;

CREATE TABLE BANKACCOUNT (
              USERNAME           VARCHAR(32) NOT NULL,
              NUMBER             VARCHAR(32) NOT NULL,
              TYPE               VARCHAR(32),
              BALANCE            DECIMAL(15,2),
            PRIMARY KEY (NUMBER)
) IN WEBASDB.WOMDB;
CREATE UNIQUE INDEX XC_BANKACCOUNT
                 ON BANKACCOUNT (NUMBER)
                 CLOSE NO;
COMMIT;

CREATE TABLE PASSWD (
              USERNAME           VARCHAR(32) NOT NULL,
              PASSWORD           VARCHAR(32) NOT NULL,
            PRIMARY KEY (USERNAME)
) IN WEBASDB.WOMDB;
CREATE UNIQUE INDEX XC_PASSWD
                 ON PASSWD (USERNAME)
                 CLOSE NO;
COMMIT;

CREATE TABLE ALOGRECORD (
              USERNAME           VARCHAR(32) NOT NULL,
              NUMBER             VARCHAR(32) NOT NULL,
              PROCTIME           VARCHAR(64),
              LONGDATE           FLOAT,
              TRANS              VARCHAR(50)
) IN WEBASDB.WOMDB;
COMMIT;
GRANT ALL ON TABLE OWNER.BANKACCOUNT TO PUBLIC;
GRANT ALL ON TABLE OWNER.PASSWD TO PUBLIC;
GRANT ALL ON TABLE OWNER.ALOGRECORD TO PUBLIC;
COMMIT;
```

Figure 25. SQL statements to create the WOMBank database

```
CREATE TABLESPACE TCDB IN WEBASDB
  USING STOGROUP WEBASTOR
  PRIQTY 20 SECQTY 20 ERASE NO BUFFERPOOL BP0 CLOSE NO;
COMMIT;

CREATE TABLE TCEVENT (
              EVNUM              INTEGER NOT NULL,
              EVNAME             CHAR(254),
              EVTHMB             CHAR(254),
             PRIMARY KEY (EVNUM)
) IN WEBASDB.TCDB;
CREATE UNIQUE INDEX XC_TCEVENT
                 ON TCEVENT (EVNUM)
                 CLOSE NO;
COMMIT;

CREATE TABLE TCPRICE (
              PRNUM              INTEGER NOT NULL,
              PRLEVEL            CHAR(1),
              PRPRC              NUMERIC(15,2) NOT NULL,
              PRSCODE            CHAR(7),
              PRSENUM            INTEGER NOT NULL,
              PRTIXAV            INTEGER NOT NULL,
              PRPID              CHAR(12),
            PRIMARY KEY (PRNUM)
) IN WEBASDB.TCDB;
CREATE UNIQUE INDEX XC_TCPRICE
                  ON TCPRICE (PRNUM)
                  CLOSE NO;
COMMIT;
CREATE TABLE TCVENUE (
              VECODE             VARCHAR(10) NOT NULL,
              VEMILES            CHAR(20),
              VEDESC             CHAR(254),
              VELOCAT            CHAR(254),
            PRIMARY KEY (VECODE)
) IN WEBASDB.TCDB;
CREATE UNIQUE INDEX XC_TCVENUE
                 ON TCVENUE (VECODE)
                 CLOSE NO;
COMMIT;
```

Figure 26. SQL statements to create the TicketCentral database (part 1 of 3)

```
CREATE TABLE TCSESSION (
              SENUM              INTEGER NOT NULL,
              SEDESC             VARCHAR(1024),
              SESTIME            TIME NOT NULL,
              SEFTIME            TIME,
              SEDATE             DATE NOT NULL,
              SEEVNUM            INTEGER NOT NULL,
              SEVECOD            CHAR(10),
              SETIXAV            INTEGER NOT NULL,
            PRIMARY KEY (SENUM)
) IN WEBASDB.TCDB;
CREATE UNIQUE INDEX XC_TCSESSION
                 ON TCSESSION (SENUM)
                 CLOSE NO;
COMMIT;

CREATE TABLE CUSTOMER (
              CUUSERNAME         VARCHAR(20) NOT NULL PRIMARY KEY,
              CUTITLE            VARCHAR(5),
              CUPASSWD           VARCHAR(20) NOT NULL,
              CUACC              INTEGER,
              CUADDRCHG          INTEGER,
              CUNEW              CHAR(1) NOT NULL
) IN WEBASDB.TCDB;
CREATE UNIQUE INDEX XC_CUSTOMER
                 ON CUSTOMER (CUUSERNAME)
                 CLOSE NO;
COMMIT;
CREATE TABLE ORDER (
              ORUSERNAME         VARCHAR(20) NOT NULL,
              ORSTAMP            INTEGER NOT NULL,
              ORSTAT             CHAR(1) NOT NULL,
              ORITEMNUM          INTEGER NOT NULL,
              ORQUANTITY         INTEGER NOT NULL,
              ORCONFIRMATIONNO   INTEGER,
              PRIMARY KEY(ORUSERNAME, ORITEMNUM, ORSTAT, ORSTAMP)
) IN WEBASDB.TCDB;
CREATE UNIQUE INDEX XC_ORDER
                 ON ORDER (ORUSERNAME, ORITEMNUM, ORSTAT, ORSTAMP)
                 CLOSE NO;
COMMIT;
```

Figure 27. SQL statements to create the TicketCentral database (part 2 of 3)

```
CREATE TABLE CCARD (
              CCUSERNAME         VARCHAR(20) NOT NULL,
              CCTYPE             VARCHAR(10) NOT NULL,
              CCNUM              VARCHAR(20) NOT NULL,
              CCEXPM             INTEGER NOT NULL,
              CCEXPY             INTEGER NOT NULL
) IN WEBASDB.TCDB;
COMMIT;

GRANT ALL ON TABLE OWNER.TCEVENT   TO PUBLIC;
GRANT ALL ON TABLE OWNER.TCSESSION TO PUBLIC;
GRANT ALL ON TABLE OWNER.TCPRICE   TO PUBLIC;
GRANT ALL ON TABLE OWNER.TCVENUE   TO PUBLIC;
GRANT ALL ON TABLE OWNER.CUSTOMER  TO PUBLIC;
GRANT ALL ON TABLE OWNER.CCARD     TO PUBLIC;
GRANT ALL ON TABLE OWNER.ORDER     TO PUBLIC;
COMMIT;
```

Figure 28. SQL statements to create the TicketCentral database (part 3 of 3)

Appendix B. Fixing the WAS 1.1-provided samples

In this chapter we describe the steps you need to perform in order to fix the samples provided with IBM WebSphere Application Server 1.1. The updated versions of the downloadable samples on the Internet already contain these changes.

B.1 Fixing the no-database samples

This section describes how to fix the samples that do not require a database. Samples that run without any changes in source or environment are not mentioned in this chapter.

B.1.1 XtremeTravel

We found that the ibmwebas.jar file in the /applicationserver_root/lib directory did not contain the correct com.ibm.servlet.servlets.personalization.util package, hence the SendMessage class could not be found together with these other classes: CheckMessage, GetMessage, GetVariableText and SetVariableText.

Solution: We needed to create "hard links" for the missing classes in the directory /applicationserver_root/servlets/com/ibm/servlet/servlets/personalization/util to the modules in the directory /applicationserver_root/servlets/com/ibm/servlet/servlets/personalization/util/IBM, as follows:

CheckMessage.class	EJSCCKMS
EmptyEnumeration.class	EJSCEE
GetVariableText.class	EJSCGVT
SendMessage.class	EJSCSM
SetVariableText.class	EJSCSVT
VariableText.class	EJSCVT

B.2 Fixing the database samples

Important

All the database samples require changes to the Java source files. We recommend doing the following:

- Copy the respective source files to your own user directory.
- Make the changes described in this chapter.
- Compile the code using the *javac filename.java* command.
- Copy the generated class files to the /*applicationserver_root*/servlets directory.

  We recommend that you put these samples in the /applicationserver_root/servlets directory, because they are part of the IBM delivery. You should put your own servlets into a directory defined by the servlets.classpath definition and not into the WebAS installation directories.

This ensures that the original source files stay intact for future reference.

B.2.1 JDBCServlet

To fix the sample, complete the following steps:

Step 1. Edit the JDBCServlet.java source file.

Copy the JDBCServlet.java source from the /*applicationserver_root*/samples/JDBCServlet directory to your directory, for example /u/*user_name*.

Find and change the following lines in the source code. The bold text highlights what we changed. The bold italic text is for information only.

```
.
.
static String DbName  = null;
static String DbOwner = null;
//static String Db      = "db2"; -----> comment out
static String Db      = "db2os390"; --> replace with this line
//static String JDBCDriver = "COM.ibm.db2.jdbc.app.DB2Driver"; --> comment out
static String JDBCDriver = "ibm.sql.DB2Driver"; ----------> replace with this line
static final String CONFIG_BUNDLE_NAME = "login";
.
.
```

Step 2. Compile the new JDBCServlet.java source file.

Go to the OMVS shell and compile the changed source code using the following command:

```
javac JDBCServlet.java
```

The compile generates two class files: JDBCServlet.class and ConInfo.class. Copy these files to the *applicationserver_root*/servlets directory using the following commands (back up the original files before copying):

```
cp JDBCServlet.class /applicationserver_root/servlets
cp ConInfo.class /applicationserver_root/servlets
```

Step 3. Change the JDBCServletForm.html file.

Back up the JDBCServletForm.html file in the /*applicationserver_root*/samples/JDBCServlet/ directory. Edit the file and issue the following ISPF edit commands to change the respective text words:

```
change 'employee' 'emp' all
change 'department' 'dept' all
change 'project' 'proj' all
change 'emp_act' 'empproject' all
```

B.2.2 IBMConnMgrTest

To fix the sample, complete the following steps:

Step 1. Edit the IBMConnMgrTest.java source file.

Copy the IBMConnMgrTest.java file from the /*applicationserver_root*/samples/database directory to your directory, for example /u/*user_name*.

Find and change the following lines in the source code. The bold text highlights what we changed. The bold italic text is for information only.

```
.
.
static String DbName      = null;      // database name
//static String Db          = "db2"; //JDBC subprotocol for DB2 ---> comment out
static String Db          = "db2os390"; ---> replace with this line
static String poolName    = "JdbcDb2"; // from Webmaster
//static String jdbcDriver  = "COM.ibm.db2.jdbc.app.DB2Driver"; ---> comment out
static String jdbcDriver  = "ibm.sql.DB2Driver"; ---> replace with this line
static String url         = null;      // constructed later
.
.
```

Edit the source code and issue the following ISPF edit commands to replace the respective text words:

```
change 'employee' 'emp' all
```

Step 2. Compile the new IBMConnMgrTest.java source file.

Go to the OMVS shell and compile the changed source code using the following command:

```
javac IBMConnMgrTest.java
```

The compile generates the IBMConnMgrTest.class class file. Copy this file to the *applicationserver_root*/servlets directory using the following command (back up the original file before copying):

```
cp IBMConnMgrTest.class /applicationserver_root/servlets
```

B.2.3 IBMDataAccessTest

To fix the sample, complete the following steps:

Step 1. Update your personal profile file to include the EnterpriseAccessBeans for DB2 of VisualAge for JAVA V2.

Edit the /u/*user_name/*.profile file. Add the following class library to the classpath in order to be able to compile the SQLJ sample:

/your_path/ivjdab.jar

Activate the changes.

Step 2. Edit the IBMDataAccessTest.java source file.

Copy the IBMDataAccessTest.java file from the /*applicationserver_root*/samples/database directory to your directory, for example: /u/*user_name*.

Find and change the following lines in the source code. The bold text highlights what we changed. The bold italic text is for information only.

```
.
.
static String DbName      = null;      // database name
//static String Db          = "db2"; //JDBC subprotocol for DB2 ---> comment out
static String Db          = "db2os390"; ---> replace with this line
static String poolName    = "JdbcDb2"; // from Webmaster
//static String jdbcDriver  = "COM.ibm.db2.jdbc.app.DB2Driver"; ---> comment out
static String jdbcDriver  = "ibm.sql.DB2Driver"; ---> replace with this line
static String url         = null;      // constructed later
.
.
```

Change the SQL query as follows. It needs to be changed to be adapted to the sample database that is delivered with DB2/390.

```
.
.
// Query string, with :idParm and :deptParm parameters.
String sqlQuery = "SELECT EMPNO, LASTNAME, WORKDEPT, COMM " +
                  "FROM " + owner + ".EMP " +
                  "WHERE EMPNO >= ? " +
                  "AND WORKDEPT =  ? " +
                  "ORDER BY EMPNO ASC";
.
.
```

The metadata statements need to be changed as well, due to different datatypes and names in the sample database for DB2/390.

```
.
.
metaData.addParameter("idParm",   String.class,     Types.CHAR);
metaData.addParameter("deptParm", String.class,     Types.CHAR);
metaData.addColumn("EMPNO",       String.class,     Types.CHAR);
metaData.addColumn("LASTNAME",    String.class,     Types.VARCHAR);
metaData.addColumn("WORKDEPT",    String.class,     Types.CHAR);
metaData.addColumn("COMM",        BigDecimal.class, Types.DECIMAL);
metaData.addTable(owner + ".EMP");
.
.
```

The conditions for the SELECT clause in the sample need to be changed as well.

```
.
.
String  wantThisDept  = "E21";
String  wantThisId    = "000100";
selectStatement.setParameter("deptParm", wantThisDept);
selectStatement.setParameter("idParm",   wantThisId);
.
.
```

Finally the HTML output needs to be adapted to names and datatypes.

```
.
.
out.println("<TD>" + (String)result.getColumnValue("EMPNO") + "</TD>");
out.println("<TD>" + (String)result.getColumnValue("LASTNAME") + "</TD>");
out.println("<TD>" + (String)result.getColumnValue("WORKDEPT") + "</TD>");
out.println("<TD>" + (BigDecimal)result.getColumnValue("COMM") + "</TD>");
out.println("</TR>");
result.previousRow();
out.println("<TR>");
out.println("<TD>" + (String)result.getColumnValue("EMPNO") + "</TD>");
out.println("<TD>" + (String)result.getColumnValue("LASTNAME") + "</TD>");
out.println("<TD>" + (String)result.getColumnValue("WORKDEPT") + "</TD>");
out.println("<TD>" + (BigDecimal)result.getColumnValue("COMM") + "</TD>");
.
.
```

Step 3. Compile the new IBMDataAccessTest.java source file.

Go to the OMVS shell and compile the changed source code using the following command:

```
javac IBMDataAccessTest.java
```

The compile generates the IBMConnMgrTest.class class file. Copy this file to the *applicationserver_root*/servlets directory using the following command (back up the original file before copying):

```
cp IBMDataAccessTest.class /applicationserver_root/servlets
```

B.2.4 WOMBank

To fix the sample, complete the following steps:

Step 1. Edit the bankserv.java source file.

Copy the bankserv.java file from the /applicationserver_root/samples/WomBank/ directory to your user directory, for example: /u/*user_name*/wm .

Find and change the following lines in the source code. The bold text highlights the changes. The bold italic text is for information only.

```
.
.
static String DbUserid = null;
static String DbPasswd = null;
//static String Db      = "db2";           ----> comment out
static String Db      = "db2os390";        ----> replace with this line
//static String DbName  = "womdb";         ----> comment out
static String DbName  = "DBS3";            ----> replace with your LOCATION in DDF
//static String JDBCDriver = "COM.ibm.db2.jdbc.app.DB2Driver";   ----> comment out
static String JDBCDriver = "ibm.sql.DB2Driver";     ----> replace with this line
static String DbSource ="jdbc:" + Db + ":" + DbName;
.
.
```

Note: For the LOCATION name in DB2 DDF, refer to Figure 14 on page 35.

```
..
 int cnt = 0;
  stmt = info.con.createStatement();
//String query = "Select number, proctime, trans from " + DbOwner + -----> comment out
1 ".alogrecord where username = '" + uname + "' order by longdate desc";
  String query = "Select number, proctime, trans, longdate from " +
1 DbOwner + ".alogrecord where username = '" + uname + "' order by longdate desc";
  System.out.println("Executing Query: " + query);
  rs = stmt.executeQuery (query);
  while (rs.next() && cnt < MAXLOGRECS)
.
```

Note: 1 This line appears on one line in the actual file; it is split for redbook printing purposes.

- Find all occurrences (there are two) of getOutputStream() and replace them with:

```
.
//ServletOutputStream out = res.getOutputStream();      ----> comment out
  res.setContentType("text/html");
  res.setHeader("Pragma","No-cache");
  res.setHeader("Cache-control","no-cache");
  res.setDateHeader("Expires",0);

//PrintWriter printWriter = new PrintWriter(out);      ----> comment out
  PrintWriter printWriter = res.getWriter();      ----> replace with this line
.
.
//ServletOutputStream out = info.res.getOutputStream();         ----> comment out
  info.res.setContentType("text/html");
  info.res.setHeader("Pragma","No-cache");
  info.res.setHeader("Cache-control","no-cache");
  info.res.setDateHeader("Expires",0);
//PrintWriter printWriter = new PrintWriter(out);               ----> comment out
  PrintWriter printWriter = info.res.getWriter();   ----> replace with this line
.
```

Note: If the getWriter/getOutputStream changes are not performed, the data will appear as EBCDIC on the browser.

Step 2. Compile the new bankserv.java source file.

Go to the OMVS shell and compile the changed source code using the following command:

```
javac bankserv.java
```

Note: Ignore the following warning message when you compile:

```
MINESHV:/u/mineshv/wm: >javac bankserv.java
Note: bankserv.java uses a deprecated API. Recompile with "-deprecation" for details.
1 warning
MINESHV:/u/mineshv/wm: >
```

The compile generates the Bankinfo.class and bankserv.class files. Copy these files to the *applicationserver_root*/servlets directory using the following commands:

```
cp bankserv.class /applicationserver_root/servlets
cp Bankinfo.class /applicationserver_root/servlets
```

B.2.5 TicketCentral

To fix the sample, complete the following steps:

Step 1. Copy the source files.

Copy all the Java source files from the /*applicationserver_root*/samples/TicketCentral/ directory to your work directory, for example: /u/*user_name*/tc. Go to the OMVS shell and issue the following commands:

```
cd /applicationserver_root/samples/TicketCentral/
cp *.java /u/mineshv/tc
```

Step 2. Edit the Java source files in /u/*user_name*/tc.

The bold text highlights the changes. The bold italic text is for information only.

Accept.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
.
  // Render the Register/Signon page
  //-------------------------------------------------------------
//ServletOutputStream out = res.getOutputStream();  ---> comment out
//PrintWriter printWriter = new PrintWriter(out);   ---> comment out
  PrintWriter printWriter = res.getWriter();     ----> replace with this line
.
.
```

ConnectionManager.java:

```
.
.
static String DBUSERID = "null";
static String DBPASSWD = "null";
//static String DBUSED = "db2";           ----> comment out
static String DBUSED = "db2os390";        ----> replace with this line
static String DBOWNER = "null";
//static String DB_NAME = "tcdb";         ----> comment out
static String DB_NAME = "DBS3";           ----> replace with LOCATION name in DDF
.
.
```

Note: For the LOCATION name in DB2 DDF, refer to Figure 14 on page 35.

Delivery.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
.
  // Render the Register/Signon page
  //-------------------------------------------------------------
//ServletOutputStream out = res.getOutputStream();  ---> comment out
//PrintWriter printWriter = new PrintWriter(out);   ---> comment out
  PrintWriter printWriter = res.getWriter();     ----> replace with this line
.
.
```

Payment.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
.
  // Render the Register/Signon page
  //-------------------------------------------------------------
//ServletOutputStream out = res.getOutputStream();  ---> comment out
//PrintWriter printWriter = new PrintWriter(out);   ---> comment out
  PrintWriter printWriter = res.getWriter();     ----> replace with this line
.
.
```

PopulateTCData.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
```

Reservation.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
  // Render the Register/Signon or appropriate error page which set in the re
  //-----------------------------------------------------------------------
//ServletOutputStream out = res.getOutputStream();    ----> comment out
//PrintWriter printWriter = new PrintWriter(out);     ----> comment out
  PrintWriter printWriter = res.getWriter();          ----> replace with this line
```

SearchResult.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
  // Render the ticket search page.
  //-------------------------------------------------------------
//ServletOutputStream out = res.getOutputStream();    ----> comment out
//PrintWriter printWriter = new PrintWriter(out);     ----> comment out
  PrintWriter printWriter = res.getWriter();          ----> replace with this line
.
.
```

ShopCart.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
  // Render the ShopCart/Delivery page
  //-------------------------------------------------------------
//ServletOutputStream out = res.getOutputStream();    ----> comment out
//PrintWriter printWriter = new PrintWriter(out);     ----> comment out
  PrintWriter printWriter = res.getWriter();          ----> replace with this line
.
.
```

TicketSearch.java:

```
.
.
// register the driver with DriverManager
//Class.forName("COM.ibm.db2.jdbc.app.DB2Driver"); ----> comment out
Class.forName("ibm.sql.DB2Driver");    ----> replace with this line
.
.
// Render the Register/Signon or appropriate error page which set in the re
  //-----------------------------------------------------------------------
//ServletOutputStream out = res.getOutputStream();    ----> comment out
//PrintWriter printWriter = new PrintWriter(out);     ----> comment out
  PrintWriter printWriter = res.getWriter();          ----> replace with this line.
.
.
.
stmt = conn.createStatement();
//select = "select DISTINCT MONTH(sedate)"           ----> comment out
//         + " from " + dbOwner + ".tcsession, " + dbOwner + ".tcvenue" -> comment out
//         + " where setixav > 0 and vecode=sevecod and"                -> comment out
//         + " MONTH(sedate) >=" + currentMonth      ----> comment out
//         + " order by MONTH(sedate)";              ----> comment out
  select = "select DISTINCT MONTH(sedate) as result"
           + " from " + dbOwner + ".tcsession, " + dbOwner + ".tcvenue"
           + " where setixav > 0 and vecode=sevecod and"
           + " MONTH(sedate) >=" + currentMonth
           + " order by result";                 ------> replace with these lines
  rs = stmt.executeQuery(select);
.
.
```

Note: If the getWriter/getOutputStream changes are not performed, the data will appear as EBCDIC on the browser and will not be legible.

Step 3. Compile the new source files.

Go to the OMVS shell and compile the changed source codes using the following commands. Ignore the following warning message when you compile:

```
Note: xxxxxxx.java  uses a deprecated API.  Recompile with
"-deprecation" for details.
1 warning
```

```
javac PopulateTCData.java
javac Reservation.java
javac Accept.java
javac Delivery.java
javac Payment.java
javac SearchResult.java
javac ShopCart.java
```

The compiles generate *.class files. Copy these files to the *applicationserver_root*/servlets directory using the following commands (back up the original files before copying):

```
cp *.class /applicationserver_root/servlets
```

Appendix C. Excerpts from GC34-4757

In this appendix we reprint some chapters of *WebSphere Application Server for OS/390 Application Server Planning, Installing, and Using Version 1.2*, GC34-4757.

We do so because a lot of people indicated to us that they want to have a printed version of this book, which is now only available online.

This is a very rapidly evolving area, and changes to the product that need to be reflected in the documentation occur frequently.

We urge you to check the library Web site frequently, especially before starting a new project in the OS/390 WebSphere area:

`http://www.ibm.com/software/webservers/appserv/library.html`

The reprinted chapters include the updates of 03/17/2000.

C.1 Performing basic servlet administration tasks

This section was originally Chapter 3 in *WebSphere Application Server for OS/390 Application Server Planning, Installing, and Using Version 1.2*, GC34-4757.

C.1.1 Basic administration tasks requiring changes to the was.conf file

For each Application Server model you are using, you can update the associated was.conf file to:

- Add directories for servlets, JavaBeans and other supported files to the Application Server.
- Specify an abbreviated name for a servlet.
- Set up aliases.
- Associate a servlet with a MIME-type.

- Change servlet configuration and initialization parameters.
- Enable/disable the automatic reloading of modified servlets.
- Enable/disable the Java compiler or change its name.

After updating an Application Server model was.conf file, you must always run the updateproperties program before any of your changes will go into affect. For instructions on running the updateproperties program, refer to Updating the Application Server model was.conf file.

C.1.1.1 Adding servlets and JavaBeans and other supported files

This section describes how to add servlets, JavaBeans and other supported files to the Application Server.

To add a servlet, JavaBean or other supported file to the Application Server you must

- Tell the Application Server where to the find the servlet and supporting class files.
- Tell the Web server how the servlets will be invoked.
- Tell the Application Server how the servlets will be invoked.

To tell the Application Server where to the find the servlet and supporting class files you must add the Java classpath information for the servlet and supporting class files to either the *ncf.jvm.classpath* or *servlets.reload.directories* properties or place the .servlet or .class files in the IBM provided server_model_root/*servlets* directory. The *ncf.jvm.classpath* or *servlets.reload.directories* properties are contained in the was.conf file which is located in the server_model_root/*properties* directory. (See Updating the Application Server model was.conf file for a description of how to update this file.)

When searching for a servlet, JavaBean, or other supported file, the Application Server:

1. First checks the Java classpath that is created from the list of directories, jar files, zip files, and ser files specified on the ncf.jvm.classpath property.
2. Next searches the directories listed on the servlets.reload.directories property in the was.conf file for the existence of .class files containing the servlet implementation.
3. Then, searches the server_model_root/servlets directory for the existence of .class files containing the servlet implementation.

Besides the Service directives described in Required Service directives, you must also add Service directives to the Web server httpd.conf file that indicate to the Web server how servlets will be invoked. This directive should have the following general format:

```
Service  request-template
applicationserver_root/lib/libadpter.so:AdapterService
```

request-template is the template value contained in a URL that indicates to the Web server that it is to call the Application Server to handle this request. (See your Web server documentation for more information about the Service directive.)

applicationserver_root is the fully qualified name of the mounted install-image of an individual execution system. The default value is /usr/lpp/WebSphere/AppServer.

One of the Service directives that is suggested looks like the following:

```
Service  /servlet/*  applicationserver_root/lib/libadpter.so:AdapterService
```

This directive tells the Web server to call the Application Server when a request contains a URL that matches the template /servlet/*. Similarly if you also want to invoke servlets using a URL that has the template /myservlets/*, add the following directive to the Web server httpd.conf file:

```
Service  /myservlets/*
applicationserver_root/lib/libadpter.so:AdapterService
```

Then when the Web server receives either of the following URLs, it will pass the URL to the Application Server for processing:

```
http://server-name/servlet/HelloWorld
http://server-name/myservlets/MyHelloWorld
```

In order for the Application Server to process the second request for a servlet when it is received from the Web server, you must also add the following line to the was.conf file in the *server_model_root* directory:

```
urltype.servlet=/myservlet
```

The was.conf file already includes a urltype.servlet property that corresponds to the /servlet/* Service directive template, so no additional property needs to be added for this template.

Once the Application Server knows where to the find the servlet and supporting class file, and both the Web server and the Application Server know how the servlet will be invoked, the servlet can be invoked by:

- Being called directly by someone specifying a URL.
- Being specified on the ACTION attribute of the `<FORM>` tag in an HTML file.
- Being specified within the `<SERVLET>` tag in an SHTML file.

Note: If your servlets are likely to be modified, see "Reloading modified servlets" on page 124 for additional information before adding their directories to the ncf.jvm.classpath property.

File types supported by the Application Server

The following table describes the file types supported by the Application Server:

File type	Description
.class files	.class files for servlets and JavaBeans are produced by a Java compiler. They contain byte code that will be executed by a Java Virtual Machine
.jhtml and .jsp files	After creating these files and placing them in a directory, you must ensure that the necessary PASS statements are included in the Web server configuration file
.servlet files	A .servlet file is an XML servlet configuration file and contains the name of the servlet class file, servlet initalization parameters, and a page list containing the URIs (universal resource identifiers) of the JSPs the servlet can call

Adding JHTML files and JavaServer Pages (JSPs)

To add JHTML, and JSP files for servlets, create and place these files in a directory of your choice. Next, add PASS statements to the Web server configuration file to point to these files.

Example:

```
PASS /jsps/* /u/john/mystuff/*
```

To invoke /u/john/mystuff/myjsp.jsp from a browser, a developer can enter the following URL:

```
http://server_name/jsps/myjsp.jsp
```

Your developers must code their servlets to use the method that returns the location of the servlet's JSP file. For example, they might specify the server configuration as:

```
ServletRequest.getRealPath("/my.jsp")
```

Adding .servlet files to the Application Server

.servlet files are XML servlet configuration files. Each file is an XML document and contains:

The name of the servlet class file.

A description of the servlet.

The servlet initialization parameters.

To add .servlet files to the Application Server, place them in either user created directories or in the IBM provided server_model_root/servlets directory. If you place them in user created directories you must also add their directories to the paths already included on one of the following properties in the server_model_root was.conf file:

ncf.jvm.classpath property.

servlets.reload.directories property.

The Application Server searches for .servlet files in the same manner as it searches for .class files containing servlet implementations. (See Adding servlets and JavaBeans and other supported files to the Application Server.)

In the following example, the .servlet file mywizard.servlet resides in the directory proj2/servlets2 which was added to the ncf.jvm.classpath property:

```
ncf.jvm.classpath=/proj1/servlets1:/proj2/servlets2:/mybeans/classes/
```

Adding servlets or JavaBeans in jar, ser, or zip files

If your servlets are in a jar, ser, or zip file, you will need to add the name of the file to the ncf.jvm.classpath property in the server_model_root was.conf file.

In the following example, the jar file myfiles/projA, the ser file myfiles/projB, and the zip file teamfiles/projC were added to the ncf.jvm.classpath property:

```
ncf.jvm.classpath=/myfiles/projA.jar:/myfiles/projB.ser:/teamfiles/projC.z
ip:/project1/favoriteservlets:/project2/otherservlets:/mybeans/classes/...
```

The lines in this example are split for printing purposes. In the actual file, these lines are on a single line.

Note: Modified servlets contained in a jar, ser, or zip file will not become effective until the Web server is stopped and then started again

C.1.1.2 Associating servlets with MIME-types

For some requests, you may want to set up a MIME-type to indicate to the Application Server that two or more servlets are to be invoked as a series (in a specific order). This is different from setting up an alias because an alias only results in all of the associated servlets being invoked. The order in which they are invoked does not matter. With a servlet series, the series starts with input from the browser being sent to the first servlet in the series. The output from the first servlet serves as input for the next servlet in the series, and so forth until the output from the last servlet in the series is sent as a response back to the browser.

To set up a MIME-type, add a line to the was.conf file of the form:

```
filter.MIME-type=<servlet_1>,<servlet_2>,...,<servlet_n>
```

When associating a series of servlets with a MIME-type, the servlets must be specified in the order in which they are to be invoked, separated by a comma.

MIME-type indicates the generic file category, such as audio, application, image, etc. (See File types supported by the Application Server for a description of the file types supported by the Application Server.)

<servlet_1> is the servlet that you want to receive the initial input.

<servlet_2> is the servlet to which you want <servlet_1>'s output to be sent.

<servlet_n> is the last servlet in the series to be invoked.

All of the servlets associated with a particular MIME type are invoked whenever a request with that MIME-type is generated.

If you need to add a servlet to an existing MIME-types or to remove a servlet from an existing MIME-type, edit the filter. property in the was.conf file that defines that MIME-type so that it only includes those servlets you now want associated with this MIME-type.

C.1.1.3 Changing servlet configuration and initialization parameters

After you have started using the Application Server, you may find that you need to update the default servlet configuration and initialization parameters to change:

Which servlets are to be loaded when the Application Server starts.

The list of {name, value} pairs that a servlet can access using servlet API calls.

To change which servlets are to be loaded when the Application Server starts, update the servlets.startup property in the was.conf file to include only those servlets that are to be automatically loaded whenever the Application Server starts. The servlet names must be separated by blank spaces as shown in the following example:

```
servlets.startup=<servlet_1> <servlet_2> ... <servlet_n>
```

To change the list of {name, value} pairs that a servlet can access using servlet API calls, update the servlet.servlet_name.initArgs property in the was.conf file to include only the valid parameters and their associated values. You must separate each parameter with a comma as shown in the following example:

```
servlet.servlet_name.initArgs=<parm_1_name>=<parm_1_value>,
<parm_2_name>=<parm_2_value>, ... ,<<parm_n_name>=<parm_n_value>
```

The lines in this example are split for printing purposes. In the actual file, these lines are on a single line.

C.1.1.4 Assigning a servlet an additional name

Occasionally you may need to assign an additional name to a servlet. For example, you may have a servlet with a long name that describes its function to which you want to assign an abbreviated name to make it easier to invoke, or you may a servlet that was given a code name while it was being developed to which you want to assign a more meaningful name.

To assign a servlet an additional name, add a new line with the format servlet.<servlet_name>.code=<servlet_class> to the was.conf file. This line will specify the additional name (servlet_name), other than its class file name, by which a servlet can be requested. <servlet_name> must be a unique name and can only contain the following types of characters:

English alphanumeric characters (upper or lowercase letters A to Z and numbers 0 to 9.

Period (.)

Underscore (_)

Hyphen (-)

<servlet_class> is the class file associated with this servlet without the .class extension.

For example, if you want the HelloWorldServlet.class file to be invoked when someone enters <server_name>/servlet/Hi on a browser, you would add the following line to the was.conf file:

```
servlet.Hi.code=HelloWorldServlet
```

C.1.1.5 Using a Java compiler with the Application Server

If your system includes a Just-In-Time (JIT) Java compiler, you may need to:

- Change the default value for the JIT dll file that the Application Server uses to determine the location of the Java compiler.
- Disable the Java compiler when it is not needed.

Changing the default value for the JIT dll file

The Application Server determines the location of the Java compiler by inserting the name of the JIT dll file into the template libJIT_dll_file.so. The default value for the JIT dll file is jitc. Given that /usr/lpp/java/J1.1 is the path for your JAVA_HOME root directory, the Application Server will look for the Java compiler in the directory /usr/lpp/java/J1.1/lib/mvs/native_threads/libjitc.so.

To change the location of the Java compiler, modify the java.compiler property in the was.conf file to reflect the correct name for your JIT dll file. The Application Server will then use this name to determine the correct directory for the Java compiler.

Disabling the Java compiler

To disable the Java compiler, remove any existing value from the java.compiler property in the was.conf file. The property would then look like the following:

```
java.compiler=
```

To enable the Java compiler again, update the java.compiler property in the was.conf file with the correct name of the JIT dll file you want the Application Server to use to determines the location of the Java compiler. The original value specified for this property was jitc

Reloading modified servlets

If your developers need to be able to modify a servlet, .class file or .servlet file and then have it automatically reloaded, the servlet or file must be either placed directly in the IBM provided server_model_root/servlets directory or the directories containing these items need to be included on the servlets.reload.directories property in the server_model_root was.conf file. If the directories containing these items are added to the ncf.jvm.classpath property instead, the changes to the modified servlets, .class files or .servlet files will not take effect until the Web server is stopped and then started again.

In addition, you must change the boolean value of the servlets.reload. property in the was.conf file to true to activate the automatic reload function. The default value for this property is false because using this function will result in additional system overhead and slower performance.

When the automatic reload function is enabled, the Application Server will monitor reloadable .class files of currently loaded servlets for changes. (Reloadable .class files are .class files that are located in the directories specified on the servlets.reload.directories property or that are located in the server_model_root/servlets directory.) When the Application Server detects that one of these .class file has been modified, it will automatically reload all currently active servlets and JavaBeans that are eligible for reloading.

C.1.1.6 Setting up aliases for a single servlet or a chain of servlets

Another way to give a servlet an abbreviated name is to assign it an alias. One advantage of an alias is that it can be used to associate a single name with either a single servlet or a chain of servlets that you want to invoke at the same time with a single request.

To set up an alias for a single servlet, add a line of the form:

```
alias.alias_name=<servlet_1>
```

to the was.conf file to specify the path-mapping rules that the Application Server is to use when invoking the servlet. alias_name is the path for the servlet relative to the Web server's location.

To set up an alias for multiple servlets, add a line of the form:

```
alias.alias_name=<servlet_1>,<servlet_2>,<servlet_3>,...,<servlet_n>
```

to the was.conf file to specify the path-mapping rules that the Application Server is to use when invoking the listed servlets.

When specifying a servlet series, using the alias. property, enter the name of each servlet in the series, separated by commas, but with no spaces. For example:

```
alias.myservlets=account,date,update
```

Once you have set up an alias, a shortcut URL can be used to invoke the associated servlets from a browser, or a Web server servlet can be invoked from within HTML documents or other Java programs by placing the alias in a server-side INCLUDE.

Using the previous example, if alias.myservlets is invoked from a browser, all three servlets, account, date, and update, will be invoked.

For servlets, servlet must be included as part of the servlet name in order for it to be invoked by a servlet alias unless you have added a service directive to the Web server's http.conf file defining another URI, and a corresponding urltype property statement to the was.conf file. For example, if you want URLs that begin with /mycompany to be processed by the Application Server as servlets, add the following directive to the http.conf file:

```
Service  /mycompany  /usr/lpp/AppServer/lib/libadpter.so:AdapterService
```

and the following urltype property statement to the was.conf file:

```
urltype.servlet=/mycompany
```

Note: You can include a wild-card character (*) as part of a servlet alias, but only at the beginning of a pathname.

Web server configuration file updates for servlet aliases
If you add a servlet alias to an Application Server model was.conf file that is not covered by an existing Web server Pass rule, you must add a Pass rule for this alias to this httpd.conf file.

For example, the request template /servlet/* tells the Web server to accept and respond to all requests starting with /servlet/. If you add an alias for /other/* using the alias property in the Application Server model was.conf file, you must add a corresponding Pass rule to the Web server httpd.conf configuration file.

If you add /servlet/other/*, you do not need to add a Pass rule to the Web server configuration file because the rule /servlet/* includes /servlet/other/*.

Related information:

- For more information on Pass rules, see your Web server documentation.
- For more information on adding a servlet alias, see C.1.1.6, “Setting up aliases for a single servlet or a chain of servlets” on page 125.

C.1.2 Security considerations

The Application Server does not provide any additional level of network security. Therefore, you must make sure that the security set up for the Web server on which you will be running the Application Server has been configured to meet any special needs of the resources that will be run on the Application Server. These resources include servlets and JavaServerPages (JSPs) that are accessed through your URLs.

To control client access to these resources, you may need to add access control directives to your Web server configuration file to set up security protection for the URL under which these resources are accessed. These directives are described in detail in your Web server documentation.

In some cases the Application Server generates files for its use. When a JSP is invoked, the files that are created (a Java source file and a class file) are created under the Web server's identity, and permissions are set so the Web server can access these files. If you start the Web server using a different identity, the Web server may not be able to access these files any longer. You can remove the /servlets/pagecompile directory and the Application Server will create these files with the appropriate ownership and permissions. Alternatively, you must change the ownership and permissions of the sub-directories and files in the pagecompile directory so that the Web server will be able to access them.

If the Web server is unable to access a resource, such as a servlet or JSP, an HTTP status code of 500 is sent to the browser, with a message indicating that an error occurred during processing of the request. An error message is also written to the Application Server's error log indicating which resource could not be accessed.

Note: The Application Server is able to service many requests for a resource after that resource has been loaded into memory. In particular, a servlet may be loaded at start-up and used for the life of the server to process many requests from a large number of clients. Therefore, you should not rely on control of the physical resources (i.e. file permissions, etc..) as an alternative for providing client access control.

C.1.3 Servlet development considerations

This section discusses servlet development considerations.

C.1.3.1 Developing servlets in a development or test environment

If you are configuring the Application Server for a development or test environment, you should consider running in Scalable Server mode to provide developers and testers the flexibility to modify and test their servlets without impacting each other. You may also want to:

- Set the value of the servlets.reload property in the was.conf file to true so that modified servlets are automatically reloaded when they are invoked. For more information about reloading modified servlets, see Reloading modified servlets.

- Create a unique application environment (ApplEnv) for each project or developer.

For example, you may want to create an ApplEnv called JOHN for John to develop and test his servlets in, and an ApplEnv called MARY for Mary to develop and test her servlets in. To set this up for John and Mary, follow these steps:

1. Configure WLM so that only one Queue Server is created per system per model of the Application Server for each ApplEnv. For more information on configuring WLM, refer to the OS/390 MVS Planning: Workload Management book. You can view this book on the Web at URL:

   `http://www.ibm.com/s390/os390/bkserv/`

2. Add the following ApplEnv directives to the Web server configuration file to indicate that servlets invoked by server_name/servlet/john/* go to John's ApplEnv, and that servlets invoked by server_name/servlet/mary/* go to Mary's ApplEnv:

   ```
   ApplEnv    /servlet/john/*    JOHN
   ApplEnv    /servlet/mary/*    MARY
   ```

   For more information on Scalable Server mode and enabling WLM support on the Web server, refer to your Web server documentation. For information on accessing Web server documentation, see Required OS/390 Web server.

3. Modify the ncf.jvm.classpath property in the Application Server model was.conf file to include the directory paths and/or jar files for John's servlets and Mary's servlets:

   `ncf.jvm.classpath=/u/john/projB.jar:/u/mary/projA.jar`

   For more information about modifying the Application Server model was.conf file, see Updating the Application Server model was.conf file.

   To invoke their servlets from a browser, John and Mary would type in the name of their Web site:

   ```
   http://server_name/servlet/john/Wonderfulservlet
   http://server_name/servlet/mary/EvenBetterservlet
   ```

   By creating unique application environments for both projects, you provide John and Mary the flexibility to modify and test their respective servlets without impacting each other.

C.1.3.2 Compiling servlets

To compile a servlet:

1. Set the Web server CLASSPATH environment variable to include the Application Server JAR files and the JDK classes.zip file:

   Example:

   ```
   export CLASSPATH=
   applicationserver_root/lib/ibmwebas.jar:
   applicationserver_root/lib/jst.jar:
   applicationserver_root/lib/jsdk.jar:
   applicationserver_root/lib/x509v1.jar:
   jdk_root/lib/classes.zip
   ```

   applicationserver_root is the fully qualified name by which the install-image is mounted to an individual execution system. The default value is

   `/usr/lpp/WebSphere/AppServer.`

2. Set the PATH environment variable to include the Java Development Kit/bin directory:

   ```
   export PATH="jdk_root"/bin:$PATH
   ```

   For jdk_root, enter the JAVA_HOME root directory of your Java Development Kit (JDK), for example, /usr/lpp/java/J1.1.

3. Issue the following command to ensure that Java Development Kit (JDK) Version 1.1.6 or higher is in your path:

   ```
   java -version
   ```

   This command should return a message stating what the JDK version is. If the command is not found, you will see an error message.

4. Issue the following command to compile the servlet:

   javac filename.java

C.2 Performing advanced administration tasks

This section was originally Chapter 4 in *WebSphere Application Server for OS/390 Application Server Planning, Installing, and Using Version 1.2*, GC34-4757.

Be sure to check the official Web site for the most up-to-date information:

`http://www.ibm.com/software/webservers/appserv/library.html`

C.2.1 Advanced administration tasks requiring changes to was.conf

For each Application Server model you are using, you must update the associated was.conf file if you want to:

Use the Connection Manager to connect to a DB2 database.

Set up session tracking.

After updating an Application Server model was.conf file, you must always run the updateproperties program before any of your changes will go into affect. For instructions on running the updateproperties program, refer to Updating the Application Server model was.conf file.

C.2.2 Connecting to a DB2 database

The Application Server uses the Connection Manager to cache and reuse connections to a DB2 database. When a servlet needs a connection, it can usually obtain one from a pool of available connections, thus eliminating the overhead required to open a new connection each time a connection is needed.

Before the Connection Manager can set up a pool of connections to a DB2 data base, you must:

- Make sure that the JDBC directory name exists on the ncf.jvm.classpath property in the server_model_root was.conf file. The default installation directory for JDBC is /usr/lpp/db2/db2510/classes/db2jdbcclasses.zip.
- Add the following line to the server_model_root was.conf file, for each pool you want to set up, defining the pool connection.

  ```
  IBMConnMgr.Pool=<pool_name>
  ```

  where <pool_name> specifies the unique name of each new connection pool you are defining.

Each connection pool you define will initially contain two connections. When a servlet requests a connection, the Connection Manager will first look for an idle connection that is already in the pool. If no idle connections are available, the Connection Manager will create a new connection as long as the number of connections already in the pool is less than 2,000.

If the pool already has 2,000 connections, and the servlet is willing to wait for a connection (its waitRetry parameter is set to true), the Connection Manager will grab the first connection that becomes available and assign it to the waiting servlet. If multiple servlets are waiting for a connection, the

Connection Manager will satisfy the requests on a first come first served basis.

Periodically, the Connection Manager examines connections that are not in use in a given pool, and removes those that have been idle for a while, until a minimum of two connections remain in the pool. The last two connections are never removed.

C.2.2.1 Collecting session tracking information

The Application Server lets you specify whether or not you want to collect session tracking information. If you do not want to capture session tracking information, set the session.enable property in the was.conf file to the boolean value f. The Application Server will then ignore all session tracking information. If you do want to capture session tracking information, leave this property at the default value t. (See Updating the Application Server model was.conf file for a description of how to edit this file.)

Using cookies with session tracking

If the session.cookies.enable property in the was.conf file is set to the boolean value t, the Session Tracker uses a unique session ID to match user requests with their HttpSession objects on the server. When the user first makes a request and the HttpSession object is created, the session ID is sent to the browser as a cookie.

On subsequent requests, the browser sends the session ID back as a cookie and the Session Tracker uses it to find the HttpSession associated with this user.

When using cookies, you may also need to modify the following properties in the was.conf file:

session.cookie.name to specify the name of the cookie. The name specified must be a unique name and can only contain the following types of characters:

English alphanumeric characters (upper or lowercase letters A to Z and numbers 0 to 9.

Period (.)

Underscore (_)

Hyphen (-)

session.cookie.comment to specify a comment about the cookie.

session.cookie.maxage to specify the amount of time in milliseconds that the cookie is to remain valid. This property should only be used to restrict

or extend how long the session cookie will live on the client browser. By default, the cookie only persists for the current invocation of the browser. When the browser is shut down, the cookie is deleted.

session.cookie.path to specify the path field that will be sent for session cookies. This property should only be used to restrict which paths on the server (servlets, jhtml files, and html files) the cookies will be sent to. If the value on this property is left at the default, the cookie will be sent on any access to the given server.

In addition, if you want to restrict the exchange of cookies to only HTTPS sessions, you can set the session.cookie.secure property to the boolean value t.

Note: You must run the updateproperties program before any of these changes to the was.conf file take effect. (See Updating the Application Server model was.conf file for information on how to run this program.)

Maintaining state without cookies

There are situations in which cookies will not work. For example, some browsers do not support cookies, while other browsers allow the user to disable cookie support. In such cases, the Session Tracker must resort to a second method, URL rewriting, to track a user session.

With URL rewriting, all links that a servlet returns to the browser or redirects have the session ID appended to them. For example, the link

```
<a href="/store/catalog">
```

can be rewritten as

```
<a href="/store/catalog;$sessionid$DA32242SSGE2">
```

Then when the user clicks this link, the rewritten form of the URL is sent to the server as part of the client's request. The Session Tracker recognizes

;$sessionid$DA32242SSGE2 as the session ID and uses it to obtain the proper HttpSession object for this user.

If Web pages are coded as Java Server Pages (JSP), the JSP processor automatically handles URL rewriting. If a JSP page contains a link, the link is wrapped in an encode URL method when the page is processed.

To maintain session information without cookies for URLs that are not invoked from a JSP file, you must enable URL rewriting by setting the session.urlrewriting.enable property in the was.conf file to the boolean value t. If you also want the Session Tracker to be able to add the session ID to

URLs when the URL requires a switch from HTTP to HTTPS or HTTPS to HTTP, you must set the session.protocolswitchrewriting.enable property in the was.conf file to the boolean value true.

Note: You must run the updateproperties program before any changes to the was.conf file take effect. (See Updating the Application Server model was.conf file for information on how to run this program.)

C.2.2.2 Using a proxy server with the Application Server

During normal servlet processing, if a servlet uses callPage to invoke a JSP, the Application Server provides the JSP with a `<base href>` tag containing the base URL which specifies the server name and port.

If you are using a proxy server to reroute requests from one port to another, the base URL that the Application Server creates will become corrupted, and future requests will fail. To prevent this from happening, when you are using a proxy server:

Set the servlets.callpage.create.href property in the was.conf file to the boolean value false and run the updateproperties program.

Note: If this property does not already exist in the was.conf file, add the following line to this file and then run the updateproperties program:

```
servlets.callpage.create.href=false
```

Manually add a `<base href>` tag to each JSP that is going to be invoked so that references within that JSP, such as references to images, do not have to be fully qualified.

C.3 Property file reference for migrating configuration settings

This section was originally Appendix 1 in *WebSphere Application Server for OS/390 Application Server Planning, Installing, and Using Version 1.2*, GC34-4757.

Be sure to check the official Web site for the most up-to-date information:

```
http://www.ibm.com/software/webservers/appserv/library.html
```

C.3.1 Before you begin

Configuration information for V1.1 Application Server and ServletExpress was saved in several property files. For V1.2 Application Server, this configuration information is consolidated into one configuration file called was.conf.

To migrate your previous configuration settings, use the information in this section to locate the files and configurable properties from V1.1 Application Server or ServletExpress, then edit your V1.2 was.conf file with the appropriate values.

Important Notes:

1. old_root is the root directory of your previous V1.1 Application Server or ServletExpress installation. Default directories are:

   V1.1 Application Server: /usr/lpp/WebSphere/AppServer

   ServletExpress: /usr/lpp/ServletExpress

2. If old_root is not the same root as your V1.2 applicationserver_root, use the V1.1 and ServletExpress file paths below.

   If old_root is the same root as your V1.2 applicationserver_root, the installation program will create backup copies of your property files in the /defaults/ directory. For example, your jvm.properties file would be in the directory:

   old_root/defaults/properties/server/servlet/servletservice/jvm.properties.date-time

   date-time is in the format YYMMDDHHMM, for example, 9909280850.

3. server_model_root is the directory of the V1.2 Application Server model.

C.3.2 Configuration settings

C.3.2.1 Basic settings

ncf.jvm.use.system.classpath

V1.1 file: old_root/properties/server/servlet/servletservice/jvm.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/jvm.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: ncf.jvm.use.system.classpath

Description: The value of this property is a boolean that indicates whether the System Classpath value is appended to the Java Classpath you specify. If you specify false, only the path specified for the ncf.jvm.classpath property is used.

ncf.jvm.classpath

V1.1 file: old_root/properties/server/servlet/servletservice/jvm.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/jvm.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: ncf.jvm.classpath

Description: The value of this property is a string that specifies the Java Classpath that the Application Server will use. This Classpath must point to the directory and JAR files that contain the necessary Class files.

ncf.jvm.libpath

V1.1 file: old_root/properties/server/servlet/servletservice/jvm.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/jvm.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: ncf.jvm.libpath

Description: The value of this property is a string that specifies the Java library path. This is a path that points to the shared libraries that the Application Server will be using, such as the jdk1.1.6/lib directory.

ncf.jvm.path

V1.1 file: old_root/properties/server/servlet/servletservice/jvm.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/jvm.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: ncf.jvm.path

Description: The value of this property is a string that specifies the Java path for the binary files (such as Java or Javac).

java.compiler

V1.1 file: old_root/properties/server/servlet/servletservice/jvm.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/jvm.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: java.compiler

Description: The value of this property is a string that specifies the name of the JIT DLL file. If no value exists on this property, the JIT is disabled.

servlets.reload

V1.1 file:
old_root/properties/server/servlet/servletservice/servlets.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/servlets.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: servlets.reload

Description: The value of this property indicates whether or not the automatic servlet reload function is active. The value for this property is either the boolean value true (to enable the function) or false (to disable the function). The default value is false.

servlets.classpath

V1.1 file:
old_root/properties/server/servlet/servletservice/servlets.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/servlets.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: servlets.reload.directories

Description: The value of this property is a string that specifies additional directories for servlets that will be automatically reloaded when modified.

C.3.2.2 Session tracking settings

enable.sessions

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.enable

Description: The value of this property is a boolean that indicates whether session tracking is enabled.

enable.urlrewriting

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.urlrewriting.enable

Description: The value of this property is a boolean that indicates whether session tracking uses rewritten URLs to carry the session IDs.

enable.cookies

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.cookies.enable

Description: The value of this property is a boolean that indicates whether session tracking uses cookies to carry the session IDs.

enable.protocolswitchrewriting

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.protocolswitchrewriting.enable

Description: The value of this property is a boolean that indicates whether the session ID is added to URLs when the URL requires a switch from HTTP to HTTPS or HTTPS to HTTP.

session.cookie.name

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.cookie.name

Description: The value of this property is a string that specifies the name of the cookie. This property must be specified if cookies are enabled and can only contain the following types of characters:

English alphanumeric characters (upper or lowercase letters A to Z and numbers 0 to 9.

Period (.)

Underscore (_)

Hyphen (-)

session.cookie.comment

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.cookie.comment

Description: The value of this property specifies a string to be used as a comment about the cookie.

session.cookie.maxage

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.cookie.maxage

Description: The value of this property is an integer that specifies the amount of time in milliseconds that a cookie will remain valid. Specify a value only to restrict or extend how long the session cookie will live on the client browser.

By default, the cookie only persists for the current invocation of thebrowser. When the browser is shut down, the cookie is deleted.

session.cookie.path

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.cookie.path

Description: The value of this property is a string that specifies the path field that will be sent for session cookies.

session.cookie.secure

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.cookie.secure

Description: The value of this property is a boolean that indicates whether session cookies include the secure field.

session.invalidationtime

V1.1 file:
old_root/properties/server/servlet/servletservice/session.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/session.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: session.invalidationtime

Description: The value of this property is an integer that specifies the amount of time in milliseconds that a session is allowed to go unused before it is invalidated.

C.3.2.3 Logging settings for error and event messages

log.error.level

V1.1 file:
old_root/properties/server/servlet/servletservice/systemDefaults.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/systemDefaults.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: log.error.level

Description: The value of this property specifies which error messages will be logged.

log.error.destination

V1.1 file:
old_root/properties/server/servlet/servletservice/systemDefaults.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/systemDefaults.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: log.error.destination

Description: The value of this property specifies where the error message output should be sent.

log.error.filename

V1.1 file:
old_root/properties/server/servlet/servletservice/systemDefaults.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/systemDefaults.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: log.error.filename

Description: The value of this property is a string that specifies the name of the log file for error messages.

log.event.level

V1.1 file:
old_root/properties/server/servlet/servletservice/systemDefaults.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/systemDefaults.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: log.event.level

Description: The value of this property specifies which event messages will be logged.

log.event.destination

V1.1 file:
old_root/properties/server/servlet/servletservice/systemDefaults.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/systemDefaults.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: log.event.destination

Description: The value of this property specifies where the event message output should be sent.

log.event.filename

V1.1 file:
old_root/properties/server/servlet/servletservice/systemDefaults.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/systemDefaults.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: log.event.filename

Description: The value of this property is a string that specifies the name of the log file for event messages.

log.event.filename

V1.1 file:
old_root/properties/server/servlet/servletservice/systemDefaults.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/systemDefaults.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: log.event.filename

Description: The value of this property is a string that specifies the name of the log file for event messages.

C.3.2.4 Managing servlets

Adding servlets:
servlet.servlet_name.code

V1.1 file:
old_root/properties/server/servlet/servletservice/servlets.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/servlets.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: servlet.servlet_name.code

Description: The value of this property specifies the name of the associated class file. The servlet_name is a string that specifies the unique name of the servlet and can only contain the following types of characters:

English alphanumeric characters (upper or lowercase letters A to Z and numbers 0 to 9.

Period (.)

Underscore (_)

Hyphen (-)

servlet.servlet_name.initArgs

V1.1 file:
old_root/properties/server/servlet/servletservice/servlets.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/servlets.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: servlet.servlet_name.initArgs

Description: The value of this property specifies a list of name, value pairs which can be accessed by the servlet using the servlet API calls. The servlet_name is a string that specifies the unique name of the servlet and can only contain the following types of characters:

English alphanumeric characters (upper or lowercase letters A to Z and numbers 0 to 9.

Period (.)

Underscore (_)

Hyphen (-)

servlets.startup

V1.1 file:
old_root/properties/server/servlet/servletservice/servlets.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/servlets.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: servlets.startup

Description: The value of this property specifies servlets to be loaded when the Web server starts.

Servlet aliases:
alias_name

V1.1 file: old_root/properties/server/servlet/servletservice/alias.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/alias.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: servlet.alias_name

Description: Use this property to specify servlet(s) to be invoked when the associated alias is recognized.

Filtering using MIME types:
filter.enable

V1.1 file:
old_root/properties/server/servlet/servletservice/mimeservlets.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/mimeservlets.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: filter.enable

Description: The value of this property is a boolean that indicates whether filtering is enabled.

MIME_type

V1.1 file:
old_root/properties/server/servlet/servletservice/mimeservlets.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/mimeservlets.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: filter.MIME_type

Description: The value of this property is a string that specifies the servlet to be invoked when the associated MIME type is recognized.

Mapping URL types:
URL_type

V1.1 file: old_root/properties/server/servlet/servletservice/rules.properties

ServletExpress file:
old_root/properties/server/ServletExpress/servletservice/rules.properties

V1.2 file: server_model_root/properties/was.conf

V1.2 property name: urltype.jsp or urltype.servlet

Description: Use this property to specify URL types that should be processed as JSPs or servlets.

C.4 Producing error logs for IBM support personnel

This section was originally Appendix 2 in *WebSphere Application Server for OS/390 Application Server Planning, Installing, and Using Version 1.2*, GC34-4757.

Be sure to check the official Web site for the most up-to-date information:

`http://www.ibm.com/software/webservers/appserv/library.html`

The Application Server can be configured to perform the following types of logging to assist IBM Service Personnel in diagnosing Application Server errors:.

- Native DLL logging: Log messages produced by the Web server C code before entering Java.
- Java standard out logging: Any System.out and System.err prints go to this log.

To enable Native DLL logging:

- Stop the Web server
- Using a text editor, open the server_model_root was.conf file in the server_model_root/properties directory.
- Set the ncf.native.logison= property to the boolean value true.
- Start the Web server

To disable Native DLL logging, stop your Web server, set the ncf.native.logison= property to the boolean value false, and then start the web server again.

To enable Java standard out logging to a file:

- Stop the Web server
- Using a text editor, open the server_model_root was.conf file.
- Set the ncf.jvm.stdoutlog.enable= property to the boolean value true.
- Set the ncf.jvm.stdoutlog.file= property to the boolean value true.
- Start the Web server

The log paths are:

```
/server_model_root/logs/native.log
/server_model_root/logs/ncf.log
```

server_model_root is the unique root directory of an Application Server model containing the server properties for that model.

Appendix D. Web delivery

Several examples mentioned in this book are available in softcopy format on the Internet from the redbooks Web server.

Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG245604`

Alternatively, you can go to the redbooks Web site at:

`http://www.redbooks.ibm.com`

Select **Additional Materials** and open the file that corresponds with the redbook form number.

Appendix E. Special notices

This publication is intended to help webmasters and OS/390 systems programmers to customize and understand the functions of the IBM WebSphere Application Server V1.2 for OS/390. The information in this publication is not intended as the specification of any programming interfaces that are provided by the IBM WebSphere Application Server. See the PUBLICATIONS section of the IBM Programming Announcement for IBM HTTP Server 5.2 for OS/390 and IBM WebSphere Application Server for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been

reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM ®	400
AIX	AS/400
AT	CICS
CICS/ESA	CT
CUA	DB2
DFSMS	DRDA
eNetwork	IBM
Language Environment	Netfinity
OS/390	RACF
RS/6000	S/390
SP	System/390
WebSphere	XT

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix F. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

F.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 157.

- *e-business Application Solutions on OS/390 Using Java: Volume I*, SG24-5342
- *e-business Application Solutions on OS/390 Using Java: Samples*, SG24-5365
- *Java Programming Guide for OS/390,* SG24-5619
- *Accessing DB2 for OS/390 Data from the World Wide Web*, SG24-5273
- *OS/390 Workload Manager Implementation and Exploitation*, SG24-5326
- *Enterprise WebServing with the Internet Connection Secure Server for OS/390*, SG24-2074 (suffix 00)
- *Enterprise Web Serving with the Lotus Domino Go Webserver for OS/390*, SG24-2074 (suffix 01)
- *OS/390 eNetwork Communications Server V2R7 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- *Integrating Java with Existing Data and Applications on OS/390*, SG24-5142

F.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs.Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849

CD-ROM Title	Collection Kit Number
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

F.3 Other resources

These publications are also relevant as further information sources:

- *WebSphere Application Server Planning, Installing and Using Version 1.2,* GC34-4757
- *IBM HTTP Server for OS/390, Planning, Installing, and Using, Version 5.2*, SC31-8690 (suffix 02)
- *IBM HTTP Server for OS/390 Release 7, Web Programming Guide, Version 5.1,* SC34-4743
- *WebSphere Application Server for OS/390 Release 2, Modification Level 0 Program Directory*, GI10-6780
- *OS/390 UNIX System Services Command Reference*, SC28-1892
- *OS/390 V2R8.0 UNIX System Services Planning*, SC28-1890 (suffix 08)
- *OS/390 V2R8.0 Language Environment for OS/390 & VM Programmer's Guide,* SC28-1939 (suffix 07)
- *OS/390 V2R8.0 Language Environment for OS/390 & VM Programmer's Reference, SC28-1940 (suffix 07)*
- *OS/390 eNetwork Communications Server - IP Configuration V2R7,* SC31-8513 (suffix 02)
- DB2 *UDB for OS/390 V6 Installation Guide,* GC26-9008
- *DB2 UDB for OS/390 V6 Application Programming Guide and Reference for Java,* SC26-9018
- *Program Directory for DB2 for OS/390,* GT02-0571
- *CICS Installation Guide*, GC33-1681
- *CICS Customization Guide*, GC33-1683
- *CICS Planning for Installation*, GC33-1789
- *CICS TS for OS/390 V1R2 CICS Internet and External Interfaces Guide,* SC33-1944

- IMS, *IMS/ESA V6 Installation Volume I: Installation and Verification,* GC26-8736
- *IMS/ESA V6 Installation Volume II: System Definition and Tailoring,* GC26-8737
- *IMS/ESA Release Planning Guide V6,* GC26-8744
- *IMS/ESA V6 OTMA Guide and Reference,* SC26-8743
- *MQ-Series Planning Guide,* GC33-1349
- *MQ-Series for OS/390 V2 Release 1 Program Directory,* GI10-2501

F.4 Referenced Web sites

These Web sites are relevant as further information sources:

- `http://www.ibm.com/software/data/db2/os390/sqlj.html`

  DB2 OS/390 SQLJ page
- `http://www.ibm.com/s390/java`

  Java for OS/390 home page
- `http://www.ibm.com/software/data/ims/imstoc.html`

  IMS TOC home page
- `http://www.ibm.com/software/data/ims/otmaci.html`

  IMS/ESA OTMA Callable Interface home page
- `http://www.ibm.com/software/ts/mqseries/txppacs/txpm1.html`

  MQSeries Support Pacs home page
- `ftp://www.redbooks.ibm.com/redbooks/SG245365/`

  e-business Application Solutions on OS/390 Using Java: Samples redbook download directory
- `ftp://www.redbooks.ibm.com/redbooks/SG245619/`

  Java Programming Guide for OS/390 redbook download directory
- `http://www.ibm.com/software/webservers/httpservers/doc52.html`

  Documentation: V5.2 IBM HTTP Server for OS/390 Releases 8 and 9
- `http://www.ibm.com/software/webservers/appserv/library.html`

  IBM WebSphere Application Server documentation
- `http://www.ibm.com/s390/java`

  Java for OS/390 reference material and download page

- http://www-4.ibm.com/software/webservers/appserv/doc/os390/v12/hejs120.htm

  Program Directory for WebSphere Application Server for OS/390 Release 2, Modification Level 0, GI10-6780

- http://www.ibm.com/software/webservers/appserv/troubleshooter.html

  WebSphere Troubleshooter for OS/390

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** http://www.redbooks.ibm.com/

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of abbreviations

ACL	Access Control List
APF	Authorized Program Facility
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASID	Address Space ID
CA	Certificate Authority
CGI	Common Gateway Interface
CLI	Call Level Interface
CORBA	Common Objects Request Broker Architecture
DASD	Direct Access Storage Device
DBA	Data Base Administrator
DDF	Distributed Data Facility (with DB2)
DGW	Domino Go Webserver (predecessor of the IBM HTTP Server)
DLL	Dynamic Link Library
DRDA	Distributed Relational Database Architecture
EJB	Enterprise Java Beans
FRCA	Fast Response Cache Accelerator
FTP	File Transfer Protocol
GWAPI	Go Webserver API
HFS	Hierarchical File System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IIIOP	Internet Inter-Orb Protocol
IMAP	Internet Message Access Protocol
IPC	Inter-Process Communication
IPCS	Inter-Process Communication Services
IPL	Initial Program Load
ISV	Independent Software Vendor
ITSO	International Technical Support Organization
JDBC	Java Data Base Connection
JDK	Java Development Kit
JIT	Just In Time compiler (in Java)
JSP	Java Server Pages
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LE	Language Environment
LPA	(OS/390) Link Pack Area
LPAR	Logical Partition (S/390 Hardware Feature)
NNTP	Network News Transfer Protocol
MVS	Old name for OS/390
OEDIT	OpenEdition Editor

OMVS	OpenEdition MVS, old name for OS/390 UNIX System Services
OpenEdition	Old name for UNIX System Services
ORB	Object Request Broker
PID	Process ID
PPID	Parent Process ID
PDS	Partitioned Data Set
PDS/E	Partitioned Data Set/Extended
POP3	Post Office Protocol 3
RACF	Resource Access Control Facility
PTF	Program Temporary Fix
SSL	Secure Socket Layer
TSO	Time Sharing Option
URL	Uniform Resource Locator
USS	UNIX System Services
VIPA	Virtual IP Addressing
WAS	WebSphere Application Server
WebAS	WebSphere Application Server
WLM	Workload Manager

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

W

X

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5604-01 OS/390 e-business Infrastructure: IBM Websphere Application Server 1.2 Customization and Usage
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

OS/390 e-business Infrastructure: IBM Websphere Application Server 1.2 Customization and Usage

(0.2”spine)
0.17”<->0.473”
90<->249 pages

OS/390 e-business Infrastructure: IBM WebSphere Application Server 1.2

Customization and Usage

Configure, set up, maintain, troubleshoot the Application Server

Sample programs and servlets - using Java in e-business applications

Enable DB2, CICS, IMS, MQ, Oracle databases for the Application Server

This redbook will help you understand, configure, and use the IBM WebSphere Application Server 1.2 for OS/390. The material for the book was developed using OS/390 R8 and the IBM HTTP Server 5.2 for OS/390.

This book is for webmasters and system programmers who install or customize the IBM WebSphere Application Server for OS/390.

We describe how to configure the Application Server and enable subsystems to use it. We also provide sample programs and servlets, and give information on more advanced WebAS (WAS) configuration and performance tuning.

In Appendix C, we include a relevant portion of the softcopy document WebSphere Application Planning, Installing and Using Version 1.2, GC34-4757 to help you perform both basic and advanced servlet administration tasks.

This redbook should be used in conjunction with OS/390 e-business Infrastructure: IBM HTTP Server 5.1 - Customization and Usage, SG24-5603. It is an update to OS/390 e-business Infrastructure: IBM WebSphere Application Server 1.1 - Customization and Usage, SG24-5604 (level 00).

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-5604-01 ISBN 073841901X