WebSphere software

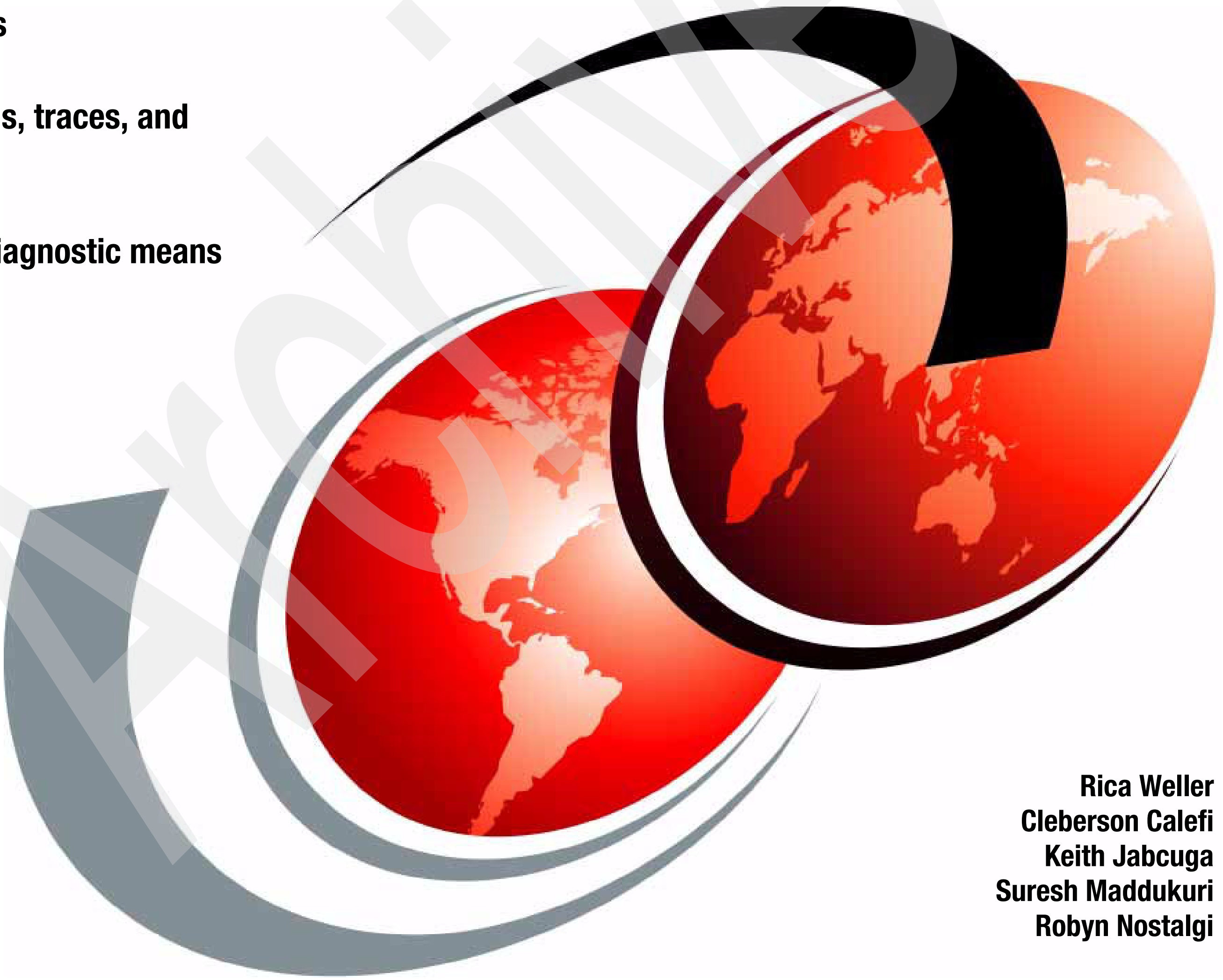
IBM

WebSphere Application Server for z/OS Problem Determination Means and Tools

Useful WebSphere for z/OS commands

Helpful logs, traces, and dumps

Guide to diagnostic means and tools



Rica Weller
Cleberson Calefi
Keith Jabcuga
Suresh Maddukuri
Robyn Nostalgi

Redpaper

ibm.com/redbooks

International Technical Support Organization

WebSphere Application Server for z/OS Problem Determination Means and Tools

March 2006

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (March 2006)

This edition applies to Version 6, Release 0, Modification 1 of WebSphere Application Server for z/OS (program number 5655-N01).

Contents

Notices

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®
@server®
Redbooks (logo) ™
alphaWorks®
z/OS®
zSeries®
AIX®
Cloudscape™
CICS®
DB2®
Informix®
IBM®
IMS™
Language Environment®
MVS™
OS/390®
Rational®
Redbooks™
RACF®
S/390®
Tivoli®
WebSphere®

The following terms are trademarks of other companies:

Enterprise JavaBeans, EJB, Java, JavaBeans, JDBC, JDK, JSP, JVM, J2EE, Solaris, SNM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Visual Studio, Windows server, Windows NT, Windows, Win32, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

For the day-to-day tasks that are involved in the process of IBM® WebSphere® Application Server for z/OS®, V6 administration, you need powerful and efficient tools for analyzing problems, for finding their root cause, and for solving them.

This Redpaper can be used as a reference for identifying the means and tools used to determine problems for WebSphere for z/OS.

We introduce and explain:

- Especially helpful commands for the WebSphere for z/OS environment
- WebSphere for z/OS logs
- Traces and dumps
- Diagnostic tools for problem determination in WebSphere for z/OS
- Other helpful tools

For each of these means, we provide information about its nature, when to use it, and how to use it. We describe the output, discuss how to interpret it, and provide an example.

Our focus in this Redpaper is on IBM products. This is no valuation; rather, it represents our experience. If you explore and use other products by various vendors for problem determination in the WebSphere for z/OS environment, the authors would welcome your comments and recommendations.

The team that wrote this Redpaper

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

Rica Weller is a Project Manager at the International Technical Support Organization (ITSO), working in New Zealand and the U.S. She was a Systems Engineer for S/390® for two years and a Senior Consultant for IBM WebSphere Business Integration on z/OS in the Competence Center with IBM Germany for three years. She also taught classes, presented at several conferences, and coauthored several Redbooks™ about WebSphere on z/OS and textbooks about zSeries® Basics. Rica holds a degree in Business Administration from TU Dresden, Germany, and a Master's Degree in Business from Massey University, New Zealand.

Cleberson Calefi is an IBM WebSphere consultant at Bank of Brazil. For the last three years, Cleberson has worked extensively with the WebSphere Application Server environment, advising on problem solving, tuning, and implementation of fail-safe runtime environments. His areas of expertise include J2EE™ application development and WebSphere Application Server administration for z/OS, z/Linux®, and Windows®. He holds a degree in Information Systems from the University Alvorada, Brazil.

Keith Jabcuga is Software Support Specialist working in the IBM Support Center in Poughkeepsie, New York. He has been working on the WebSphere for z/OS support team for four years, and his areas of expertise include defect support and application diagnostics. Keith holds an M.S. in Computer Science from the University of Buffalo.

Suresh Maddukuri is an IT Specialist assisting customers in the United States. He worked as an administrator for IBM WebSphere Application Server on z/OS and distributed platforms. His main responsibilities are troubleshooting problems, performance monitoring, and application server tuning. His areas of expertise include IBM WebSphere MQ and WebSphere Business Integration Message Broker. He holds a Post Graduate Diploma in Computer Applications and a degree in Mechanical Engineering from Nagarjuna University, India.

Robyn Nostalgi is a IT Software Support Specialist working in the IBM Support Center in Sydney, Australia, and has been in this role for over 10 years. She has worked with the zSeries Software Support team doing defect and non-defect support for all z/OS operating system related software components. More recently, she has specialized in supporting customers that are running WebSphere Application Server on z/OS.

Thanks to all the contributors of the previous Redbooks *WebSphere for z/OS V5 Problem Determination*, SG24-6880, and the support team of the IBM International Technical Support Organization, Poughkeepsie Center, in particular (in no specific order):

Tamas Vilaghy, Patrick C. Ryan, Andrew Lam, Youn Chin Mah, Ralph Schipani Jr., Brent Watson, Ron Allan, Dave Clarke, James Bai, Paola Bari, DongJune Choi, Mike Cox, Alberto González Dueñas, John Hutchinson, Wilhelm Michel, Theresa Tai, Egon Terwedow, Ella Buslovitch, Rich Conway, Don Bagwell, Keith Kopycinski, Nancy Trent, Michael Stephen, Hany Salem, Peter Bertolozzi, Michael Schwartz, Dave Griffiths, Christopher Vignola, Timothy Spewak, Stephen J Kinder, Mark Dinges, David Follis, Timothy Kaczynski, Teddy J Torres, Scott Kurz, Louis Wilen, Maria Clarke, Edward McCarthy, Kenneth Irwin, and many other IBM employees working in z/OS and WebSphere for z/OS Development and Technical Support worldwide.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYJ Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Commands

There are various commands that can be useful when you are trying to determine the root cause of problems in WebSphere for z/OS and are preparing to analyze logs, traces, dumps, and diagnostic tool output. This chapter is intended to be a quick reference guide to these commands.

Although they are not specific to problem determination for WebSphere for z/OS, we consider them very useful and powerful for performing day-to-day administrative tasks with WebSphere for z/OS and for identifying problems.

In the sections that follow, we introduce you to:

- A few command-line tools
- z/OS commands for WebSphere such as MODIFY, DISPLAY, and TRACE
- Some TCP/IP commands
- Related UNIX System Services (USS) for z/OS (OMVS) commands such as **`df`**, **`du`**, **`ps`**, **`pid`**
- The WASgrep.sh command for searching string patterns,
- The Windows FTP command

1.1 Commands for administering WebSphere for z/OS

There are several command-line tools that you can use to start, stop, and monitor WebSphere server processes and nodes. These tools only work on local servers and nodes. They cannot operate on a remote server or node.

To use these command-line tools, perform the following steps:

1. Go to the `bin` directory.
2. Issue one of the following commands:

```
START appserver_proc_name
STOP appserver_proc_name
START dmgr_proc_name
STOP dmgr_proc_name
START nodeagent_proc_name
STOP nodeagent_proc_name
addNode
serverStatus
removeNode
cleanupNode
syncNode
backupConfig
restoreConfig
EARExpander
GenPluginCfg
```

For more information about the command line tools and syntax, search for the "Using command line tools" topic in *WebSphere for z/OS Information Center*, available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

For other MVS™ system commands that can assist with the display, modification, and monitoring of z/OS subsystems, such as WLM, RRS and the WebSphere for z/OS components, refer to *z/OS V1R5 MVS System Commands*, GC28-1781.

1.2 z/OS MODIFY commands

You can use the MODIFY (short command: F) command to send requests to the WebSphere for z/OS controller region. In this section, we discuss how to use the MODIFY command and then show a combination of display commands for requesting information about the server and options for requesting trace and diagnostic data.

You must use the server name, not the started task (STC) name, with the MODIFY command. You can find the correct server name either in the JCL for your WebSphere for z/OS server, as shown in Example 1-1, or you can issue the **`d a,l`** system command.

Example 1-1 WebSphere for z/OS server JCL

```
//PDACR  PROC ENV=,PARMS=' ',Z=PDACRZ
// SET ROOT='/waspdconfig/pdcell'
// SET FOUT='properties/service/logs/applyPTF.out'
//APPLY   EXEC PGM=BPXBATCH,REGION=0M,
// PARM='SH &ROOT./&ENV..HOME/bin/applyPTF.sh inline'
//STDOUT   DD PATH='&ROOT./&ENV..HOME/&FOUT.',
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
//STDERR   DD PATH='&ROOT./&ENV..HOME/&FOUT.',
// PATHOPTS=(OWRONLY,OCREAT,OAPPEND),PATHMODE=(SIRWXU,SIRWXG)
```

```
//BBOCTL  EXEC PGM=BBOCTL,COND=(8,EQ),REGION=OM,TIME=MAXIMUM,
// PARM='TRAP(ON,NOSPIE),ENVAR("_EDC_UMASK_DFLT=007") / &PARMS.'
//BBOENV DD PATH='&ROOT/&ENV/was.env'
// INCLUDE MEMBER=&Z
//
```

The message for your commands is issued to the job log of your WebSphere for z/OS server and to your z/OS system log.

As with other z/OS system commands, you can issue them either on a system console or you can use a system command interface such as System Display and Search Facility (SDSF) or JES33 Spooler Interface (EJES).

The general syntax for the system commands is:

```
MODIFY <Server_Name>,DISPLAY,<Options>
```

The syntax is broken down as follows:

Server NAME	Server name as specified in the JCL
DISPLAY	Fixed keyword
Options	HELP, SERVERS, TRACE, WORK

The MODIFY command can be abbreviated by using the “F” character, for example:

```
f bboasr2a,display,trace
```

Note that the system commands are not case-sensitive.

`MODIFY <Server_Name>,DISPLAY` returns this information:

STC/server name	Started task name and server name
Status	ACTIVE
System name	SYSID of your system on which WebSphere for z/OS is active
Level	Build level of your WebSphere for z/OS server

1.2.1 z/OS DISPLAY commands

We found the commands shown in the following examples especially useful.

MODIFY <Server_Name>,DISPLAY

Example 1-2 shows the result of this MODIFY command.

Example 1-2 MODIFY <Server_Name>,DISPLAY

```
F PDSR01A,DISPLAY
BBOO0173I SERVER PDSR01/PDSR01A ACTIVE ON SC49 AT LEVEL W510004.
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY
```

MODIFY <Server_Name>,DISPLAY,HELP

Example 1-3 shows the result of this MODIFY command.

Example 1-3 MODIFY <Server_Name>,DISPLAY,HELP

```
F PDSR01A,DISPLAY,HELP
BBOO0178I THE COMMAND DISPLAY, MAY BE FOLLOWED BY ONE OF THE FOLLOWING 042 KEYWORDS:
BBOO0179I SERVERS - DISPLAY ACTIVE CONTROL PROCESSES
```

```
BBOO0179I SERVANTS - DISPLAY SERVANT PROCESSES OWNED BY THIS CONTROL 044 PROCESS
BBOO0179I SESSIONS - DISPLAY INFORMATION ABOUT COMMUNICATIONS SESSIONS
BBOO0179I TRACE - DISPLAY INFORMATION ABOUT TRACE SETTINGS
BBOO0179I JVMHEAP - DISPLAY JVM HEAP STATISTICS
BBOO0179I WORK - DISPLAY WORK ELEMENTS
BBOO0179I ERRLOG - DISPLAY THE LAST 10 ENTRIES IN THE ERROR LOG
BOO0188I END OF OUTPUT FOR COMMAND DISPLAY,HELP
```

MODIFY <Server_Name>,DISPLAY,SERVERS

Example 1-4 shows the result of this MODIFY command.

Example 1-4   MODIFY <Server_Name>,DISPLAY,SERVERS

```
F PDSR01A,DISPLAY,SERVERS
BBOO0182I SERVER            ASID   SYSTEM    LEVEL
BBOO0183I PDCELL  /SC49      3F5x  SC49      W510004
BBOO0183I PDAGNTB /PDAGNTB     78x  SC42      W510004
BBOO0183I PDSR01  /PDSR01A    3F6x  SC49      W510004
BBOO0183I PDCELL  /SC42        72x  SC42      W510004
BOO0183I PDDMGR  /PDDMGR     3EBx  SC49      W510004
BBOO0183I PDAGNTA /PDAGNTA    3ECx  SC49      W510004
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,SERVERS
```

The MODIFY <Server_Name>,DISPLAY,SERVERS command returns the following information:

SERVER	STC and server name for all active servers
ASID	Address space ID (ASID) for all active servers
SYSTEM	System ID (SYSID) of the system on which the server is active
LEVEL	Build level of each active server

MODIFY <Server_Name>,DISPLAY,TRACE

Example 1-5 shows the result of this MODIFY command.

Example 1-5   MODIFY <Server_Name>,DISPLAY,TRACE

```
F PDSR01A,DISPLAY,TRACE
BBOO0224I TRACE INFORMATION FOR SERVER PDSR01/PDSR01A/STC05755
BBOO0197I LOCATION = SYSPRINT BUFFER
BBOO0197I AGGREGATE TRACE LEVEL = 1
BBOO0197I EXCEPTION TRACING = RAS(0), Common Utilities(1), COMM(3), 059
ORB(4), OTS(6), Shasta(7), OS/390 Wrappers(9), Daemon(A), Security(E),
Externalization(F), JRAS(J), J2EE(L), Logging(M)
BBOO0197I BASIC TRACING =
BBOO0197I DETAILED TRACING =
BBOO0197I TRACE SPECIFIC = NONE SPECIFIED
BBOO0197I TRACE EXCLUDE SPECIFIC = NONE SPECIFIED
BBOO0225I TRACE INFORMATION FOR SERVER PDSR01/PDSR01A/STC05755 064
COMPLETE
```

The MODIFY <Server_Name>,DISPLAY,TRACE command returns the following information:

SERVER/STC	Started task name and server name
LOCATION	Target location for trace data
LEVEL	Trace level

OPTIONS   Trace options

MODIFY <Server_Name>,DISPLAY,ERRLOG

You can use this command as a quick way to display the last 10 messages in the error log even if you are not routing them to a log stream. Example 1-6 shows only the last three entries in the log.

Example 1-6   WebSphere for z/OS DISPLAY,ERRLOG command

```
F DISPLAY,ERRLOG
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,ALL,SRS
F PDSR01A,DISPLAY,ERRLOG
BBOO0266I (STC05755) BossLog: { 0002} 2004/09/21 22:13:44.668 01 138
SYSTEM=SC49 SERVER=PDSR01A  PID=0X03080069 TID=0X216AC930 00000000
c=UNK ./bborjtr.cpp+830 ... BBOO0222I TRAS0017I: The startup trace
state is *=all=disabled.
BBOO0266I (STC05755) BossLog: { 0003} 2004/09/21 22:13:53.044 01 139
SYSTEM=SC49 SERVER=PDSR01A  PID=0X03080069 TID=0X216AC930 00000000
c=UNK ./bborjtr.cpp+830 ... BBOO0222I SECJ0231I: The Security
component's FFDC Diagnostic Module com.ibm.ws.security.core.SecurityDM
registered successfully: true
BBOO0266I (STC05755) BossLog: { 0004} 2004/09/21 22:13:53.485 01 140
9} 2004/09/21 22:14:51.908 01 145
SYSTEM=SC49 SERVER=PDSR01A  PID=0X03080069 TID=0X2173C430 0X00001A
c=UNK ./bborjtr.cpp+842 ... BBOJ0087W MDB Workload Classification
Support is not enabled
BBOO0188I END OF OUTPUT FOR COMMAND
DISPLAY,ERRLOG
```

DISPLAY WLM,APPLENV=*

To display the WLM application environment names for WebSphere for z/OS (V5), use this command as shown in Example 1-7.

*Example 1-7   DISPLAY WLM,APPLENV=**

```
D WLM,APPLENV=*
IWM029I  08.28.43  WLM DISPLAY 768
  APPLICATION ENVIRONMENT NAME     STATE     STATE DATA
  BBOASR1                          AVAILABLE
  BBOASR2                          AVAILABLE
  CBINTFRP                         AVAILABLE
  CBNAMING                         AVAILABLE
  CBSYSMGT                         AVAILABLE
  C1INTFRP                         AVAILABLE
  C1NAMING                         AVAILABLE
  C1OASR1                          AVAILABLE
  C1OASR2                          AVAILABLE
  C1SYSMGT                         AVAILABLE
  DBD7MWLM                         AVAILABLE
  DBD8MWLM                         AVAILABLE
  DB7AODBA                         AVAILABLE
  DB7EUTIL                         AVAILABLE
  DB7EWLM                          AVAILABLE
  DB7LSQL                          AVAILABLE
  DB7LUTIL                         AVAILABLE
  DB7LWLM                          AVAILABLE
  DB7LWLM2                         QUIESCED
  DB7PDBUG                         AVAILABLE
  DB7PJAVS                         AVAILABLE
  DB7PODBA                         AVAILABLE
```

DISPLAY WLM,DYNAPPL=*

To list all the dynamic application environment names (for WebSphere for z/OS V6), use this command as shown in Example 1-8.

*Example 1-8 DISPLAY WLM,DYNAPPL=**

```
D WLM,DYNAPPL=*
IWM029I  08.39.26  WLM DISPLAY 854
  DYNAMIC APPL. ENVIRON. NAME      STATE     STATE DATA
  PDSR01                           AVAILABLE
  ATTRIBUTES: PROC=PDASR    SUBSYSTEM TYPE: CB
  SUBSYSTEM NAME: PDSR01A   NODENAME: PDCELL
  PDDMGR                           AVAILABLE
  ATTRIBUTES: PROC=PDASR    SUBSYSTEM TYPE: CB
  SUBSYSTEM NAME: PDDMGR    NODENAME: PDCELL
  CLU491                           AVAILABLE
  ATTRIBUTES: PROC=WS5491S  SUBSYSTEM TYPE: CB
  SUBSYSTEM NAME: WS491     NODENAME: CL491
```

Other useful DISPLAY commands

The following DISPLAY commands are also useful:

```
MODIFY <Server_Name>,DISPLAY,WORK,SERVLET
MODIFY <Server_Name>,DISPLAY,WORK,SERVLET,SRS
MODIFY <Server_Name>,DISPLAY,WORK,SUMMARY
MODIFY <Server_Name>,DISPLAY,WORK,SUMMARY,SRS
MODIFY <Server_Name>,DISPLAY,WORK,ALL
MODIFY <Server_Name>,DISPLAY,WORK,ALL,SRS
MODIFY <Server_Name>,DISPLAY,WORK,CLINFO
```

1.2.2 Basic TRACE commands

Because trace or diagnostic data is not documented and is only useful to the IBM support team, you should only use these traces at the request of IBM. For this reason, we only list a few commands here and do not cover them in detail:

- To turn tracing to SYSPRINT on and off:

  ```
  MODIFY <server_name>,TRACETOSYSPRINT=YES | NO
  ```

- To change the overall trace level (F is short for MODIFY):

  ```
  F <server_name>,TRACEALL=0 | 1 | 2 | 3
  ```

- To turn on basic or detailed tracing for specified components (non-Java):

  ```
  F <server_name>,TRACEBASIC=(0,1,2...)
  F <server_name>,TRACEDETAIL=(0,1,2..)
  ```

  See Table 1-1 for explanations of the codes.

Table 1-1 TRACEBASIC/TRACEDETAIL codes

Code	Explanation	Code	Explanation
0	RAS	A	Daemon
1	Common utilities	E	Security

Code	Explanation	Code	Explanation
3	COMM	F	Externalization
4	ORB	J	JRAS
6	OTS	L	J2EE
7	Shastat		
9	z/OS wrappers		

- To turn off all tracing:

```
F <server_name>,TRACENONE
```

Other useful trace commands are:

```
F <control_region_JOBNAME>,TRACESPECIFIC
F <control_region_JOBNAME>,TRACE_EXCLUDE_SPECIFIC
F <control_region_JOBNAME>,TRACETOTRCFILE
F <control_region_JOBNAME>,MDBSTATS
```

1.2.3 Dynamic Java TRACE

One of the most important tools is the MVS Modify command that dynamically turns Java™ tracing on and off. It is well documented in the *WebSphere for z/OS Information Center*, but can be overlooked among all the other tracing tools.

The z/OS MODIFY command does not require the server to be recycled.

- To turn on Java tracing for specified components such as com.ibm.ws.security, enter:

```
F <server_name>,TRACEJAVA='com.ibm.ws.security.yyy.*=all=enabled
```

- To reset to trace settings in your configuration (such as in was.env), enter:

```
F <server_name>,TRACEINIT
```

- To turn off all tracing, enter:

```
F <server_name>,TRACENONE
```

For more information, search for the “Dynamic Java Trace” topic in the *WebSphere for z/OS Information Center*, available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

1.3 TCP/IP-related commands

We describe several TCP/IP commands that can be very powerful for performing day-to-day administrative and problem determination tasks for WebSphere for z/OS.

As a multi-address-space application, WebSphere for z/OS is very dependent on communication services. Any interruption or wrong configuration prevents communication between the components (for example, between the node agent and deployment manager).

During the development test and deployment process, you also rely on the communication between the client on one platform and the WebSphere for z/OS environment. Tools such as Rational® Application Developer have to establish a connection between the workstation and z/OS for full functionality.

As with any communication scenario, you must check both sides that want to exchange information for functionality. If both are running on the same platform, you can use the same tool to check the server and client functionality.

Because TCP/IP is a standard protocol and its usage on various platform is very similar, the commands and their returned information only vary slightly. Although there are many different commands for analyzing communication problems, we are presenting only the ones that we consider particularly useful:

- `netstat`
- `nslookup`
- `ping`
- `tracert`

We have assumed that you use a TSO Command panel with z/OS and a simple Windows command prompt for the remote system.

Note: There are a number of tools available that issue the same commands under the cover, but provide the information in a more sophisticated way.

For a complete description of all tools, options, and return codes, refer to *TCP/IP V3.2 for MVS: Users Guide*, SC13-7136.

1.3.1 The netstat command

This command provides information about all external connections to your system and about all servers waiting for inbound connections.

To issue the nslookup command, you use a TSO Command panel on z/OS and a Windows command prompt on the remote system.

When you issue the command in TSO, you receive the following details:

User ID	The user ID of the local and remote task of the active connection.
Conn	The connection ID in TCP/IP.
Local Socket	The IP address and port number in the local system. An IP address of 0.0.0.0 refers to the default stack.
Foreign Socket	The IP address and port number of the remote system. The IP addresses of the local and the foreign systems can be the same. For an ESTABLISHED connection in the same system, there are two lines with reversed local and foreign socket information, representing an active connection.
State	The state of the socket. Some important states are LISTEN (server is listening for incoming connections), ESTABLISHED (a connection is established), SYNC_SENT (actively attempting to establish a connection), TIME_WAIT (waiting after close for remote shutdown), FIN_WAIT1(socket is closed and shutting down), and so on.

You should be most interested in the normal session states of LISTEN (a server waiting for work), ESTABLISHED (a communication between client and server is in progress), and SYNC_SENT (usually means an attempt to establish a connection is being blocked by a firewall).

For an explanation of the other states, refer to *TCP/IP V3.2 for MVS: Users Guide*, SC13-7136.

Example 1-9 shows the output after we issued the netstat command in our z/OS system.

Example 1-9 The netstat command and its response.

```
>tso netstat

MVS TCP/IP NETSTAT CS V1R6      TCPIP Name: TCPIP          15:29:47
User Id  Conn     Local Socket           Foreign Socket       State
-------  ----     ------------           --------------       -----
FTPMVS1  0000002A 9.12.4.28..21          0.0.0.0..0           Listen
FTPOE1   0000002B 9.12.4.29..21          0.0.0.0..0           Listen
INETD1   00000033 9.12.4.29..512         0.0.0.0..0           Listen
INETD1   00000030 9.12.4.29..23          0.0.0.0..0           Listen
INETD1   00000032 0.0.0.0..513           0.0.0.0..0           Listen
INETD1   00000031 9.12.4.29..514         0.0.0.0..0           Listen
NFSMVS   00000050 0.0.0.0..10001         0.0.0.0..0           Listen
NFSMVS   0000004F 0.0.0.0..10000         0.0.0.0..0           Listen
NFSMVS   00000055 0.0.0.0..2049          0.0.0.0..0           Listen
NFSMVS   00000052 0.0.0.0..10002         0.0.0.0..0           Listen
PMAP     00000028 0.0.0.0..111           0.0.0.0..0           Listen
REXECD   00000025 9.12.4.28..512         0.0.0.0..0           Listen
REXECD   00000026 9.12.4.28..514         0.0.0.0..0           Listen
TCPIP    00000016 127.0.0.1..1024        127.0.0.1..1025      Establsh
TCPIP    000206E2 9.12.4.28..23          9.12.6.132..1438     Establsh
TCPIP    0000001B 0.0.0.0..23            0.0.0.0..0           Listen
TCPIP    0000000C 127.0.0.1..1024        0.0.0.0..0           Listen
TCPIP    00000015 127.0.0.1..1025        127.0.0.1..1024      Establsh
TCPIP    0001492A 9.12.4.28..23          9.12.6.136..2330     Establsh
WASPDCTG 000017E3 0.0.0.0..2006          0.0.0.0..0           Listen
WS551    00000275 0.0.0.0..19080         0.0.0.0..0           Listen
WS551    0000024D 0.0.0.0..10003         0.0.0.0..0           Listen
WS551    00000249 0.0.0.0..18880         0.0.0.0..0           Listen
WS551    00000276 0.0.0.0..19443         0.0.0.0..0           Listen
WS551    0000024B 0.0.0.0..12809         0.0.0.0..0           Listen
WS551D   0000023B 0.0.0.0..15655         0.0.0.0..0           Listen
WS6552   0001385F 0.0.0.0..29080         0.0.0.0..0           Listen
WS6552   00013839 0.0.0.0..10044         0.0.0.0..0           Listen
WS6552   00013837 0.0.0.0..22809         0.0.0.0..0           Listen
WS6552   00013835 0.0.0.0..28880         0.0.0.0..0           Listen
WS6552   00013860 0.0.0.0..29443         0.0.0.0..0           Listen
WS6552D  0001381F 0.0.0.0..25655         0.0.0.0..0           Listen
WS6552S  00014871 9.12.4.28..10054       9.12.4.30..38050     ClosWait
NFSCLNT  00000022 0.0.0.0..1017          *..*                 UDP
NFSCLNT  00000021 0.0.0.0..1018          *..*                 UDP
NFSCLNT  00000020 0.0.0.0..1019          *..*                 UDP
NFSCLNT  0000001F 0.0.0.0..1020          *..*                 UDP
NFSCLNT  0000001E 0.0.0.0..1021          *..*                 UDP
NFSCLNT  0000001D 0.0.0.0..1022          *..*                 UDP
NFSCLNT  0000001C 0.0.0.0..1023          *..*                 UDP
NFSMVS   00000054 0.0.0.0..2049          *..*                 UDP
NFSMVS   0000004E 0.0.0.0..10001         *..*                 UDP
NFSMVS   00000051 0.0.0.0..10002         *..*                 UDP
NFSMVS   0000004D 0.0.0.0..10000         *..*                 UDP
NFSMVS   00000053 0.0.0.0..10003         *..*                 UDP
PMAP     00000027 0.0.0.0..111           *..*                 UDP
SYSLOGD4 00000024 0.0.0.0..514           *..*                 UDP
```

1.3.2 The nslookup command

The name server lookup command tells you the name and the IP address of the registered name server and uses this connection to give you the IP address of a given server.

To issue the nslookup command, you use a TSO Command panel for z/OS and a Windows command prompt for the remote system.

The command provides the following information:

Server	Host name of DNS server
Address	IP address of the DNS server
Name	Host name of the requested server
Address	IP address of the requested server

Example 1-10 shows output from the nslookup command on the workstation.

Example 1-10 The nslookup command and its response

```
WASPD2 @ SC55:/u/waspd2>nslookup wtsc55.itso.ibm.com
Defaulting to nslookup version 4
Starting nslookup version 4
Server:  itsodns.itso.ibm.com
Address:  9.12.6.7

Name:    wtsc55.itso.ibm.com
Address:  9.12.4.28
```

1.3.3 The ping command

The ping command sends a request to a remote system and monitors the response time and the number of lost replies.

The ping command is mainly used on the client to check communication and the proper name and server configuration. If you can ping your WebSphere for z/OS system by using the name of the system, you can be almost certain that your client network configuration is fine. If you cannot ping your system, you should ping the IP address and use the nslookup command to check the name and server configuration.

To issue the ping command, you use a TSO Command panel on z/OS and a Windows command prompt on the remote system.

The ping command provides the following information:

No. of responses	The ping command counts the number of responses. On z/OS, the ping command only sends one request. On other client systems, you should specify a count to limit the number of requests; otherwise, you must interrupt the process by pressing **Ctrl+C**.
Response time	How many seconds elapsed until the reply came back.
Successes	How many tries were successful and many requests were lost.

Example 1-11 on page 11 shows sample output from the ping command on the workstation.

Example 1-11 The ping command and its response

```
C:\Documents and Settings\TOT188>ping wtsc55.itso.ibm.com

Pinging wtsc55.itso.ibm.com [9.12.4.28] with 32 bytes of data:

Reply from 9.12.4.28: bytes=32 time=4ms TTL=63
Reply from 9.12.4.28: bytes=32 time=4ms TTL=63
Reply from 9.12.4.28: bytes=32 time=4ms TTL=63
Reply from 9.12.4.28: bytes=32 time=7ms TTL=63

Ping statistics for 9.12.4.28:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 7ms, Average = 4ms
```

1.3.4 The tracert command

The trace route (**`tracert`**) command follows all intermediate network hops until it reaches a given IP address and informs you about any delays in this process.

The tracert command can be used from both ends of the communication to check for routers or bridges that are preventing communication or causing unnecessary delays.

Note: The trace route command in z/OS TSO/E is called TRACERTE and not tracert.

The tracert command provides the following information:

Address	Address of the target server
Delay	One line per hop, with IP address and delay in ms

Example 1-12 shows sample output from the tracert command on a workstation.

Example 1-12 The tracert command and its response

```
C:\Documents and Settings\TOT188>tracert plpsc.pok.ibm.com

Tracing route to plpsc.pok.ibm.com [9.56.214.1]
over a maximum of 30 hops:

  1     3 ms     3 ms     3 ms  pok6509r.itso.ibm.com [9.12.6.92]
  2     4 ms     4 ms     4 ms  9.56.1.189
  3     4 ms     4 ms     4 ms  pok-ud-2a-v993.pok.ibm.com [9.56.126.3]
  4     4 ms     4 ms     4 ms  pok-co-a-v808.pok.ibm.com [9.56.2.33]
  5     4 ms     4 ms     4 ms  pok-bd-b-ge0-4.pok.ibm.com [9.56.2.6]
  6     4 ms     4 ms     4 ms  pok-sc-a-v256.pok.ibm.com [9.56.1.6]
  7     7 ms     4 ms     5 ms  pok-sd-5b-ge2-1.pok.ibm.com [9.56.208.4]
  8     8 ms     5 ms     4 ms  plpsc.pok.ibm.com [9.56.214.1]

Trace complete.
```

1.4 USS and OMVS commands

Some USS and OMVS commands can be very powerful tools for performing day-to-day WebSphere for z/OS administrative tasks. We introduce five of them here:

- The df command for displaying a file system
- The du command for displaying usage of disk space
- The ps command for displaying process and thread information
- The DISPLAY command for thread details in a server region
- The WASgrep.sh command for searching string patterns in XML files

1.4.1 Display file system with df

In a z/OS USS (OMVS) environment, you allocate a hierarchical file system (HFS) data set with a disk space allocation of a specific fixed size. You can mount the HFS data set to a certain mount point and make it available to the application.

Using **df** (display file) as a common UNIX® command is particularly useful in a situation where you want to find out whether any HFS file system is full.

To use the command, first you go to a subdirectory that you would like to check. Then, enter the following command:

```
df -k .
```

where **-k** is the option flag request for the output to be in kilobytes and the **"."** means "this directory."

The df command displays:

- Mount-point name
- HFS data set name
- Total size of the file system and the remaining disk space that is still available

Figure 1-1 shows the df command output when executed from the OMVS command line.

.

```
WASPD2 @ SC55:/waspdconfig/pdcell>df -k ./*
Mounted on     Filesystem                Avail/Total    Files Status
/waspdconfig/pdcell (OMVS.WAS.PDCELL.CONFIG.HFS) 139200/288000  4294949456
Available
/waspdconfig/rescell (OMVS.WAS.RESCELL.CONFIG.HFS) 222224/288000  4294960903
Available
```

Figure 1-1 USS command df display

This display shows that there are two file systems mounted under the /waspdconfig subdirectory:

- ./pdcell has 143560 K available out of a total of 288000 K (50% full).
- ./rescell has 222224 K available out of a total of 288000 K (23% full).

1.4.2 Display disk space usage with du

The disk usage (**du**) command is used to show the amount of disk space that is consumed by one or more directories (or directory trees).

In a USS environment, we often encounter disk full or file system full problems. In such situations, it is very useful to know which file or directory has consumed most of the disk space.

In this section, we look at a simple tool that can display the files or subdirectories that use most of the disk space. To achieve this, we show how to “pipe” the output from the du command into a sort command and then “pipe” the sorted results into a head command.

First, you go to the file system that encountered the file system full problem. Then, you issue:

```
du -k . | sort -r | head -n 20
```

where:

- **du -k .** displays list that consists of all the files and subdirectories.
- **sort -r** sorts the list in reverse or descending order.
- **head -n 20** displays only the top 20 rows of the sorted list.

These three commands are connected with the UNIX “|” piping function.

This tool displays a list of rows sorted in descending order. Each row consists of two columns: the size in kilobyte block and the name of the subdirectories as shown in Example 1-13.

Example 1-13 The du command output

```
WASPD2 @ SC55:/waspdconfig/pdcell>du -k | sort -r | head -n 20
119580 .
   84644 ./DeploymentManager
   43436 ./DeploymentManager/installedApps
   43428 ./DeploymentManager/installedApps/pdcell
   43276 ./DeploymentManager/installedApps/pdcell/adminconsole.ear
   42596./DeploymentManager/installedApps/pdcell/adminconsole.ear/adminconsole.war
   29576./DeploymentManager/installedApps/pdcell/adminconsole.ear/adminconsole.war/WEB-INF
   27000 ./AppServerNodeA
   16144 ./DeploymentManager/wstemp
   14316 ./DeploymentManager/config
   13900 ./AppServerNodeA/config
   13724 ./DeploymentManager/config/cells
   13708 ./DeploymentManager/config/cells/pdcell
   13128 ./DeploymentManager/config/cells/pdcell/applications
   12392 ./AppServerNodeA/config/backup
   12384 ./AppServerNodeA/config/backup/base
   12188 ./DeploymentManager/config/cells/pdcell/applications/adminconsole.ear
   11920 ./AppServerNodeA/config/backup/base/cells
   11912 ./AppServerNodeA/config/backup/base/cells/pdcella
   11660 ./AppServerNodeA/config/backup/base/cells/pdcella/applications
```

1.4.3 Display thread information with ps

You can use the UNIX ps command to find out how many threads there are in an application servant region. To do this, you have to find out the process ID (PID) of the servant region as follows:

1. Using the z/OS SDSF display, find out the ASIDX (address space ID in hexadecimal).
2. With ASIDX, use the z/OS command from the console command to get the PID:

```
/d omvs,asid=xxx
```

3. From the OMVS command line, use the **ps** command:

```
ps -p <pid> -m | wc -l
```

Example 1-14 shows the result.

Example 1-14 The ps command to find the number of threads

```
SDSF DA SC49  SC49     PAG    0 SIO    53 CPU   4/  4  LINE 1-7 (7)
COMMAND INPUT ===>                                          SCROLL ===> CSR
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO   CPU% ASID ASIDX
     PDDEMN   PDDEMN   BBODAEMN STC12605 WSDMNCR1   NS  FE 7680   0.00   0.00   0.00   89  0059
     PDAGNTA  PDAGNTA  BBOCTL   STC12612 ASCR1      NS  FE  50T   0.00   1.18   0.02   92  005C
     PDSR01AS PDSR01AS BBOSR    STC18036 ASSR1      IN  FB  55T   0.00   0.00   0.00   94  005E
     PDDMGRS  PDDMGRS  BBOSR    STC15651 DMSR1      IN  FE 125T   0.00  30.84   0.02   97  0061
     PDSR01AS PDSR01AS BBOSR    STC18038 ASSR1      IN  FE  55T   0.00   0.01   0.02  101  0065
     PDDMGR   PDDMGR   BBOCTL   STC13312 ASCR1      NS  FE  54T   0.00   1.22   0.02 1005  03ED
     PDSR01A  PDSR01A  BBOCTL   STC18031 ASCR1      NS  FE  40T   0.00   0.01   0.04 1011  03F3

COMMAND INPUT ===> /D OMVS,ASID=65                          SCROLL ==
RESPONSE=SC49
 BPXO040I 18.21.20 DISPLAY OMVS 833
 OMVS     000E ACTIVE          OMVS=(5A)
 USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
 ASSR1    PDSR01AS 0065   84410478          1 HR---- 17.53.49     28.11
   LATCHWAITPID=         0 CMD=BBOSR

From OMVS command line:
>ps -p 84410478 -m | wc -l
     25
```

In this sample, you can see how we issued commands based on the result of the previous command:

1. **`SDSF`** shows the ASIDX for PDSR01AS as 0065.
2. **`/D OMVS,ASID=65`** shows the PID as 84410478.
3. **`ps -p 84410478 -m | wc -l`** gives the number of threads as 25.

1.4.4 Display thread details with DISPLAY

You can use the DISPLAY OMVS z/OS command to find out detailed information about threads in a particular application servant region as follows. This information includes:

- Thread Control Block (TCB) address
- Whether it is a Workload Manager (WLM) or a non-WLM thread
- CPU time usage

1. Obtain the PID of the servant region:
   a. From the z/OS SDSF display, you can find out the ASIDX (address space ID in hexadecimal).
   b. With ASIDX, use the z/OS command from the console command to get the PID:

      ```
      /d omvs,asid=xxx
      ```

2. Then, from the OMVS command line, use the z/OS command:

   ```
   /d omvs,pid=yyy
   ```

Figure 1-2 on page 15 shows the result of these commands.

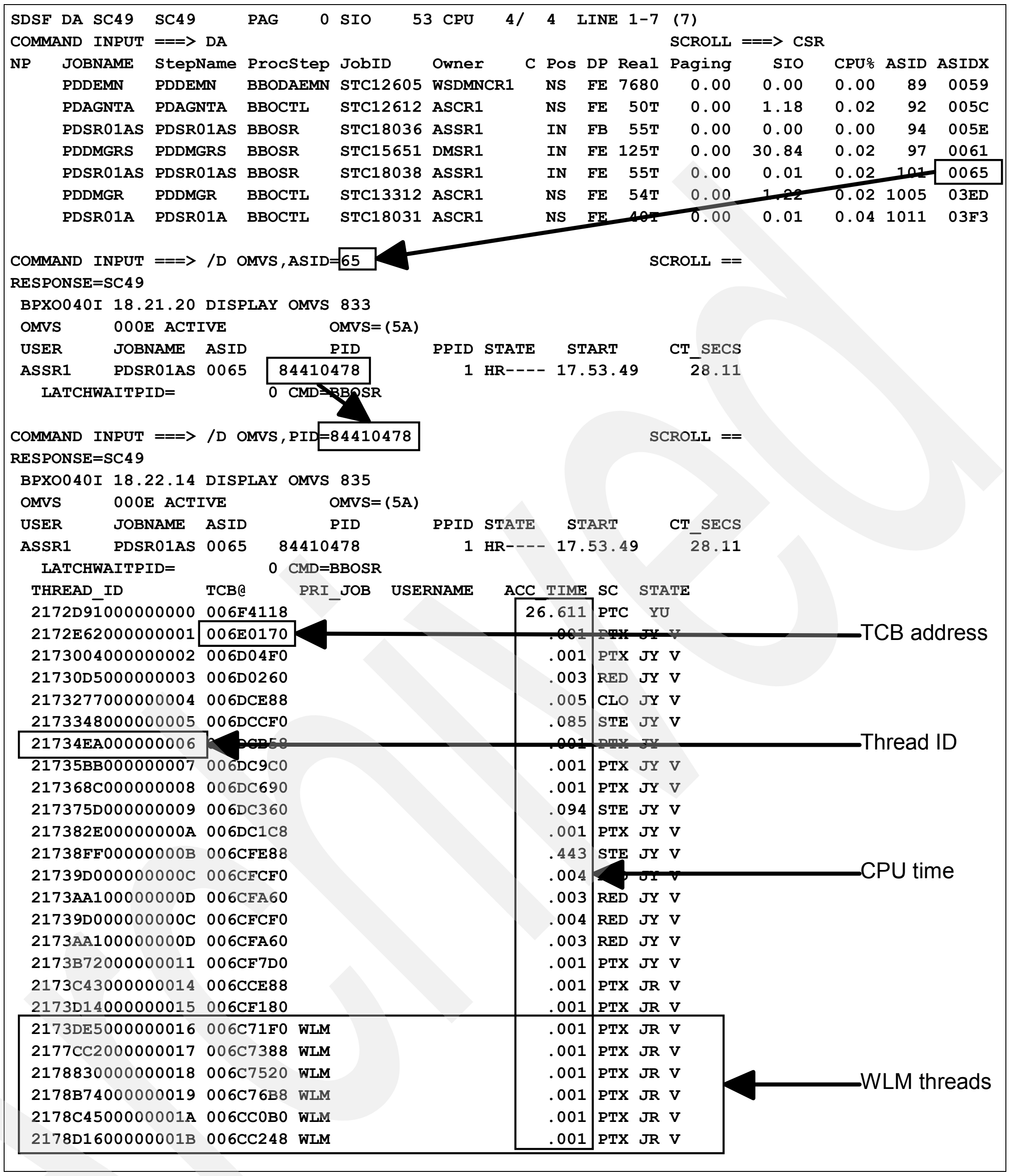

```
SDSF DA SC49   SC49      PAG    0 SIO    53 CPU   4/  4  LINE 1-7 (7)
COMMAND INPUT ===> DA                                        SCROLL ===> CSR
NP    JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO   CPU% ASID ASIDX
      PDDEMN   PDDEMN   BBODAEMN STC12605 WSDMNCR1   NS  FE 7680   0.00   0.00   0.00   89  0059
      PDAGNTA  PDAGNTA  BBOCTL   STC12612 ASCR1      NS  FE  50T   0.00   1.18   0.02   92  005C
      PDSR01AS PDSR01AS BBOSR    STC18036 ASSR1      IN  FB  55T   0.00   0.00   0.00   94  005E
      PDDMGRS  PDDMGRS  BBOSR    STC15651 DMSR1      IN  FE 125T   0.00  30.84   0.02   97  0061
      PDSR01AS PDSR01AS BBOSR    STC18038 ASSR1      IN  FE  55T   0.00   0.01   0.02  101  0065
      PDDMGR   PDDMGR   BBOCTL   STC13312 ASCR1      NS  FE  54T   0.00   1.22   0.02 1005  03ED
      PDSR01A  PDSR01A  BBOCTL   STC18031 ASCR1      NS  FE  40T   0.00   0.01   0.04 1011  03F3

COMMAND INPUT ===> /D OMVS,ASID=65                             SCROLL ==
RESPONSE=SC49
 BPXO040I 18.21.20 DISPLAY OMVS 833
 OMVS     000E ACTIVE          OMVS=(5A)
 USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
 ASSR1    PDSR01AS 0065   84410478          1 HR---- 17.53.49     28.11
   LATCHWAITPID=         0 CMD=BBOSR

COMMAND INPUT ===> /D OMVS,PID=84410478                        SCROLL ==
RESPONSE=SC49
 BPXO040I 18.22.14 DISPLAY OMVS 835
 OMVS     000E ACTIVE          OMVS=(5A)
 USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
 ASSR1    PDSR01AS 0065   84410478          1 HR---- 17.53.49     28.11
   LATCHWAITPID=         0 CMD=BBOSR
  THREAD_ID        TCB@     PRI_JOB  USERNAME   ACC_TIME SC  STATE
  2172D91000000000 006F4118                       26.611 PTC  YU
  2172E62000000001 006E0170                         .001 PTX JY V
  2173004000000002 006D04F0                         .001 PTX JY V
  21730D5000000003 006D0260                         .003 RED JY V
  2173277000000004 006DCE88                         .005 CLO JY V
  2173348000000005 006DCCF0                         .085 STE JY V
  21734EA000000006 006DCB58                         .001 PTX JY
  21735BB000000007 006DC9C0                         .001 PTX JY V
  217368C000000008 006DC690                         .001 PTX JY V
  217375D000000009 006DC360                         .094 STE JY V
  217382E00000000A 006DC1C8                         .001 PTX JY V
  21738FF00000000B 006CFE88                         .443 STE JY V
  21739D000000000C 006CFCF0                         .004 RED JY V
  2173AA100000000D 006CFA60                         .003 RED JY V
  21739D000000000C 006CFCF0                         .004 RED JY V
  2173AA100000000D 006CFA60                         .003 RED JY V
  2173B72000000011 006CF7D0                         .001 PTX JY V
  2173C43000000014 006CCE88                         .001 PTX JR V
  2173D14000000015 006CF180                         .001 PTX JR V
  2173DE5000000016 006C71F0 WLM                     .001 PTX JR V
  2177CC2000000017 006C7388 WLM                     .001 PTX JR V
  2178830000000018 006C7520 WLM                     .001 PTX JR V
  2178B74000000019 006C76B8 WLM                     .001 PTX JR V
  2178C4500000001A 006CC0B0 WLM                     .001 PTX JR V
  2178D1600000001B 006CC248 WLM                     .001 PTX JR V
```

Figure 1-2 Using DISPLAY OMVS to show thread information

1.4.5 Search string patterns with WASgrep.sh

The WASgrep.sh command searches for a particular string pattern in all of the ASCII-based XML files in the current directory and all subdirectories as follows:

1. The script does a **`find`** at the current directory to obtain a list of all the XML files in the current directory and subsequent subdirectories.
2. It converts each ASCII-based XML file into an EBCDIC-based temporary file.
3. It searches for the string pattern in the temporary file.

Example 1-15 on page 16 shows the contents of the shell script.

Example 1-15   WASgrep shell script

```
#!/bin/sh

ascii=${A2E_ASCII_CODEPAGE:-ISO8859-1}
ebcdic=${A2E_EBCDIC_CODEPAGE:-IBM-1047}

cmd="iconv -f $ascii -t $ebcdic"

if [ "$1" = "?" ]
then
  echo "This script searches for *.xml files in curent dir and subdirs"
  echo "This script will convert a file (ascii) to ebcdic and then"
  echo "search it for the specified string"
  echo "the original file(ascii)  is untouched"
  exit 0
fi

tempfile=/tmp/checkfile
for file in `find . -name "*.xml"`
do
    echo '--------------------------------------------'
    echo $file
    $cmd $file > $tempfile    # convert the file
    grep $1 $tempfile
    rm $tempfile
done

exit
```

To use the tool, you **cd** to an appropriate directory. Then, you run the shell script with a string pattern that you want to search:

```
<script-directory>/WASgrep.sh jmsQCF2
```

This tool displays the XML file name and all the text lines that have the string pattern. Example 1-16 shows that only the server.xml file contains the was.wlmTimeout search string.

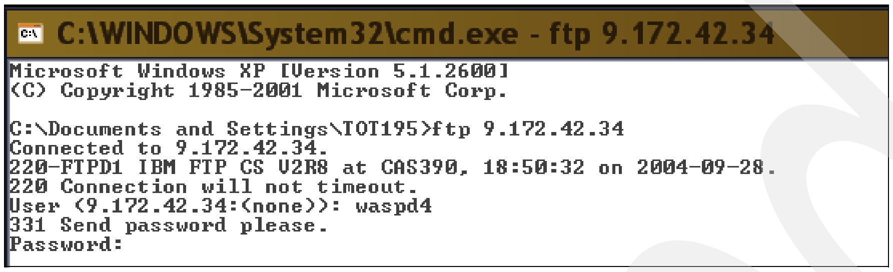
Example 1-16   Search string in server.xml file

```
cd /waspdconfig/pdcell/AppServerNodeB/config/cells/pdcell/nodes/pdnodeb/servers/pdsr02b/

/u/waspd2/WASgrep.sh was.wlmTimeout
-------------------------------------------
./namestore-cell.xml
-------------------------------------------
./namestore-node.xml
-------------------------------------------
./resources.xml
-------------------------------------------
./server.xml
  <properties xmi:id="Property_24" name="was.wlmTimeout" value="300"/>
-------------------------------------------
./variables.xml
-------------------------------------------
./ws-security.xml
```

1.5 Windows FTP command

The FTP client that is delivered with Windows offers the functionality of a standard FTP client with a character-based user interface. You can use it to transfer files to and from z/OS systems. Figure 1-3 shows a DOS window with this command:

```
ftp 9.172.42.34
```



```
C:\WINDOWS\System32\cmd.exe - ftp 9.172.42.34
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\TOT195>ftp 9.172.42.34
Connected to 9.172.42.34.
220-FTPD1 IBM FTP CS V2R8 at CAS390, 18:50:32 on 2004-09-28.
220 Connection will not timeout.
User (9.172.42.34:(none)): waspd4
331 Send password please.
Password:
```

Figure 1-3 FTP client delivered with Windows

After supplying the user ID and password you can issue PUT and GET commands to transfer files between the two systems.

2

Logs for problem determination in WebSphere for z/OS

In this chapter, we explain the following logs:

- Job logs and system log
- WebSphere error log (BBORBLOG)
- First Failure Data Capture
- Java Logging API
- IBM HTTP Server logs and trace.

For each log we provide information about the nature of the log, discuss when to use it, and demonstrate how to use it. We describe the output, show how to interpret it, and provide an example.

2.1 Job logs and system log

WebSphere is a collection of server instances working together. Each server instance is made up of a controller region and some number of servant regions. Each of these regions is an address space that, as with any other address space, has a job log that you can view through SDSF. If you run JES3, then you might be using EJES, the JES2 spool.

The system log (SYSLOG) is a record of the console messages. WebSphere messages issued to SYSLOG also show up in the job log. In most problem cases, these logs show information about exceptions, information about abnormal situations, or simply warning messages in the system. These are normally the first logs that you should examine when software problems occur.

2.1.1 When to use system log and job logs

The JES2 spool writes the information to the job logs of the address spaces for the different regions. There are several types of regions that are of interest:

- For each server instance, there is one control region and 0 or more servant regions (except for the daemon, for which there is only a control region).
  - Control regions, to put it simply, handle communication, receiving requests from clients and sending back responses.
  - Servant regions are given the requests to process, so they do the actual work.
- You might also have address spaces from local clients and your HTTP server.

Now, given all the job logs available, how do you know which ones have useful information? The answer depends on what is taking place at the time of the error.

In a typical situation, a client sends requests to the servant to process an action against some object. This client might be a local batch program, a remote Java application, a browser-based application, or the administrative console.

The requests are sent to the control region of the appropriate application server. Then, the requests are processed in a servant. Therefore, the best place to look for messages regarding the error is in the application servant region. This information might be supported by more detailed messages in the error log.

The application control region is of most interest in communication failures. For naming registration failures, the application server regions have messages regarding the failure. Again, the error log has more detailed messages to go along with the ones in the address space. Some failures result in a CEEDUMP being generated in the job log. Any regions with a CEEDUMP are always of interest. You should determine what is processing the work when the failure occurs.

As for the SYSLOG, in addition to the system messages, there are WebSphere for z/OS messages, but there could be messages from other products invoked by or related to WebSphere. Indications of dumps are also found in the SYSLOG.

2.1.2 How to set up system log and job logs

When the job output of an address space is viewed with a tool such as SDSF or EJES, it can be viewed either as one piece of output or as different sections.

Use the JOB statement MSGLEVEL parameter to request that the job control statements be printed in the job log output listing. Use **MSGLEVEL=(1,1)** to receive the maximum amount of information in the following order:

1. JES messages and job statistics
2. All job control statements in the input stream and procedures
3. Messages about job control statements
4. JES and operator messages about the processing of the job: allocation of devices; volumes, execution, and termination of job steps and the job; and disposition of data sets

2.1.3 System log and job log output and their interpretation

Each job log output section contains certain types of information:

- JESMSGLG: This section contains start-up messages, including a list of environment variable values and server settings, and the service level of WebSphere, for example:

  ```
  BBOM0007I CURRENT CB SERVICE LEVEL IS build level cf20523.06 release  829
  WAS601.ZNATV date 06/07/05 10:24:12.
  ```

  It also lists the Java service level when in a J2EE server region, for example:

  ```
  BBOJ0011I JVM Build is J2RE 1.4.2 IBM z/OS Persistent Reusable VM  070
  build cm142sr1a-20050209 (JIT enabled: jitc).
  ```

- JESJCL: This section lists the job control language (JCL) of the procedure running the address space. This is a useful place to look for incorrect STEPLIBs and other JCL-related issues.
- JESYSMSG: This section might list more messages, dump information, and provide a list of environment variables and server settings.
- CEEDUMP: An exception in the address space might cause this section to be generated. It lists failure information including trace backs (a trace back shows which functions were last called prior to the program failure).
- System output (SYSOUT): During normal processing, the SYSOUT should be empty, but there are situations that cause output to be written to this section. If the error log stream cannot connect, the messages set to be written to the error log are written to CERR, which goes to SYSOUT. Trace from the JVM™ occurs when you set these environment variables:
  - `control_region_jvm_logfile`
  - `server_region_jvm_logfile`,
  - `server_region_use_java_g` to `SYSPRINT`.
- SYSPRINT: The WebSphere for z/OS trace output can be written to SYSPRINT if the environment variable is:

  ```
  ras_trace_outputLocation=SYSPRINT.
  ```

Important: When an IBM service asks for a trace, always send the *entire* job output, because each section might have useful information that can help debug the problem.

The job log can have a variety of information based on environment variable settings. As a default, it obtains information about the job itself, such as life-cycle messages (when it started, finished initiating, and so on), the JCL used to run the job, data set utilization, and other typical JES messages.

There are also WebSphere messages, which start with the BBO prefix. The messages appearing on the console (what we refer to as SYSLOG messages) are typically related to configuration failures of other products, unrecoverable WebSphere configuration errors, and WebSphere life-cycle messages. Messages written explicitly to the job log are more general failure and warning messages. The more detailed messages supporting these can be found in the error log.

Other important pieces of useful information that always come out in the job log are the configuration messages. These list the values of the environment variables and the server properties.

You also have the option of using various traces and managing their output. For more information about traces, refer to Chapter 4 in *WebSphere Application Server for z/OS V6, Troubleshooting and support*, GA22-7964-03.

Example 2-1 shows a part of SYSLOG, where information generated at start-up for WebSphere for z/OS is captured. In this example, the release, build level, cell name, node name, and procedure name are displayed.

Example 2-1 SYSLOG sample at start-up

```
+BBOO0239I WEBSPHERE FOR Z/OS SERVANT PROCESS cl6422/nd6422/ws6422 IS
 STARTING.
+BBOM0007I CURRENT CB SERVICE LEVEL IS build level cf20523.06 release
 WAS601.ZNATV date 06/07/05 10:24:12.
```

For more information about the messages you can see in the SYSLOG and job log, refer to *WebSphere Application Server for z/OS V6 Messages and Codes in the infocenter.* Also refer to the messages appendix in this book.

2.2 WebSphere error log (BBORBLOG)

This section describes the WebSphere for z/OS error log. This error log is where WebSphere for z/OS saves the detailed error messages from its runtime servers. The WebSphere Application Server for z/OS error log stream is a system logger application. The system logger is a z/OS component that allows applications to log data in a sysplex.

The System logger creates and manages log streams that are written first to a coupling facility or local in-memory buffer and then transferred to log data sets on a direct access storage device (DASD) for longer term access. Log streams that are written to local buffers rather than to a coupling facility are called DASD-only log streams. For more information about the system logger and sysplex refer to *z/OS V1R6.0 MVS Setting Up a Sysplex,* SA22-7625-10.

Note: There is a significant performance penalty when DASD-only error logging is used.

2.2.1 When to use BBORBLOG

The WebSphere for z/OS error log is a log stream data set that uses the system logger to record messages. These messages are typically more detailed than those from the system console or from the job output of the server address space.

The advantage of the BBORBLOG log is that the error logs from the various WebSphere Application Server address spaces are consolidated into one log. The consolidation includes control region, server region, and daemon error logs. This is very useful when you are trying to compare timestamps and errors from the different servers.

If the error log is a DASD-only log stream, this provides single system-only error logging. If you are using the Coupling Facility for the system logger, then action is sysplex \-wide and all the systems in the sysplex can log data in one place.

2.2.2 How to set up BBORBLOG

The BBORBLOG is set up during the server installation using the customization dialogs and all the jobs required for the setup are created for you. If the BBORBLOG was not set up during initial installation, then it can be enabled later. The detailed information can be found in the *WebSphere Application Server for z/OS, Version 6, Installing your application serving environment*, GA22-7957. Here is a brief summary of the required steps:

1. Create a new CFRM policy with the new structure definition. Refer to sample job BBOWCFRM.
2. Define a WebSphere error log stream by updating the LOGR policy. Refer to sample job BBOERRLG.
3. Check for security authorization. Refer to sample jobs BBODBRAC and BBOWBRAC.
4. Update the ras_log_logstreamName environment variable using the Administrative Console:

   Environment → **WebSphere Variables** → Scope node.

   a. Locate the variable ras_log_logstreamName with the logstream name.
   b. Once updated click **OK**.
   c. On the tool bar, select **Save** → **Save.**

   Note: Timestamps will appear in GMT unless changed by setting the WebSphere Application Server variable ras_time_local to 1.

The change takes effect when the server is recycled.

During server initialization, an attempt is made to connect to the appropriate log stream data set. If this connection is successful, you see messages such as these, which indicates that the name of the data set being used:

```
+BBOM0001I   ras_log_logstreamName: WAS.SC42.ERROR.LOG
+BBOO0024I ERRORS WILL BE WRITTEN TO WAS.SC42.ERROR.LOG LOG STREAM FOR
 JOB WS6422S.
+BBOO0153I THE FOLLOWING NUMBER OF MESSAGES WERE WRITTEN TO CERR PRIOR
 TO CONNECTING TO LOGSTREAM: 1.
```

If, however, the server cannot connect to the log stream, the message is instead written to CERR, which puts it in the SYSOUT of the job output. This is indicated by the message:

```
BBOO0024I ERRORS WILL BE WRITTEN TO CERR FOR JOB <server name>
```

Important: Even if the server successfully connects to the log stream, there will still be a message saying that errors will be written to CERR. This is because during initialization, before the connection to the log stream is made, errors are written to CERR.

When they are written to CERR, or SYSOUT, messages have a header that looks like that shown in Example 2-2 on page 24. Notice that it is prefaced with BossLog.

Example 2-2 BBORBLOG sample header

```
BossLog: { 0001} 2005/08/08 13:06:36.265 01 SYSTEM=SC42 SERVER=<none>

You can view the error log stream output using the BBORBLOG browser. To invoke the browser,
go to ISPF option 6 and enter:

ex ’BBO.SBBOEXEC(BBORBLOG)’ ’WAS.SC42.ERROR.LOG’
```

In this example, BBORBLOG resides in BBO.SBBOEXEC, and WAS.SC42.ERROR.LOG is the LOG_STREAM_NAME that was configured in the administrative console. Other examples include:

```
ex ’BBO.SBBOEXEC(BBORBLOG)’ ’WAS.SC42.ERROR.LOG 80’
ex ’BBO.SBBOEXEC(BBORBLOG)’ ’WAS.SC42.ERROR.LOG noformat’
```

The browser creates a data set named USERID.LOG_STREAM_NAME (for example, `WASPD3.WAS.SC42.ERROR.LOG`), which contains the formatted contents of the log stream. When the browser is started, it:

1. Allocates the USERID.LOG_STREAM_NAME data set, overwriting any duplicates
2. Populates the data set with the contents of the log stream
3. Puts the user in browse mode on the data set

Important: Each time BBORBLOG is invoked, a static file is created that overwrites the existing file. To refresh the file, it is necessary to re-issue BBORBLOG. If you want to keep the last log, you must rename it before running the tool again.

You can also invoke the BBORBLOG utility from the OMVS shell or Telnet using the shell script shown in Example 2-3.

Example 2-3 Shell script to format error log stream from MVS shell or telnet

```
/* REXX */
/* trace r */
parse arg logstrm format .
if logstrm = '' then logstrm = "WAS.ERROR.LOG"
if format = '' then format = "80"
qual =userid()
file_name = "/tmp/" || qual || ".errorlog"
"rm " || file_name
"touch " || file_name
call syscalls 'SIGOFF'
call bpxwdyn "alloc fi(bbolog) path('" || file_name || "')"
address LINKMVS "BBORBLOG logstrm format"
call bpxwdyn "free fi(bbolog)"
"vi " || file_name
exit(0)
```

2.2.3 BBORBLOG output and its interpretation

The output of BBORBLOG looks similar to that shown in Example 2-4.

Example 2-4 Error logstream output with line numbers

```
1| 2005/08/08 20:11:04.658 01 SYSTEM=SC42 SERVER=WS6422   JobName=WS6422
2| ASID=0X0403 PID=0X0301014D TID=0X22172FA0 0X000019 c=UNK
3| ./bbooboat.cpp+3152 ... BBOO0011W The function
4| ACR_ExecutionThread::ProcessInboundRequest(acrwObj *, ThreadCleanUp
5| *, BOSS_Object_Key &, Internal_CORBA_Request &)+3152 received CORBA
6| system exception CORBA::INTERNAL.  Error code is C9C2102F.
```

The line numbers together with the output listed in Table 2-1 can help you analyze and understand the output of this log.

Table 2-1 Parts of server log stream record output

Line number	Component	Description
1	`2005/08/08 20:11:04.658 01`	Date, timestamp, 2-digit record version number
1	`SYSTEM=SC42`	System name
1	`SERVER=WS6422`	Server name
2	`ASID=0X0403`	ASID
2	`PID=0X0301014D`	PID
2	`TID=0X22172FA0 0X000019`	Thread identifier (TID)
2	`c=UNK`	Request correlation information
Lines 1 and 2 help identify when and where the error occurred.		
3	`./bbooboat.cpp+3152`	File name and line
3	`BBOO0011W`	Log message number
The message number can provide detailed information about the error.		
3, 4, 5, 6	`The function`	Log message
It shows you which function was active in the moment of error; useful information for describing the problem to the IBM Support Center.		
6	`Error code is C9C2102F`	Error code
Sometimes, the error code is more meaningful than the message number.		

2.2.4 Related references

The following references can provide further details about BBORBLOG:

- *WebSphere Application Server for z/OS V6, Troubleshooting and support*, GA22-7964-03
- For more information about messages, refer to the *WebSphere for z/OS Information Center* at:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp
- For assistance with error messages refer to Appendix A, “Messages and codes” on page 99.

2.3 First Failure Data Capture

This section describes the First Failure Data Capture ((FFDC) facility that is included as part of WebSphere Application Server V5 and later for all platforms. With the FFDC tool, system administrators can identify runtime issues with their WebSphere Application Server in any environment. If an exception occurs, the FFDC tool traps and logs the exception, and then returns control to the thread that threw the exception.

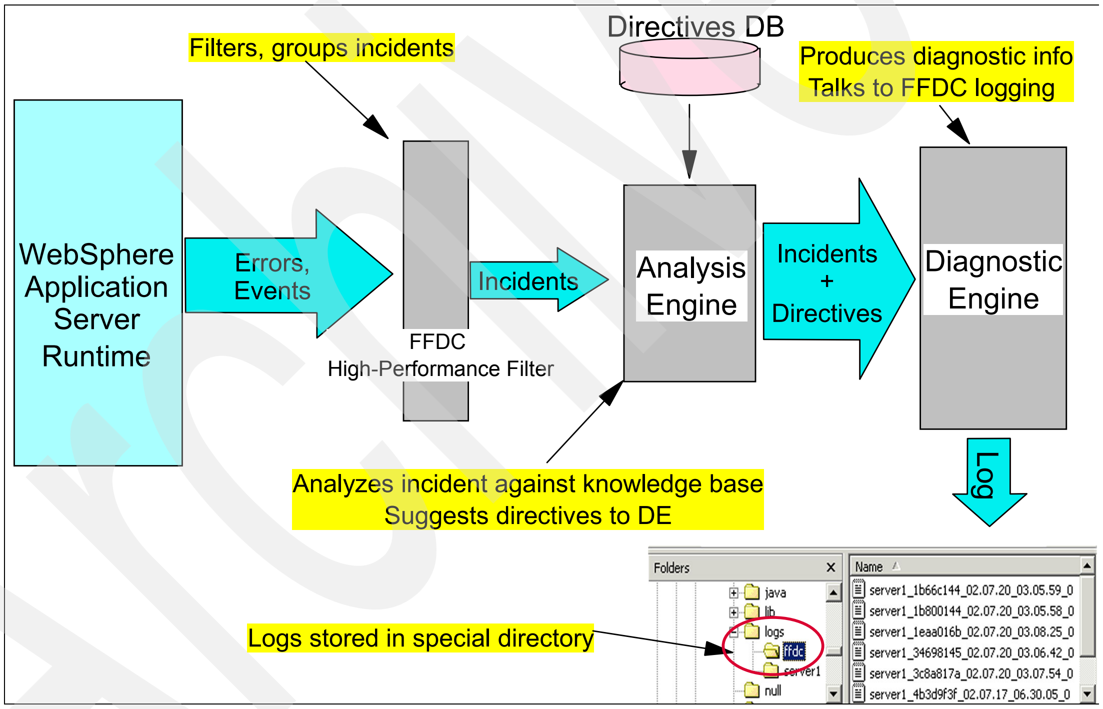
In WebSphere for z/OS V6.0.2, FFDC is enabled by default; in previous versions it was disabled.

Important: Because each WebSphere address space writes to its own FFDC file, the FFDC files can accumulate over time and take up a significant amount of space. We recommend that, if FFDC is enabled, the FFDC directory be pruned periodically to remove old FFDC files that are no longer required.

2.3.1 When to use FFDC

Because the I/O of the information captured by the FFDC tool is infrequent, FFDC has virtually no runtime performance impact. The artifacts produced by the FFDC tool are incremental rather than repeated, which results in very little use of the file system resources.

The FFDC tool is delivered as part of any WebSphere Application Server Base or Network Deployment installation and consists of a set of components. These components are the error filter, the analysis engine, the knowledge base that feeds the analysis engine, the diagnostic engine, and a set of diagnostic modules as shown in Figure 2-1.



```
server1_1b66c144_02.07.20_03.05.59_0
server1_1b800144_02.07.20_03.05.58_0
server1_1eaa016b_02.07.20_03.08.25_0
server1_34698145_02.07.20_03.06.42_0
server1_3c8a817a_02.07.20_03.07.54_0
server1_4b3d9f3f_02.07.17_06.30.05_0
```

Figure 2-1 FFDC tool architectural overview

The servant regions of WebSphere for z/OS have been instrumented with calls to the FFDC tool. When an exception occurs, the event is passed through the error filter to the diagnostic engine. If the analysis engine is enabled, it retrieves any information that relates to the event from the symptom database.

Each component of WebSphere Application Server on z/OS contains a symptom database for the FFDC tool. This symptom database is located in the <install_root>/properties/logbr/ffdc/adv/ directory installed under all members of the cell, including the deployment manager and all installed nodes.

When the analysis engine tool is run against an exception log, the symptom database, *ffdcdb.xml*, is used in an attempt to provide a solution to the exceptions that are captured by the FFDC tool. The analysis engine uses the exception name, the source ID, and the probe ID from the exception as the key to the symptom database. If a match exists, the solution is displayed as a result of running the analysis engine.

2.3.2 How to set up the FFDC tool

There is no centralized control of the FFDC tool in WebSphere for z/OS. It must be activated for each member in the cell through the ffdcRun.properties file for that cell member. This activation is done through the ffdcRun.properties file, which is found in the properties directory for all members of a z/OS cell. This includes the deployment manager in a Network Deployment installation and all nodes in the cell.

If a particular cell member is of interest during a runtime problem determination session, only that particular FFDC capability has to be activated. The only property that needs to be reset in the ffdcRun.properties file is the Level property. For a WebSphere for z/OS V5 installation, the Level property is set to a value of “0” by default. Figure 2-2 shows the possible Level values and their effect.

```
# Level of processing to perform
#  0 - none
#  1 - monitor exception path
#  2 - dump the call stack, with no advanced processing
#  3 - 2, plus object interspecting the current object
#  4 - 2, plus use DM to process the current object
#  5 - 4, plus process the top part of the call stack with DMs
#  6 - perform advanced processing the entire call stack
```

Figure 2-2 ffdcRun.properties level values

In this example, the default value is set to *4*. We recommend that you use the same value as a starting point. However, if you want to set it to another logging level, use the following process:

1. Locate the ffdcRun.properties file for the address space of interest.
2. Open the ffdcRun.properties file for editing in any ASCII-capable editor.
3. Set the Level key of the ffdcRun.properties to the level of your choice, see Figure 2-2.
4. Save the ffdcRun.properties file.
5. Restart the address space.

Important: In z/OS, each address space produces its own set of FFDC files. Therefore, it is important to determine which address space is of interest during the runtime debug process to enable the FFDC tool for that specific address space.

2.3.3 FFDC output and its interpretation

The FFDC tool produces a set of files including:

- An index file that references all of the exceptions logged by FFDC, as shown in Example 2-5
- An exception file for each exception type from each probe

Example 2-5 FFDC index file

```
Index  Occur Time of last Occurence   Exception SourceId ProbeId
```

```
       ences
------------------------------------------------------------------------
1      1     04.09.27 14:58:58:371 GMT java.lang.IllegalStateException
com.ibm.ws.webcontainer.servlet.ServletManager.doService 3891
2      1     04.09.27 14:58:58:435 GMT java.lang.IllegalStateException
com.ibm.ws.webcontainer.servlet.ServletInstance.service 2821
3      1     04.09.27 14:58:58:391 GMT java.lang.IllegalStateException
com.ibm.ws.webcontainer.servlet.StrictLifecycleServlet._service 1901
4      334   04.09.27 14:58:59:577 GMT java.lang.ClassNotFoundException
com.ibm.ws.classloader.CompoundClassLoader.loadClass 248
5      1     04.09.27 14:58:58:553 GMT java.lang.IllegalStateException
com.ibm.ws.webcontainer.webapp.WebAppRequestDispatcher.handleWebAppDispatch 746 6     1
04.09.27 14:58:58:590 GMT
com.ibm.ws.webcontainer.servlet.exception.UncaughtServletException
com.ibm.ws.webcontainer.webapp.WebAppRequestDispatcher.dispatch 428
```

The index file uses a naming convention of *<server name>*_exception.log and has the following columns:

- Index
  This column is used to determine the number of rows in the table. A plus sign (+) in front of this value signifies that this value has been updated since the last persistence of the information to the file system.
- Occurrences
  This column signifies the number of times that the exception has occurred.
- Time of last Occurrence
  A time stamp of the last occurrence of this exception.
- Exception
  The name of the Java exception class that was captured by the FFDC tool.
- SourceId
  The unique identifier for the exception source.
- ProbeId
  The unique identifier for the probe used in the data capture.

The exception file in Example 2-6 was produced by a 4 setting for Level in the ffdcRun.properties file. It has the stack trace for a *java.lang.IllegalStateException* and a dump of the object *this and* its properties. This was not captured in the z/OS system log, so the information captured by FFDC was over and above the normal logging.

Example 2-6 FFDC exception file

```
------Start of DE processing------ = [04.09.27 14:58:58:371 GMT] , key = java.lang.IllegalStateException
com.ibm.ws.webcontainer.servlet.ServletManager.doService 3891
Exception = java.lang.IllegalStateException
Source = com.ibm.ws.webcontainer.servlet.ServletManager.doService
probeid = 3891
Stack Dump = java.lang.IllegalStateException: Context has not been prepared for next connection
   at com.ibm.ws.webcontainer.srt.NilSRPConnection.getHeaderNames(SRTConnectionContext.java:482)
   at com.ibm.ws.webcontainer.srt.SRTServletRequest.prepareHeader(SRTServletRequest.java(Compiled Code))
   at com.ibm.ws.webcontainer.srt.SRTServletRequest.getHeader(SRTServletRequest.java:307)
   at
.
.
.
   at com.ibm.ws390.orb.ORBEJSBridge.invoke(ORBEJSBridge.java:170)

Dump of callerThis =
Object type = com.ibm.ws.webcontainer.servlet.StrictServletInstance
com.ibm.ws.webcontainer.servlet.StrictServletInstance@4045fee5

Exception = java.lang.IllegalStateException
Source = com.ibm.ws.webcontainer.servlet.ServletManager.doService
probeid = 3891
Dump of callerThis =
Object type = com.ibm.ws.webcontainer.servlet.StrictServletInstance
  class$com$ibm$ws$webcontainer$servlet$StrictServletInstance =
    serialPersistentFields = {}
    serialVersionUID = 3206093459760846163
    allPermDomain = null
    getPDperm = null
    have_extensions = true
  _servicingCount = 0
  _servletClassname = com.ibm.ws.cache.servlet.ServletWrapper
  _servletName = action
  _servlet =
    class$com$ibm$ws$cache$servlet$ServletWrapper =
      serialPersistentFields =
this.class$com$ibm$ws$webcontainer$servlet$StrictServletInstance.serialPersistentFields
      serialVersionUID = 3206093459760846163
      allPermDomain = null
      getPDperm = null
      have_extensions = true
    applicationUnAvailList =
      class$java$lang$Object = null
      size = 0
      elementData = [Ljava.lang.Object;@3b07e96
      serialVersionUID = 8683452581122892189
      modCount = 0
    firstTime = true
    wrapsCacheableServlet = false
cacheEntrySet = true
    cacheEntry = null
    proxied =
      definitionsFactory = org.apache.struts.tiles.definition.ReloadableDefinitionsFactory@25807ee1
      lStrings = java.util.PropertyResourceBundle@743b3e83
      LSTRING_FILE = javax.servlet.http.LocalStrings
      HEADER_LASTMOD = Last-Modified
      HEADER_IFMODSINCE = If-Modified-Since
      METHOD_TRACE = TRACE
      METHOD_PUT = PUT
      METHOD_POST = POST
      METHOD_OPTIONS = OPTIONS
      METHOD_GET = GET
      METHOD_HEAD = HEAD
      METHOD_DELETE = DELETE
config = this._config
    tc =
      ivLogger = null
      ivResourceBundleName = com.ibm.ws.cache.resources.dynacache
      ivDumpEnabled = false
      defaultMessageFile = com.ibm.ejs.resources.seriousMessages
      ivEntryEnabled = false
      ivEventEnabled = false
      ivDebugEnabled = false
      ivName = com.ibm.ws.cache.servlet.ServletWrapper
  tc =
    ivLogger = null
    ivResourceBundleName = com.ibm.ejs.resources.seriousMessages
    ivDumpEnabled = false
    defaultMessageFile = com.ibm.ejs.resources.seriousMessages
    ivEntryEnabled = false
    ivEventEnabled = false
    ivDebugEnabled = false
    ivName = com.ibm.ws.webcontainer.servlet.StrictServletInstance
  syncObject = java.lang.Object@40457ee5
  servicingCount = 1
  _implementsSTM = false
  _config =
    _servletName = action
_initParams =
      hexDigit = [C@523dfeb5
      whiteSpaceChars =
      specialSaveChars = =:
      #!
      strictKeyValueSeparators = =:
      keyValueSeparators = =:
      defaults = null
      serialVersionUID = 4112578634029874840
      class$java$util$Hashtable$Entry = java.lang.Class@68cbe4b
      emptyIterator = java.util.Hashtable$EmptyIterator@522cbeb5
      emptyEnumerator = java.util.Hashtable$EmptyEnumerator@522d3eb5
      ENTRIES = 2
      VALUES = 1
      KEYS = 0
      values = null
      entrySet = null
      keySet = null
      modCount = 11
      loadFactor = 0.75
      threshold = 17
      count = 10
      table = [Ljava.util.Hashtable$Entry;@40657ee5
    _servletContext = com.ibm.ws.webcontainer.webapp.WebApp@16503e92
  _unavailableUntil = -1
_servicingState =
    _instance = this._servicingState
  _state =
    _instance = this._state
  PERMANENTLY_UNAVAILABLE_FOR_SERVICE_STATE =
    _instance = this.PERMANENTLY_UNAVAILABLE_FOR_SERVICE_STATE
  UNAVAILABLE_FOR_SERVICE_STATE =
    _instance = this.UNAVAILABLE_FOR_SERVICE_STATE
  AVAILABLE_FOR_SERVICE_STATE = this._servicingState
  ERROR_STATE =
    _instance = this.ERROR_STATE
  DESTROYED_STATE =
    _instance = this.DESTROYED_STATE
  DESTROYING_STATE =
    _instance = this.DESTROYING_STATE
  STM_SERVICING_STATE =
    _instance = this.STM_SERVICING_STATE
  SERVICING_STATE = this._state
  IDLE_STATE =
    _instance = this.IDLE_STATE
  INITIALIZING_STATE =
    _instance = this.PRE_INITIALIZED_STATE
  PRE_INITIALIZED_STATE =
_instance = this.PRE_INITIALIZED_STATE
```

The information dumped from the object *this* provides extra context information for the stack, including the member variables, calling objects, and so on. This information is of limited value when you are doing a problem determination by inspection. However, it can provide valuable information to the analysis engine when the exception log can be correlated to the symptom database. The real value of this exception log is the capture of the exception, during runtime, of the exception stack trace.

The combination of the exception, source ID, and the probe ID form an index key that is used to identify the exception log that has the FFDC exception information.

The exception file for each exception uses a naming convention of *<server name>_<thread Id>_yy.MM.dd_HH.mm.ss_<unique id>*.txt and contains information that is relative to the value of the ffdcRun.properties Level property value. The higher the value of the Level property, the greater the amount information in the exception file. Refer to "How to set up the FFDC tool" on page 27 for an explanation of the information produced by each logging level.

Once enabled, the FFDC tool produces the index and exception logs that are associated with the address space and persist the to the <install base>/logs/ffdc directory. Retrieval of these log files can be done by using an FTP client from any other environment. As an example, the index and exception logs could be retrieved with the ASCII setting for the FTP client on a Microsoft® Windows host. Because the index and exception logs are text files, they can be viewed in any ASCII-capable text editor or viewer.

2.3.4 Example: Using the FFDC tool for problem determination

This section describes a real-life scenario where the FFDC tool was used to resolve an issue at a customer site.

We encountered the following problem: When multiple administrators were logged on to the administrative console, a synchronization error for a shared resource occurred as a result of contention over a shared context object. The symptom was an HTTP 500 error on one of the administrative consoles.

We ran the FFDC tool and captured the exception:

```
java.lang.IllegalStateException: Context has not been prepared for next connection
```

This information signified that an investigation of a thread synchronization issue was appropriate.

The FFDC tool was enabled, as described in "How to set up the FFDC tool" on page 27. In particular, the ffdcRun.properties file was edited and the Level property was changed from a value of `0` to a value of `4` for this problem determination session. After the value was changed, the edited file was put in place and the deployment manager address space was restarted.

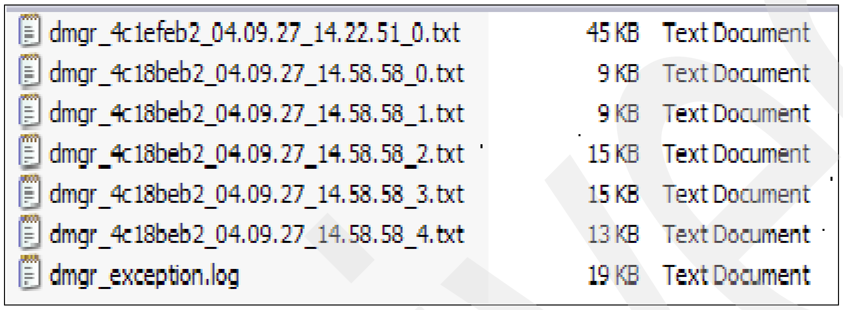
Preliminary investigation

Having retrieved the ASCII FFDC index file and exception logs, several steps were followed in order to determine that a `java.lang.IllegalStateException` had occurred:

1. The dmgr_exception.log was opened using WordPad to format the data.

Note: Figure 2-3 has been edited to eliminate all occurrences of the `com.ibm.ws.classloader.CompoundClassLoader.loadClass` exception. Only the last of these exceptions is shown.

2. Upon visual inspection of the dmgr_exception.log file, it was clear that the `IllegalSateException` was of interest. Note how the `java.lang.Illegal state` exception is originating from multiple classes. This is because the exception is being trapped by the FFDC tool as it traverses the call stack.
3. Having the information, we began to inspect the exception logs that the FFDC tool had produced and that we had retrieved from the z/OS system. Here is where we noted a weakness between the index file and the naming of the exception logs: It was not clear which exception log contained the `java.lang.IllegalStateException` without opening each file and inspecting them.
4. Once again, we used WordPad on a Windows client to open each of these files in a formatted manner. The set of files had the names shown in Figure 2-3.



dmgr_4c1efeb2_04.09.27_14.22.51_0.txt	45 KB	Text Document
dmgr_4c18beb2_04.09.27_14.58.58_0.txt	9 KB	Text Document
dmgr_4c18beb2_04.09.27_14.58.58_1.txt	9 KB	Text Document
dmgr_4c18beb2_04.09.27_14.58.58_2.txt	15 KB	Text Document
dmgr_4c18beb2_04.09.27_14.58.58_3.txt	15 KB	Text Document
dmgr_4c18beb2_04.09.27_14.58.58_4.txt	13 KB	Text Document
dmgr_exception.log	19 KB	Text Document

Figure 2-3 Index and exception Logs

As you can see, the dmgr_4c8beb2_04.09.27_14.58.58_0.txt file had the first instance of the java.lang.IllegalStateException.

In addition to the exception and the call stack, information in the exception log includes class and state information. This interpretation of the information over and above the stack trace is meant to be used by IBM to debug the problem and has little meaning outside of the IBM support network. Therefore, forward the exception logs to IBM Support to obtain further information about the captured exception.

Running the FFDC analysis engine on the exception log

In Example 2-7, the results of running the analysis engine against the dmgr_4c18beb2_04.09.27_14.58.58_0.txt exception log with the java.lang.IllegalState exceptions are displayed. As you can see, no solution was found in the symptom database. This is the expected result because the issue that was uncovered was unknown at the time and resulted in a change to the code base.

The analysis engine was run on a Windows server™ where WebSphere Application Server V5.1 had been previously installed. Example 2-7 illustrates the command line and the arguments used to invoke the analysis engine. Note that the analysis engine functionality is in the ffdc.jar library.

Example 2-7 Analysis engine output

```
C:\ffdclogs>java -classpath c:/WebSphere/AppServer/lib/ffdc.jar -Djava.
ext.dirs=c:/WebSphere/AppServer/lib com.ibm.ws.ffdc.AnalysisEngineTool
dmgr_4c18beb2_04.09.27_14.58.58_0.txt

c:/WebSphere/AppServer/properties/logbr/ffdc/adv/ffdcdb.xml
----------------------------------------------------------------------
Key Value :  java.lang.IllegalStateException com.ibm.ws.webcontainer.servlet.ServletManager.doService 3891
Exception Name : java.lang.IllegalStateException
```

```
Solution : ****** NOT FOUND -- See attached call stack for more details about the problem.
java.lang.IllegalStateException com.ibm.ws.webcontainer.servlet.ServletManager.doService 3891
Stack Dump = java.lang.IllegalStateException: Context has not been prepared for next connection
at com.ibm.ws.webcontainer.srt.NilSRPConnection.getHeaderNames(SRTConnectionContext.java:482)
        at com.ibm.ws.webcontainer.srt.SRTServletRequest.prepareHeader(SRTServletRequest.java(Compiled
Code))
        at com.ibm.ws.webcontainer.srt.SRTServletRequest.getHeader(SRTServletRequest.java:307)
.
.
.
        at com.ibm.ws390.orb.ORBEJSBridge.invoke(ORBEJSBridge.java:170) ******
```

In this case, the analysis engine did not find a solution in the symptom database, as reported by this statement:

```
Solution: ****** NOT FOUND
```

The conclusion of the analysis, by inspection, resulted in further investigation by the WebSphere development team. Based on the java.lang.IllegalStateException, synchronized code that was responsible for managing the administrative console was studied, and it was determined that a race condition existed for the shared connection resource. The code was then updated so that the appropriate synchronization occurred for the shared resource.

It took less than one hour to run the FFDC tool in the customer environment and, in this case, it provided an important clue in resolving the problem.

2.4 Java Logging API

Developing applications and maintaining them are complex tasks. When a running application encounters an unexpected condition, it might not be able to complete a requested operation. In such a case, you want the application to inform the administrator that the operation has failed and explain why. Those who develop or maintain applications need to gather detailed information relating to the execution path of a running application to determine the root cause of a failure that is due to a code bug. The facilities that are used for these purposes are typically referred to as message logging and diagnostic trace. Java Logging API implements this facility.

Previous versions of WebSphere for z/OS exposed an API called JRas. Java logging and JRas have similar functionality, but Java Logging API makes application logging portable to other compliant containers. The JRas API is still exposed to applications.

Internally, WebSphere for z/OS still uses JRas in conjunction with Java Logging, although that functionality is deprecated in V6.

WebSphere also supports Jakarta Commons-Logging, which defines a common programming model for logging and framework that enables applications to bind different logger implementations. It provides the APIs plus thin configuration file implementations.

For more detailed information about Jakarta Commons-Logging, refer to the “Jakarta Commons Logging” topic in WebSphere for z/OS Information Center, available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

2.4.1 What is the Java Logging API and when to use it

The Java Logging API is a standard logging API for your applications provided by the java.util.logging package.

One of the drawbacks of using System.out.println statements is that the application only has two channels: Standard output or Standard error. Piping all debugging messages into a single point results in a deficiency of error granularity. Once an application goes into production, many of the logging messages can be either redundant or unnecessary, causing production logs to become extremely large.

Java Logging does not distinguish between tracing and message logging, although previous versions of WebSphere for z/OS have made a clear distinction between them. In WebSphere for z/OS, the differences between tracing and message logging are:

- Tracing messages are messages with lower severity.
- Tracing messages generally are not localized.
- When tracing is enabled, a much higher volume of messages is produced.
- Tracing messages provide information for problem determination.

WebSphere for z/OS and application code use the Logger object to put data into the logstream. The Logger objects can use one or more Handler. Each Handler is an output device. The Filters are used to reduce the amount of information sent to the stream. The Formatter is used to format log data. Figure 2-4 shows the elements of the Java Logging Architecture in context.

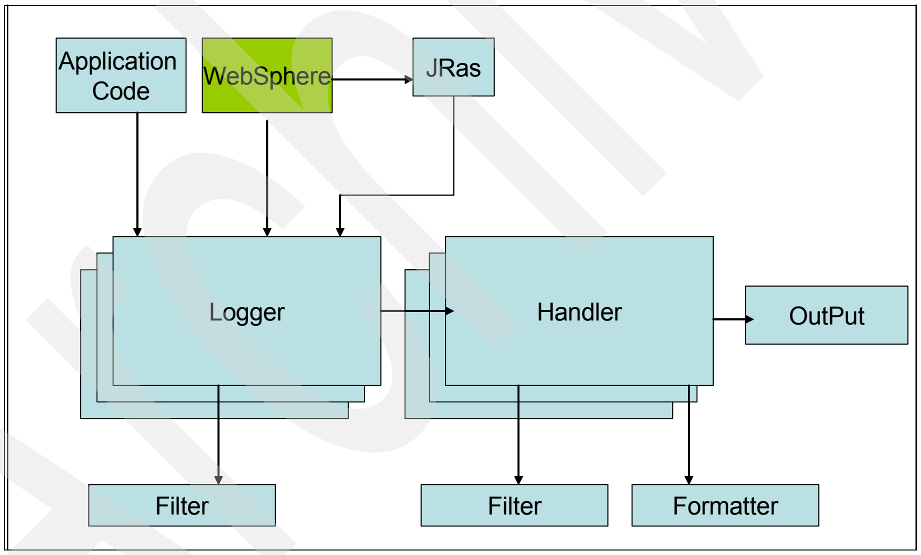


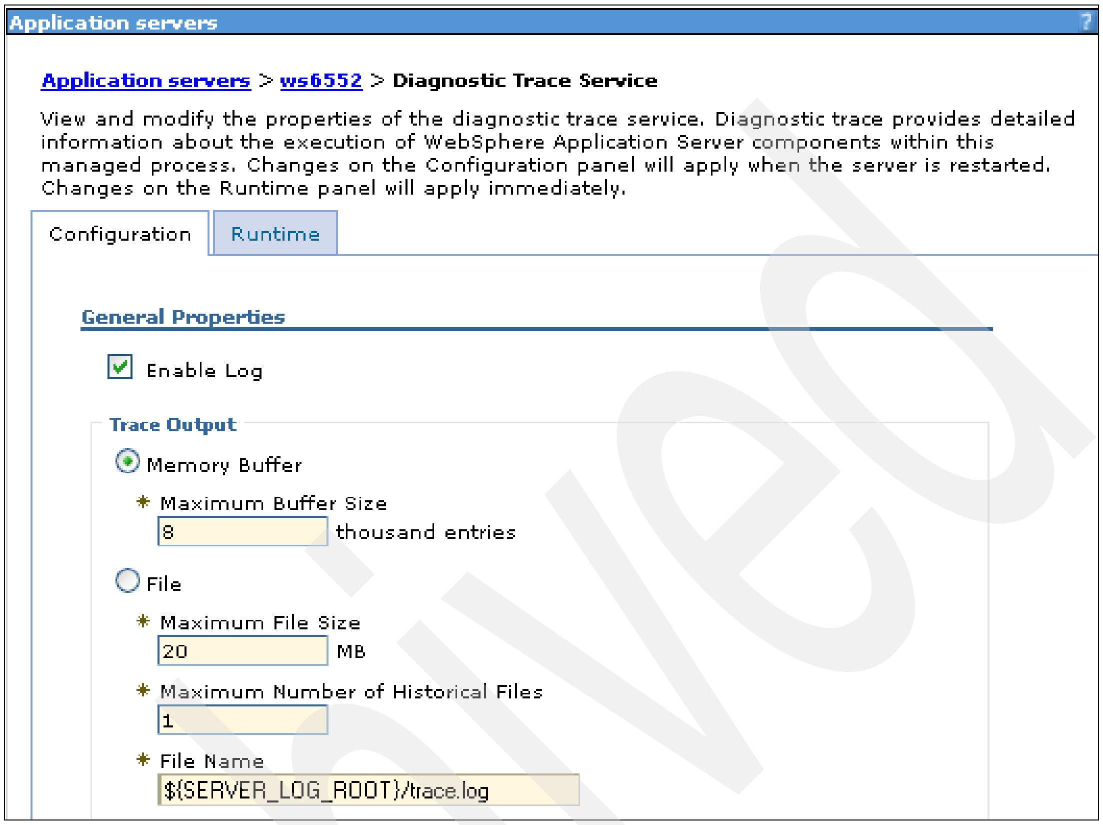
Figure 2-4 Java logging architecture

2.4.2 Setting up the Java Logging API

To collect traces and logs of WebSphere for z/OS using the Java Logging API:

1. Log in to the Administrative Console.
2. In the navigation pane, click **Servers** and **Application Servers**.
3. Click the name of the server that you want to work with.
4. Click Diagnostic **Trace Service**.
5. Select **Enable Log.**
6. Select either **Memory buffer** or **file**.
7. Specify your configurations.

In Figure 2-5 shows **Enable log in Diagnostic Trace Service** and options to enable logs and trace.


Application servers

Application servers > ws6552 > Diagnostic Trace Service

View and modify the properties of the diagnostic trace service. Diagnostic trace provides detailed information about the execution of WebSphere Application Server components within this managed process. Changes on the Configuration panel will apply when the server is restarted. Changes on the Runtime panel will apply immediately.

Configuration | Runtime

General Properties

[✓] Enable Log

Trace Output

[● Memory Buffer]

* Maximum Buffer Size: 8 thousand entries

○ File

* Maximum File Size: 20 MB

* Maximum Number of Historical Files: 1

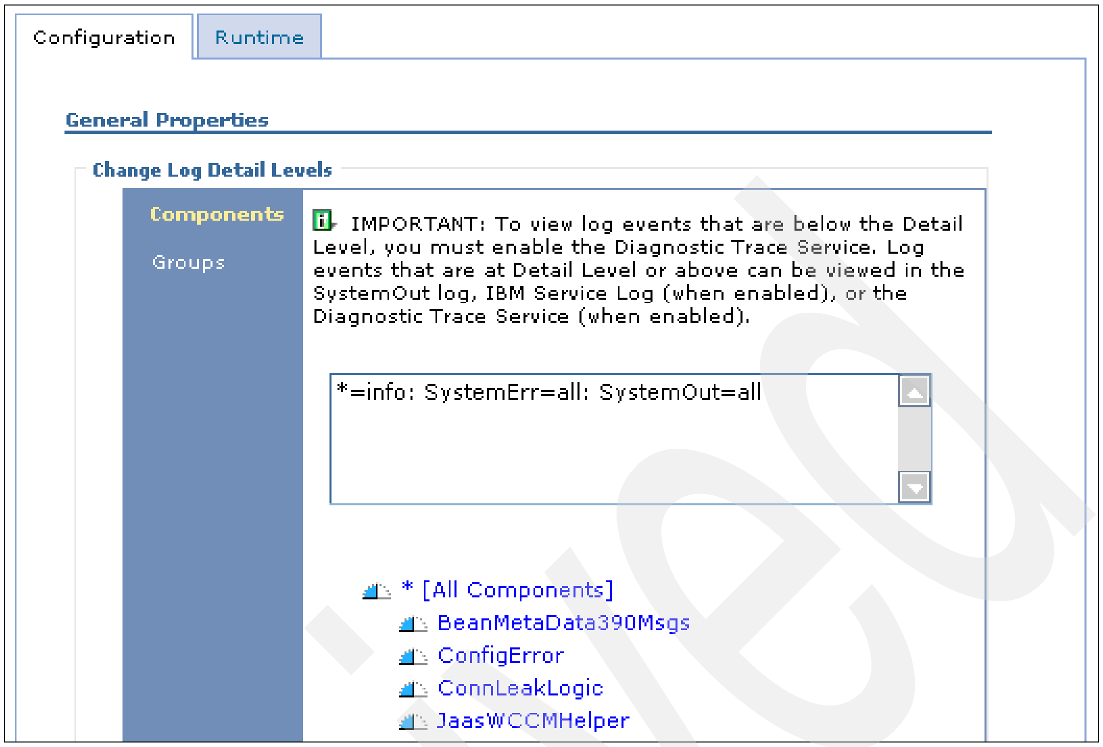
* File Name: ${SERVER_LOG_ROOT}/trace.log

Figure 2-5 Enable log in Diagnostic Trace Service

The difference between V6 and older versions of WebSphere for z/OS is that the log level string has its own panel in V6. To set traces and logs level details:

1. In the navigation pane, click **Servers** and **Application Servers**.
2. Click the name of the server that you want to work with.
3. Under Troubleshooting, select **Logging and tracing**.
4. Click **Change Log Detail levels**.
5. To make a static change to the configuration, click the Configuration tab. A list of well-known components, packages, and groups is displayed.
6. To change the configuration dynamically, click the Runtime tab.

   Figure 2-6 on page 36 shows the Administrative Console panel for changing Log Levels Details. The list of components, packages, and groups shows all the components that are currently registered to the running server.



IMPORTANT: To view log events that are below the Detail Level, you must enable the Diagnostic Trace Service. Log events that are at Detail Level or above can be viewed in the SystemOut log, IBM Service Log (when enabled), or the Diagnostic Trace Service (when enabled).

```
*=info: SystemErr=all: SystemOut=all
```

* [All Components]
- BeanMetaData390Msgs
- ConfigError
- ConnLeakLogic
- JaasWCCMHelper

Figure 2-6 WebSphere Change Log Detail Levels

7. Select a component, package, or group to set a logging level.
8. Click **Apply**.
9. Click **OK**.

The default log level is *=info. To modify it, you can type another level or set it using the graphical menu.

Table 2-2 describes the fields in the first line of the trace.

Table 2-2 Log Details Level

Level	Consequence
Off	No events are logged
Fatal	Task cannot continue and component cannot function
Severe	Task cannot continue but component can still function
Warning	Potential error or impending error
Audit	Significant event affecting server state or resources
Info	General information outlining overall task progress
Config	Configuration change or status
Detail	General information detailing subtask progress
Fine	Trace information: General trace
Finer	Trace information: Detailed trace
Finest	Trace information: A more detailed trace (includes all the details that are needed to debug problems

Level	Consequence
All	All events are logged. Inclusive custom logs

The syntax of strings that are used in the log detail level is specified by the Java Logging specification. The string is defined by the component or group that you want to trace, followed by an equal sign (=) and the level for detail. For example, to enable fine trace level for all classes in the com.ibm.ws.classloades package, use:

```
com.ibm.ws.classloader.*=fine
```

To enable detailed trace level for all components in the EJBContainer group, use:

```
EJBContainer=finest
```

Note: Tracing components have an impact on performance. A trace of all components (com.ibm.*=all) causes the system to run very slowly.

2.4.3 Java Logging output and interpretation

The logging output differs depending on the parameters that you set in the administrative console. It has a heading (first line) and a brief description (second line) in each entry of the trace. Example 2-8 shows the records of a sample entry in a trace.

Example 2-8 An entry in a trace

```
Trace: 2005/08/03 13:43:44.723 01 t=7CB4F8 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 932
  error message: BBOO0222I: TRAS0018I: The trace state has changed. The new trace state is
*=config.
```

In Table 2-3 you can see the following fields in the first line.

Table 2-3 First line in trace sample

2005/08/03 13:43:44.723	Date and hour of the entry in the trace.
01	Version ID.
t=7CB4F8	TCB.
c=UNK	Represents correlation information that consists of internal runtime information (session ID and request ID) that is used to identify trace entries related to a particular client request
key=P8	Represents which state and key the code is running in, for example, code running in a control process is running in supervisor state, key 2 (s2) and code running in a servant process is in problem state, key 8 (p8)
(0000000A)	Trace point ID that is used to locate trace in code and which follows the ccmmmttt, structure, where **cc** is the component ID from include/private/bborras.h, **mmm** identifies the module in the component in include/private/ras/bboXcrd.h, and **ttt** is the unique trace point within the source (this value is in hex; the value in the code is decimal)

In the second line, you can see a brief description of the entry recorded in the trace:

```
Description: Log Boss/390 Error
```

Table 2-2 on page 36 lists the events and description.

2.4.4 Java Logging API example

In Example 2-9, you can see the log of a change detail trace level when the Java Logging API was used.

Example 2-9 Change log detail level

```
Trace: 2005/08/03 14:58:22.257 01 t=7C9690 c=UNK key=S2 (13007002)
  ThreadId: 000001d7
  FunctionName: com.ibm.ejs.ras.ManagerAdmin
  SourceId: com.ibm.ejs.ras.ManagerAdmin
  Category: INFO
  ExtendedMessage: BBOO0222I: TRAS0018I: The trace state has changed. The new trace state
is *=config:com.ibm.ws.db2.logwriter=fine:
com.ibm.ws.database.logwriter=fine:itso.db2.cmp.j2ee.*=fine:com.ibm.ws.webcontainer.servlet
.ServletWrapper=fine:com.ibm.ws.webcontainer.srt.SRTServletRequest=fine:com.ibm.ws.webconta
iner.srt.SRTServletResponse=fine.
```

2.5 IBM HTTP Server logs and trace

IBM HTTP Server writes various kinds of logs into HFS files. Depending on the specific function, various logs are provided:

- Error log
- Access log
- Fast Response Cache Accelerator (FRCA) access log
- Proxy and cache access logs
- Agent log
- Referer log
- CGI error log

We focused on the server error log and the server access log, which are particularly useful WebSphere for z/OS problem determination.

It is also possible to activate the IBM HTTP Server trace called very verbose trace (-vv trace) to collect the activity of the server. This trace is written directly in the job log of the IBM HTTP Server started task.

The Web server plug-in in IBM HTTP Server is used to forward requests to a WebSphere for z/OS Web container. This HTTP plug-in component usually writes out logs and traces for informational or problem diagnosis use.

Figure 2-7 on page 39 shows the HTTP server logs and traces in relation to WebSphere for z/OS.

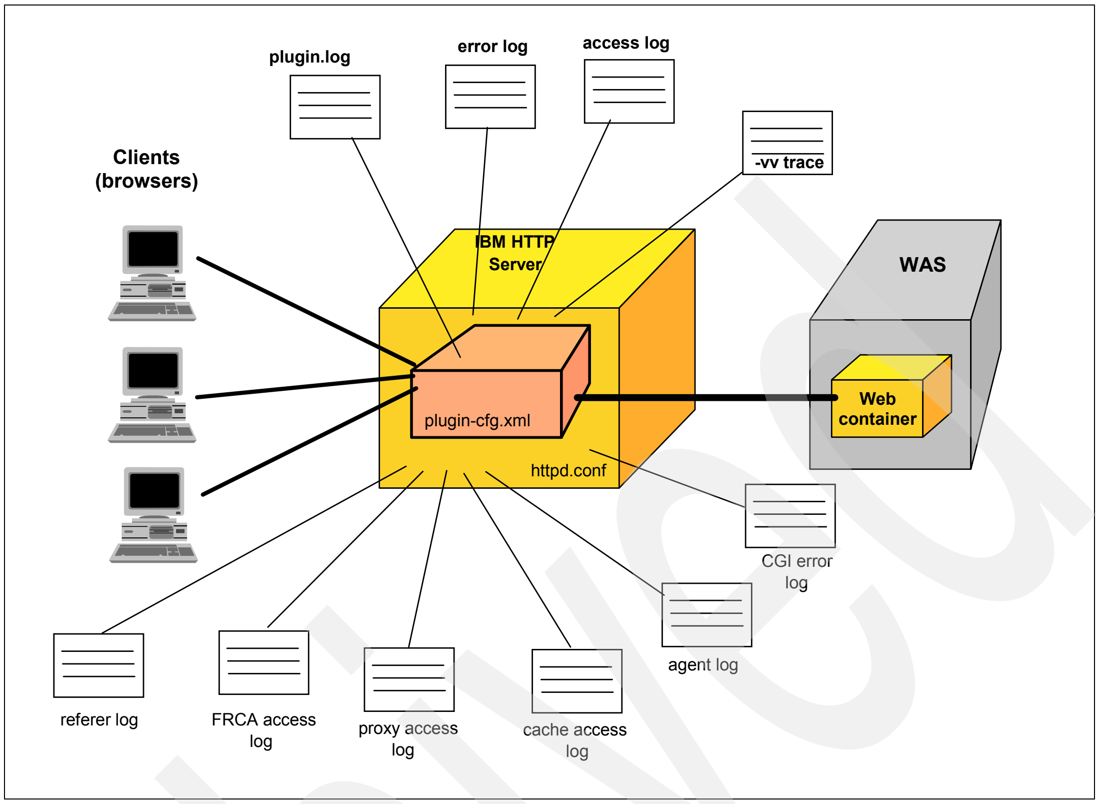


Figure 2-7 IBM HTTP Server logs and trace

2.5.1 Server error log

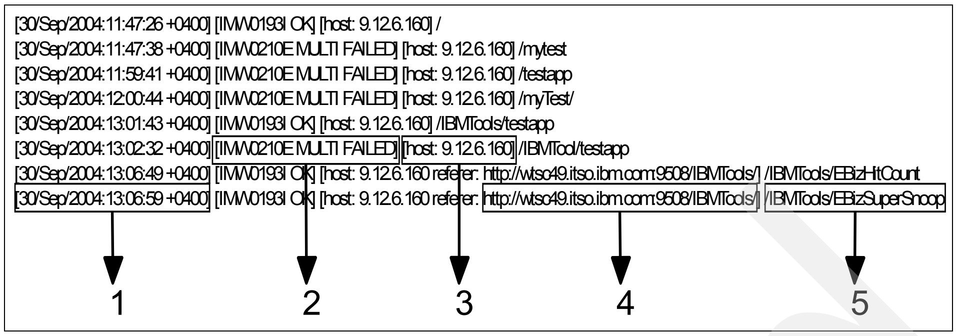
The IBM HTTP Server creates an error log that includes errors that were encountered by the clients for the server, such as timing out or not getting access. After you determine that there are problems (for example, the client and the server are not communicating), you should refer to this log because it might indicate what is wrong.

The server saves the error log in an HFS file. You can direct the path and the name of this file to where you want to log internal server errors by using a logging directive in the server configuration file. It is called httpd.conf by default. The name of the directive is ErrorLog, for example:

```
ErrorLog /web/logs/errorlog
```

When creating the file, the server uses the file name that you specify and appends a date suffix.

The server error log displays all requests that experienced problems. Figure 2-8 on page 40 illustrates a section of a server error log with each number representing a different field.



```
[30/Sep/2004:11:47:26 +0400] [IMW0193I OK] [host: 9.12.6.160] /
[30/Sep/2004:11:47:38 +0400] [IMW0210E MULTI FAILED] [host: 9.12.6.160] /mytest
[30/Sep/2004:11:59:41 +0400] [IMW0210E MULTI FAILED] [host: 9.12.6.160] /testapp
[30/Sep/2004:12:00:44 +0400] [IMW0210E MULTI FAILED] [host: 9.12.6.160] /myTest/
[30/Sep/2004:13:01:43 +0400] [IMW0193I OK] [host: 9.12.6.160] /IBMTools/testapp
[30/Sep/2004:13:02:32 +0400] [IMW0210E MULTI FAILED] [host: 9.12.6.160] /IBMTool/testapp
[30/Sep/2004:13:06:49 +0400] [IMW0193I OK] [host: 9.12.6.160 referer: http://wtsc49.itso.ibm.com:9508/IBMTools/] /IBMTools/EBizHitCount
[30/Sep/2004:13:06:59 +0400] [IMW0193I OK] [host: 9.12.6.160 referer: http://wtsc49.itso.ibm.com:9508/IBMTools/] /IBMTools/EBizSuperSnoop
```

Figure 2-8 Server error log sample

The fields (delineated by numbers in the illustration) in the server error log represent the following information:

1. Date and time when the entry of the request was recorded in the server
2. Error message (a description of this message is provided in *IBM HTTP Server Planning, Installing, and Using*, SC34-4826)
3. The IP address of the client that accessed the server
4. The URL that the client requested
5. The context root and the file requested by the client

Note: If you access the server from a PC client, the IP address in the server error log might not be the same as the IP address of your PC. This depends on the network configuration that you use (proxies, gateways, and so on).

2.5.2 Server access log

The IBM HTTP Server records activities in the access log files and stores them each day. Each day at midnight, the server closes the current access log and creates a new access log file for the next day. The access log has entries for page requests made to the server.

For each access request your server receives, an entry is made in the access log showing:

- What was requested
- When it was requested
- Who requested it
- The method of the request
- The type of file that your server sent in response to the request
- The return code, which indicates whether the request was honored

To enable the server access log, set another logging directive in the httpd.conf file called AccessLog that points to the file where you want the access log to be saved, for example:

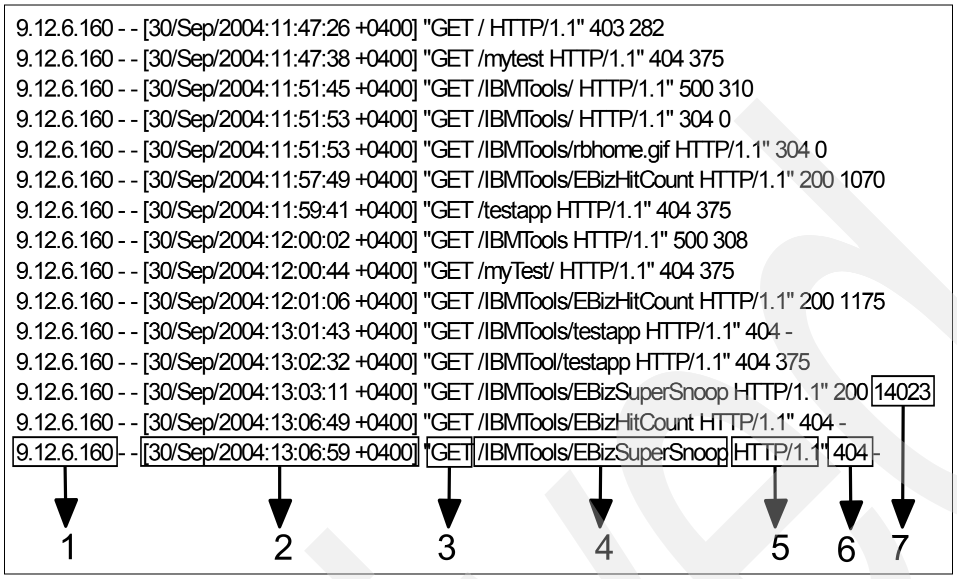
```
AccessLog /web/logs/accesslog
```

The server then creates the file in ServerRoot/web/logs/:

```
accesslog.datesuffix.file_extension
```

where ServerRoot is a path that can be configured with another directive in the httpd.conf, and datesuffix.file_extension is the date when the log was created with the Web server-generated file extension.

The server records one line per request that arrives. Figure 2-9 illustrates the fields that each line contains in an example.



```
9.12.6.160 - - [30/Sep/2004:11:47:26 +0400] "GET / HTTP/1.1" 403 282
9.12.6.160 - - [30/Sep/2004:11:47:38 +0400] "GET /mytest HTTP/1.1" 404 375
9.12.6.160 - - [30/Sep/2004:11:51:45 +0400] "GET /IBMTools/ HTTP/1.1" 500 310
9.12.6.160 - - [30/Sep/2004:11:51:53 +0400] "GET /IBMTools/ HTTP/1.1" 304 0
9.12.6.160 - - [30/Sep/2004:11:51:53 +0400] "GET /IBMTools/rbhome.gif HTTP/1.1" 304 0
9.12.6.160 - - [30/Sep/2004:11:57:49 +0400] "GET /IBMTools/EBizHitCount HTTP/1.1" 200 1070
9.12.6.160 - - [30/Sep/2004:11:59:41 +0400] "GET /testapp HTTP/1.1" 404 375
9.12.6.160 - - [30/Sep/2004:12:00:02 +0400] "GET /IBMTools HTTP/1.1" 500 308
9.12.6.160 - - [30/Sep/2004:12:00:44 +0400] "GET /myTest/ HTTP/1.1" 404 375
9.12.6.160 - - [30/Sep/2004:12:01:06 +0400] "GET /IBMTools/EBizHitCount HTTP/1.1" 200 1175
9.12.6.160 - - [30/Sep/2004:13:01:43 +0400] "GET /IBMTools/testapp HTTP/1.1" 404 -
9.12.6.160 - - [30/Sep/2004:13:02:32 +0400] "GET /IBMTool/testapp HTTP/1.1" 404 375
9.12.6.160 - - [30/Sep/2004:13:03:11 +0400] "GET /IBMTools/EBizSuperSnoop HTTP/1.1" 200 14023
9.12.6.160 - - [30/Sep/2004:13:06:49 +0400] "GET /IBMTools/EBizHitCount HTTP/1.1" 404 -
9.12.6.160 - - [30/Sep/2004:13:06:59 +0400] "GET /IBMTools/EBizSuperSnoop HTTP/1.1" 404 -
```

Figure 2-9 Server access log sample

The numbered fields represent the following information:

1. The IP address of the client that made the request
2. The date and time of the request
3. The method of the request
4. The file that the client requested
5. The protocol and version
6. The value of the HTTP return code
7. The size of the file (in bytes) being requested

For more information about the server logs see *IBM HTTP Server Planning, Installing, and Using*, SC34-4826, especially Chapter 15.

2.5.3 Very verbose trace

The server trace has several levels of debugging (verbose, much too verbose, verbose cache, and debug). The most common is the -vv (very verbose) trace.

Note: Activation of the -vv trace results in a large amount of information that is recorded in the job log. It directly impacts server performance.

There are two ways to start the -vv trace:

- Use the -vv parameter in the started procedure of the server, as shown in Example 2-10.

Example 2-10 Configuration of -vv trace in started procedure in PROCLIB

```
//IMWPROC PROC LEPARM=,ICSPARM='-vv -r /web/httpd.conf'
//*********************************************************************
//WEBSRV EXEC PGM=IMWHTTPD,REGION=OK,TIME=NOLIMIT,
// PARM=(©&LEPARM/&ICSPARM©)
//*********************************************************************
//SYSIN DD DUMMY
//OUTDSC OUTPUT DEST=HOLD
```

```
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSERR DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDOUT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDERR DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSOUT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//CEEDUMP DD SYSOUT=*,OUTPUT=(*.OUTDSC)
```

The server is in a very verbose mode when it is restarts. To stop the trace, you either change the parameter in the procedure and restart the server or issue a MODIFY command.

- Alternatively, dynamically modify the server with the following console command:

  ```
  /f imwebsrv,appl=-vv
  ```

  where imwebsrv is the name of your IBM HTTP Server.

  The following message appears in the console log:

  ```
  30Sep04 10:28:08: IMW3518I Second level tracing (-vv) enabled.
  ```

  To stop the trace, launch this command:

  ```
  /f imwebsrv,appl=-nodebug
  ```

  The following message appears in the console log:

  ```
  30Sep04 10:38:38: IMW3508I Debug has been disabled for all modules.
  ```

Important: Because of the large amount of data that the -vv trace generates and the impact on performance, we recommend that you start the trace dynamically, reproduce the error, and then stop the trace. That way, you have a short -vv trace, which makes it easier to find the section of the log that relates to the problem.

The -vv trace provides more detailed information than the server error log or the access log. For this reason, the trace is more helpful if you determine that the problem occurred inside IBM HTTP Server and you need detailed step-by-step processing information to rectify the problem. Example 2-11 displays only a portion of the trace, showing a request of the /IBMTools/EBizSuperSnoop file from a browser. It contains the following information:

- The method of the request
- The file requested (GET //IBMTools/EBizSuperSnoop)
- The protocol and version (HTTP/1.1)
- The browser and the operative system of the client (Mozilla 4.0; compatible Microsoft Internet Explorer 6.0; Microsoft Windows NT® 5.1)
- The IP address and port of the host (wtsc49.itso.ibm.com:9508)

Example 2-11 Very verbose trace sample

```
[21646C48 30/Sep/2004:14:06:12.854728]: 30Sep04 14:06:12: IMW3518I Second level tracing (-vv) enabled.
[21660778 30/Sep/2004:14:06:22.556968]: Read 460 bytes from socket 10.
[21660778 30/Sep/2004:14:06:22.557027]: After AcceptEx nAcceptThds: 75 and nSSLAcceptThds: 0.
[21660778 30/Sep/2004:14:06:22.557046]: server_loop... Accepted socket: 10.
[21660778 30/Sep/2004:14:06:22.557131]: KEEPALIVE... set.
[21660778 30/Sep/2004:14:06:22.557156]: HTSession... starting for socket=10; STHD=21932DD8
[21660778 30/Sep/2004:14:06:22.557187]: Keep-Alive.. Starting HTTPD 1.1 loop.
[21660778 30/Sep/2004:14:06:22.557218]: HTTimer... setting timer off->set (1) on socket 10.
[21660778 30/Sep/2004:14:06:22.557236]: HTTimer... set, old=0, cur=0, new=1
[21660778 30/Sep/2004:14:06:22.557314]: Client sez.. GET /IBMTools/EBizSuperSnoop HTTP/1.1
[21660778 30/Sep/2004:14:06:22.557340]: Protocol version.... 1.1
[21660778 30/Sep/2004:14:06:22.557355]: Persistent Connection has been established
[21660778 30/Sep/2004:14:06:22.557378]: Client sez.. Accept: */*
```

```
[21660778 30/Sep/2004:14:06:22.557395]: Accept...... */* (q=1.00,mxb=0.0,mxs=0.0)
[21660778 30/Sep/2004:14:06:22.557437]: Client sez.. Referer: http://wtsc49.itso.ibm.com:9508/IBMTools/
[21660778 30/Sep/2004:14:06:22.557456]: Referer..... http://wtsc49.itso.ibm.com:9508/IBMTools/
[21660778 30/Sep/2004:14:06:22.557476]: Client sez.. Accept-Language: en-us
[21660778 30/Sep/2004:14:06:22.557492]: Language.... en-us (q=1.00)
[21660778 30/Sep/2004:14:06:22.557517]: Client sez.. Accept-Encoding: gzip, deflate
[21660778 30/Sep/2004:14:06:22.557533]: Encoding.... gzip (q=1.00)
[21660778 30/Sep/2004:14:06:22.557552]: Encoding.... deflate (q=1.00)
[21660778 30/Sep/2004:14:06:22.557580]: Client sez.. If-Modified-Since: Thu, 30 Sep 2004 17:03:11 GMT
[21660778 30/Sep/2004:14:06:22.557600]: Format...... Wkd, 00 Mon 0000 00:00:00 GMT
[21660778 30/Sep/2004:14:06:22.557621]: TimeZone.... 04 hours from GMT
[21660778 30/Sep/2004:14:06:22.557638]: Time string. Thu, 30 Sep 2004 17:03:11 GMT; offset = 0 seconds
[21660778 30/Sep/2004:14:06:22.557662]: Parsed...... to 1096563791 seconds, Thu Sep 30 13:03:11 2004
[21660778 30/Sep/2004:14:06:22.557687]: Give only... if modified since (localtime) Thu Sep 30 13:03:11 2004
[21660778 30/Sep/2004:14:06:22.557722]: Client sez.. User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows
NT 5.1; .NET CLR 1.1.4322)
[21660778 30/Sep/2004:14:06:22.557746]: User-Agent.. Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;
.NET CLR 1.1.4322)
[21660778 30/Sep/2004:14:06:22.557771]: Client sez.. Host: wtsc49.itso.ibm.com:9508
[21660778 30/Sep/2004:14:06:22.557789]: Host........ wtsc49.itso.ibm.com
[21660778 30/Sep/2004:14:06:22.557804]: Host Port... 9508
```

2.5.4 HTTP plug-in log

The HTTP plug-in log usually has error messages regarding forward requests to WebSphere for z/OS and configuration errors in the plugin-cfg.xml file. In addition, in the plugin-cfg.xml file, you can turn on tracing to provide more details about the routing of a request, such as:

- URL and Uniform Resource Indicator (URI) matching
- Virtual host matching
- Request header and session affinity cookie

These logs can help you determine where the problem exists during an initial search. It is useful to follow the request from a browser on the client side to WebSphere for z/OS, and also in the other direction, from WebSphere for z/OS to the client. By looking at the logs, you can determine if the requests from the client arrive correctly at IBM HTTP Server, and if they are mapped correctly according to the httpd.conf directives and plugin-cfg.xml routing table.

However, the -vv trace provides a level of messaging that can provide information about the problem if it is occurring in the server. If it is not occurring inside the server, the -vv trace can provide detailed information about what is “traveling” through it.

For more information about the server logs and the server trace, see *IBM HTTP Server Planning, Installing, and Using*, SC34-4826.

For more information about the HTTP plug-in log, see *WebSphere Application Server for z/OS V5.1: Servers and Environment*, GA22-7958.

To enable logging for the WebSphere z/OS plug-in, specify the location of the plugin-cfg.xml file in the httpd.conf file:

```
ServerInit /<...>/ihs390WAS50Plugin_http.so:init_exit /<...>/plugin-cfg.xml
```

If your plug-in initialized successfully, the following messages appear in SYSOUT, indicating which plugin-cfg.xml file is used by IBM HTTP Server:

```
WebSphere HTTP Plug-in for z/OS and OS/390  initializing with configuration file :
/web/andy1/plugin-cfg.xml
WebSphere HTTP Plug-in for z/OS and OS/390  initialization went OK :-)
```

Then, in the plugin-cfg.xml file, specify the LogLevel and Name of the plug-in log file (plugin.log) where all logging output should go, as shown in the following example:

```
<Log LogLevel="Error" Name="/<...>/plugin.log"/>
```

Plug-in logging allows logging at many level of detail to suit various situations. You can specify one of the followings levels:

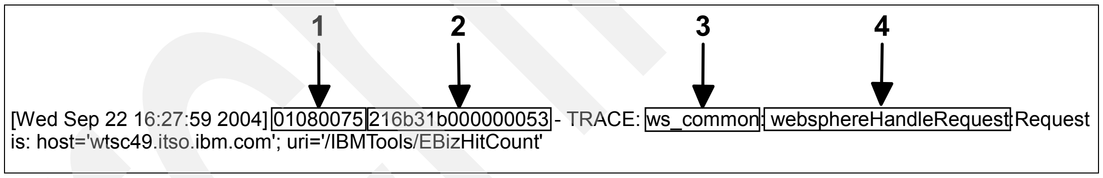
Trace	All of the steps in the request process are logged in detail.
Stats	The server selected for each request and other load balancing information that is related to request handling is logged.
Warn	All warning and error messages that result from abnormal request processing are logged.
Error	Only error messages that result from abnormal request processing are logged.

The default level of logging is Error.

> **Note:** Specifying `LogLevel="Trace"` generates a large amount of data that might impact performance. The authors recommend that you specify `LogLevel="Error"`.

The server records one line per request that arrives. The example in Figure 2-10 illustrates the fields in each record line:

1. Process ID
2. PThread ID
3. IBM Software source code file name
4. Function name



```
[Wed Sep 22 16:27:59 2004] 01080075 216b31b000000053 - TRACE: ws_common: websphereHandleRequest: Request
is: host='wtsc49.itso.ibm.com'; uri='/IBMTools/EBizHitCount'
```

Figure 2-10 A plug-in trace record and some of the important fields

Figure 2-11 shows the plug-in trace records that resulted after the following request to find matches for the virtual host group (VhostGroup) and URI group (UriGroup) was made:

`http://wtsc49.itso.ibm.com:9508/IBMTools/EBizHitCount`

```
TRACE: ws_common: websphereVhostMatch: Comparing 'wtsc49.itso.ibm.com:9508' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_hos
TRACE: ws_common: websphereVhostMatch: Found a match 'wtsc49.itso.ibm.com:9508' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default
TRACE: ws_common: websphereVhostMatch: Comparing '*:9559' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_host
TRACE: ws_common: websphereVhostMatch: Comparing '*:9558' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_host
TRACE: ws_common: websphereVhostMatch: Comparing '*:9549' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_host
TRACE: ws_common: websphereVhostMatch: Comparing '*:9548' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_host
TRACE: ws_common: websphereVhostMatch: Comparing '*:80' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_host
TRACE: ws_common: websphereVhostMatch: Comparing 'wtsc49.itso.ibm.com:9519' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_hos
TRACE: ws_common: websphereVhostMatch: Comparing 'wtsc49.itso.ibm.com:9518' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_hos
TRACE: ws_common: websphereVhostMatch: Comparing '*:9519' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_host
TRACE: ws_common: websphereVhostMatch: Comparing '*:9518' to 'wtsc49.itso.ibm.com:9508' in VhostGroup: default_host
TRACE: ws_common: websphereUriMatch: Comparing '/admin' to '/IBMTools/EBizHitCount' in UriGroup: default_host_dmgr_pddmnode_Cluster_
TRACE: ws_common: websphereUriMatch: Comparing '/admin/*' to '/IBMTools/EBizHitCount' in UriGroup: default_host_dmgr_pddmnode_Cluste
TRACE: ws_common: websphereUriMatch: Comparing '/adminservlet' to '/IBMTools/EBizHitCount' in UriGroup: default_host_dmgr_pddmnode_C
TRACE: ws_common: websphereUriMatch: Comparing '/FileTransfer' to '/IBMTools/EBizHitCount' in UriGroup: default_host_dmgr_pddmnode_C
TRACE: ws_common: websphereUriMatch: Comparing '/adminservlet/*' to '/IBMTools/EBizHitCount' in UriGroup: default_host_dmgr_pddmnode
TRACE: ws_common: websphereUriMatch: Comparing '/FileTransfer/*' to '/IBMTools/EBizHitCount' in UriGroup: default_host_dmgr_pddmnode
TRACE: ws_common: websphereUriMatch: Failed to match: /IBMTools/EBizHitCount
```

Figure 2-11 Plug-in traces

The plug-in first attempts to find a match for the wtsc49.itso.ibm.com virtual host and port 9508 in the defined virtual host group. Then, it compares the /IBMTools/EBizHitCount URI with the defined URI group entries. The last line shows that there was no matching URI definition.

3

WebSphere for z/OS traces and dumps

In this chapter, we explain the following:

- CTRACE for WebSphere
- Java Database Connectivity (JDBC) trace
- SVCDump
- CEEDUMP
- Java Transaction Dump
- Javadump
- Heapdump

For each trace or dump we provide information about its nature, about when to use it, and about how to use it. We describe the output, show how to interpret it, and provide an example.

3.1 CTRACE for WebSphere

WebSphere for z/OS uses z/OS CTRACE facilities to manage the collection and storage of trace data. Unless you configure specific CTRACE controls, WebSphere for z/OS records its trace data in address space buffers in private (pageable) storage. This data is not accessible unless a dump of the address space is taken.

CTRACE data is primarily output for the IBM service team to use, but it is possible that you might be asked to provide IBM with CTRACE output. Therefore, it is important for you to know how to use CTRACE in your environment to obtain additional trace data that is available when a problem first occurs.

The CTRACE efficiently uses system resources so that you can collect valuable trace data with a minimal impact on performance. WebSphere for z/OS identifies itself to CTRACE with the component name that is determined by the short cell name. With CTRACE, you can:

- Use one or more data sets for capturing trace data so that you can manage I/O more effectively.
- Merge multiple traces, including other components such as TCP/IP and z/OS USS, using the interactive problem control system (IPCS).
- Write trace data to a data set rather than SYSPRINT, keeping spool space use to a minimum.
- Allow trace data to wrap, providing for better management of system resources.
- Funnel trace data from multiple address spaces to one data set, or have CTRACE send the trace data from each address space to separate data sets or the same one.
- Start and stop tracing without stopping and restarting WebSphere for z/OS address spaces.

3.1.1 Setting up and taking a CTRACE

The procedure of setting up and taking a CTRACE is available in *WebSphere Application Server for z/OS V6, Troubleshooting and support*, GA22-7964-03

More information is available if you search for CTRACE at the *WebSphere for z/OS Information Center*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

3.1.2 Using the IPCS dialog to format CTRACE data

Perform the following steps to use the IPCS dialog to format application trace data for error analysis:

1. From the IPCS Primary Option Menu panel, select option `6` (COMMAND).
2. On the IPCS Sub-command Entry panel:
   a. Issue the SETDEF sub command to determine the default values for routing displays.
   b. Enter the CTRACE command, with the following required parameters:

      ```
      CTRACE COMP(cell_short_name)
      ```

      where cell_short_name is the value specified through the ISPF Customization Dialog to identify the location of server configuration files (eight or fewer characters and all uppercase).

If you are interested in only JRas data, enter the following command and specify additional parameters as necessary:

```
CTRACE COMP(cell_short_name )USEREXIT(JRAS)
```

For more details about CTRACE, see *z/OS MVS IPCS Commands*, SA22-7594.

3. View your application CTRACE data, based on the options that you chose for the location of the data.

To navigate through the trace data on the Dump Display Reporter panel, use the commands and PF keys listed in *z/OS MVS IPCS User's Guide*, SA22-7596.

3.1.3 CTRACE output and its interpretation

CTRACE contains important information that helps IBM provide a high level of support. IBM might ask you to provide a CTRACE when the IBM Support Center requires a deeper level of information to analyze your problem.

Example 3-1 shows WebSphere for z/OS CTRACE output.

Example 3-1 WebSphere for z/OS CTRACE output

```
SY1       OBOAT008  04000002  00:14:57.268258  Dispatch Method
      ASID.... 0039
      TCB..... 009E34A0 PSW1.... 078D2400 SESS.... 00000008 REQI.... 0000006C
      Class Name         = JPolicyEmSQLMO
      Method Name        = _get_policyNo
      object             = 0x260ED1F8
      objectPtr refcount = 3           0x00000003
      objectPtr classname= JPolicyEmSQLMO

An entry contains an undefined ID: 13007002  , hex format will be used.
 SY1       N/A       13007002  00:14:57.272682  N/A
      0002009E  34A0078D  24000039  00000008  | ................ |
      0000006C  000C0302  97969389  83A8D596  | ...%....policyNo |
      00120402  6DD1D796  938983A8  C2D6C994  | ...._JPolicyBOIm |
      97930009  0A02C1E4  C4C9E300  2E0B02C2  | pl....AUDIT....B |
      C2D6D1F0  F0F0F240  D7969389  83A84095  | BOJ0002 Policy n |
      A4948285  9940F3F3  6BF3F3F3  409682A3  | umber 33,333 obt |
      81899585  844B4040  40                  | ained.           |
Trace: 2004/10/12 00:14:57.268 01 t=9E34A0 c=8.6C key=P8 (04000002)
  Description: Dispatch Method
  Class Name: JPolicyEmSQLMO
  Method Name: _get_policyNo
  object: 260ED1F8
  objectPtr refcount: 3
  objectPtr classname: JPolicyEmSQLMO
Trace: 2004/10/12 00:14:57.272 01 t=9E34A0 c=8.6C key=P8 (13007002)
  FunctionName: policyNo
  SourceId: _JPolicyBOImpl
  Category: AUDIT
  ExtendedMessage: BBOJ0002 Policy number 33,333 obtained.
```

3.1.4 Viewing CTRACE and JRas data through IPCS

Once activated, WebSphere for z/OS always writes trace data into memory buffers. The number and size of these buffers is controlled with WebSphere variables. You can get this trace data from a dump, which might be taken by the system or requested by the operator with DUMP or SLIP commands. To view messages or application trace data from component trace, you must use the IPCS to format the data.

For information about viewing CTRACE and JRas data through IPCS, refer to the *WebSphere Application Server for z/OS V6, Troubleshooting and support*, GA22-7964-03.

Also, you can visit the *WebSphere for z/OS Information Center* at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp`

3.2 JDBC trace

Tracing the JDBC is useful when you are experiencing problems with applications that use a WebSphere for z/OS data source to connect to databases that support traces.

There are two traces that you can use. The first is based on Logging Java for WebSphere for z/OS and is a JVM trace. The other is specifically for DB2®.

3.2.1 Setting up the JDBC trace

You use the WebSphere for z/OS administrative console to turn on the Java Virtual Machine (JVM) trace.

Turn on JDBC tracing by using the trace strings that are shown in Table 3-1. For information about how to enable trace strings to look for error messages, see 2.4, “Java Logging API” on page 33.

Table 3-1 JDBC trace strings

com.ibm.ws.database.logwriter	Trace string for databases that use the GenericDataStoreHelper that can also be used for unsupported databases.
com.ibm.ws.db2.logwriter	Trace string for DB2 databases
com.ibm.ws.oracle.logwriter	Trace string for Oracle databases
com.ibm.ws.cloudscape.logwriter	Trace string for Cloudscape™ databases
com.ibm.ws.informix.logwriter	Trace string for Informix® databases
com.ibm.ws.sqlserver.logwriter	Trace string for Microsoft SQL Server databases
com.ibm.ws.sybase.logwriter	Trace string for Sybase databases

If this trace cannot help you and you are connecting to DB2 databases, the DB2 SQLJ/JDBC trace might. The following steps describe the procedure to obtain this JDBC DB2 trace:

1. Log in to the Administrative Console and expand the Environment item in the menu and select **WebSphere Variables.** Select the Scope for which you want to enable the trace and click **Apply**.
2. Click **New** and add this variable name and its value:

   DB2SQLJPROPERTIES=/mydb2dir/wsccb_db2sqljjdbc.properties

3. In this properties file, called /mydb2dir/wsccb_db2sqljjdbc.properties, we set up the variable DB2SQLJ_TRACE_FILENAME to enable the SQLJ/JDBC trace and specify the name of the file to which the trace is written:

   DB2SQLJ_TRACE_FILENAME=/tmp/IVP2_jdbctrace

4. The JDBC trace produces two HFS files:
   - /tmp/IVP2_jdbctrace

     This file is in binary format. You must format it using the db2sqljtrace command (as shown in the following step).
   - /tmp/IVP2_jdbctrace.JTRACE

     The second file contains readable text.

5. To format the binary trace data, we use the following db2sqljtrace command in the USS environment:

   ```
   db2sqljtrace fmt|flw TRACE_FILENAME > OUTPUT_FILENAME
   ```

   where **fmt|flw** are sub commands that ensure that the output race contains:

fmt	a record every time a function is entered or exited before a failure
flw	the function flow before a failure

   and OUTPUT_FILENAME is the name of the file that the new formatted trace is written to.

   To run **db2sqljtrace** correctly, ensure that the JDBC path and libraries are the PATH and LIBPATH environment variables.

   You can change them with the following commands:

   ```
   export PATH=$PATH:/usr/lpp/db2/db2810/bin
   export LIBPATH=$LIBPATH:/usr/lpp/db2/db2810/lib
   ```

   Note: The IBM default path is /usr/lpp/db2/db2810/, but you might have another path, depending on your installation.

   You can verify that they are correct with the following commands:

   ```
   echo $PATH
   echo $LIBPATH
   ```

For more information, refer to DB2 documentation.

3.2.2 Output and interpretation

JDBC trace information is helpful, especially for IBM support. It shows Java methods, database names, plan names, user names, or connection pools.

Example 3-2 shows the formatted JDBC trace; in our case, the file name is /tmp/IVP2_jdbctrace.JTRACE.

Example 3-2 Formatted JDBC trace sample

```
Timestamp> <Trace Point> <Method Name> <Class/ObjectId> <Thread Name> <Optional Parms>
 <2004.10.04 20:15:02.810> <Entry>   <printHeader> <COM.ibm.db2os390.sqlj.util.DB2SQLJTrace>
<P=253767:O=0:CT>
    -- <p#1=Start of DB2 SQLJ/JDBC Tracing  <2004.10.04 20:15:02.810>>
    -- <p#2=DB2 for OS/390 SQLJ/JDBC Driver build version is: DB2 7.1 UQ85384 JDBC 2.0>
 <2004.10.04 20:15:02.949> <Entry>   <Constructor> <COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection@5254a29a>
<P=253767:O=0:CT>
```

```
    -- <p#1=source=DSN7>
    -- <p#2=parser=COM.ibm.db2os390.sqlj.jdbc.parser.DB2JDBCParser@5c10229d>
    -- <p#3=planname=DSNJDBC>
    -- <p#4=pooledConnection=com.ibm.db2.jcc.DB2PooledConnection@6c54e29a>
 <2004.10.04 20:15:02.9
<92> <Entry>   <setTransactionIsolation> <COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection@5254a29a>
<P=253767:O=0:CT>
    -- <p#1=Current Isolation=2>
    -- <p#2=New Isolation=2>
    -- <p#3=COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection@5254a29a[pCONN=3767fec8]> <2004.10.04
20:15:03.123> <Exit>    <setTransactionIsolation> <COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection@5254a29a>
<P=253767:O=0:CT>
    -- <p#1=COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection@5254a29a[pCONN=3767fec8]>
 <2004.10.04 20:15:03.139> <Exit>    <Constructor> <COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection@5254a29a>
<P=253767:O=0:CT>
    -- <p#1=COM.ibm.db2os390.sqlj.jdbc.DB2SQLJConnection@5254a29a[pCONN=3767fec8]>
 <2004.10.04 20:15:03.157> <Entry>   <getConnection> <com.ibm.db2.jcc.DB2PooledConnection@6c54e29a>
<P=253767:O=0:CT>
    -- <p#1=com.ibm.db2.jcc.DB2PooledConnection@6c54e29a>
 <2004.10.04 20:15:03.164> <Entry>   <constructor> <com.ibm.db2.jcc.DB2LogicalConnection>
<P=253767:O=0:CT>
 <2004.10.04 20:15:03.164> <Exit>    <constructor> <com.ibm.db2.jcc.DB2LogicalConnection>
<P=253767:O=0:CT>
    -- <p#1=com.ibm.db2.jcc.DB2LogicalConnection@4ba6629c[mClosed=false;mConnection=5254a29a]>
```

Example 3-3 shows a JDBC™ trace formatted with the fmt sub command.

Example 3-3   JDBC trace formatted with fmt sub command

```
Trace Version            : DB2 7.1
Driver Build Version     : DB2 7.1 UQ85384 JDBC 2.0
Trace Captured at        : Mon Oct  4 20:15:02 2004
Trace buffer size        : 262144 bytes
Records to keep          : LAST
Trace truncated          : NO
Trace wrapped            : NO
Shared Memory Address    : 0x1E5CA568
First empty slot         : 7604
Trace Table Address      : 0x1E681030
Size of trace            : 7592 bytes
Records in trace         : 134
1       SQLJ fnc_entry    sqlj_JDBC_Driver DB2SQLJ_sqlj_driver_native_init (2.1.7.1)
        pid 0x007fb620; tid 0x007fb620; time 1096935302; tpoint 0
        0000 0000                          ....
2       SQLJ fnc_entry    sqlj_JDBC_AttachMgr sqlj_Attach_Global_Init (2.1.14.1)
        pid 0x007fb620; tid 0x007fb620; time 1096935302; tpoint 0
        0000 0000                          ....
3       SQLJ fnc_data     sqlj_JDBC_AttachMgr sqlj_Attach_Global_Init (2.3.14.1)
        pid 0x007fb620; tid 0x007fb620; time 1096935302; tpoint 1
        0000 0001 0000 0004 37ac 75d0      ...........}
4       SQLJ fnc_entry    sqlj_Native_Util sqlj_memAlloc (2.1.3.1)
        pid 0x007fb620; tid 0x007fb620; time 1096935302; tpoint 0
        0000 0000                          ....
```

Example 3-4 shows a JDBC trace formatted with the flw sub command.

Example 3-4 JDBC trace formatted with flw sub command

```
Trace Version           : DB2 7.1
Driver Build Version    : DB2 7.1 PQ56655
Trace Captured at       : Wed Sep 18 17:12:00 2002
Trace buffer size       : 262144 bytes
Records to keep         : LAST
Trace truncated         : NO
Trace wrapped           : NO
Shared Memory Address   : 0x236D3568
First empty slot        : 184452
Trace Table Address     : 0x2377C030
Size of trace           : 184440 bytes
Records in trace        : 2298
pid = 0x007f9358;
1  DB2SQLJ_sqlj_driver_native_init fnc_entry   ...
2   |sqlj_Attach_Global_Init fnc_entry   ...
3   |sqlj_Attach_Global_Init fnc_data    ...
4   | |sqlj_memAlloc fnc_entry   ...
5   | |sqlj_memAlloc fnc_data    ...
6   | |sqlj_memAlloc fnc_retcode 0
```

3.3 SVC dumps

An SVC dump is a dump that is initiated by the z/OS operating system generally when a programming exception occurs. SVC dump processing stores data in dump data sets that you preallocate or that the system allocates automatically as needed.

An SVC dump is an unformatted dump and is not readable without the use of a formatting tool. The z/OS dump formatting tool is the IPCS.

3.3.1 How to obtain an SVC dump

There are three ways to obtain and SVC dump:

- It is initiated by the system because an exception or problem occurred
- You can initiate an SVC dump from the MVS console to gather diagnostic data using the MVS dump command. SVC dumps initiated this way are called console dumps.
- Set SLIP to trigger an SVCDUMP when a particular condition is met.

Console dump

A console dump is an SVC dump. It is referred to as a console dump because of the way that it is triggered. It is taken using the MVS DUMP command run from the console or sdsf log.

You initiate a console dump when:

- You want an SVC dump of a servant region or a dump of the servant controller region.
- You suspect a particular servant region to be the source of a problem. Dump the controller region and all of its servant regions.
- You detect a hang or high CPU utilization for a particular address space.

A sample PARMLIB member that determines the information to be included in a console dump can be found in SBBOSLIB(BBODMCCB). The sample contains instructions about its installation and use.

The standard SDATA expected in a SVC dump is:

```
SDATA=(ALLNUC,CSA,GRSQ,LPA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT)
```

Slip dump

A slip dump is an SVC dump, but it is called a slip dump because of the way it is triggered by the MVS SLIP command. You can use a slip dump when there is an error and no SVC dump is being produced because the SVC dump is probably being suppressed by the Dump Analysis and Elimination (DAE) facility. You can also use a slip dump when you want a dump to be triggered when a certain error message occurs or you have been asked for one by IBM support.

An example of slip dump that we used to capture an EC3/04130007 is shown in Example 3-5.

Example 3-5 Slip dump for capturing EC3/04130007

```
SLIP SET,A=SVCD,COMP=EC3,REASON=04130007,ID=WEC3,
MATCHLIM=20,ASIDLST=(0,H,I,P,S),
SDATA=(ALLNUC,CSA,GRSQ,LPA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT),END
```

3.3.2 Problems obtaining an SVC dump

If you cannot find an SVC dump for a specific abnormal end (abend), your installation might be using Dump Analysis and Elimination (DAE) to suppress the dump. If this is the case, you can change DAE to let the dump be taken, or you set a SLIP on the specific abend for a particular job name if the timeout is happening consistently.

If SYSLOG has a message indicating that the maximum space limit was reached for this dump, the SVC dump might be a partial one. The partial SVC dump might no contain the data that you need to diagnose the timeout. This limit means that the data set that is used for the SVC dump is not large enough, and you must change the size to capture a complete dump. For example:

```
IEA043I SVC DUMP REACHED MAXSPACE LIMIT - MAXSPACE=00000500MEG
```

To change the size of dump storage, issue:

```
CHNGDUMP SET,SDUMP,MAXSPACE=nnnnnM
```

3.3.3 Formatting an SVC dump using IPCS

IPCS is used to format an SVC dump. The formatting commands that you use depend on the problem being investigated. We have listed some IPCS commands that we have found to be especially useful for formatting an SVC dump in Table 3-2.

Table 3-2 Useful IPCS commands for formatting an SVC dump

IPCS commands	Explanation
ip st regs faildata worksheet	This command formats the dump title, date, and time and shows the default ASID) for the dump, registers, and psw information.
ip systrace time(local)	This command formats the System Trace entries for the TCBs in the ASID.

IPCS commands	Explanation
ip select list all	This generates a list off all the address spaces in the system at the time of the dump with ASID and JOBNAME.
ip summary format	This command formats all the address space and task-related control blocks for a particular ASID. Some of the control blocks formatted are the ASCB, TCBs, RBs, and RTM2WA.
ip verbx ledata ‘nthreads(*)’	This command formats all the C-stacks (DSAs) for threads in the process for the default ASID.
ip verbx ledata ‘nthreads(*) asid(aaaa)’	This command formats all the C-stacks (DSAs) for threads in the process for the ASID specified.
ip verbx ledata ‘tcb(tttttttt) nthreads(*)’	This command formats all the C-stacks (DSAs) for a particular TCB and includes traceback information.
ip omvsdata process detail asid(x’hhhh’)	This command generates a report for the process that show the thread status from a USS perspective.
ip analyze resource	This command generates a report showing resource contention. This is useful in a hang situation.
ip verbx vsmdata ‘summary’	This command generates a report that shows the virtual storage usage for the system. This is useful when the system is experiencing storage problems.
ip verbx logdata	This command formats the available erep detail reports.
ip verbx mtrace	This command formats the available master trace which holds syslog information.

3.3.4 When to use an SVC dump

SVC dumps are useful in diagnosing many WebSphere for z/OS problems, including:

- An abend issued by the application server
- An abend in the operating system or subsystem component
- An application server timeout
- An application server hang
- An application server that is using high CPU

3.3.5 Related references

For additional information about DAE, see *z/OS V1R6.0 MVS Diagnosis: Tools and Service Aids*, GA22-7589.

Refer to *z/OS V1R6.0 MVS System Commands*, SA22-7627, for details about the SLIP and DUMP commands.

For information about how to use IPCS refer to *z/OS V1R6.0 MVS IPCS Commands SA22-7594-05 and z/OS V1R2.0 MVS IPCS User's Guide,* SA22-7596-01.

3.4 CEEDUMP

Generally, a CEEDUMP is generated if any region fails or there is an abend, for example, an error occurring in the z/OS Language Environment® or Java Runtime Environment (JRE). Typically, CEEDUMP can be found in the job logs of the different servers.

CEEDUMP can help you identify the failing module in the *Traceback* section of the dump. Search for Traceback at the top of the CEEDUMP. The result is a sequence of modules as shown in Figure 3-1. The last modules in action are at the top, and underneath then are the oldest, in order. Look for the term *Exception* in the Status column. Usually, an exception is in one of the last modules in action, so it is likely to be near the top of the Status column.

```
CEE3DMP V1 R3.0: Condition processing resulted in the unhandled condition.       09/18/02 5:19:06 PM
Page:    1

Information for enclave main

  Information for thread 23B00F1000000000

Traceback:

DSA Addr  Program Unit  PU Addr   PU Offset  Entry         E Addr    E  Offset   Statement  Load Mod  Service  Status
236D4768                06CB6B48  +00000806  __zerros      06CB6B48  +00000806              CEEEV003           Call
236D3C08  CEEHDSP       06E7C2B0  +00002BE6  CEEHDSP       06E7C2B0  +00002BE6              CEEPLPKA           Call
                            /src/share/java/runtime/jni.c
236D37B8                26091830  +00000528  JNI_CreateJavaVM 26091830  +00000528       4432  *PATHNAM Exception
236D32C0                1C2FBEE0  +00001270  loadAndInitVM(JavaVM_**,JNIEnv_**,SOMException*)
                                                           1C2FBEE0  +00001270        411  BBOLRT    CB30038 Call
236D3210                1C301E88  +000002BE  getJavaEnv(SOMException*)
                                                           1C301E88  +000002BE       1679  BBOLRT    CB30036  Call
236D30F8                1C302860  +00000092  buildJavaClass(const char*,SOMException*)
                                                           1C302860  +00000092       1921  BBOLRT    CB30036  Call
236D3030                1C30A5F0  +000001A4  __cdecl _NewObject(SOMClassRef*,SOMException*)
```

Figure 3-1   CEEDUMP sample

The name of the entry with the exception (JNI_CreateJavaVM in the Entry column) is the most important string in this CEEDUMP, because it is the search argument that you use for researching known problems and their solutions in APARs, PMRs, or on your favorite problem search site on the Internet.

See *Problem Determination Methodology for WebSphere on z/OS*, REDP6001, for information about IBM resources and using the exception entry to search problem databases.

A CEEDUMP is a formatted dump and therefore IPCS is not required to read it. Depending on the problem a CEEDUMP may not have enough information formatted and so IBM support may require an SVC dump. For CEEDUMP parameter settings, see *z/OS V1R6.0 Language Environment Debugging Guide*, GA22-7560

If you have an SVC dump, it is possible to view CEEDUMP contents in a SVC dump using the IPCS verbexit LEDATA with the CEEDUMP or NTHREADS options. This formats the Language Environment control blocks to help in analysis. For additional information, see the *z/OS V1R6.0 Language Environment Debugging Guide*, GA22-7560, to learn more about using IPCS to format and analyze dumps.

3.5 Java Transaction Dump (TDUMP)

When an out of memory errors (OutOfMemoryErrors) occurs, a Java Transaction Dump (TDUMP) is produced. The TDUMP is generated from the IEATDUMP MVS service by default when there is a program check or exception in the JVM.

You can use IPCS to inspect a TDUMP. You can also inspect a TDUMP using a Java application such as svcdump if the dump data set has been transferred in binary mode to the inspecting system. A TDUMP can have multiple Address Spaces. It is important to work with the correct address space associated with the failing Java process.

In the server region joblog and syslog, you see messages like IEA822I indicating that a transaction dump has been taken as shown in Example 3-6.

Example 3-6   Transaction dump message in joblog

```
IEA822I COMPLETE TRANSACTION DUMP WRITTEN TO ASSR1.JVM.TDUMP.WS6422S.D050809.T190153
```

The joblog will also show the JVM messages in the trace part of the joblog as shown in Example 3-7

Example 3-7   Java OutOfMemoryErrors shown in joblog

```
JVMDG217: Dump Handler is Processing OutOfMemory - Please Wait.
JVMHP002: JVM requesting System Transaction Dump
JVMHP012: System Transaction Dump written to ASSR1.JVM.TDUMP.WS6422S.D050809.T1
JVMDG315: JVM Requesting Heap dump file
JVMDG318: Heap dump file written to /SC42/tmp/HEAPDUMP.20050809.190559.16843095
JVMDG303: JVM Requesting Java core file
JVMDG304: Java core file written to /SC42/tmp/JAVADUMP.20050809.190613.16843095
JVMDG274: Dump Handler has Processed OutOfMemory.
JVMST109: Insufficient space in Javaheap to satisfy allocation request

...Trace: 2005/08/09 19:06:19.729 01 t=7CC148 c=11.1 key=P8 (13007002)
  ThreadId: 00000029
  FunctionName: com.ibm.ws.webcontainer.servlet.ServletWrapper
  SourceId: com.ibm.ws.webcontainer.servlet.ServletWrapper
  Category: SEVERE
  ExtendedMessage: BBOO0220E: SRVE0068E: Could not invoke the service() method
on servlet MemLeak. Exception thrown : java.lang.OutO
fMemoryError: JVMXE006:OutOfMemoryError, stAllocArray for executeJava failed
```

In the case of an OutOfMemory (OOM) error looking for the memory leak using the transaction dump in IPCS is not useful. In the case of OOM error using java tools like Heaproots or the Memory dump diagnostic for Java are more effective.

You can disable the generation of a TDUMP, but IBM does not recommend it.

For more information about TDUMP, refer to *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2Diagnostics Guide* SC34-6358-01 for your version of Java on z/OS.

3.6 Javadump

A Javadump produces files with diagnostic information that is related to the JVM and a Java application captured at a point during execution. For example, the information can be about the operating system, the application environment, threads, native stack, locks, and memory. The exact contents depend on the platform that you are running.

By default, a Javadump occurs when the JVM terminates unexpectedly. A Javadump can also be triggered by sending specific signals to the JVM.

Note: Javadump is also known as Javacore. This is *not* the same as a core file (that is, an operating system feature that can be produced by any program, not just the JVM).

For more information on the Javadump, refer to *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2Diagnostics Guide* SC34-6358-01 for your version of Java on z/OS.

3.7 Heapdump

Heapdump is an IBM JVM facility that generates a dump of all of the live objects that are on the Java heap, that is, those that are used by the Java application. It shows the objects that are using large amounts of memory on the Java heap and what is preventing them from being collected by the Garbage Collector.

For more information about the Heapdump refer to *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.2Diagnostics Guide* SC34-6358-01 for your version of Java on z/OS.

4

Diagnostic tools for WebSphere for z/OS

In this chapter, we explain the following diagnostic tools for WebSphere for z/OS that can support you in the problem determination process:

- JVM dump and heap analysis tools
- Memory Dump Diagnostic Tool for Java
- Trace Analyzer for WebSphere Application Server
- Java Garbage Collection Formatter
- dumpNameSpace tool
- Rational Application Developer V6
- Tivoli® Performance Viewer
- Omegamon

For each tool, we provide information about its nature, discuss when to use it, and explain how to use it. We describe the output, demonstrate how to interpret it, and provide an example.

4.1 JVM dump and heap analysis tools

This section describes three problem analysis tools for the WebSphere for z/OS production environment:

- Svcdump.jar
- HeapRoots
- Dumpviewer GUI and jformat

The JVM sits at the core of the execution environment for WebSphere. The many benefits of using Java as a development language and a JVM as a targeted run time also bring with them a few issues. The diagnosis of problems in a production environment can seem both more difficult than, and very remote from, the original development environments, where real-time, GUI-based debugging is the standard. However, this real-time approach can transfer to IBM @server® zSeries and production servers, at least from a technical, "it is possible" perspective. For example, it is possible to load a server with a JVM in debug mode and attach a remote debugger such as WebSphere Studio Integrated Edition.

There are other tools available that make use of the various JVM interfaces such as JVM Profiling Interface ((JVMP) and JVM Monitoring Interface (JVMMI), to provide real-time monitoring and profiling of the JVM that is attached to WebSphere servant processes. These techniques work well when the transaction throughput is low, or when you want to target behavior on a server that is being used on z/OS in development or testing before deploying it to a production server. They are usually unsuitable for production usage, for a variety of reasons:

- There is a requirement to load a debug version of the JVM. The debug version is built with many asserts for invalid conditions turned on. This means that the performance of the JVM is not adequate for production usage.
- The quantity of data that is generated by profiling tools from a server with a high transaction throughput is generally much larger than can be accommodated by the system or than can be meaningfully post processed from the profiling tool.
- Profiling tools are typically fairly platform agnostic and therefore not sensitive enough to platform peculiarities or problems where the problem needs to be researched through middleware or OS components.

The analysis of heap-related issues, such as OutOfMemoryError and other crashes, hangs, or loops in WebSphere for z/OS address spaces can be similar to that for the level that is possible with the earlier deployment environments, such as CICS® and IMS™. You use the relatively low impact tools of unformatted dumps and JVM internal information for analysis, which means that you must draw together knowledge of the JVM internals with the less invasive diagnostic approaches (such as SDUMP) that are typically used to diagnose problems in z/OS production environments. These tools can be used to diagnose problems that affect your important production workload, but can only be recreated in these high-transaction environments.

These approaches should be driven initially by the systems programming staff, because they have authority to access the SVC dumps or transaction dumps that are taken during failures or have the authority to request console dumps of hung or looping servers. After the unformatted dumps are available, the post processing and interpretation of the data can be done by either systems programming or by development staff, because the tools are Java-based and therefore not tied to z/OS.

4.1.1 Svcdump.jar

The svcdump.jar file allows direct access to the binary SVC dump or transaction dumps created on z/OS without the need for intermediate software such as IPCS. There are three packages that are shipped in svcdump.jar:

- Dump utility: com.ibm.jvm.svcdump.Dump package

  This formats native and Java stacks for threads in dumped processes that include an instantiated JVM. The dump utility includes a function to print out other useful information, for example, in core trace buffers maintained by the JVM and the system trace that mimic or extend the information that can be obtained with IPCS.

- FindRoots utility: com.ibm.jvm.findroots.* package

  This provides multiple ways of formatting the object graphs that are present in the Java-managed heap. This is critical for the sometimes difficult tasks of pinning down object leaks and making sense of heap occupancy in other ways.

- Java API

  This can be used to write ad hoc utilities. For example, you could write small programs that report on objects in the Java heap that maintain state data about a business application.

How to use svcdump.jar

The svcdump.jar file is available publicly from this Web site (requires IBM registration):

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=diagjava

Note: This tool is under active development, so it would be helpful to IBM if you provide feedback about your experiences with it.

You need three files:

- The svcdump.jar file itself
- The doc.jar file, which contains documentation for the exposed API
- libsvcdump.so, a DLL that allows the Java code to access an unformatted dump in an MVS data set rather than in the HFS (when you use this DLL, you do not have to provide a large HFS data set to copy the dump to, meaning that on z/OS, the original dump can be analyzed instead)

Attention: We used Version 20041012 of the code. Later versions might offer additional functions or different output.

To use the dump utility, copy the three files in binary format to a suitable location in the HFS. In our example, the files are in /u/dclarke.

Use the following command to confirm the version of the utility that you are running:

```
java -cp svcdump.jar com.ibm.jvm.svcdump.Dump –version
```

Example 4-1 on page 67 shows the results that we obtained after we used the command.

Example 4-1 Determining the dump utility version

```
You are using
jar:file:/C:/Documents%20and%20Settings/Administrator/My%20Documents/SVCDumps/svcdump200410
07.jar!/com/ibm/jvm/svcdump/Dump.class
which was last modified on Tue Oct 12 14:53:29 BST 2004
```

This uses introspection to identify when this code was last modified.

Parameters and options to set up the dump utility

Table 4-1 on page 67 is an overview of the options for the dump utility. To use the dump utility with these options, use the following command:

```
java com.ibm.jvm.svcdump.Dump [options] <filename>
```

Table 4-1 Parameters and options for the dump utility

Option	Description
`-debug`	Print internal debugging information
`-verbose`	Print extra information
`-heap`	Print a table that shows which classes have the most objects allocated
`-cache`	Print alloc cache
`-exception`	Print old exception objects
`-dis <addr> <n>`	Disassemble <n> instructions starting at <addr> (hex)
`-dump <addr> <n>`	Dump <n> words of storage starting at <addr> (hex)
`-dumpapars`	Print the APARs installed
`-dumpclasses`	Print information about all classes
`-dumpclass <addr>`	Print information about class at given address
`-dumpobject <addr>`	Print information about object at given address
`-dumpmdata <addr>`	Print information about mdata at given address
`-dumpprops`	Print the system properties
`-dumpnative`	Dump all the native methods
`-dumpverbosegc`	Dump the verbosegc
`-heapstats`	Print stats about heap usage
`-tcbsummary`	Print a summary of what the TCBs are doing
`-systrace`	Print the system trace
`-hpitrace`	Print the HPI trace
`-caa <addr>`	Specify the CAA to use when disassembling
`-r<n>`	Include saved register <n> in stack trace
`-args`	Print first four function arguments
`-verifysubpools`	Verify subpools
`-verifyheap`	Verify heap
`-printdosed`	Print pinned and dosed objects
`-printroots`	Print garbage collection (GC) roots
`-version`	Print the version and exit
`-fullversion`	Print the version of the jvm in the dump and exit
`-title`	Print title of the dump and exit

Option	Description
-time	Print time of the dump and exit-dis <addr> <n> disassemble <n> instructions starting at <addr> (hex)

The most important parameters for problem determination in WebSphere for z/OS are described in more detail in *WebSphere for z/OS V5 JVM Dump and Heap Analysis Tools*, REDP-3950.

Example 4-2 shows a simple shell script that can be used to execute the utility.

Example 4-2   Shell script for the dump utility

```
#!/bin/sh
#TZ=EST5EDT
set -x
DUMPNAME=£1
SVCDUMPJARFILE=/u/dclarke/svcdump20041007.jar
SVCDUMPLIBPATH=/u/dclarke

java -Xmx348m -Dsvcdump.libpath=£SVCDUMPLIBPATH
-Xbootclasspath/p:£SVCDUMPJARFILE \
   -Dsvcdump.default.jvm=0 \
   com.ibm.jvm.svcdump.Dump -exception \
   £DUMPNAME \
   >>£DUMPNAME.svcdump.txt

java -Xmx348m -Dsvcdump.libpath=£SVCDUMPLIBPATH
-Xbootclasspath/p:£SVCDUMPJARFILE \
   -Dsvcdump.default.jvm=0 \
   com.ibm.jvm.svcdump.Dump -hpitrace  \
   £DUMPNAME \
   >>£DUMPNAME.svcdump.txt

java -Xmx348m -Dsvcdump.libpath=£SVCDUMPLIBPATH
-Xbootclasspath/p:£SVCDUMPJARFILE \
   -Dsvcdump.default.jvm=0 \
   com.ibm.jvm.svcdump.Dump -systrace  \
   £DUMPNAME \
   >>£DUMPNAME.svcdump.txt
```

The shell script can then be invoked as shown in Example 4-3.

Example 4-3   invoke shell script

```
/u/dclarke:==>svcdump.sh ONTOP.GS031.P10316.C724.JVMDMP
+ DUMPNAME=ONTOP.GS031.P10316.C724.JVMDMP
+ SVCDUMPJARFILE=/u/dclarke/svcdump20041007.jar
+ SVCDUMPLIBPATH=/u/dclarke
+ java -Xmx348m -Dsvcdump.libpath=/u/dclarke -
Xbootclasspath/p:/u/dclarke/svcdump20041007.jar -
Dsvcdump.default.jvm=0 com.ibm.jvm.svcdump.Dump -exception
ONTOP.GS031.P10316.C724.JVMDMP
+ 1>> ONTOP.GS031.P10316.C724.JVMDMP.svcdump.txt
```

Analysis of the dump can take some time, especially for the first execution. The tool stores some heap information in a small .cache file that make subsequent executions faster. Use this simple JCL to run the utility in batch:

```
//STEP1    EXEC PGM=BPXBATCH,REGION=0M,
// PARM='SH /u/dclarke/svcdump.sh ONTOP.GS031.P10316.C724.JVMDMP'
```

What can I find out with com.ibm.jvm.svcdump.Dump?

If your application crashed, you can use the dump utility to analyze the problem by establishing:

- Which thread the crash occurred on
- What the native stack was on the failing thread
- What the Java stack was for the failing thread

If you experience an application loop or hang, you can use the utility to identify:

- The thread under which a loop is occurring
- The threads that are contending for resources or that are involved in a lockout
- A thread waiting for some operation that is external to the server

In all of these cases, the reports can help attach the failure to a particular component or subcomponent before you raise an incident with the IBM service team.

Use of com.ibm.jvm.findroots.* on z/OS

FindRoots is a cross-platform tool for analyzing memory leaks in Java applications. Although the package is still called FindRoots for historical reasons, it is now divided into a number of smaller tools that do the same thing; this is mainly to separate the extraction of the heap from the subsequent analysis:

1. Run the convert tool to extract the heap dump and create a Portable Heap Dump (a .phd file).
2. Run another tool, try PrintDomTree before the others, to analyze the .phd file. The advantage of this approach is that you can pass the smaller .phd files rather than large SVC or Transaction dumps. The recommended approach for analyzing memory leaks is to run PrintDomTree.

The quintessential memory leak in Java is created by an application or middleware bug that is causing a reference to an object to retained. Because this object is still reachable through all of its referents, it can render a large portion of an object graph still reachable and, therefore, not eligible for garbage collection.

For example, consider the case where some state data for a transaction is kept as an XML data object. The design is one where the data is only relevant for the life of the transaction, and the intent is that this object should become eligible for garbage collection after the transaction has ended. There is a bug in this code that is causing some other global object to maintain a reference to our XML data object after the end of the transaction. Although the object itself is small, it contains a reference to non-trivial numbers of objects that is created by XML parsing.

A successful diagnosis of the problem might be as follows:

1. A still reachable XML data object exists after each transaction runs.
2. The remaining reachable data gradually increases over time after each garbage collection cycle.
3. You can observe this from the verbose:gc output (or from the incore verbosegc data).
4. Eventually, the Java heap is exhausted and an OutOfMemoryError is thrown.
5. You obtain a console dump of the server at the time when the heap usage is high.
6. You run the PrintDomTree tool and establish from the reports that there is an unexpectedly high number of these XML data objects.

7. You find the unexpected reference in the reports from the global object.
8. A review of the logic reveals that this reference is not nulled out after the transaction as the design anticipated.

More information about the different FindRoots utilities and the reports they produce is available in *WebSphere for z/OS V5 JVM Dump and Heap Analysis Tools*, REDP-3950.

4.1.2 HeapRoots

The HeapRoots utility is shipped in the HR204.jar file and is derived from the same requirement as that for being able to map the heap object graphs, but was originally developed for the JVM shipped with IBM AIX®. It is now possible to use this code seamlessly with binary SVC or transaction dumps, providing a range of additional functions are available with the FindRoots utility in svcdump.jar.

HeapRoots is available through the alphaWorks® Web site, at:

http://www.alphaworks.ibm.com/tech/heaproots

To use HeapRoots with .phd files, use the following command:

```
java -classpath svcdump.jar;HR204.jar HR.main.Launcher
```

Examples and details about HeapRoots are available in *WebSphere for z/OS V5 JVM Dump and Heap Analysis Tools*, REDP-3950.

4.1.3 Dumpviewer GUI and jformat

The Dumpviewer GUI and jformat are cross-platform utilities that allow you to use graphs to review native and Java stacks and that provide facilities to help diagnose locking-related and heap-related problems. On platforms other than z/OS, the tools work with platform-independent dump files that are extracted from the native dump. On z/OS, the tools work directly with SVC or transaction dump, using the code from the svcdump.jar file.

The tools are shipped with the IBM Java Development Kits (JDKs) for all platforms. The GUI is described in some detail in *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.3.1, Diagnostics Guide*, SC34-6200, and *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.1, Diagnostics Guide*, SC34-6309, which is shipped with the IBM JDKs 1.3.1 and 1.4.1.

The Dumpviewer can be invoked with the following command:

```
java -Xmx512m -cp svcdump.jar com.ibm.jvm.dump.format.DvConsole -g
```

When the GUI initializes, select **File** from the menu to locate the dump for initialization.

Restriction: To use this tool with z/OS, you must export the DISPLAY environment variable to a valid X Server display on a Win32® or Linux system.

With the Win32 JDK™ shipped with WebSphere for Windows, the formatter can be invoked with the jformat command, found in C:\Program Files\IBM\Java142\bin\jformat.

The **–g** switch starts the GUI:

```
"C:\Program Files\IBM\Java142\bin\jformat" -J-Xmx512m -g
```

For more information, see *IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.3.1, Diagnostics Guide*, SC34-6200, (used with WebSphere V5.0), and *IBM*

Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.1, Diagnostics Guide, SC34-6309, (used with WebSphere V5.1). These manuals and detailed documentation for the garbage collector used by the IBM JVM, can be downloaded from:

`http://www.ibm.com/developerworks/java/jdk/diagnosis/`

4.2 Memory Dump Diagnostic Tool for Java

The Memory Dump Diagnostic for Java tool analyzes common memory dump formats (heap dumps) from the JVM that is running the WebSphere Application Server.

In z/OS, when out of memory errors (OutOfMemoryErrors) occur, a Java transaction dump is produced. This dump can be viewed with IPCS, but the traditional diagnostic information available is not helpful for finding the OutOfMemoryError. When a java transaction dump is produced you will see message IEA822I in the server region joblog.

The analysis of memory dumps is targeted toward identifying regions within the Java heap that might be root causes of memory leaks. The tool is capable of analyzing very large memory dumps that are obtained from production environment application servers that have encountered OutOfMemoryErrors.

The tool is available for download from the developerworks Web site under WebSphere downloads at:

`http://www.ibm.com/developerworks/websphere/downloads/memory_dump.html`

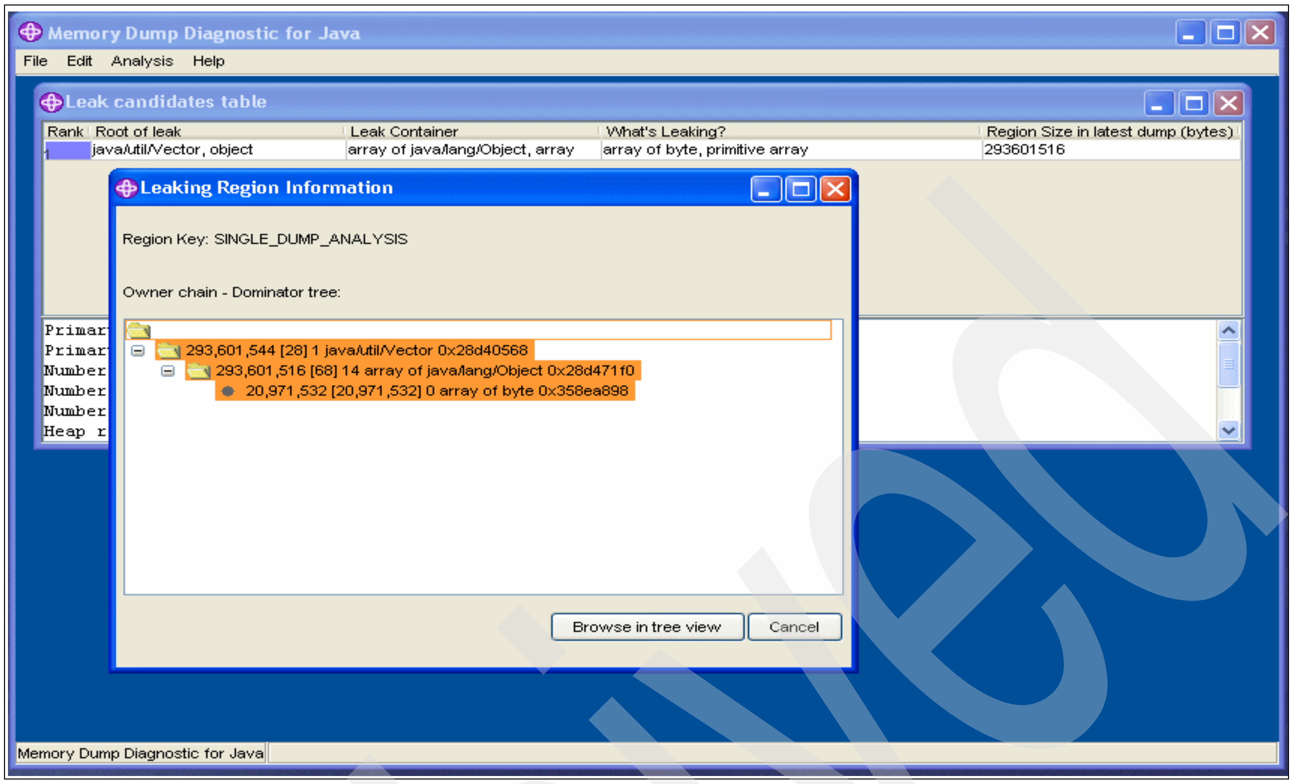
There is documentation with the tool that contains several screen captures of the functions that the tool provides.

Two types of analysis mechanisms are available: a single dump analysis and a comparative analysis of two dumps. The tool displays the contents of the memory dump in a graphical format while highlighting regions that are identified as memory leak suspects. The GUI provides browsing capabilities for verifying suspected memory leaking regions and for understanding the data structures that comprise the leaking regions.

The following formats of memory dumps are supported by this tool:

- IBM Portable Heap Dump (.phd) format
- IBM text heap dump format
- HPROF heap dump format
- SVC Dumps (on IBM z-series)

An example of the output produced by the tool is shown in Figure 4-1 on page 67.


Memory Dump Diagnostic for Java
File
Edit
Analysis
Help
Leak candidates table
Leaking Region Information
Browse in tree view
Cancel
Memory Dump Diagnostic for Java

Rank	Root of leak	Leak Container	What's Leaking?	Region Size in latest dump (bytes)
1	java/util/Vector, object	array of java/lang/Object, array	array of byte, primitive array	293601516

Region Key: SINGLE_DUMP_ANALYSIS

Owner chain - Dominator tree:

```
293,601,544 [28] 1 java/util/Vector 0x28d40568
  293,601,516 [68] 14 array of java/lang/Object 0x28d471f0
    20,971,532 [20,971,532] 0 array of byte 0x358ea898
```

Figure 4-1 Memory Dump Diagnostic tool for Java screens

4.3 Trace Analyzer for WebSphere Application Server

WebSphere for z/OS produces primary diagnostic information in the form of text-based trace logs. The trace logs are used throughout the product to diagnose failures and confirm correct code execution. However, reading a trace log in raw format can be a tedious task. Moreover, WebSphere for z/OS is being deployed in increasingly complex environments and problems are becoming more difficult to diagnose and resolve.

Trace Analyzer for WebSphere Application Server eases the process of reading diagnostic information from the WebSphere for z/OS trace logs by showing sequential, easy to read event oriented traces. Trace Analyzer for WebSphere provides:

- Visual trace presentation
- Search and filter capabilities
- Trace highlighting and mark-up
- Entry and exit record pairing

Trace Analyzer is written in Java and requires JVM 1.3.1 or later. To use it, follow this procedure:

1. Download the Trace Analyzer tool from the IBM alphaWorks Web site at:

   http://www.alphaworks.ibm.com/tech/ta4was

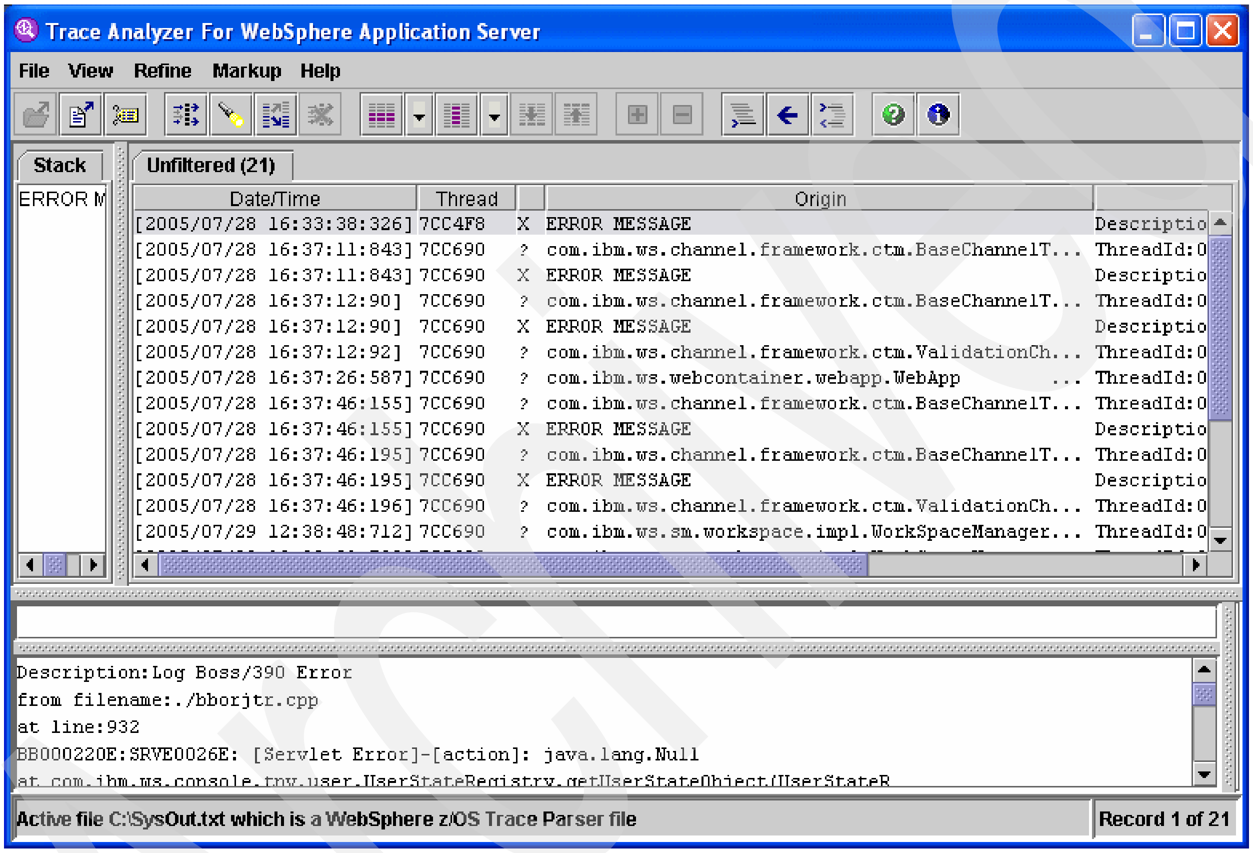
2. Download the JAR file to the selected directory on your workstation and run the application by issuing the following command:

   ```
   java -jar traceanalyzer.jar traceFile
   ```

where traceFile is the name of the trace file you want to open for analysis. This is an optional feature. You can open also files from the GUI by selecting **File Open**. Figure 4-2 shows a the Trace Analyzer for WebSphere z/OS window.

3. From the Refine menu, you can select **Filter and Search**. By selecting any entry from the trace pane, you can see its full contents in the bottom console.
4. For help using the program, see the integrated help system by selecting **Help** → **Using Trace Analyzer**.

This utility makes it relatively easy to read the diagnostic information, even when the user is not very familiar with the component that is being debugged or tested.



```
ERROR M

Date/Time                 Thread     Origin                                         
[2005/07/28 16:33:38:326] 7CC4F8  X  ERROR MESSAGE                                  Descriptio
[2005/07/28 16:37:11:843] 7CC690  ?  com.ibm.ws.channel.framework.ctm.BaseChannelT... ThreadId:0
[2005/07/28 16:37:11:843] 7CC690  X  ERROR MESSAGE                                  Descriptio
[2005/07/28 16:37:12:90]  7CC690  ?  com.ibm.ws.channel.framework.ctm.BaseChannelT... ThreadId:0
[2005/07/28 16:37:12:90]  7CC690  X  ERROR MESSAGE                                  Descriptio
[2005/07/28 16:37:12:92]  7CC690  ?  com.ibm.ws.channel.framework.ctm.ValidationCh... ThreadId:0
[2005/07/28 16:37:26:587] 7CC690  ?  com.ibm.ws.webcontainer.webapp.WebApp       ... ThreadId:0
[2005/07/28 16:37:46:155] 7CC690  ?  com.ibm.ws.channel.framework.ctm.BaseChannelT... ThreadId:0
[2005/07/28 16:37:46:155] 7CC690  X  ERROR MESSAGE                                  Descriptio
[2005/07/28 16:37:46:195] 7CC690  ?  com.ibm.ws.channel.framework.ctm.BaseChannelT... ThreadId:0
[2005/07/28 16:37:46:195] 7CC690  X  ERROR MESSAGE                                  Descriptio
[2005/07/28 16:37:46:196] 7CC690  ?  com.ibm.ws.channel.framework.ctm.ValidationCh... ThreadId:0
[2005/07/29 12:38:48:712] 7CC690  ?  com.ibm.ws.sm.workspace.impl.WorkSpaceManager... ThreadId:0
```

```
Description:Log Boss/390 Error
from filename:./bborjtr.cpp
at line:932
BB000220E:SRVE0026E: [Servlet Error]-[action]: java.lang.Null
at com.ibm.ws.console.tpv.user.UserStateRegistry.getUserStateObject(UserStateR
```

Active file C:\SysOut.txt which is a WebSphere z/OS Trace Parser file   Record 1 of 21

Figure 4-2 Trace Analyzer for WebSphere for z/OS

4.4 Java Garbage Collection Formatter

The JVM heap stores all objects that are created by a running Java application. Garbage collection is the process of automatically freeing objects that are no longer referenced by the Java program. This tool displays the Java garbage collection statistics in a tabular format.

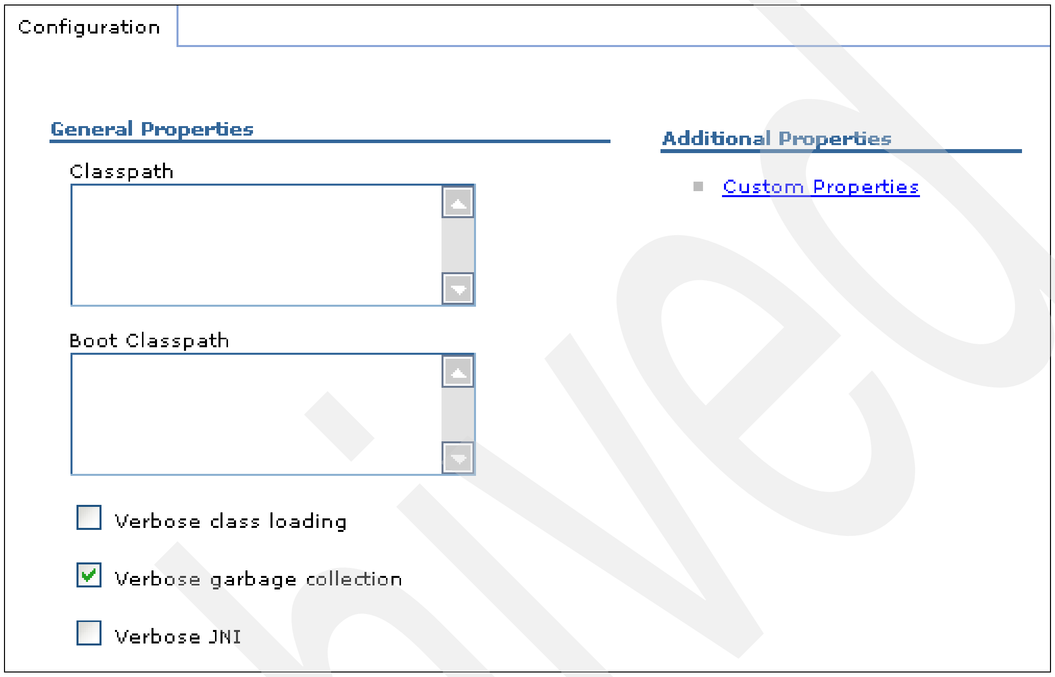
There is a Java garbage collection formatter script by John Rankin, called VGC131v7.awk, in the Techdocs TD101216, that you can download from the IBM Web site:

`http://www.ibm.com/support/techdocs`

You first need to capture the Java Garbage Collection statistics. To turn on verbose garbage collection through the Administrative Console, follow these steps:

1. Expand the Servers node in the left-hand menu and select **Application Servers**.

2. From the list of servers, select the application server for which you want to enable verbose garbage collection.
3. From the Java and Process Management menu, select **Process Definition.**
4. From the processType list, select the appropriate servant.
5. From the Additional Properties menu, select **Java Virtual Machine**.



[ ] Verbose class loading
[✓] Verbose garbage collection
[ ] Verbose JNI

Figure 4-3 Enable GC Verbose in Advanced Java virtual machine settings

6. Find **Verbose garbage collection** in the list of General Properties and select it.

   Figure 4-3 shows the Advanced Java Virtual Machine settings to enable Verbose mode.
7. Click **Apply** to apply your changes. Click **Save** to save your configuration.
8. Run transactions through your server for some time. The results are a log file that is similar to the one shown in Example 4-4.

Example 4-4 Java garbage collection trace sample

```
<AF[1]: Allocation Failure. need 528 bytes, 122081 ms since last AF>
<AF[1]: managing allocation failure, action=1 (0/255012224) (13421696/13421696)
<GC(1): GC cycle started Mon Aug  1 17:44:10 2005
<GC(1): freed 183448600 bytes, 73% free (196870296/268433920), in 400 ms>
<GC(1): mark: 332 ms, sweep: 68 ms, compact: 0 ms>
<GC(1): refs: soft 0 (age >= 32), weak 275, final 1348, phantom 0>
<AF[1]: completed in 403 ms>
```

9. Using a REXX or AWK script, format the output of the Java garbage collection trace to get a semicolon-delimited condensed file such as Example 4-5:

Example 4-5 Semicolon-delimited Java garbage collection trace

```
afnum ; timeSinceLastAF ; aftime ; afsize ; gcnum ; conGCnumb; timeSinceLastConGC; conGCtime; ConRes ;
ConTarget ; ConTraced ; ConFree ;gcstart ; gcfreed ; freespace ; heapsize ; gctime ;threadStopTime ;
threadStartTime ; marktime ; sweeptime ; compacttime ; msmin ; msmax ; msavg ; moved ; bytes ; reason ;
1;122081;403;528;1;;;;;;;;17:44:10;183448600;196870296;268433920;400;;;332;68;0;;;;;;
2;102844;514;528;2;;;;;;;;17:45:53;139549048;152714712;268433920;513;;;462;51;0;;;;;;
```

```
3;53149;526;5184;3;;;;;;;;17:46:47;134965264;147782608;268433920;526;;;464;62;0;;;;;;
4;431101;525;528;4;;;;;;;;17:53:58;140140296;150365624;268433920;524;;;474;50;0;;;;;;
5;346417;446;528;5;;;;;;;;17:59:45;156671872;164212856;268433920;446;;;392;54;0;;;;;;
6;9495859;525;528;6;;;;;;;;20:38:01;140242632;145579056;268433920;525;;;462;63;0;;;;;;
7;12308;539;528;7;;;;;;;;20:38:14;138183336;140835416;268433920;539;;;482;57;0;;;;;;
```

10. FTP the output file (in ASCII) to your workstation and import into a spreadsheet. An example is shown in Figure 4-4.

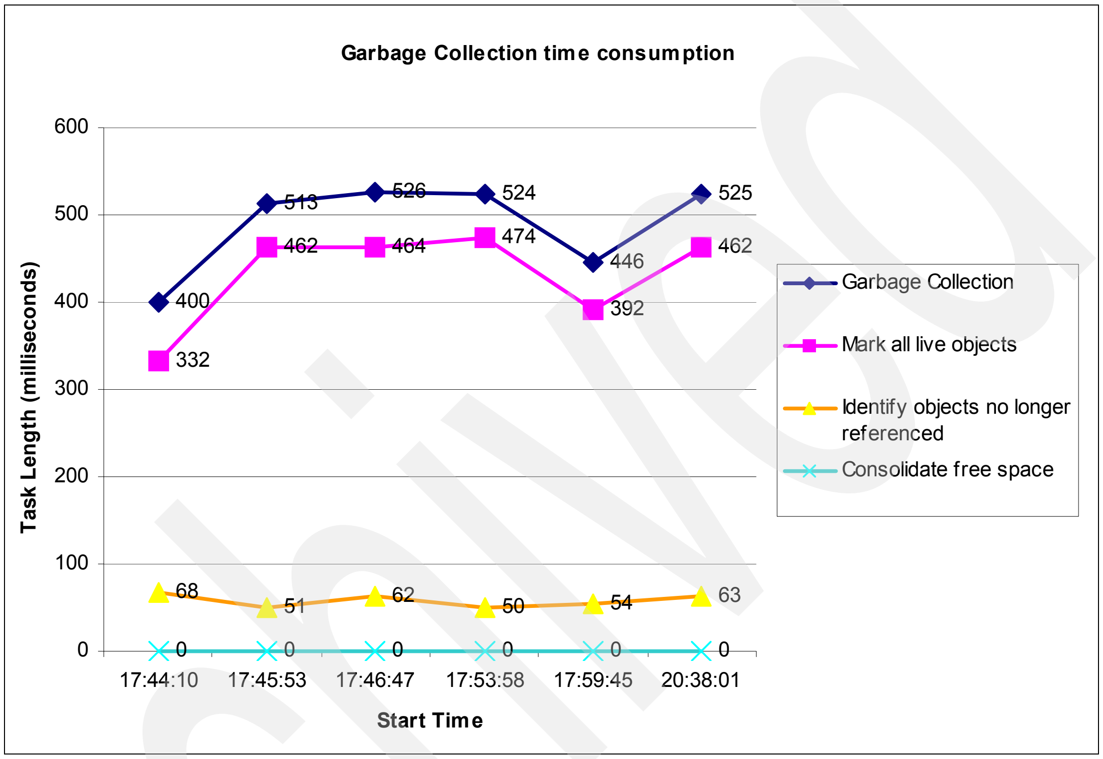


Figure 4-4 Graphical representation of garbage collection records

For more information, refer to the “Java memory tuning tip” topic in WebSphere for z/OS Information Center, available at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp`

4.5 dumpNameSpace tool

Problems can surface when you are accessing resources through the namespace. The namespace is a collection of references to resources, such as connection pools, Enterprise JavaBeans™ (EJBs), and message listeners. In the WebSphere for z/OS environment, the namespace is federated among all servers in the cell, with each server process containing its own name server.

The dumpNameSpace tool obtains the contents of a namespace in a name server. The tool can be invoked with a UNIX shell script or from its interface in the WebSphere Application Server API. It can only dump namespaces from remote name servers that are not local to the server process.

The dumpNameSpace tool can be run with a number of parameters specified. Invoke the tool from the OMVS command line by following these steps:

1. Change the present directory to <Installation_directory>/<Profiles>/<Profiles_Name>/bin, where:
   - <Installation_directory> is the directory into which WebSphere was installed.
   - <Profiles> is the directory bellow WebSphere for z/OS installation directory.
   - <Profiles_Name> is the name of the profile for which you want to obtain the namespace dump.
2. Run this command:

   ```
   dumpNameSpace.sh -port 22809 > nameSpaceDumpFile.txt
   ```

   The port parameter indicates the bootstrap port which, if not specified, defaults to 2809.
3. Log on to the Administrative Console. Select **Servers** → **Application servers** → **server** → **Ports.** You can see the bootstrap port used in your WebSphere for z/OS.
4. Set the final parameter to output to the appropriate file and directory.

For more detailed information about all available parameters and their uses, refer to the "dumpNameSpace tool" topic in WebSphere for z/OS Information Center, available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

The dumpNameSpace tool generates a list of all resources and their types. This can be useful for diagnosing problems when resources are referenced and not found in an application. This information can also help resolve of ClassCastException messages. Example 4-6 shows a sample of output from the dumpNameSpace tool. The resource is displayed in the left column, the type in the right column.

Example 4-6 dumpNameSpace output

```
Getting the initial context
Getting the starting context
==============================================================================
Name Space Dump
   Provider URL: corbaloc:iiop:localhost:22809
   Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
   Requested root context: cell
   Starting context: (top)=cl6552
   Formatting rules: jndi
   Time of dump: Mon Aug 01 12:09:10 EDT 2005
==============================================================================
==============================================================================
Beginning of Name Space Dump
==============================================================================
    1 (top)
    2 (top)/legacyRoot                                  javax.naming.Context
    2    Linked to context: cl6552/persistent
    3 (top)/domain                                      javax.naming.Context
    3    Linked to context: cl6552
    4 (top)/persistent                                  javax.naming.Context
    5 (top)/persistent/cell                             javax.naming.Context
    5    Linked to context: cl6552
    6 (top)/cellname                                    java.lang.String
    7 (top)/cell                                        javax.naming.Context
    7    Linked to context: cl6552
    8 (top)/nodes                                       javax.naming.Context
    9 (top)/nodes/nd6552                                javax.naming.Context
   10 (top)/nodes/nd6552/domain                         javax.naming.Context
   10    Linked to context: cl6552
   11 (top)/nodes/nd6552/servers                        javax.naming.Context
```

```
   12 (top)/nodes/nd6552/servers/ws6552               javax.naming.Context
================================================================================
End of Name Space Dump
================================================================================
```

4.6 Rational Application Developer V6

Rational Application Developer V6 is a very comprehensive development environment. There are several perspectives in Rational Application Developer. The Debug Perspective is a very useful tool for looking for problems in application code.

The Debug Perspective is used for testing and debugging Java applications, XSL transforms, and other components that are developed within Rational Application Developer. You can debug applications locally on a test server or remotely on a server such as WebSphere for z/OS.

4.6.1 When to use Rational Application Developer

With the debugger, you can control the execution of your program by setting breakpoints, suspending launches, stepping through your code, and examining the contents of variables.

Breakpoints are temporary markers that you put in your program to tell the debugger to stop your program at a given point. When the workbench is running a program and encounters a breakpoint, it suspends execution. The corresponding thread is suspended (that is, temporarily stops running) so that you can see the stack for the thread.

Execution suspends at the breakpoint before the statement is executed. You can check the contents of variables and the stack. You can then step over statements, step into other methods or classes, continue running until the next breakpoint is reached, or continue running until you reach the end of the program.

Application developers can use the Rational Application Developer remote debugger to ensure that their application is working as designed on the WebSphere for z/OS platform. It is used to identify any WebSphere for z/OS-specific bugs that cannot be tested during function testing in the WebSphere test environment in Rational Application Developer.

The local system runs the debugger, and the remote system runs both the debugging engine and your program. The person debugging the program on the workstation interacts with the program as usual (except where breakpoints or step commands introduce delays) and is able to control the program and observe the internal behavior of the remote program from the local system.

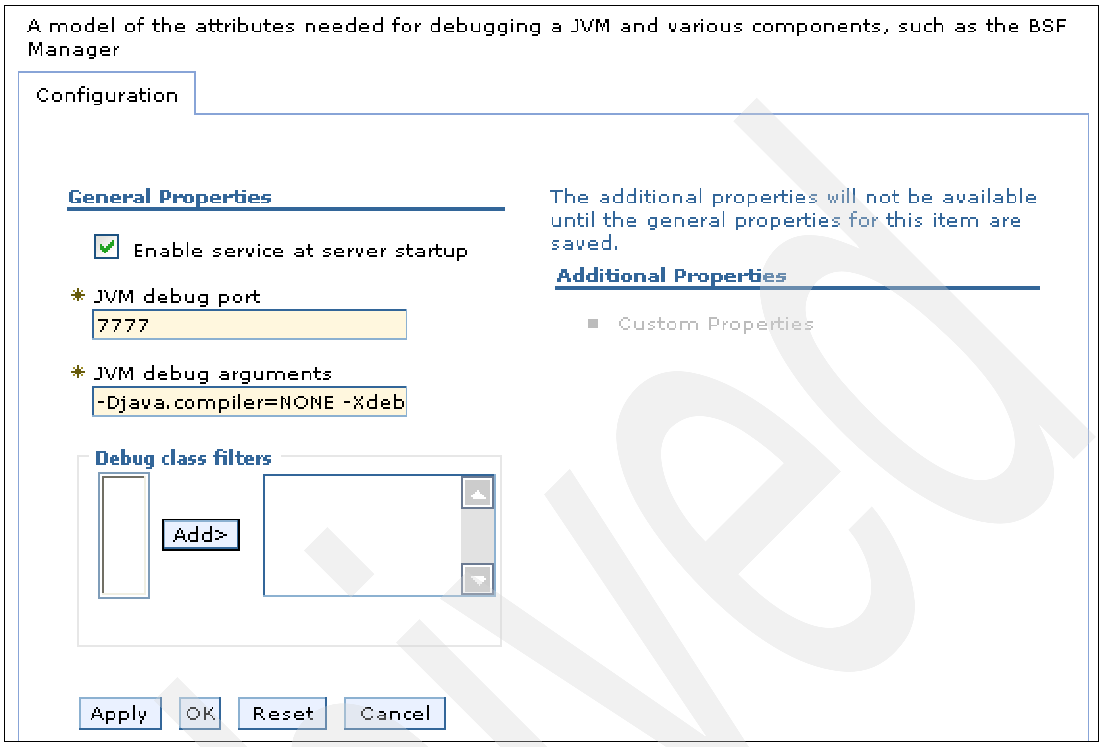
4.6.2 Setting up Rational Application Developer

To use the Rational Application Developer Debug Perspective, you must first set up the tool and then interact with the tool while it is connected to the running application.

Configuring Rational Application Developer for remote debugging requires changes in the WebSphere for z/OS environment and a connection to the remote application server from Rational Application Developer. Follow these steps to enable remote debugging:

1. Log in to the administrative console for your WebSphere for z/OS.
2. Expand the **Servers** item in the menu and select **Application Servers**. Select the server for which you want to enable debugging.

3. Under Additional Properties, select **Debugging Service**. Figure 4-5 shows a sample of the Debugging Service panel.


A model of the attributes needed for debugging a JVM and various components, such as the BSF Manager

Configuration

General Properties

✓ Enable service at server startup

* JVM debug port: 7777

* JVM debug arguments: -Djava.compiler=NONE -Xdeb

The additional properties will not be available until the general properties for this item are saved.

Additional Properties

- Custom Properties

Figure 4-5 Enable Debugging Service in Administrative Console

4. You can specify the **JVM debug port**; port 7777 as the default.
5. Verify the **JVM debug arguments**. The default settings are:

```
-Djava.compiler=NONE
-Xdebug
-Xnoagent
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7777
```

   Debug filters do not need to be set initially. You might find them useful as you gain experience with the debugger.

6. Click **Apply** and **Save** to apply your modifications and save the configuration.
7. Stop and start the server for which you enabled debugging.
8. Now in Rational Application Developer, open the Debug Perspective by selecting **Window** → **Open Perspective** → **Debug**.
9. Open the Debug configurations by selecting the debug icon and choosing **Debug** from the menu as show in Figure 4-6.

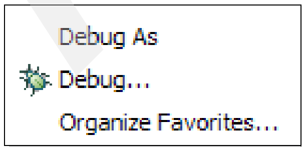
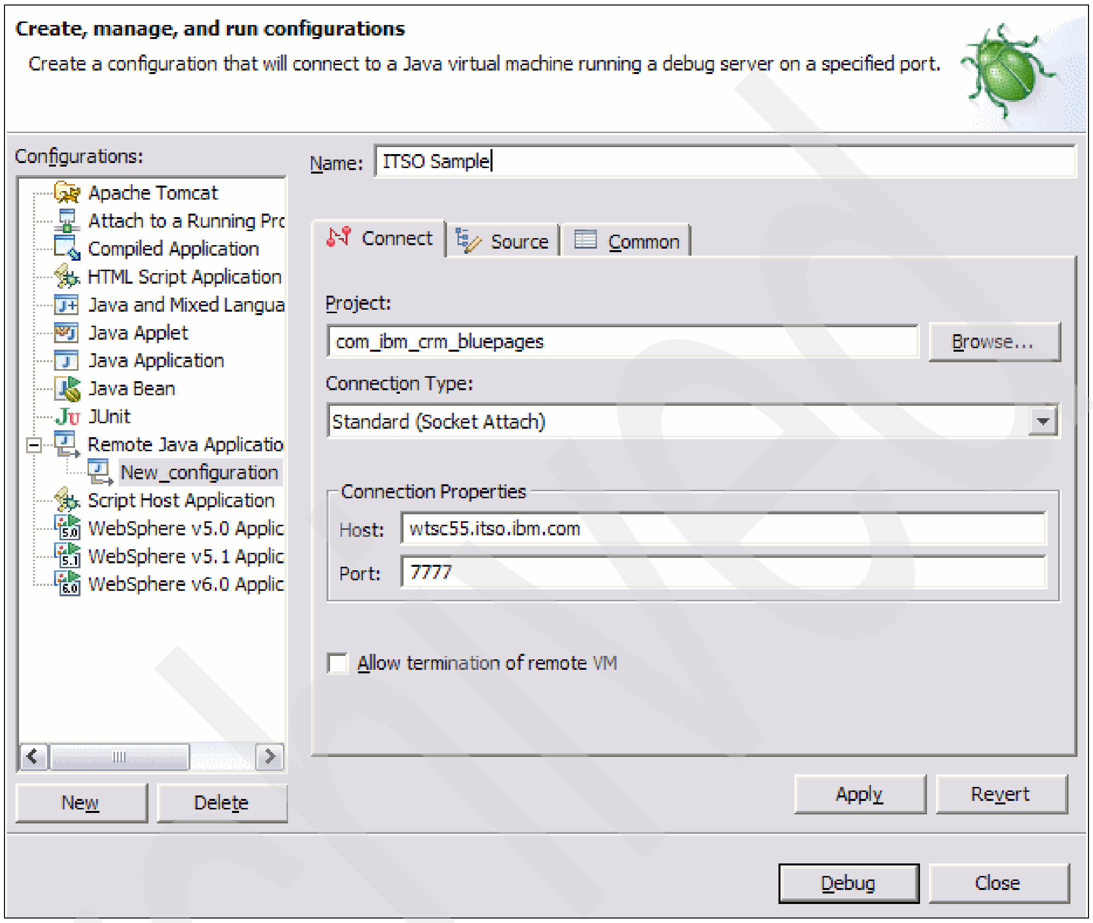


Figure 4-6 Debug menu from the Debug icon

10. Select **Remote Java Application** from the list of configurations and click **New** to create a new configuration. You should see a panel similar to the one shown in Figure 4-7 on page 74.


Create, manage, and run configurations

Create a configuration that will connect to a Java virtual machine running a debug server on a specified port.

Name: ITSO Sample

Project: com_ibm_crm_bluepages

Connection Type: Standard (Socket Attach)

Connection Properties

Host: wtsc55.itso.ibm.com

Port: 7777

☐ Allow termination of remote VM

Figure 4-7 Remote Java Application Debug configuration

11. Enter a Name for your configuration. Select the project you want to debug by clicking **Browse** and choosing it from the list of projects in the work space.
12. Enter the host address of your WebSphere for z/OS application server and enter the JVM debug port that you specified in the administrative console.
13. Click **Apply** to save your changes.
14. Click **Debug**. Now, you can start debugging the application on the remote application server. A debug engine daemon is listening for a connection.

To debug Java source code, you set breakpoints in the source code. You can set a breakpoint on a line of code to be triggered when a certain exception occurs. Add breakpoints in your code by double-clicking the gray area next to the line of code that you want to break. Right-click the breakpoint and select **Breakpoint Properties** to specify more detailed properties.

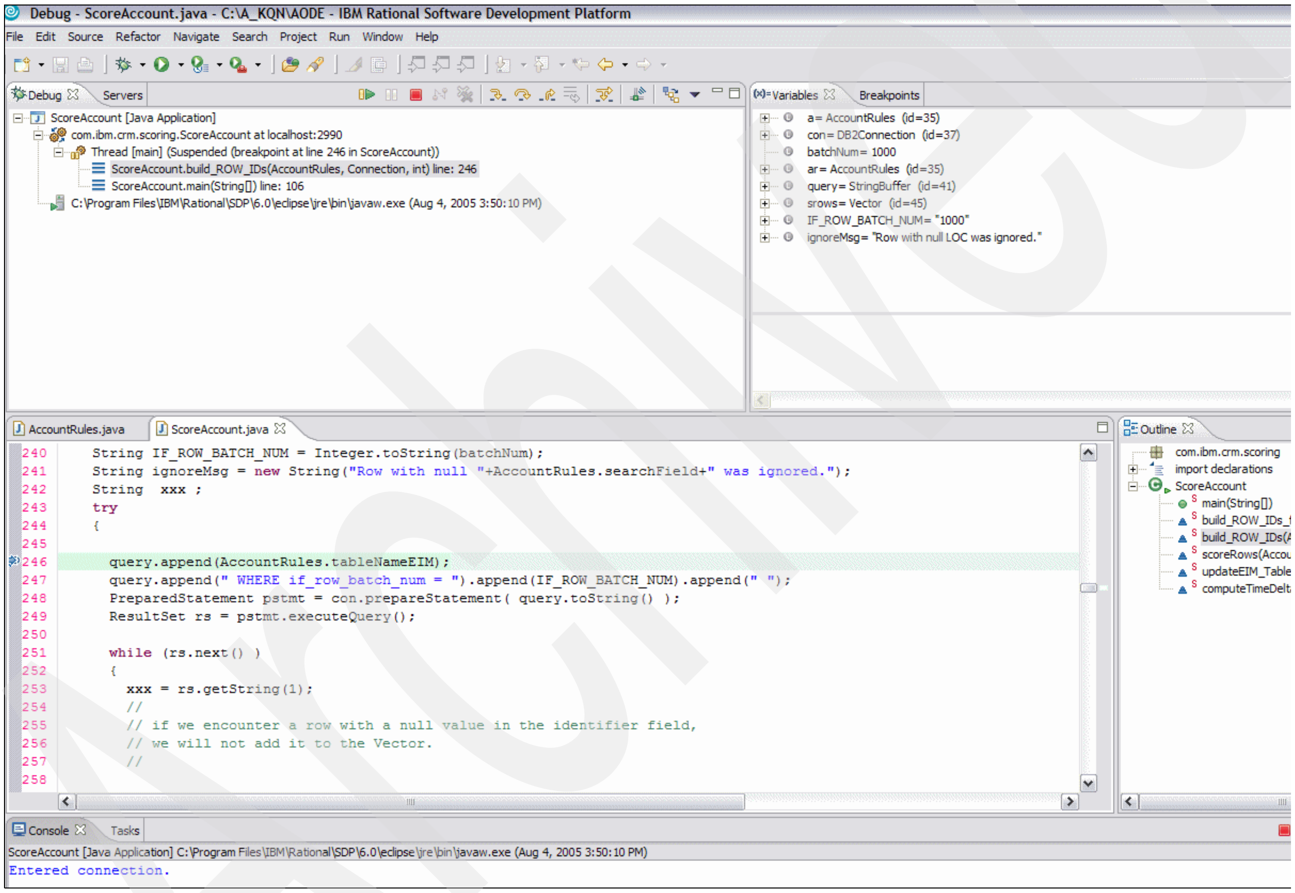
To start a program in debug mode, click the Debug icon, and the Debug Perspective opens. If you have multiple debug configurations, you can choose which to debug by selecting the arrow next to the Debug icon and selecting it from the menu.

In the Debug Perspective, you use icons to step into a line of code, to step over a line of code, or to run to the end of a method (step return). There are multiple views to aid you in debugging your application, including the Variables, Inspector, Debug, and Outline views.

4.6.3 Rational Application Developer output and interpretation

The debugger provides information interactively. As the application runs, you can view data as it is loaded, as it is manipulated, and as it flows through the application.

The Rational Application Developer Debug Perspective features different views to provide different information. Figure 4-8 show some separate views. You can access views by selecting **Window** → **Show View**. Change between the currently open views by selecting a tab from the bottom of each area in the Rational Application Developer window.



Debug - ScoreAccount.java - C:\A_KQN\AODE - IBM Rational Software Development Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Servers
Variables
Breakpoints
AccountRules.java
ScoreAccount.java
Outline
Console
Tasks

```
ScoreAccount [Java Application]
  com.ibm.crm.scoring.ScoreAccount at localhost:2990
    Thread [main] (Suspended (breakpoint at line 246 in ScoreAccount))
      ScoreAccount.build_ROW_IDs(AccountRules, Connection, int) line: 246
      ScoreAccount.main(String[]) line: 106
  C:\Program Files\IBM\Rational\SDP\6.0\eclipse\jre\bin\javaw.exe (Aug 4, 2005 3:50:10 PM)
```

```
a= AccountRules  (id=35)
con= DB2Connection  (id=37)
batchNum= 1000
ar= AccountRules  (id=35)
query= StringBuffer  (id=41)
srows= Vector  (id=45)
IF_ROW_BATCH_NUM= "1000"
ignoreMsg= "Row with null LOC was ignored."
```

```
240     String IF_ROW_BATCH_NUM = Integer.toString(batchNum);
241     String ignoreMsg = new String("Row with null "+AccountRules.searchField+" was ignored.");
242     String  xxx ;
243     try
244     {
245
246       query.append(AccountRules.tableNameEIM);
247       query.append(" WHERE if_row_batch_num = ").append(IF_ROW_BATCH_NUM).append(" ");
248       PreparedStatement pstmt = con.prepareStatement( query.toString() );
249       ResultSet rs = pstmt.executeQuery();
250
251       while (rs.next() )
252       {
253         xxx = rs.getString(1);
254         //
255         // if we encounter a row with a null value in the identifier field,
256         // we will not add it to the Vector.
257         //
258
```

```
com.ibm.crm.scoring
import declarations
ScoreAccount
  main(String[])
  build_ROW_IDs_f
  build_ROW_IDs(A
  scoreRows(Accou
  updateEIM_Table
  computeTimeDelt
```

```
ScoreAccount [Java Application] C:\Program Files\IBM\Rational\SDP\6.0\eclipse\jre\bin\javaw.exe (Aug 4, 2005 3:50:10 PM)
Entered connection.
```

Figure 4-8 The Debug Perspective in Rational Application Developer

Another Rational Application Developer tool is the XML editor and perspective that can be used to properly create and maintain valid XML files. An XML validator ensures that the XML is in a valid format and can be useful in fixing files that have become incorrectly formatted.

The Rational Application Developer can import application client JAR, EJB™ JAR, EAR, and WAR files, which can be helpful in problem determination. Sometimes, it is important to be able to view the code of the application that you are deploying. These tools can take the place of a decompiler because they can import an archive file into an editable project.

4.7 Tivoli Performance Viewer

Tivoli Performance Viewer is a tool integrated in Administrative Console, which is used to look for bottlenecks in WebSphere for z/OS configuration.

Tivoli Performance Viewer can be used to fine-tune the performance of an enterprise system by optimizing resources. You can view the current performance activity of a server using the Tivoli Performance Viewer.

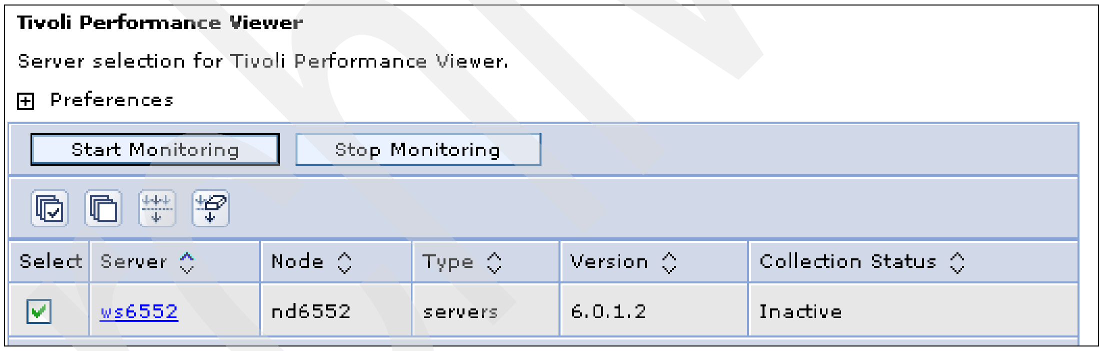
With Tivoli Performance Viewer, administrators and programmers can monitor the current health of WebSphere Application Server. Because the collection and viewing of data occurs in the application server, performance is affected. To minimize performance impacts, monitor only those servers whose activity needs investigation.

4.7.1 Setting Tivoli Performance Viewer up

Tivoli Performance Viewer no longer has to be installed in WebSphere for z/OS V6. You can use it from within the Administrative Console by selecting **Monitoring and Tuning** → **Performance Viewer** → **Current Activity** → **server**:

1. Click the server name to view the current activity for that specific server.
2. Select one or more servers from the list.
3. Click **Start Monitoring** to start the Tivoli Performance Monitor for the selected servers.

Figure 4-9 shows the panel for selecting servers for monitoring.


Tivoli Performance Viewer

Server selection for Tivoli Performance Viewer.

⊞ Preferences

Select	Server	Node	Type	Version	Collection Status
✓	ws6552	nd6552	servers	6.0.1.2	Inactive

Figure 4-9 Start Tivoli Performance Viewer in Administrative Console

The performance viewer consists of two panels: the Resource Selection panel and the Data Monitoring panel. The Resource Selection panel, located on the left, provides a view of resources for which performance data can be displayed. The Data Monitoring panel located on the right, displays numeric and statistical data for the resources in the Resource Selection panel.

4.7.2 Tivoli Performance Viewer output and its interpretation

In Tivoli Performance Viewer you can see what is happening with the WebSphere for z/OS servers in real time. You can see, for example, if the number of concurrent threads in the Web container pool is high enough or if you must add more.

You must analyze the modules shown in Table 4-2 to identify problems in your WebSphere for z/OS configuration.

Table 4-2 Modules and description to verify in Tivoli Performance Viewer

Modules	**Description**
Average response time	Includes statistics such as servlet or enterprise beans response time
Number of request	Enables understanding of how much traffic is processed by WebSphere for z/OS, thus helping determine the capacity to manage
Web and EJB Thread Pools Database and connection pool size	Interpret together because these thread pools might constrain performance due to their size
JVM Memory	Use to understand the JVM heap dynamics, including the frequency of garbage collection

Figure 4-10 shows the Tivoli Performance Viewer integrated in the Administrative Console.

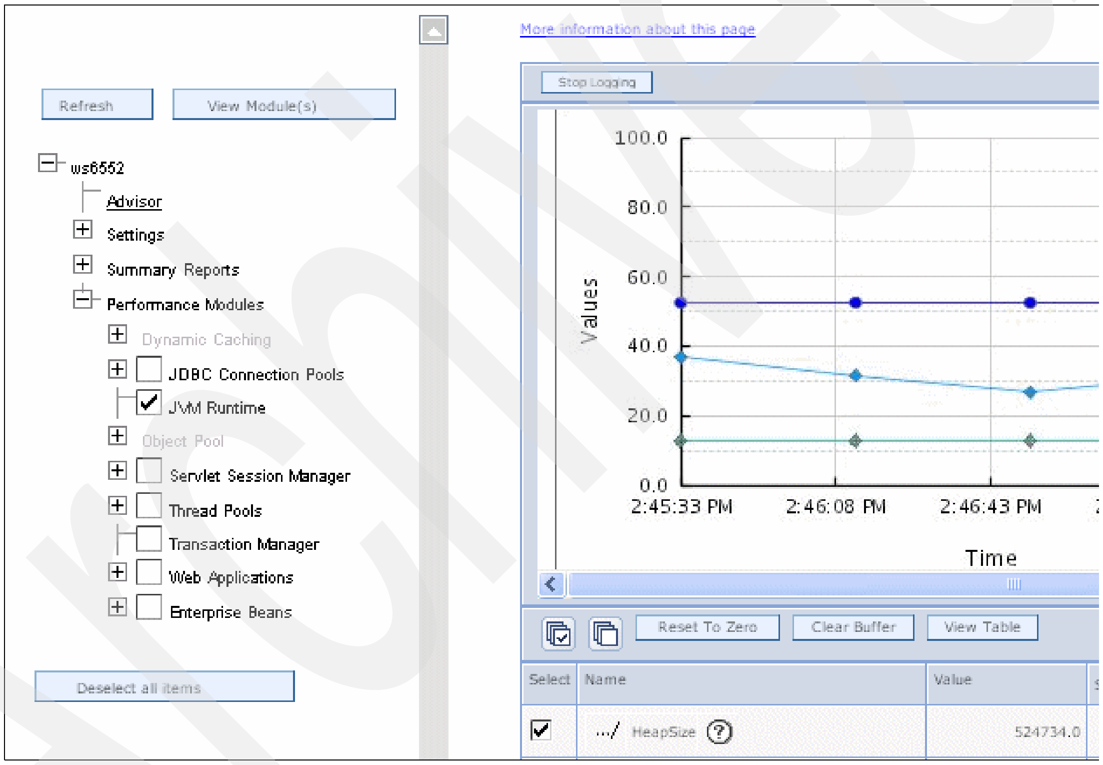


Figure 4-10 Tivoli Performance Viewer

For more detailed information about all available parameters, refer to the “Tivoli Performance Viewer” topic in WebSphere for z/OS Information Center, available at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

4.8 Omegamon XE for WebSphere

IBM Tivoli Omegamon XE for WebSphere for z/OS is a performance and availability monitoring tool. It provides real-time and historical management of WebSphere resources. It lets you monitor the performance of specific servlets, Java Server Pages (JSP™), and EJBs and take direct action to tune performance.

For further details about Omegamon XE for WebSphere Application Server on z/OS see:

http://www.ibm.com/software/tivoli/products/omegamon-xe-was-zos/

5

Other helpful tools

In this chapter, we describe other tools that we found very useful so that interested system administrators and application programmers for z/OS can have a quick reference guide to these tools. Although they are not directly related to problem determination for WebSphere Application Server for z/OS, the tools can be very powerful for performing day-to-day administrative tasks and problem determination for WebSphere for z/OS.

We describe TCP/IP-related tools such as TCP/IP checkout program (InetInfo.java) and TCP/IP network packet tracing with Ethereal. We also introduce MXI.

To avoid problems caused by bottlenecks or programming issues, the authors recommend that you run load and stress tests with various tools for your applications and for your WebSphere for z/OS environment before you go into production. Examples of these tools include:

- System Management Facilities Record Interpreter and Browser
- WebSphere Studio Workload Simulator for z/OS and OS/390,
- Microsoft Web Application Stress tool

For the day-to-day administration tasks for WebSphere for z/OS, you also need powerful and efficient tools for managing Java, XML, or HTML files. Therefore, we finish the chapter by introducing FTP, Telnet, and editor tools such as

- TeraTerm Pro
- Microsoft Windows FTP client
- WS_FTP Professional
- Directing SYSPRINT output to an HFS File
- UltraEdit

5.1 TCP/IP related tools

Some of the TCP/IP-based tools can be very powerful for performing day-to-day administrative tasks and problem determination for WebSphere for z/OS. We describe the TCP/IP checkout program (InetInfo.java) and TCP/IP network packet tracing in this section.

It is always useful to know if the client actually sent what is expected and received what was sent from the server. The same is true for the server side.

Usually, using tracing at the application server and container level in order is sufficient to obtain this message content. However, there are times when it is not possible to be certain, that is, if the server does not receive the request that was sent from the client. In these scenarios, the network packet tracer is an effective tool to apply.

TCP/IP packet tracing and analysis is a common approach for problem diagnosis at a network transport level. Various tools and techniques are available for handling all kinds of protocols and formats. These tools are usually provided as an operating system utility such as the TCP/IP for z/OS packet trace facility. Windows workstations do not have such a utility included in the OS, but various freeware utility packages are available, such as Ethereal[1].

5.1.1 TCP/IP checkout program (InetInfo.java)

InetInfo.java is a Java program that performs the following functions to display the Internet addressability for the host system:

getLocalHost	Obtain host IP address
getHostName	Using previously obtained IP address
getHostAddress	Using previously obtained host name

To use this tool, download the InetInfo.java program to your working directory in z/OS UNIX System Services (USS). Compile the Java program into the class file as follows:

```
>/usr/lpp/java/J1.4/bin/javac InetInfo.java
```

When you run the Java class, you get the following results, as shown in Figure 5-1:

1. The function getLocalHost returns an IP address.
2. The function getHostName by address returns a host name.
3. The function getHostAddress by name returns the IP address correctly.

```
>/Z16RA1/usr/lpp/java/J1.4/bin/javac InetInfo.java
>/Z16RA1/usr/lpp/java/J1.4/bin/java InetInfo

get Local Host
  IP Address: 9.12.4.28

get Host Name By Address using 9.12.4.28
   Host Name: wtsc55.itso.ibm.com

get Host Address By Name using wtsc55.itso.ibm.com
Host Address: 9.12.4.28
```

Figure 5-1 InetInfo.java program output

[1] Ethereal is a registered trademark of Ethereal, Inc.

If any of the functions fail, the following message is displayed:

```
Unknown Host, result: <returned message>
```

where <returned message> is the reason for the exception.

You can copy or download this Java program from the Techdoc *Java program to test TCP/IP setup - InetInfo.java*, TD100609, available at:

http://www.ibm.com/support/techdocs/

5.1.2 TCP/IP network packet tracing with Ethereal

Ethereal is a network packet analyzer that is used for displaying and analyzing capture packet data as detailed as possible with a GUI or command-line interface. Ethereal is open source software under the GPL. Ethereal is available for UNIX, Linux, Windows, and Apple workstation environments.

Ethereal has the following attractive features:

- Analyses data is captured in real time “off the wire” or read from a capture file.
- Real-time data can be from various network hardware devices for Ethernet, FDDI, PPP, Token-Ring, IEEE 802.11, Classical IP over ATM, and loopback interfaces.
- Data capture files can be from other tools such as UNIX tcpdum, Microsoft Network Monitor, and other hardware sniffers.
- Currently, it is able to dissect up to 706 network protocols.

To use Ethereal in a Microsoft Windows environment, you must download the Ethereal and WinPcap software packages from Ethereal Web site and install them on your target workstation. Ethereal is a network packet analyzer and WinPcap is a network packet captor for Windows. WinPcap is only required if you also want to capture data on your workstation.

To use the Ethereal GUI program to start capturing packet data and analyzing what is captured, follow these steps:

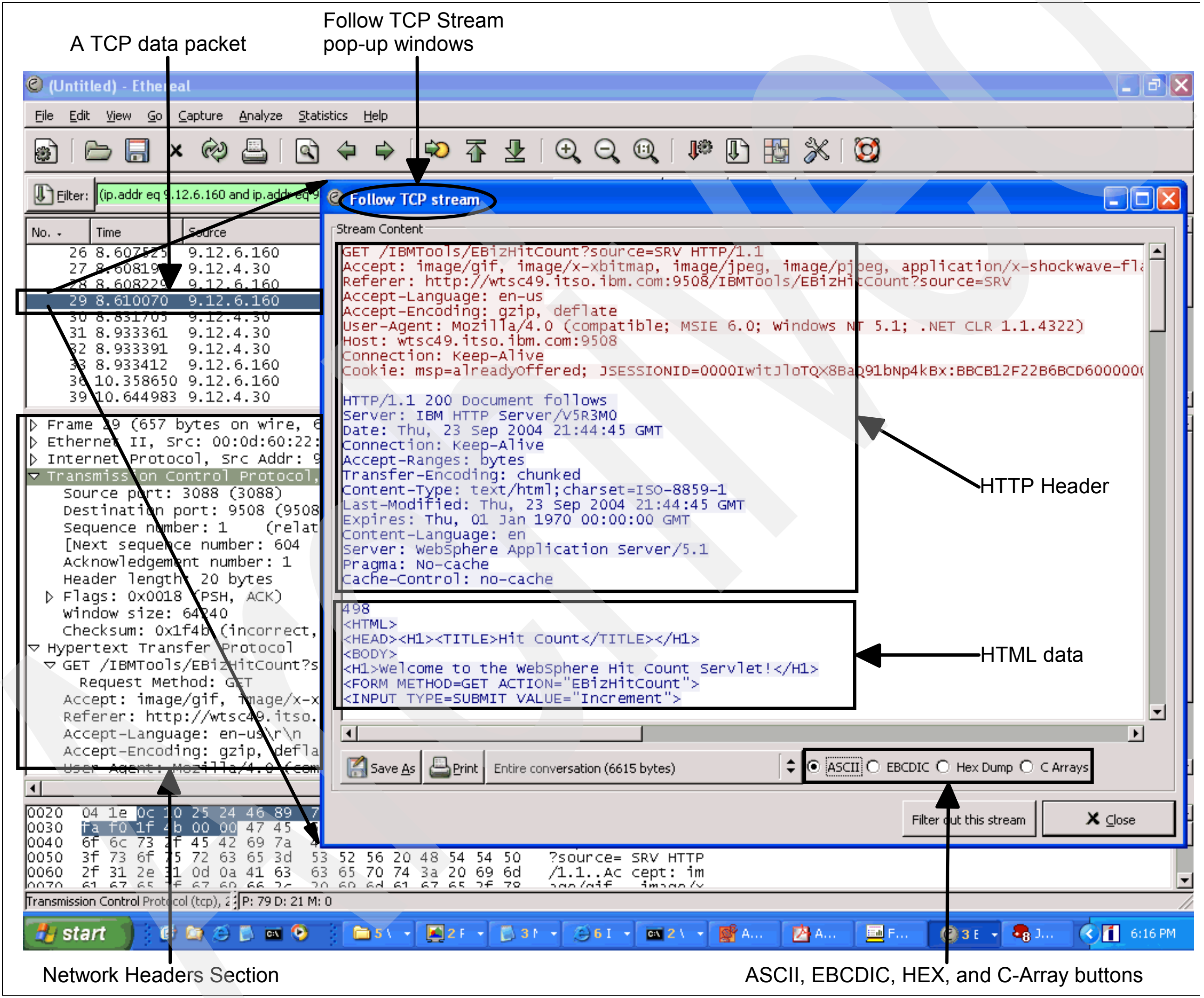
1. Start your Ethereal program. Select **Start** → **Programs** → **Ethereal** → **Ethereal**.
2. Select **Capture** → **Start**.
3. This opens a Capture Options dialog box. Make sure that you select the correct interface card if you have more than one. Then, click **OK**. Another dialog box opens that shows you some capturing statistics. There is also a Stop button for stopping the capture.
4. Recreate the problem scenario, using the Web browser to access a page.
5. To stop capturing, click the Stop button on the Capture Info dialog box.

After the data is captured, you receive a list of network data packets. Ethereal provides many ways to analyze the captured traces. For example, you can:

- Control the amount of data displayed by selecting the display that interests you:
  - Attribute columns (hardware address, IP address, port, and so on)
  - Format of times (absolute, relative, delta, and so on)
  - Sort the list by any column
  - Apply your own display filter for selecting packet data
- Use the statistics to gain many kinds of higher perspective views, for example:
  - Overall summary of transmission
  - Summary by different endpoints (by hardware, IP, and so on)
  - Summary by protocol hierarchy

- Look into the formatted display of each data packet. Ethereal formats all kinds of protocol headers according to standard specifications.
- One of the most powerful tools is a feature called “Follow TCP Stream.” When looking at a TCP data packet, you can instruct Ethereal to analyze the user data portion of the packet according to a higher-level application protocol specification such as HTTP. The data is formatted according to the HTTP header into separate sections inside other windows.
- In the data windows, you can display in ASCII, EBCDIC, hexadecimal, or C-array.

Figure 5-2 shows an Ethereal example window analyzing a TCP data packet. The section display shows the details of all the formatted network headers of the TCP data packet. The window for the “Follow TCP Stream” shows the formatted HTTP header and HTML data.



Filter: (ip.addr eq 9.12.6.160 and ip.addr eq 9

```
No. -  Time       Source
   26 8.607525   9.12.6.160
   27 8.608197   9.12.4.30
   28 8.608229   9.12.6.160
   29 8.610070   9.12.6.160
   30 8.831705   9.12.4.30
   31 8.933361   9.12.4.30
   32 8.933391   9.12.4.30
   33 8.933412   9.12.6.160
   36 10.358650  9.12.6.160
   39 10.644983  9.12.4.30
```

```
▷ Frame 29 (657 bytes on wire, 6
▷ Ethernet II, Src: 00:0d:60:22:
▷ Internet Protocol, Src Addr: 9
▽ Transmission Control Protocol,
    Source port: 3088 (3088)
    Destination port: 9508 (9508
    Sequence number: 1    (relat
    [Next sequence number: 604
    Acknowledgement number: 1
    Header length: 20 bytes
  ▷ Flags: 0x0018 (PSH, ACK)
    Window size: 64240
    Checksum: 0x1f4b (incorrect,
▽ Hypertext Transfer Protocol
  ▽ GET /IBMTools/EBizHitCount?s
      Request Method: GET
    Accept: image/gif, image/x-x
    Referer: http://wtsc49.itso.
    Accept-Language: en-us\r\n
    Accept-Encoding: gzip, defla
    User-Agent: Mozilla/4.0 (com
```

```
0020   04 1e 0c 10 25 24 46 89  7
0030   fa f0 1f 4b 00 00 47 45  5
0040   6f 6c 73 2f 45 42 69 7a  4
0050   3f 73 6f 75 72 63 65 3d  53 52 56 20 48 54 54 50    ?source= SRV HTTP
0060   2f 31 2e 31 0d 0a 41 63  63 65 70 74 3a 20 69 6d    /1.1..Ac cept: im
```

Transmission Control Protocol (tcp), 2 P: 79 D: 21 M: 0

```
GET /IBMTools/EBizHitCount?source=SRV HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-fla
Referer: http://wtsc49.itso.ibm.com:9508/IBMTools/EBizHitCount?source=SRV
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Host: wtsc49.itso.ibm.com:9508
Connection: Keep-Alive
Cookie: msp=alreadyOffered; JSESSIONID=0000IwitJloTQX8BaQ91bNp4kBx:BBCB12F22B6BCD600000

HTTP/1.1 200 Document follows
Server: IBM HTTP Server/V5R3M0
Date: Thu, 23 Sep 2004 21:44:45 GMT
Connection: Keep-Alive
Accept-Ranges: bytes
Transfer-Encoding: chunked
Content-Type: text/html;charset=ISO-8859-1
Last-Modified: Thu, 23 Sep 2004 21:44:45 GMT
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Language: en
Server: WebSphere Application Server/5.1
Pragma: No-cache
Cache-Control: no-cache

498
<HTML>
<HEAD><H1><TITLE>Hit Count</TITLE></H1>
<BODY>
<H1>Welcome to the WebSphere Hit Count Servlet!</H1>
<FORM METHOD=GET ACTION="EBizHitCount">
<INPUT TYPE=SUBMIT VALUE="Increment">
```

Display: [✓ ASCII] EBCDIC Hex Dump C Arrays

Figure 5-2 Ethereal data analysis with Follow TCP Stream windows

For more informal information about Ethereal, see:

http://www.ethereal.com

The source code and installers for Windows, Red Hat Linux, Solaris™, IBM AIX, SuSE Linux, and more, can be found on this Web site.

http://www.ethereal.com/download.html

For more information about WinPcap, see:

http://www.winpcap.org

5.1.3 TCP/IP for z/OS packet trace

TCP/IP for z/OS packet trace is a z/OS facility for network packet capture and analysis. It uses a z/OS external writer to obtain component trace data for the TCP/IP stack, packet trace, and other stack information. After the data is captured, you can use the z/OS IPCS tool to format the trace data for further analysis.

To capture trace data:

1. Set up an external writer JCL procedure (in SYS1.PROCLIB) to be used by the TCP/IP component trace, as shown in Example 5-1.

Example 5-1 External writer

```
//CTWTRPD  PROC
//IEFPROC  EXEC PGM=ITTTRCWR,REGION=32M,TIME=1440
//TRCOUT01 DD  DSNAME=WAS5PD.SC49.CTRACE,DISP=OLD
//SYSPRINT DD  SYSOUT=*
```

2. Start the external writer using the following command:

   ```
   /trace ct,wtrstart=CTWTRPD
   ```

3. Start the TCP/IP packet trace with filtering to pick up only one IP address:

   ```
   /v tcpip,TCPIP,pkttrace,full,ip=9.12.6.160
   ```

4. When the system responds with a prompt, reply as follows:

   ```
   /r xx,WTR=CTWTRPD,end
   ```

5. Use the DISPLAY command to check the external writer status, as shown in Example 5-2.

Example 5-2 Display trace status command

```
/d trace,comp=systcpda,sub=(TCPIP)
RESPONSE=SC49
 IEE843I 15.02.26  TRACE DISPLAY 267
         SYSTEM STATUS INFORMATION
  ST=(ON,0256K,00512K) AS=ON  BR=OFF EX=ON  MT=(ON,024K)
   TRACENAME
   =========
   SYSTCPDA
                       MODE BUFFER HEAD SUBS
                       =====================
                       OFF         HEAD    1
      NO HEAD OPTIONS
   SUBTRACE            MODE BUFFER HEAD SUBS
  ------------------------------------------------------
TCPIP              ON   0016M
      ASIDS      *NONE*
      JOBNAMES   *NONE*
      OPTIONS    MINIMUM
      WRITER     CTWTRPD
```

6. Recreate the problem scenario using the Web browser to access a page.
7. Stop the packet trace:

   ```
   /trace ct,off,comp=systcpda,sub=(TCPIP)
   ```

8. Stop the external writer:

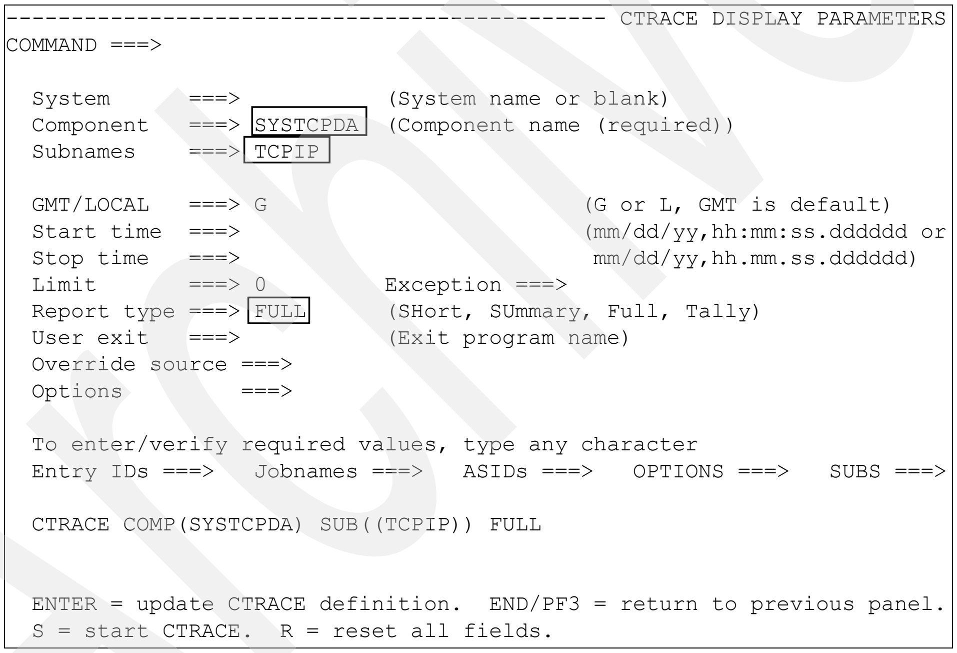
```
/trace ct,wtrstop=CTWTRPD
```

You use the IPCS utility to format the captured trace data into a user friendly format. During the format process, you have a choice of three levels of details: Summary, Short, and Full. Complete the following steps to format the trace data:

1. Access IPCS.
2. Select option `2` (ANALYSIS) from the option list.
3. Select option `0` (DEFAULT) from the option list and enter the trace data set name to be used as default source:

   ```
   Source  ==> DSNAME('WAS5PD.SC49.CTRACE')
   ```

   Press **PF3** to return to previous panel.
4. Select option `7` (TRACES) from the option list.
5. Select option `1` (CTRACE) from the option list.
6. Select option `D` (DISPLAY) from the option list. Enter the component name, subsystem name, and trace detail level as shown in Figure 5-3. To start formatting, type `S` on the command line and press Enter.



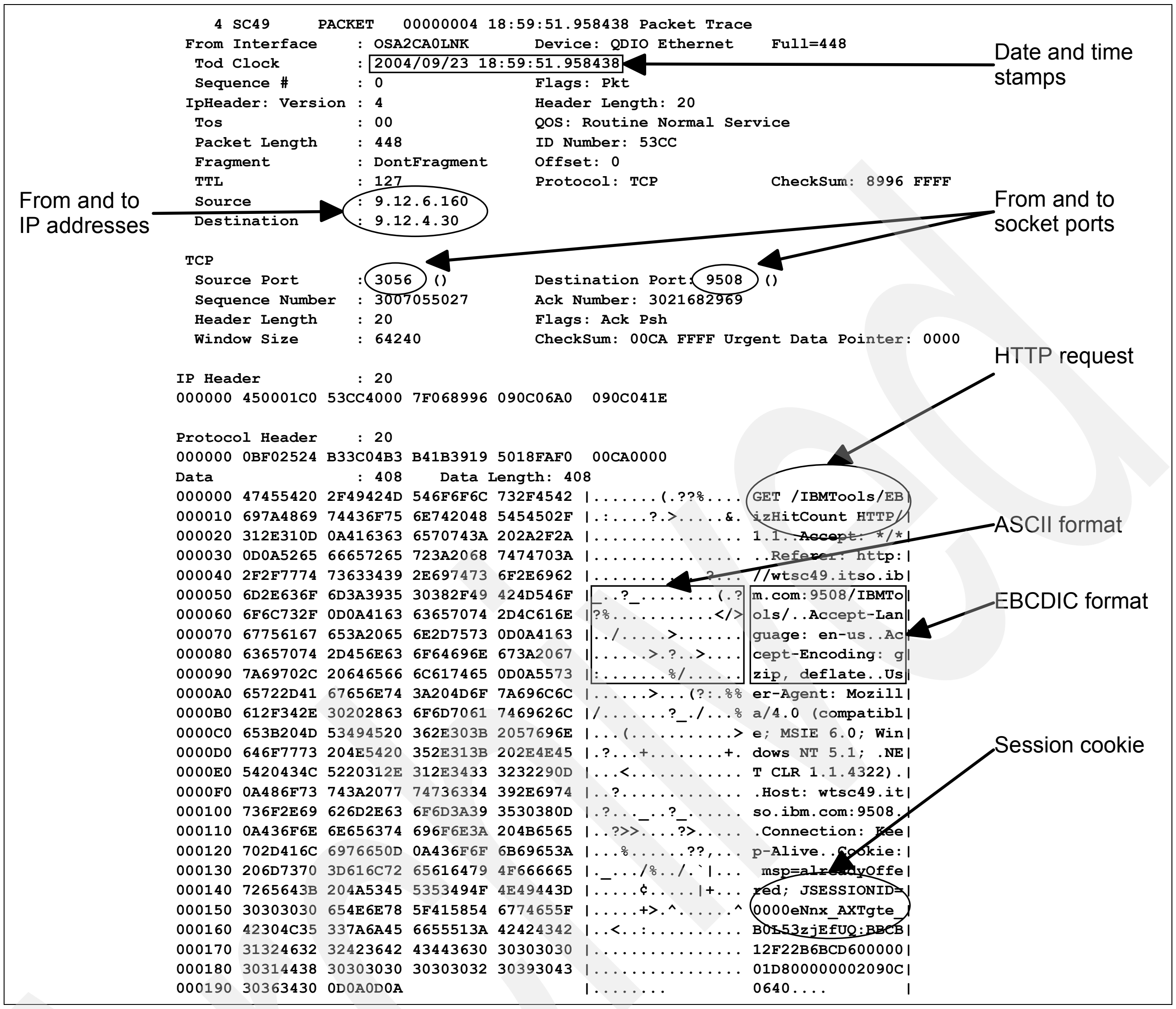
```
---------------------------------------------- CTRACE DISPLAY PARAMETERS
COMMAND ===>

  System      ===>             (System name or blank)
  Component   ===> SYSTCPDA    (Component name (required))
  Subnames    ===> TCPIP

  GMT/LOCAL   ===> G                          (G or L, GMT is default)
  Start time  ===>                            (mm/dd/yy,hh:mm:ss.dddddd or
  Stop time   ===>                             mm/dd/yy,hh.mm.ss.dddddd)
  Limit       ===> 0          Exception ===>
  Report type ===> FULL        (SHort, SUmmary, Full, Tally)
  User exit   ===>             (Exit program name)
  Override source ===>
  Options         ===>

  To enter/verify required values, type any character
  Entry IDs ===>   Jobnames ===>   ASIDs ===>   OPTIONS ===>   SUBS ===>

  CTRACE COMP(SYSTCPDA) SUB((TCPIP)) FULL


  ENTER = update CTRACE definition.  END/PF3 = return to previous panel.
  S = start CTRACE.  R = reset all fields.
```

Figure 5-3 IPCS CTRACE display parameters

Formatting the trace data with a FULL detail level results in information in the following sections:

- Interface device
- IP header
- TCP header
- Message data

Figure 5-4 on page 85 shows a generated report of one captured trace data packet. In the IP header, note the IP addresses (source and destination) and the date and timestamp. In the TCP header section, note the socket ports (source and destination).



```
   4 SC49     PACKET   00000004 18:59:51.958438 Packet Trace
 From Interface     : OSA2CA0LNK       Device: QDIO Ethernet    Full=448
  Tod Clock         : 2004/09/23 18:59:51.958438
  Sequence #        : 0                Flags: Pkt
 IpHeader: Version  : 4                Header Length: 20
  Tos               : 00               QOS: Routine Normal Service
  Packet Length     : 448              ID Number: 53CC
  Fragment          : DontFragment     Offset: 0
  TTL               : 127              Protocol: TCP           CheckSum: 8996 FFFF
  Source            : 9.12.6.160
  Destination       : 9.12.4.30

 TCP
  Source Port       : 3056 ()          Destination Port: 9508 ()
  Sequence Number   : 3007055027       Ack Number: 3021682969
  Header Length     : 20               Flags: Ack Psh
  Window Size       : 64240            CheckSum: 00CA FFFF Urgent Data Pointer: 0000

IP Header           : 20
000000 450001C0 53CC4000 7F068996 090C06A0  090C041E

Protocol Header     : 20
000000 0BF02524 B33C04B3 B41B3919 5018FAF0  00CA0000
Data                : 408    Data Length: 408
000000 47455420 2F49424D 546F6F6C 732F4542 |.......(.??%.... GET /IBMTools/EB|
000010 697A4869 74436F75 6E742048 5454502F |.:....?.>.....&. izHitCount HTTP/|
000020 312E310D 0A416363 6570743A 202A2F2A |................ 1.1..Accept: */*|
000030 0D0A5265 66657265 723A2068 7474703A |................ ..Referer: http:|
000040 2F2F7774 73633439 2E697473 6F2E6962 |..........?..... //wtsc49.itso.ib|
000050 6D2E636F 6D3A3935 30382F49 424D546F |_..?_........(.? m.com:9508/IBMTo|
000060 6F6C732F 0D0A4163 63657074 2D4C616E |?%...........</> ols/..Accept-Lan|
000070 67756167 653A2065 6E2D7573 0D0A4163 |../.....>....... guage: en-us..Ac|
000080 63657074 2D456E63 6F64696E 673A2067 |......>.?..>.... cept-Encoding: g|
000090 7A69702C 20646566 6C617465 0D0A5573 |:.......%/...... zip, deflate..Us|
0000A0 65722D41 67656E74 3A204D6F 7A696C6C |......>...(?:.%% er-Agent: Mozill|
0000B0 612F342E 30202863 6F6D7061 7469626C |/.......?_./...% a/4.0 (compatibl|
0000C0 653B204D 53494520 362E303B 2057696E |...(...........> e; MSIE 6.0; Win|
0000D0 646F7773 204E5420 352E313B 202E4E45 |.?...+........+. dows NT 5.1; .NE|
0000E0 5420434C 5220312E 312E3433 3232290D |...<............ T CLR 1.1.4322).|
0000F0 0A486F73 743A2077 74736334 392E6974 |..?............. .Host: wtsc49.it|
000100 736F2E69 626D2E63 6F6D3A39 3530380D |.?..._..?_...... so.ibm.com:9508.|
000110 0A436F6E 6E656374 696F6E3A 204B6565 |..?>>....?>..... .Connection: Kee|
000120 702D416C 6976650D 0A436F6F 6B69653A |...%......??,... p-Alive..Cookie:|
000130 206D7370 3D616C72 65616479 4F666665 |._.../%../.`|...  msp=alreadyOffe|
000140 7265643B 204A5345 5353494F 4E49443D |.....¢.....|+... red; JSESSIONID=|
000150 30303030 654E6E78 5F415854 6774655F |.....+>.^......^ 0000eNnx_AXTgte_|
000160 42304C35 337A6A45 6655513A 42424342 |..<..:.......... B0L53zjEfUQ:BBCB|
000170 31324632 32423642 43443630 30303030 |................ 12F22B6BCD600000|
000180 30314438 30303030 30303032 30393043 |................ 01D80000002090C|
000190 30363430 0D0A0D0A                   |........         0640....        |
```

Figure 5-4 TCP/IP network packet trace report

For more information about the TCP/IP for z/OS packet trace, see *z/OS V1R5.0 Communication Server: IP Diagnosis Guide*, GC31-8782.

For more information about the IPCS tool, see *OS/390 V2R10.0 MVS Interactive Problem Control System (IPCS) User's Guide*, GC28-1756.

5.2 MVS Extended Information

MVS Extended Information (MXI) is an ISPF-based application that displays important configuration information about active OS/390® or z/OS systems. MXI is free software that is available from the following site:

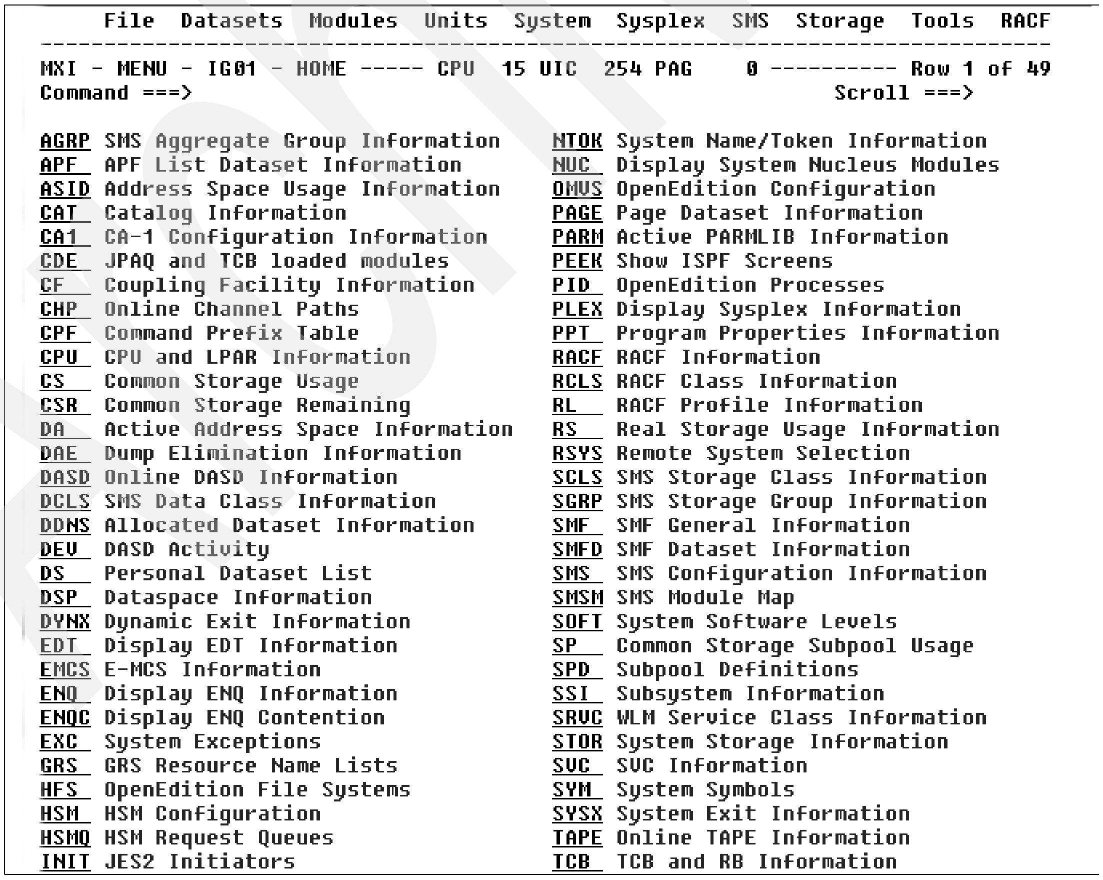
http://www.rocketsoftware.com/portfolio/mxi

While mainly used online, MXI also has a REXX interface and can be run in batch mode. It has a TCP/IP server application for issuing commands to remote systems and viewing the results locally.

MXI can display a wealth of information from your system, including:

- APF, LNKLST, and LPA data sets
- Active address spaces and ASVT slot usage
- Allocated data sets for any address space
- Master and user catalogs
- Common storage usage by address space or subpool
- Orphaned common storage
- Cross-memory connections
- CPU and LPAR information
- Online DASD and tape units
- Enqueue requests and contention
- HSM request queues
- ISPF screen images of any user
- LLA module statistics
- Memory contents of any address space
- Memory delete queue
- Real and auxiliary storage usage
- SMS classes
- SMS storage groups
- Subsystems
- SVCs and PC routines
- Sysplex information
- WLM information

Figure 5-5 shows an excerpt of the MXI Primary Option menu.



```
      File  Datasets  Modules  Units  System  Sysplex  SMS  Storage  Tools  RACF
--------------------------------------------------------------------------------
MXI - MENU - IG01 - HOME ----- CPU  15 UIC  254 PAG    0 ---------- Row 1 of 49
Command ===>                                                         Scroll ===>

AGRP SMS Aggregate Group Information    NTOK System Name/Token Information
APF  APF List Dataset Information       NUC  Display System Nucleus Modules
ASID Address Space Usage Information    OMVS OpenEdition Configuration
CAT  Catalog Information                PAGE Page Dataset Information
CA1  CA-1 Configuration Information     PARM Active PARMLIB Information
CDE  JPAQ and TCB loaded modules        PEEK Show ISPF Screens
CF   Coupling Facility Information      PID  OpenEdition Processes
CHP  Online Channel Paths               PLEX Display Sysplex Information
CPF  Command Prefix Table               PPT  Program Properties Information
CPU  CPU and LPAR Information           RACF RACF Information
CS   Common Storage Usage               RCLS RACF Class Information
CSR  Common Storage Remaining           RL   RACF Profile Information
DA   Active Address Space Information   RS   Real Storage Usage Information
DAE  Dump Elimination Information       RSYS Remote System Selection
DASD Online DASD Information            SCLS SMS Storage Class Information
DCLS SMS Data Class Information         SGRP SMS Storage Group Information
DDNS Allocated Dataset Information      SMF  SMF General Information
DEV  DASD Activity                      SMFD SMF Dataset Information
DS   Personal Dataset List              SMS  SMS Configuration Information
DSP  Dataspace Information              SMSM SMS Module Map
DYNX Dynamic Exit Information           SOFT System Software Levels
EDT  Display EDT Information            SP   Common Storage Subpool Usage
EMCS E-MCS Information                  SPD  Subpool Definitions
ENQ  Display ENQ Information            SSI  Subsystem Information
ENQC Display ENQ Contention             SRVC WLM Service Class Information
EXC  System Exceptions                  STOR System Storage Information
GRS  GRS Resource Name Lists            SVC  SVC Information
HFS  OpenEdition File Systems           SYM  System Symbols
HSM  HSM Configuration                  SYSX System Exit Information
HSMQ HSM Request Queues                 TAPE Online TAPE Information
INIT JES2 Initiators                    TCB  TCB and RB Information
```

Figure 5-5 MXI Primary Option menu

Most of the displays can be filtered using ISPF-like masking characters, and many display fields have “point-and-shoot” functionality that drills down to a more detailed display.

5.3 Stress test tools

Although load and stress tests are not part of the problem determination processes in this book, we describe them in this section. This is because some problems only occur when you test the applications in WebSphere for z/OS under load (stress). To avoid problems caused by bottlenecks or programming issues, we recommend that you run load and stress tests with various tools for your applications and for your WebSphere for z/OS environment before you go into production. Consider the following tools for this process:

- System Management Facilities Record Interpreter and Browser
- WebSphere Studio Workload Simulator for z/OS and OS/390
- Microsoft Web Application Stress tool

5.3.1 System Management Facilities Record Interpreter and Browser

The System Management Facilities (SMF) can be used to collect performance and accounting information. SMF data recording can be configured individually and must be enabled.

To make that you are not collecting more SMF data than you need, review SMFPRMxx to ensure that only the minimum number of records are being collected. Use SMF 92 or 120 only for diagnostics. SMF 92 records are created each time an HFS file is opened, closed, deleted, and so forth. Almost every Web server request references HFS files, so thousands of SMF 92 records are created. Unless you specifically need this information, turn off SMF 92 records.

You might find that running SMF 120 records in production is appropriate, because these records provide information that is specific to WebSphere applications, such as response time for J2EE artifacts and bytes transferred. If you do choose to run with SMF 120 records enabled, we recommend that you use the server interval SMF records and container interval SMF records rather than the server activity records and container activity records.

The Java SMF Record Interpreter is provided in the form of a JAR file named bbomsmfv.jar. To use this file from the z/OS or OS/390 UNIX environment, follow these steps:

1. Verify that the JAVA_HOME environment variable refers to the current Java installation, for example, `JAVA_HOME=../usr/bin/java/J1.3`. This should be at least Java 1.3, because this release is the first to implicitly contain the necessary record support needed by the interpreter.
2. Download the SMF Record Interpreter from the WebSphere for z/OS Web site at:

   http://www.ibm.com/software/webservers/appserv/zos_os390/index.html
3. Copy the file bbomsmfv.jar to your tools directory. Be sure that any edits made to the file in the future are made to both copies of the file, or simply run them from the installation directory.
4. To interpret SMF data from a cataloged z/OS or OS/390 sequential file, issue:

   ```
   java -cp bbomsmfv.jar com.ibm.ws390.sm.smfview.Interpreter “USER.SMFDATA”
   ```

   It is implicit in the Java command parameterization that your current working directory is the tools directory. If this is not the case, you will receive a NoClassDefFoundError on com.ibm.ws390.sm.smfview.Interpreter. Java does not generate a diagnostic when it does not find the bbomsmfv.jar file in the current directory.

The SMF ViewTool has been successfully installed and invoked when you do not receive any Java error messages after the invocation and the browser output is shown on the screen.

The SMF Browser, available on the WebSphere for z/OS download site, is able to display SMF record type 120. To download the SMF Browser, go to the following Web site (requires registration):

http://www6.software.ibm.com/dl/websphere20/zosos390-p

For more information about the SMF Browser, download the browser package and read the associated documentation and see *Performance Summary Report for SMF 120 records from WAS V.5 for z/OS*, PRS752, at the WebSphere for z/OS Support page:

http://www.ibm.com/software/webservers/appserv/zos_os390/support/

Another SMF reporting tool is the Tivoli Decision Support for OS/390 plus System Performance Feature Version 1.6, Program Number 5695-101 (the Performance Reporter for MVS, SLR, and EPDM).

This can be used to collect systems management data into a DB2 database from SMF, including SMF 120 records for WebSphere Application Server V5.1 for z/OS , and to generate graphic and tabular reports from its DB2 database.

5.3.2 WebSphere Studio Workload Simulator for z/OS and OS/390

WebSphere Studio Workload Simulator is an automated test tool that simulates the numbers of Web browser users or “virtual users” and generates Web traffic to test Web applications and Web servers such as WebSphere.

WebSphere Studio Workload Simulator consists of two components:

- Controller
- Engine

The WebSphere Studio Workload Simulator Controller is installed on Windows-based hardware. It provides the control function and monitor capability for WebSphere Studio Workload Simulator.

The WebSphere Studio Workload Simulator Engine is installed on an IBM @server zSeries server. It is a UNIX daemon that acts as a load generator and runs as a started task. It receives instructions from the monitor, generates the HTTP requests for the simulation, sends them to the Web server, and then returns statistics to the monitor at the end of the run.

Setting up the Workload Simulator

To set up a workload simulation using WebSphere Studio Workload Simulator, follow these steps:

1. Create or record a test script for WebSphere Studio Workload Simulator. A user can record a script using a Windows GUI for WebSphere Studio Workload Simulator. The script is series of HTTP operations that the engine on z/OS uses to run the simulation. To create a record of a test script:

   a. Select **File** → **New** → **Capture**. The capture session starts. A pop-up window and a browser window open. See Figure 5-6 on page 89.

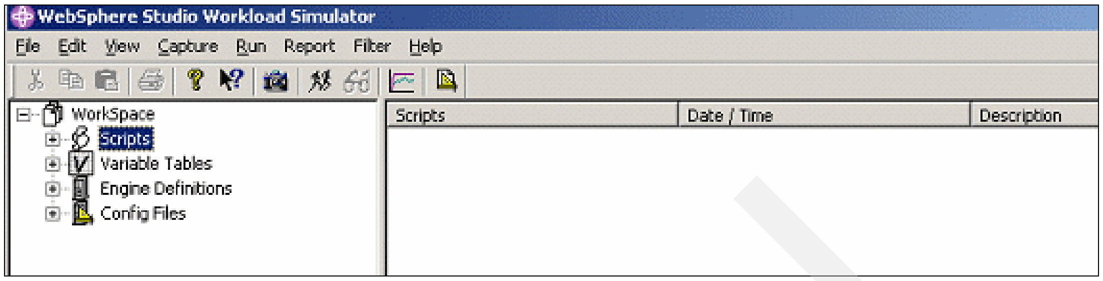


Figure 5-6 WebSphere Studio Workload Simulator window

b. In the browser, type in the URL of the Web site from which you want to capture session data. Click **Start** in your Capture window to begin recording a script, as shown in Figure 5-7. The capture session starts and the data stream for the Web session is recorded.

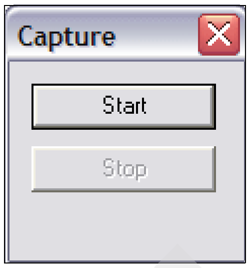
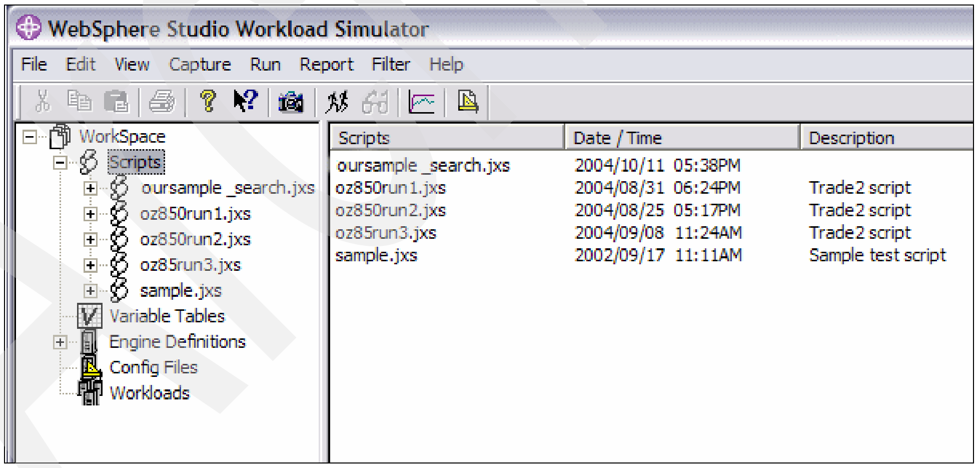


Figure 5-7 Pop-up window to start recording

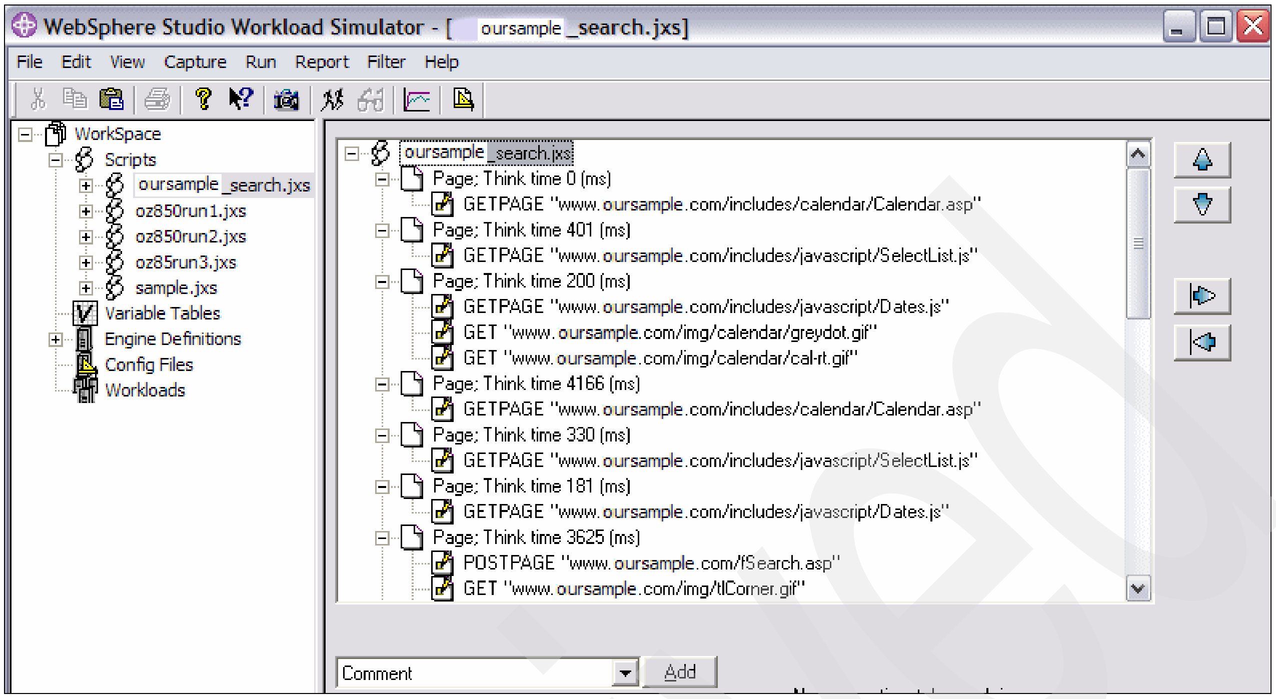
c. Click **Stop** to end the recording. When the capture session ends, WebSphere Studio Workload Simulator prompts you to enter a script name and description. See Figure 5-8.



Scripts	Date / Time	Description
oursample _search.jxs	2004/10/11 05:38PM	
oz850run1.jxs	2004/08/31 06:24PM	Trade2 script
oz850run2.jxs	2004/08/25 05:17PM	Trade2 script
oz85run3.jxs	2004/09/08 11:24AM	Trade2 script
sample.jxs	2002/09/17 11:11AM	Sample test script

Figure 5-8 WebSphere Studio Workload Simulator with scripts of captured sessions

d. The script shows a list of HTTP interactions (Web session elements) that make up the script. You can edit or change the value of these interactions, as shown in Figure 5-9 on page 90.


WebSphere Studio Workload Simulator - [oursample _search.jxs]

File Edit View Capture Run Report Filter Help

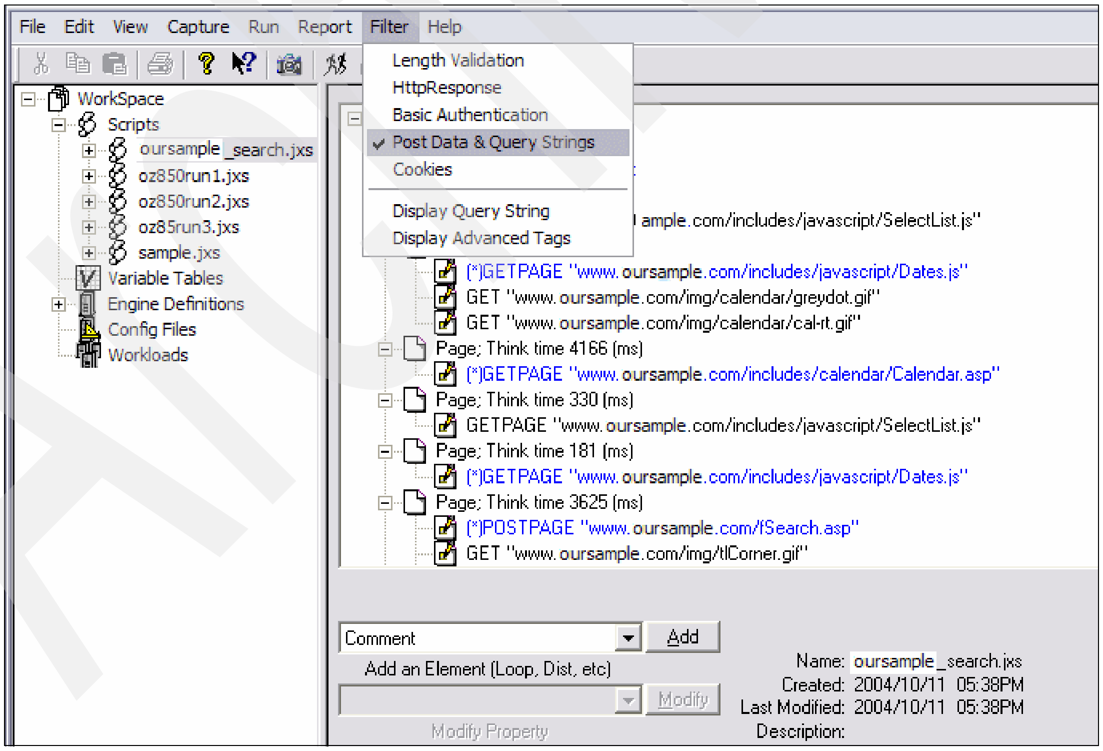
```
WorkSpace
  Scripts
    oursample _search.jxs
    oz850run1.jxs
    oz850run2.jxs
    oz85run3.jxs
    sample.jxs
  Variable Tables
  Engine Definitions
  Config Files
  Workloads
```

```
oursample _search.jxs
  Page; Think time 0 (ms)
    GETPAGE "www.oursample.com/includes/calendar/Calendar.asp"
  Page; Think time 401 (ms)
    GETPAGE "www.oursample.com/includes/javascript/SelectList.js"
  Page; Think time 200 (ms)
    GETPAGE "www.oursample.com/includes/javascript/Dates.js"
    GET "www.oursample.com/img/calendar/greydot.gif"
    GET "www.oursample.com/img/calendar/cal-rt.gif"
  Page; Think time 4166 (ms)
    GETPAGE "www.oursample.com/includes/calendar/Calendar.asp"
  Page; Think time 330 (ms)
    GETPAGE "www.oursample.com/includes/javascript/SelectList.js"
  Page; Think time 181 (ms)
    GETPAGE "www.oursample.com/includes/javascript/Dates.js"
  Page; Think time 3625 (ms)
    POSTPAGE "www.oursample.com/fSearch.asp"
    GET "www.oursample.com/img/tlCorner.gif"
```

Comment Add

Figure 5-9 WebSphere Studio Workload Simulator window: Web session elements

e. Variable elements in the script are revealed through a filter as shown in Figure 5-10.


File Edit View Capture Run Report Filter Help

```
WorkSpace
  Scripts
    oursample _search.jxs
    oz850run1.jxs
    oz850run2.jxs
    oz85run3.jxs
    sample.jxs
  Variable Tables
  Engine Definitions
  Config Files
  Workloads
```

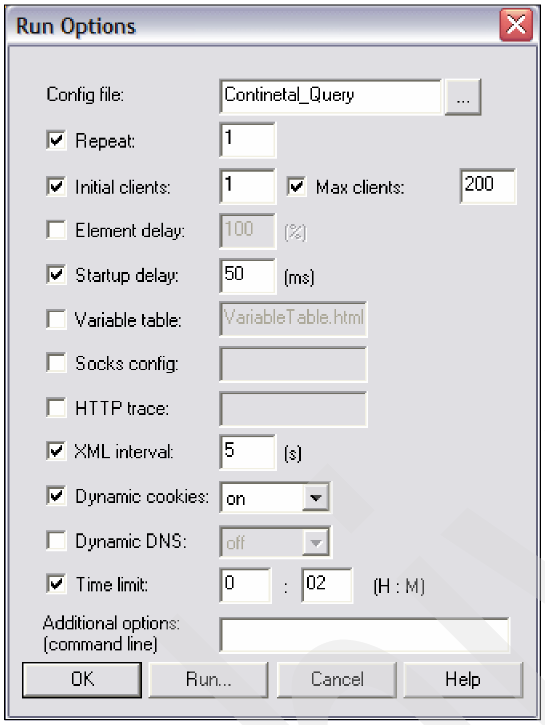
```
    ample.com/includes/javascript/SelectList.js"
    [*]GETPAGE "www.oursample.com/includes/javascript/Dates.js"
    GET "www.oursample.com/img/calendar/greydot.gif"
    GET "www.oursample.com/img/calendar/cal-rt.gif"
  Page; Think time 4166 (ms)
    [*]GETPAGE "www.oursample.com/includes/calendar/Calendar.asp"
  Page; Think time 330 (ms)
    GETPAGE "www.oursample.com/includes/javascript/SelectList.js"
  Page; Think time 181 (ms)
    [*]GETPAGE "www.oursample.com/includes/javascript/Dates.js"
  Page; Think time 3625 (ms)
    [*]POSTPAGE "www.oursample.com/fSearch.asp"
    GET "www.oursample.com/img/tlCorner.gif"
```

Comment Add
Add an Element (Loop, Dist, etc)
Modify
Modify Property

Name: oursample _search.jxs
Created: 2004/10/11 05:38PM
Last Modified: 2004/10/11 05:38PM
Description:

Figure 5-10 Variable elements through a filter

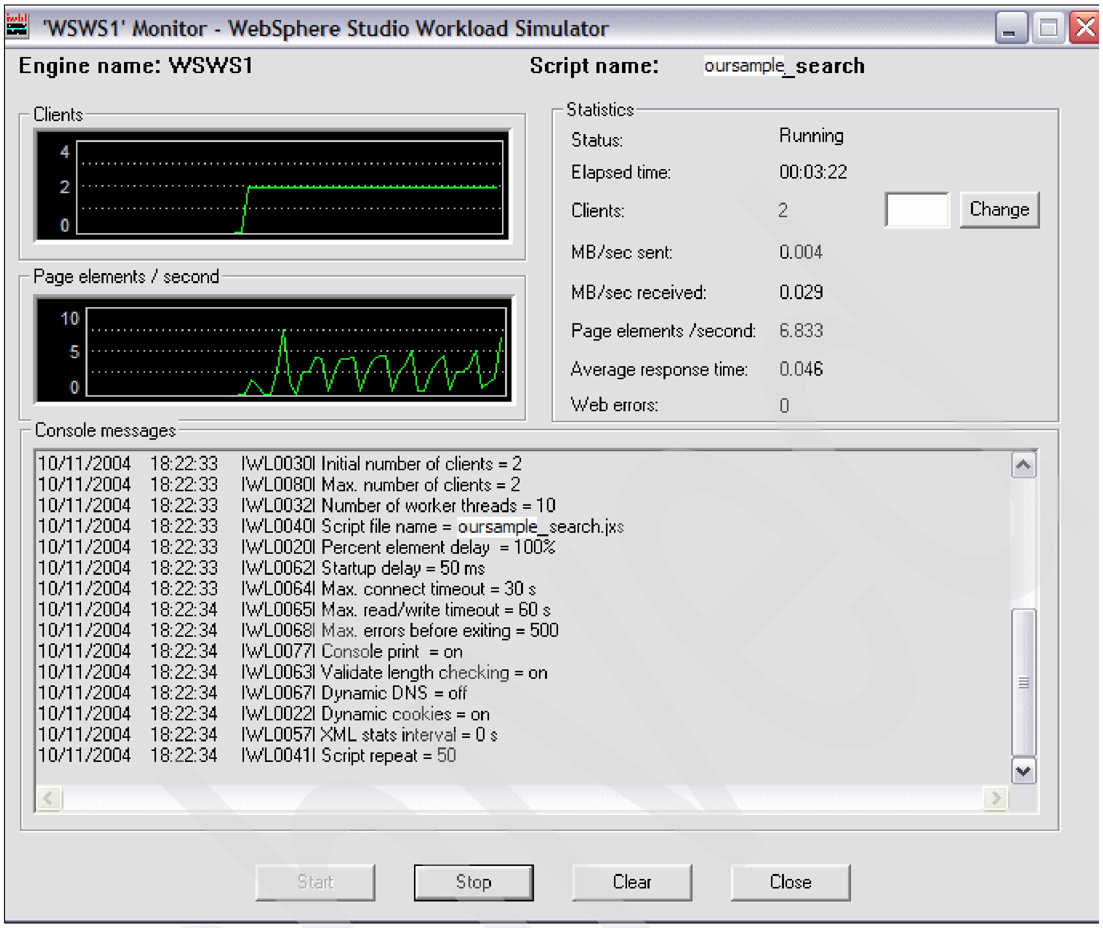
2. Set various runtime parameters, such as number of clients, number of times to repeat the script, delay controls, turn dynamic cookies on or off, a time limit for the test, HTTP trace, and Socks support before executing the script as shown in Figure 5-11. The runtime parameters can be saved in a configuration file for reuse.



Config file:	Continetal_Query
[✓] Repeat:	1
[✓] Initial clients:	1
[✓] Max clients:	200
[ ] Element delay:	100 (%)
[✓] Startup delay:	50 (ms)
[ ] Variable table:	VariableTable.html
[ ] Socks config:	
[ ] HTTP trace:	
[✓] XML interval:	5 (s)
[✓] Dynamic cookies:	on
[ ] Dynamic DNS:	off
[✓] Time limit:	0 : 02 (H : M)
Additional options: (command line)	

Figure 5-11 Various runtime parameters

3. Run the script and monitor the test. When the script runs, the test engine can be monitored in real time through a Windows GUI, as shown in Figure 5-12 on page 92.



```
10/11/2004   18:22:33   IWL0030I Initial number of clients = 2
10/11/2004   18:22:33   IWL0080I Max. number of clients = 2
10/11/2004   18:22:33   IWL0032I Number of worker threads = 10
10/11/2004   18:22:33   IWL0040I Script file name = oursample_search.jxs
10/11/2004   18:22:33   IWL0020I Percent element delay  = 100%
10/11/2004   18:22:33   IWL0062I Startup delay = 50 ms
10/11/2004   18:22:33   IWL0064I Max. connect timeout = 30 s
10/11/2004   18:22:34   IWL0065I Max. read/write timeout = 60 s
10/11/2004   18:22:34   IWL0068I Max. errors before exiting = 500
10/11/2004   18:22:34   IWL0077I Console print  = on
10/11/2004   18:22:34   IWL0063I Validate length checking = on
10/11/2004   18:22:34   IWL0067I Dynamic DNS  = off
10/11/2004   18:22:34   IWL0022I Dynamic cookies = on
10/11/2004   18:22:34   IWL0057I XML stats interval = 0 s
10/11/2004   18:22:34   IWL0041I Script repeat = 50
```

Figure 5-12   WebSphere Studio Workload Simulator Monitor GUI

Workload Simulator output and its interpretation

When the run finishes, reports are saved in the hierarchical file system on z/OS. Run statistics are recorded in the form of an XML record. These records can be used by the monitor to graph results.

There is also a log file that shows messages that are generated by the test engine during operation. This can be useful if you need to debug engine problems.

The results of the simulation are displayed as a graph for analysis as shown in Figure 5-13 on page 93. You can see various graphs for performance measurements while it is running, such as CPU or memory utilization, response time, data read, page elements, transactions, or written transfer (throughput).

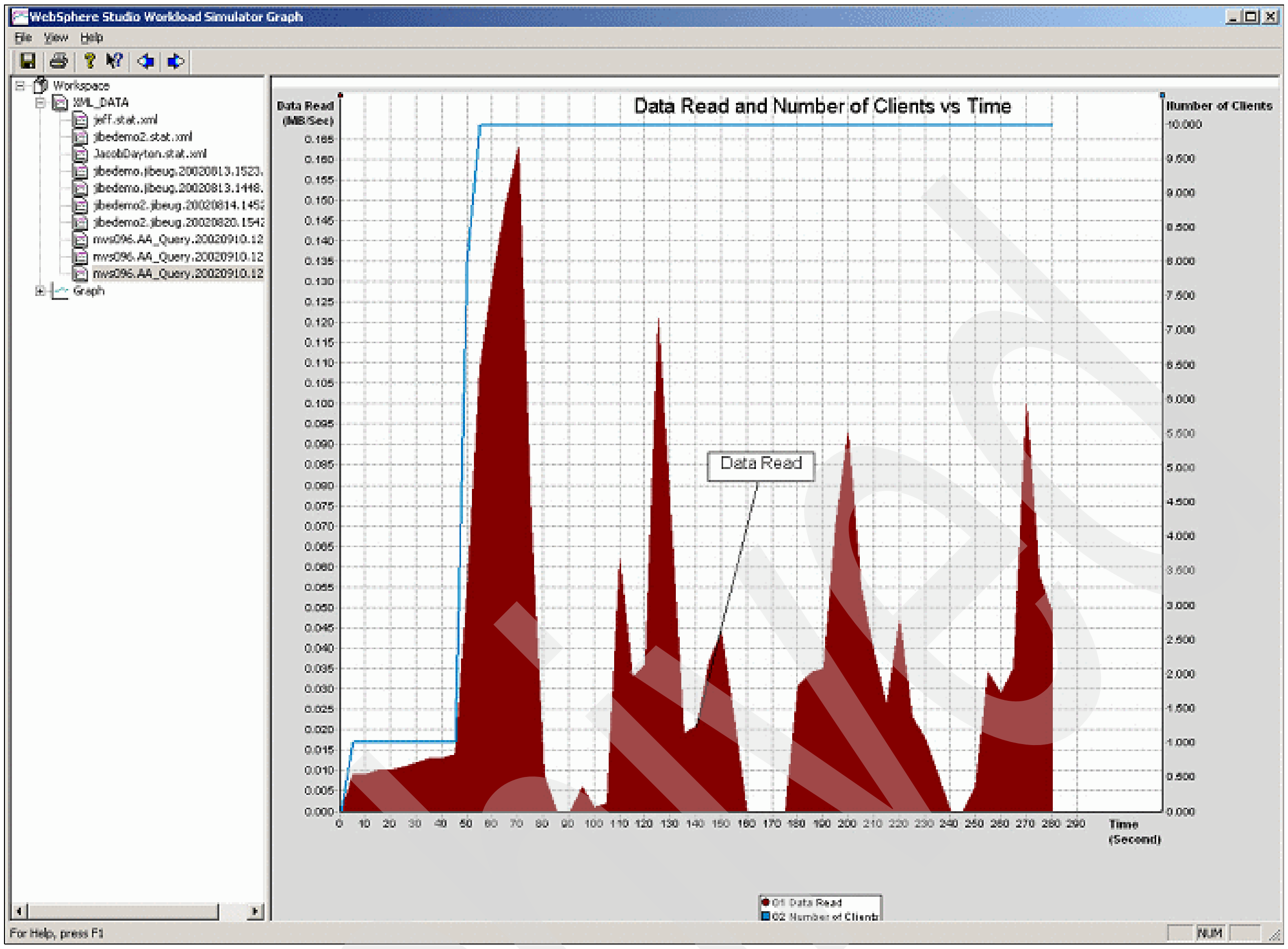


Figure 5-13 Sample simulation graph

See *WebSphere Studio Workload Simulator User's Guide*, SC31-6307, and *WebSphere Studio Workload Simulator Getting Started*, SC31-6383, on the WebSphere Studio Workload Simulator Library page for more information:

http://www.ibm.com/software/awdtools/studioworkloadsimulator/library

5.3.3 Microsoft Web Application Stress Tool

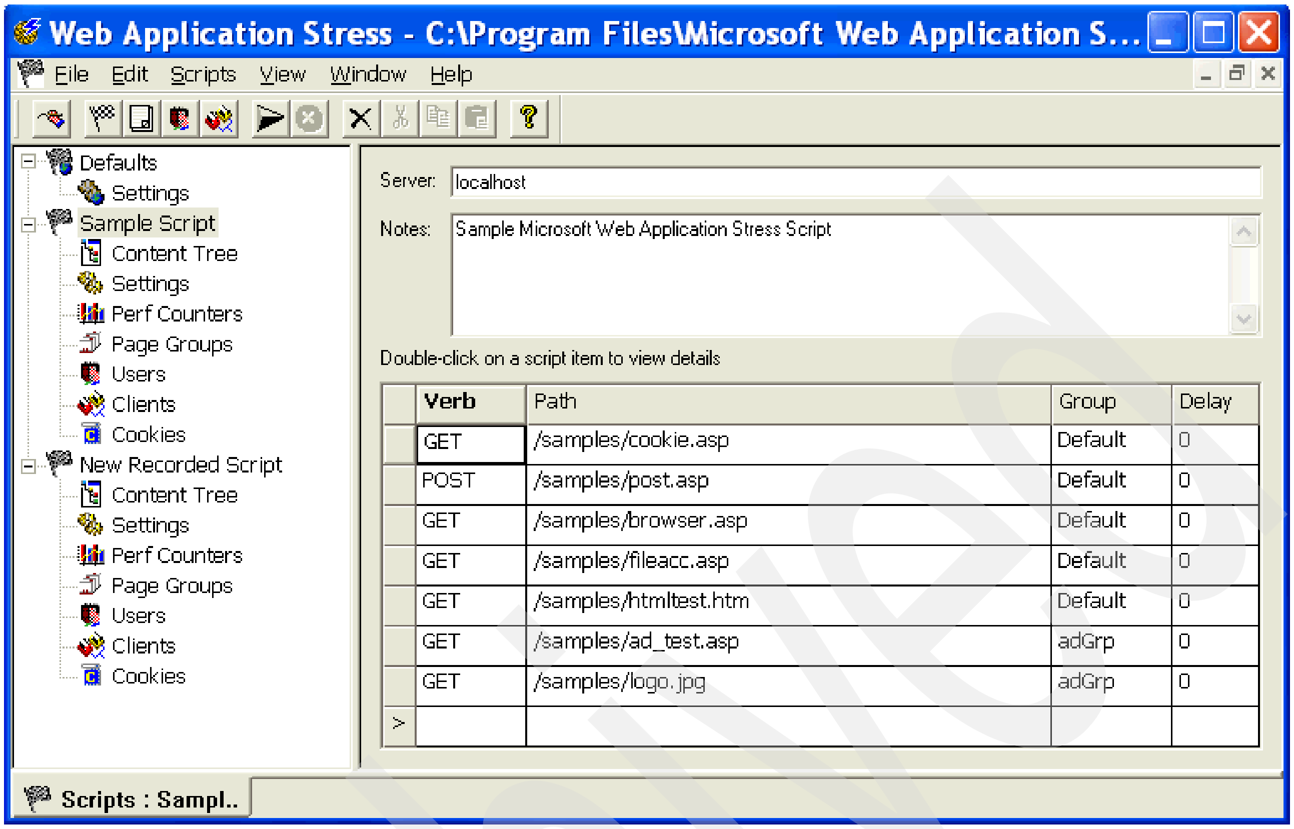
The Microsoft Web Application Stress Tool is designed to simulate multiple browsers requesting pages from a Web application. You can use this tool to gather performance and stability information about your Web application. It is extremely important to use a tool such as this to test an application and eliminate problems prior to deploying the application in a production environment.

You can download the Microsoft Web Application Stress Tool, which is free (licensed pursuant to an End User License Agreement which is available during the setup process), from this Microsoft Web site:

http://www.microsoft.com/technet/archive/itsolutions/intranet/downloads/webstres.mspx

Figure 5-14 on page 94 shows a screen capture[2] of the Microsoft Web Application Stress Tool in use.

[2] Microsoft product screen shot reprinted with permission from Microsoft Corporation.



Server: localhost

Notes: Sample Microsoft Web Application Stress Script

Double-click on a script item to view details

Verb	Path	Group	Delay
GET	/samples/cookie.asp	Default	0
POST	/samples/post.asp	Default	0
GET	/samples/browser.asp	Default	0
GET	/samples/fileacc.asp	Default	0
GET	/samples/htmltest.htm	Default	0
GET	/samples/ad_test.asp	adGrp	0
GET	/samples/logo.jpg	adGrp	0

Figure 5-14 Microsoft Web Application Stress Tool

Microsoft Visual Studio® .NET Edition ships with a license for a tool called Application Center Test 1.0 that has similar functionality and that is easy to use. To learn more, visit:

`http://msdn.microsoft.com/library/default.asp?url=/library/en-us/act/htm/actml_main.asp`

5.4 FTP, Telnet, and editors

For the day-to-day tasks in WebSphere for z/OS administration, you need powerful and efficient tools for even small tasks, such as transferring or editing files. There are numerous tools available on the Web. Several of them are especially useful for managing Java, XML, or HTML files in the WebSphere for z/OS environment. Although not directly related to problem determination, for your convenience, we provide more details about the following tools:

- TeraTerm Pro
- WS_FTP Professional
- Directing SYSPRINT output to an HFS File
- UltraEdit

5.4.1 TeraTerm Pro

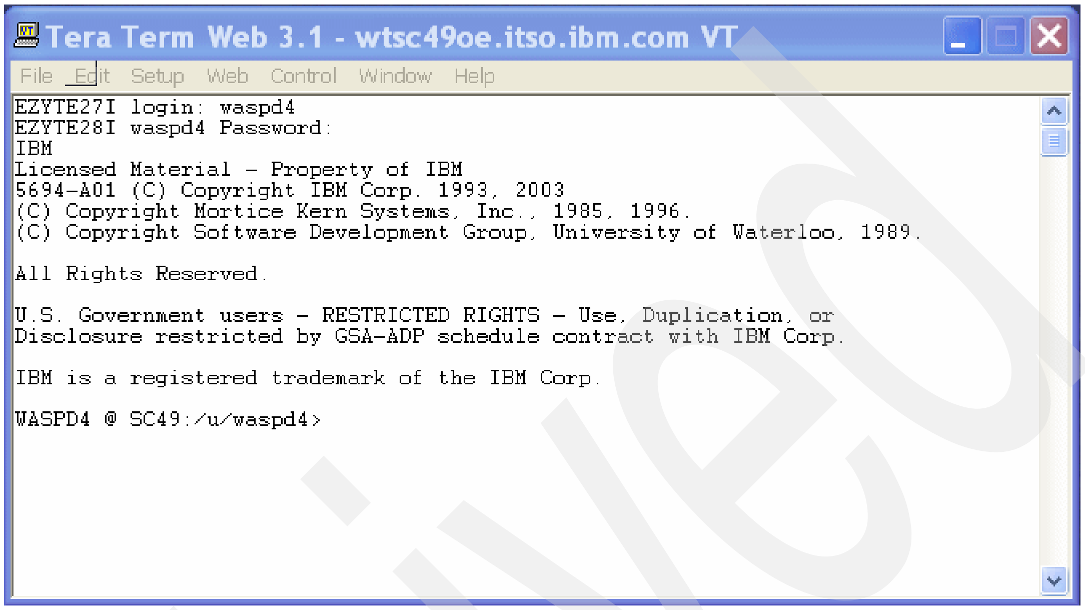
TeraTerm Pro is a free software terminal emulator (communication program) for Microsoft Windows. It provides:

- VT100 emulation
- Selected VT200/300 emulation
- TEK4010 emulation
- Kermit, XMODEM, ZMODEM, B-PLUS, and Quick-VAN file transfer protocols

You can download the application from the following URL:

`http://www.tucows.com/preview/195282.html`

Figure 5-15 shows the TeraTerm Pro emulator window.


Tera Term Web 3.1 - wtsc49oe.itso.ibm.com VT
File Edit Setup Web Control Window Help

```
EZYTE27I login: waspd4
EZYTE28I waspd4 Password:
IBM
Licensed Material - Property of IBM
5694-A01 (C) Copyright IBM Corp. 1993, 2003
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

WASPD4 @ SC49:/u/waspd4>
```

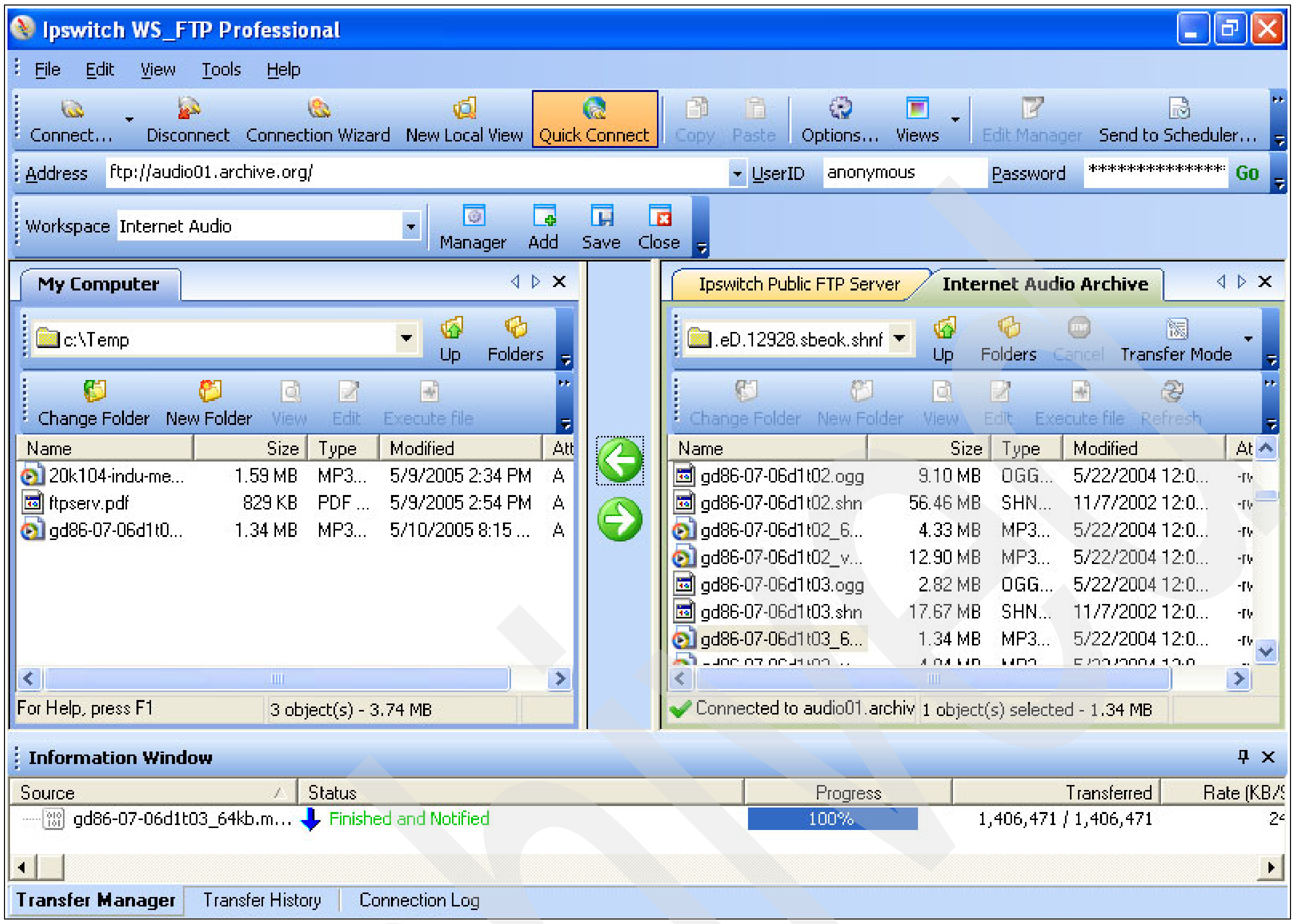
Figure 5-15 TeraTerm Pro

5.4.2 WS_FTP Professional

Ipswitch WS_FTP Professional is an FTP client program and a smart and secure way to share and transfer files. It is a fully licensed product. A free 30-trial evaluation version is available from the Web site:

`http://www.ipswitch.com`

You can use it to transfer files to and from z/OS easily, as shown in Figure 5-16 on page 96.



Address: ftp://audio01.archive.org/ UserID: anonymous Password: **************

Workspace: Internet Audio

c:\Temp

Name	Size	Type	Modified	Att
20k104-indu-me...	1.59 MB	MP3...	5/9/2005 2:34 PM	A
ftpserv.pdf	829 KB	PDF ...	5/9/2005 2:54 PM	A
gd86-07-06d1t0...	1.34 MB	MP3...	5/10/2005 8:15 ...	A

For Help, press F1 3 object(s) - 3.74 MB

.eD.12928.sbeok.shnf

Name	Size	Type	Modified	At
gd86-07-06d1t02.ogg	9.10 MB	OGG...	5/22/2004 12:0...	-rv
gd86-07-06d1t02.shn	56.46 MB	SHN...	11/7/2002 12:0...	-rv
gd86-07-06d1t02_6...	4.33 MB	MP3...	5/22/2004 12:0...	-rv
gd86-07-06d1t02_v...	12.90 MB	MP3...	5/22/2004 12:0...	-rv
gd86-07-06d1t03.ogg	2.82 MB	OGG...	5/22/2004 12:0...	-rv
gd86-07-06d1t03.shn	17.67 MB	SHN...	11/7/2002 12:0...	-rv
gd86-07-06d1t03_6...	1.34 MB	MP3...	5/22/2004 12:0...	-rv

Connected to audio01.archiv 1 object(s) selected - 1.34 MB

Source	Status	Progress	Transferred	Rate (KB/
gd86-07-06d1t03_64kb.m...	Finished and Notified	100%	1,406,471 / 1,406,471	2

Figure 5-16 Example of WS_FTP Professional)

5.4.3 Directing SYSPRINT output to an HFS File

Many WebSphere for z/OS customers that are familiar with a UNIX or Microsoft Windows NT environment are somewhat reluctant to use SDSF to view SYSPRINT output from their application server regions. They would much rather use a familiar editor (such as VI) in a Telnet session to view the STDOUT and STDERR information directed to SYSPRINT. For information about how to redirect SYSPRINT to HFS files so that they can be viewed with common editors and related topics, refer to *Directing SYSPRINT Output to an HFS File in WebSphere for z/OS*, TD101087, on the IBM Techdocs Web site:

`http://www.ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs`

5.4.4 UltraEdit

UltraEdit is a text editor, hexadecimal editor, HTML editor, and programmer editor. You can download it from the Web site for IDM Computer Solutions, Inc.:

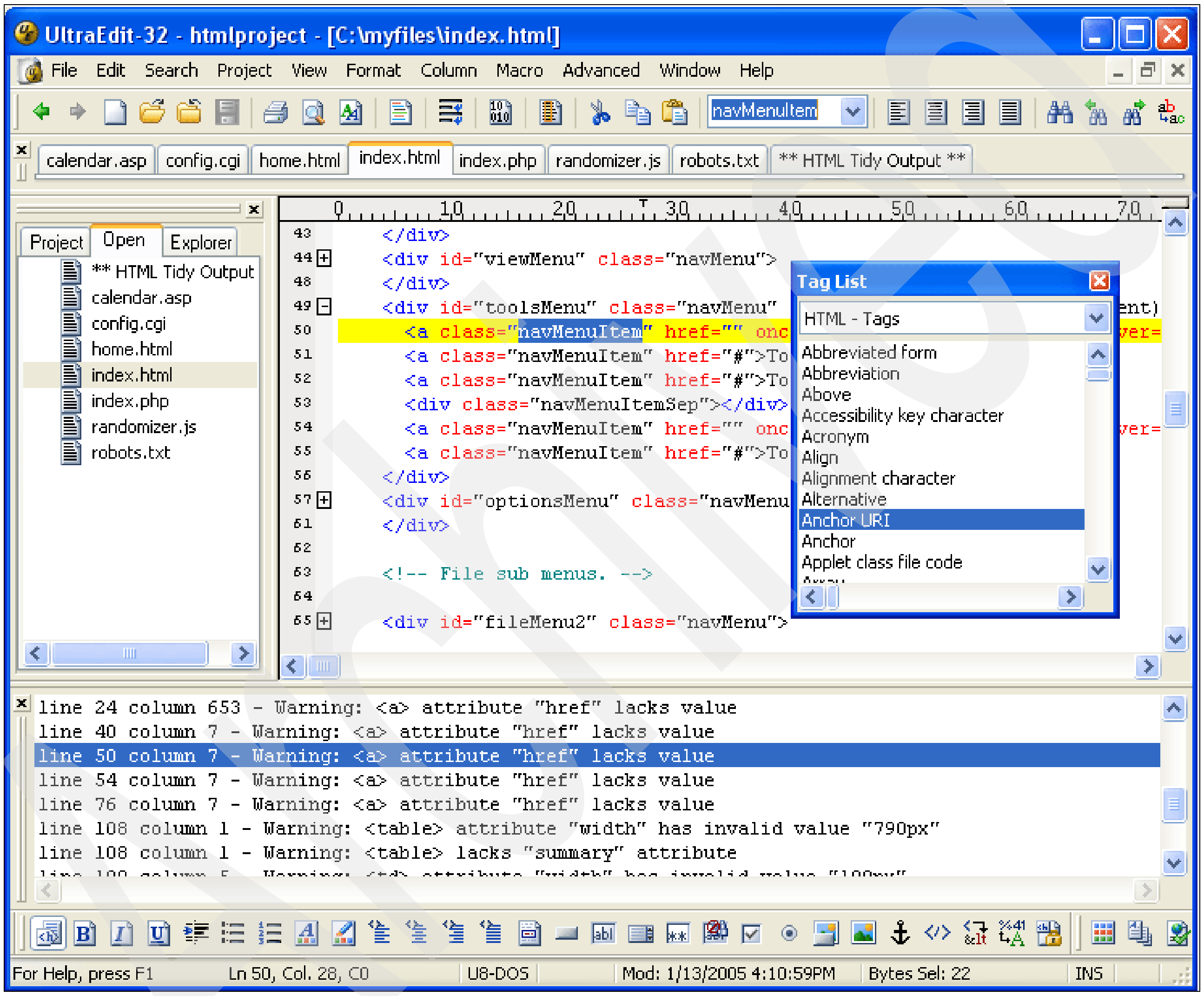
`http://www.ultraedit.com/index.php`

Some of the more popular features of this editor are:

- You can edit files remotely using FTP. This is especially useful when working with WebSphere for z/OS. You can easily edit traces, logs, configuration files, and so on that are data sets or HFS files in z/OS.

- You can edit or compare files in binary, hexadecimal, ASCII, and so on with:
  - Easy management of the search utility (when you look for a string in a log, a window shows all the lines that contains the string)
  - User-configurable syntax highlighting specific to the language that is being edited (Java, HTML, XML, C/C++, and so on).
  - Column mode and useful macros

Figure 5-17 shows an HTML file in the UltraEdit editor.



```
</div>
<div id="viewMenu" class="navMenu">
</div>
<div id="toolsMenu" class="navMenu"                              ent)
  <a class="navMenuItem" href="" onc                             ver=
  <a class="navMenuItem" href="#">To
  <a class="navMenuItem" href="#">To
  <div class="navMenuItemSep"></div>
  <a class="navMenuItem" href="" onc                             ver=
  <a class="navMenuItem" href="#">To
</div>
<div id="optionsMenu" class="navMenu
</div>

<!-- File sub menus. -->

<div id="fileMenu2" class="navMenu">
```

```
line 24 column 653 - Warning: <a> attribute "href" lacks value
line 40 column 7 - Warning: <a> attribute "href" lacks value
line 50 column 7 - Warning: <a> attribute "href" lacks value
line 54 column 7 - Warning: <a> attribute "href" lacks value
line 76 column 7 - Warning: <a> attribute "href" lacks value
line 108 column 1 - Warning: <table> attribute "width" has invalid value "790px"
line 108 column 1 - Warning: <table> lacks "summary" attribute
line 109 column 5 - Warning: <td> attribute "width" has invalid value "100px"
```

```
For Help, press F1    Ln 50, Col. 28, C0    U8-DOS    Mod: 1/13/2005 4:10:59PM    Bytes Sel: 22    INS
```

Figure 5-17 UltraEdit

A

Messages and codes

This appendix provides messages and codes for WebSphere Application Server components and subsystems for z/OS to support you in analyzing errors and problems.

We explain the format of WebSphere for z/OS message codes, list specific Java component messages, mention minor codes, and provide WebSphere for z/OS related abend codes. We also name the most common non-WebSphere for z/OS-related message prefixes with details about where they come from.

The following tables are summaries from the *WebSphere Application Server for z/OS V5.1: Messages and Codes*, GA22-7915, and the *WebSphere for z/OS Information Center*, available from:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

A.1 WebSphere for z/OS message codes

The prefix for WebSphere for z/OS messages is BBO. The format is BBOcnnnnt. Table A-1 provides the WebSphere for z/OS message formats.

Table A-1 WebSphere for z/OS message formats

Message format	Description
BBO	Identifies it as a WebSphere for z/OS message
DYNA	Identifies it as a WebSphere for z/OS Dynamic Fragment Cache message
c	Indicates the component
nnnn	A unique identifier
t	Severity (Information, Warning, or Error)

Table A-2 gives an overview of where BBO messages come from and where they appear.

Table A-2 WebSphere for z/OS messages overview

Prefix	Come from	Appear on or in
BBOJnnnnt	Java Virtual Machine (JVM)	Operator's console error log, job log
BBOMnnnnt	Runtime environment	Operator's console, error log, job log
BBOOnnnnt	Control process, servant process, daemon, CORBA (these are general messages)	Operator's console, error log, job log
BBOSnnnnt	Security system	Operator's console, error log, job log
BBOTnnnnt	Transaction service	Operator's console, error log, job log
DYNAnnnnt	Dynamic Fragment Cache.\	Job log

To look up specific message codes follow these steps:

1. In the Information Center navigation panel, click **WebSphere Application Server for z/OS V6** to see the table of contents.
2. Select **Reference** → **Troubleshooter** → **Messages**.
3. Then select the tab according to the first few letters in your message code.

You can also search for the specific message or code with the search function at the top of the window.

A.1.1 Specific Java component messages

We include the Java component messages that are prefixed by the BBOO0222I message in Table A-3 on page 101 to provide a quick reference for your convenience. (Mgmt stands for Management.)

Table A-3 BBOO0222I message components

Msg	Component	Msg	Component	Msg	Component
ACIN	Access Intent	**CWSIY**	Y SIBus Mediation Handlers	**PMON**	PMI, Tivoli Performance Viewer
ACWA	Work Area	**CWSIZ**	Z SIBus Mediation Framework	**PMRM**	Performance Monitoring Request Metrics
ADFS	Mgmt File Service Subsystem	**CWSJA**	A Admin	**PMWC**	PME Edition Support
ADMA	Application Deployment	**CWSJB**	B inter-bus messaging engine	**PROC**	Process Mgmt and Spawning Facility
ADMB	Mgmt Config Archive Subsystem	**CWSJC**	C SIBus Core SPI	**PROX**	Proxy
ADMC	Mgmt Connector Subsystem	**CWSJD**	D Admin	**SCHD**	Scheduler
ADMD	Mgmt Process Discovery				
ADME	Mgmt Event Subsystem	**CWSJO**	O SDO Repository Component	**SECG**	WEBUI SecurityCenter
ADMF	Mgmt Command Framework	**CWSJQ**	Q MFP MQ interoperability component	**SECJ**	Security
ADMG	Mgmt Connector Subsystem	**CWSJR**	R SIBus	**SESN**	Session and User Profiles
ADMK	Mgmt Utilities	**CWSJU**	U Jetstream Message Tracing	**SIEG**	Example
ADML	Mgmt Process Launching Tool	**CWSJW**	W WLM Classifier	**SOAP**	SOAP Support
ADMN	Activity Service	**CWSWS**	S SIBus Web Services	**SRMC**	Service Reference ManagerTransactions
ADMR	Mgmt Repository			**SRVE**	Transactions
ADMS	Mgmt Subsystem			**SSLC**	SSL Channel
ADMU	Mgmt Utilities	**CWUDD**	Web Services UDDI Deployment & Removal	**STFF**	Staff Support Service
ADNT	Adaptive Entity	**CWUDG**	UDDI User Console	**STUP**	Startup Beans
APPR	Application Profile	**CWUDM**	UDDI Mgmt Interface	**TCPC**	TCP Channel
ASYN	Asynchronous Beans	**CWUDN**	UDDI Node Manager	**TRAS**	Trace Facility
BBOJ	EJB Container	**CWUDQ**	UDDI Migration	**TUNE**	Perform Auto-Tuning Support
BBOM	Naming	**CWUDR**	UDDI Logging and Tracing	**UDAI**	UDDI API
BBOO	Runtime, Web	**CWUDS**	UDDI SOAP Interface	**UDCF**	UDDI Configuration
BBOS	Security			**UDDA**	UDDI Data Types
BBOT	OTS and RRS			**UDDM**	UDDI DOM
BBZW	WBI SF Install	**CWUDT**	UDDI Registry Transaction Manager	**UDEJ**	UDDI EJB Interface
BCDS	Business Context Data Service for Event Infrastructure	**CWUDU**	UDDI Utility Tools	**UDEX**	UDDI Exceptions
BNDE	Binding EJB References	**CWUDV**	UDDI Value Set Tools	**UDIN**	UDDI Installation
CHFW	Channel Framework	**CWUDX**	Web Services JAXR	**UDLC**	UDDI Local API
CHKC	Event Infrastructure Validation	**CWWCW**	W Validation	**UDPR**	UDDI Persistence
CHKP	PME Validation	**CWWDR**	R Data Replication Service	**UDRS**	UDDI Logging
		CWWSG	G Web Service Gateway	**UDSC**	UDDI Security
		DCSV	DCS	**UDSP**	UDDI SOAP Interface
				UDUC	UDDI User Console
CHKS	SIB Validation	**DSRA**	Resource Adapters	**UDUT**	UDDI Utility Tools
CHKW	Validation	**DWCT**	Dynamic Workload Mgmt Client	**UDUU**	UDDI UUID
CHKX	XD Validation	**DYNA**	Dynacache	**UTLS**	Utilities
CMPN	Compensation	**EAAT**	Placeholder	**WACS**	Activity Session Service
CNTR	EJB Container	**ECNS**	Entity Change Notification Service	**WACT**	Activity Service
CONM	Connection Manager	**ESOP**	State Observer Plug-in for Event Infrastructure	**WASX**	Non WSCP Scripting
CSCP	CScope Service			**WBIA**	Support for Business Integration Adapters
CWRCB	B Core Group Bridge			**WHFW**	Handler Framework
CWSIA	A Service Iintegration Bus				

Msg	Component	Msg	Component	Msg	Component
CWSIB	B SIBus Common	GWIN	Web Services Gateway	WKSP	Work Space
CWSIC	C Communications	HMGR	HA Manager	WKSQ	Workspace Query Utilities
CWSID	D Admin	HTPC	HTTP Channel	WLTC	Transaction Monitor
CWSIE	E SIBus Externals	I18N	Internationalization Service	WMSG	Messaging Service
CWSIF	F SIBus MFP	ILMC	Instance Location Manager	WSBB	WsByteBuffer
CWSIH	H Jetstream MatchSpace	INST	Install	WSCL	WebSphere Client
CWSII	I Security	IVTL	Installation Verification Tool	WSCP	Non WSCP Scripting
CWSIJ	J COmmunications	J2CA	J2EE Connector	WSEC	Web Services Security
CWSIK	K SIBus Return Codes	JSAS	Security Association	WSGW	Web Services Gateway
CWSIL	L PSB	JSFG	jsf (bean class type)	WSIF	Web Services Invocation Framework
CWSIM	M SIBus Mediations SIMediationSession Interface	JSPG	Java Server Pages	WSSC	SOAP Channels
CWSIN	N SIBus Mediations Framework	JSSL	ORB SSL Extensions	WSSK	Web Services Security Kerberos
CWSIO	O SIBus Migration	LTXT	Localizable Text	WSVM	Validation Manager Implementation
CWSIP	P Jetstream Message Processor	MIGR	Release-to-Release	WSVR	Server Runtime
		MSGS	Migration Tooling	WSWS	Web Services
		NMSV	JMS Server		
		OBPL	Naming Service		
		ODCF	ObjectPool		
CWSIQ	Q MQFap Channel	ORBX	On Demand ConFiguration	WTRN	Transaction recovery
CWSIR	R SIBus Core	PLGC	ORB Extensions	WUDU	WebUI Deployment Descriptor Utilities
CWSIS	S MessageStore	PLGN	Plug-in Configuration Generator	WUPD	Update Installer
CWSIT	T TRM	PLPR	Transactions Plug-in Processor	WVER	Product History Information
CWSIU	U Utilities	PMGR	Persistence Manager	WWLM	WLM Client
CWSIV	V SIBus Resource Adapter	PMI	PMI	XMEM	XMem Channel
CWSIW	W SIBus Mediations				
CWSIX	X SIBus Mediations				

A.1.2 Minor codes

A minor code is shown in the WebSphere for z/OS error log and SYSPRINT. It is a hexadecimal value with the format `C9C2nnnn`, where:

C9C2	Identifies the code as WebSphere for z/OS
nnnn	Uniquely identifies the code

A minor code is often associated with an exception, as shown in Example A-1.

Example: A-1 Exception with minor code

```
Trace: 2004/10/10 13:37:59.801 01 t=8BD0F0 c=UNK key=S2 (00000004)
   Description: Throw CORBA system exception
   exception id: CORBA::INTERNAL
   minor code: c9c2110f
   from filename: ./bboosyse.cpp
   at line: 719
```

Some of the minor code meanings are described in *WebSphere for z/OS Information Center* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp

Important: Error (minor) codes not listed in the *WebSphere for z/OS Information Center* should be reported directly to the IBM Support Center.

A.1.3 Abends

Table A-4 shows the WebSphere for z/OS related abend codes.

Table A-4 WebSphere related abend codes

Abend code	Issuer
CC3	Daemon processing failure
DC3	Controller region processing failure
EC3	Servant region processing failure

Some reason codes are also passed along with these abend codes. They are described in detail at the Information Center. Search for "Abend (reason) codes."

Table A-5 shows an example.

Table A-5 Example abend code and related reason code

Abend code	Abend reason	Explanation	Suggested action
`CC3`	000C0009	An exception occurred on the main thread of execution, probably during initialization. The address space is abended with this code to cause the space to terminate.	Further information about the exception should be found in the joblog for the space and also possibly in the error log.

If no explanation is given in the reason code, and no indication is found in any information source, the problem should be reported to IBM.

A.2 System and component message table

Table A-6 shows the most common non-WebSphere for z/OS-related message prefixes and where they come from.

Table A-6 System and component messages

Prefix	Product	Message structure
DSN	IBM DB2 UDB for z/OS	DSNcnnnt *c*: Subcomponent identifier *nnn*: Unique numeric identifier *t*: Type with I - information, A - immediate action, D - immediate decision, E - eventual action Example: `DSNB209I` Source: *DB2 UDB for z/OS Version 8 Messages and Codes*, GC18-7422
DB2 Information Center topic "Messages and Codes," available at: `http://publib.boulder.ibm.com/infocenter/dzichelp/index.jsp`		
EZA EZB EZD EZY EZZ SNM ™	Communications Server (TCP/IP)	pppnnnnt *ppp*: Prefix *nnnn*: Unique identifier *t*: Type with A - immediate action, E - eventual action, D - immediate decision, I - information Example: `EZZ0902I` Source: *z/OS V1.6 Communications Server: IP Messages: Volume 1-4*, GC31-8783/4/5/6

Prefix	Product	Message structure
ICH	Security Server (RACF®)	ICHcnnt *c*: Identifies the RACF function, where: 0: SAF initialization 3: RACROUTE REQUEST=VERIFY macro 4: RACF processing 5: RACF initialization 7: RACF status 8: RACROUTE REQUEST=AUTH macro 9: RACROUTE REQUEST=DEFINE macro *nn*: Message serial number *t*: Type, where: A - action; operator must perform a specific action. D - decision; operator must choose an alternative. E - eventual action required. I - information. W - Wait; processing stops. Example: `ICH500I` Source: *z/OS V1R6.0 Security Server RACF Messages and Codes*, SA22-7686
ISP ISR FLM	ISPF, PDF SCLM	pppannna *ppp*: Prefix *a*: Alphabetic character *nnn*: Unique identifier Example: `ISPA001` Source: *z/OS V1R6.0 ISPF Messages and Codes*, SC34-4815
IKJ	TSO/E	pppccnnnt *ppp*: Prefix *cc*: System module prefix (in decimal) *nnn*: Message serial number identifying the program that issued the message *t*:- Type, where: A - action; the terminal user must perform the action specified in the message text. E - error; processing terminates. I - information; no action is required. Example: `IKJ55112E` Source: *z/OS V1R6.0 TSO/E Messages*, SA22-7786
IRX	TSO REXX processing	pppccnnt *ppp*: Prefix *cc*: System module prefix (in decimal) *nnn*: Message serial number identifying the program that issued the message *t*: Type, where: E - error; processing terminates. I - information; no action is required. Example: `IRX0042I` Source: *z/OS V1R6.0 TSO/E Messages*, SA22-7786

Prefix	Product	Message structure
IWM	Workload Manager (WLM)	pppnnnt *ppp*: Prefix *nnn*: Message serial number *t*: Type, where: A - action by operator, D - decision by operator, E - eventual action by operator, I - information for operator/programmer, S - severe error, W - wait for operator action Example: `IWM003I` Source: *z/OS V1R6.0 MVS System Messages, Vol 9 (IGF-IWM)*, SA22-7639
CEE EDC	z/OS Language Environment Runtime C/C++ Runtime	pppnnnnt *ppp*: Prefix *nnnn*: Message serial number *t*: Type, where: I - informational message, W - warning message, E - error message, S - severe error message, C - critical error message Example: `CEE0252W` Source: *z/OS V1R1 Language Environment Run-Time Messages*, SA22-7566
GIM	SMP/E	pppnnnnnt *ppp*: Prefix *nnnnn*: Message serial number *t*: Type, where: I - informational, W - warning, E - error, S - severe, T - terminating Example: `GIM20101S` Source: *z/OS V1R1 SMP/E Messages, Codes, and Diagnosis*, GA22-7770
FDB FOM FSUM	UNIX System Services (USS) Debugger USS Shell & Utilities	pppcnnnn *ppp*: Prefix *c*: Component identifier *nnnn*: Unique identifier Example: `FOMC1013` Source: *z/OS V1R6.0 UNIX System Services Messages and Codes*, SA22-7807
IXG	System Logger	pppnnnt *ppp*: Prefix *nnnnn*: Message serial number *t*: Type, where: I - informational message, E - recoverable error, W - warning, S - serious error, T - terminating Example: `IXG004I` Source: *z/OS V1R6.0 MVS System Messages, Vol 10 (IXC-IZP),* SA22-7640
ATR	Resource Recovery Services (RRS)	pppnnnt *ppp*: Prefix *nnnn*: Message serial number *t*: Type, where: I - informational message, E - recoverable error, W - warning, S - serious error, T - terminating Example: `ATR120I` Source: *z/OS V1R6.0 MVS System Messages, Vol 3 (ASB-BPX),* SA22-7633

Prefix	Product	Message structure
IMW	HTTP Server	pppnnnnnt *ppp*: Prefix *nnnnn*: Message serial number *t*: Type, where: I - informational message, E - recoverable error, W - warning, S - serious error Message ID ranges: Components: IMW0001-IMW2000 - IMWHTTPD IMW2000-IMW2500 - Proxy Server IMW3501-IMW3700 - CONSOLE IMW3701-IMW3999 - HTCounter IMW4000-IMW5000 - HTIMAGE IMW5001-IMW6000 - HTADM IMW6100-IMW6900 - SSL Security Example: `IMW0442E` Source: *IBM HTTP Server Planning, Installing, and Using*, SC34-4826
z/OS Internet Library, available at: `http://www.ibm.com/servers/eserver/zseries/zos/bkserv/`		

Index

WebSphere Application Server for z/OS Problem Determination Means and Tools

Redpaper

Useful WebSphere for z/OS commands

Helpful logs, traces, and dumps

Guide to diagnostic means and tools

For the day-to-day administration tasks for IBM WebSphere for z/OS, you need powerful and efficient tools for analyzing problems, for finding their root causes, and for solving them.

This IBM Redpaper is intended to be a reference guide to the means and tools that can be used in the problem determination process for WebSphere for z/OS.

In this paper, we introduce especially helpful commands for the WebSphere for z/OS environment; WebSphere for z/OS logs, traces and dumps; diagnostic tools for problem determination in WebSphere for z/OS; and other helpful tools.

For each of these means, we provide information about its nature, explain when to use it, and demonstrate how to use it. We describe the output, show how to interpret it, and provide an example.